

An open source library for quantitative finance

Version 0.3.0b1-20020322

Generated by Doxygen 1.2.14

22 Mar 2002

Contents

I	User Manual	1
1	An introduction to QuantLib	3
1.1	Introduction	3
1.2	Project overview	4
1.3	Where to get QuantLib	5
1.4	Installation	6
1.5	Usage	7
1.6	Supported platforms	8
1.7	Version history	9
1.8	Todo List	12
1.9	Additional resources	13
1.10	The QuantLib Group	14
1.11	Copyright and License	15
2	QuantLib components	17
2.1	Core classes	17
2.2	Date and time calculations	18
2.3	The finite differences framework	19
2.4	The Monte Carlo framework	24
2.5	The interest rate modelling framework	30
2.6	Currencies and FX rates	32
2.7	Instruments and pricers	33
2.8	Math tools	34
2.9	Design patterns	35
2.10	Term structures	36
2.11	Utilities	37
3	Examples	39
3.1	QuantLib Examples	39

II	Reference Manual	41
4	QuantLib Module Index	43
4.1	QuantLib Modules	43
5	QuantLib Namespace Index	45
5.1	QuantLib Namespace List	45
6	QuantLib Hierarchical Index	47
6.1	QuantLib Class Hierarchy	47
7	QuantLib Compound Index	55
7.1	QuantLib Compound List	55
8	QuantLib File Index	61
8.1	QuantLib File List	61
8.2	QuantLib Related Pages	67
9	QuantLib Module Documentation	69
9.1	Deprecated classes	69
9.2	Global QuantLib macros	70
9.3	Math functions	71
9.4	Numeric limits	72
9.5	Time functions	73
9.6	Character functions	74
9.7	Input/output functions	75
9.8	Min and max functions	76
9.9	Template capabilities	77
9.10	Iterator support	78
10	QuantLib Namespace Documentation	79
10.1	QuantLib Namespace Reference	79
10.2	QuantLib::Calendars Namespace Reference	87
10.3	QuantLib::CashFlows Namespace Reference	88
10.4	QuantLib::DayCounters Namespace Reference	89
10.5	QuantLib::FiniteDifferences Namespace Reference	90
10.6	QuantLib::Indexes Namespace Reference	92
10.7	QuantLib::Instruments Namespace Reference	93
10.8	QuantLib::Math Namespace Reference	94

10.9 QuantLib::MonteCarlo Namespace Reference	96
10.10 QuantLib::Patterns Namespace Reference	98
10.11 QuantLib::Pricers Namespace Reference	99
10.12 QuantLib::RandomNumbers Namespace Reference	102
10.13 QuantLib::Solvers1D Namespace Reference	103
10.14 QuantLib::TermStructures Namespace Reference	104
10.15 QuantLib::Utilities Namespace Reference	105
11 QuantLib Class Documentation	107
11.1 QuantLib::DayCounters::Actual360 Class Reference	107
11.2 QuantLib::DayCounters::Actual365 Class Reference	108
11.3 QuantLib::DayCounters::ActualActual Class Reference	109
11.4 QuantLib::Pricers::AnalyticalCapFloor Class Reference	110
11.5 QuantLib::Arguments Class Reference	111
11.6 QuantLib::MonteCarlo::ArithmeticAPOPathPricer Class Reference	112
11.7 QuantLib::MonteCarlo::ArithmeticASOPathPricer Class Reference	113
11.8 QuantLib::Array Class Reference	114
11.9 QuantLib::AssertionFailedError Class Reference	117
11.10 QuantLib::Indexes::AUDLibor Class Reference	118
11.11 QuantLib::Pricers::BarrierOption Class Reference	119
11.12 QuantLib::MonteCarlo::BasketPathPricer Class Reference	120
11.13 QuantLib::Math::BilinearInterpolation Class Template Reference	121
11.14 QuantLib::Pricers::BinaryOption Class Reference	122
11.15 QuantLib::Solvers1D::Bisection Class Reference	123
11.16 QuantLib::Pricers::BlackCapFloor Class Reference	124
11.17 QuantLib::InterestRateModelling::BlackKarasinski Class Reference	125
11.18 QuantLib::InterestRateModelling::BlackModel Class Reference	126
11.19 QuantLib::BlackScholesProcess Class Reference	127
11.20 QuantLib::Pricers::BlackSwaption Class Reference	128
11.21 QuantLib::FiniteDifferences::BoundaryCondition Class Reference	129
11.22 QuantLib::RandomNumbers::BoxMullerGaussianRng Class Template Reference	130
11.23 QuantLib::Solvers1D::Brent Class Reference	131
11.24 QuantLib::FiniteDifferences::BSMOperator Class Reference	132
11.25 QuantLib::Indexes::CADLibor Class Reference	133
11.26 QuantLib::Calendar Class Reference	134
11.27 QuantLib::Calendar::CalendarImpl Class Reference	137
11.28 QuantLib::Calendar::WesternCalendarImpl Class Reference	138

11.29	QuantLib::CapFlatVolatilityStructure Class Reference	139
11.30	QuantLib::Volatilities::CapFlatVolatilityVector Class Reference	140
11.31	QuantLib::Instruments::CapFloorParameters Class Reference	141
11.32	QuantLib::Pricers::CapFloorPricingEngine Class Template Reference	142
11.33	QuantLib::Instruments::CapFloorResults Class Reference	143
11.34	QuantLib::CapletForwardVolatilityStructure Class Reference	144
11.35	QuantLib::CashFlow Class Reference	145
11.36	QuantLib::Indexes::CHFLibor Class Reference	146
11.37	QuantLib::RandomNumbers::CLGaussianRng Class Template Reference	147
11.38	QuantLib::Pricers::CliquetOption Class Reference	148
11.39	QuantLib::Utilities::combining_iterator Class Template Reference	149
11.40	QuantLib::CompositeMarketElement Class Template Reference	151
11.41	QuantLib::Optimization::ConjugateGradient Class Reference	152
11.42	QuantLib::Pricers::ContinuousGeometricAPO Class Reference	153
11.43	QuantLib::Optimization::CostFunction Class Reference	154
11.44	QuantLib::Utilities::coupling_iterator Class Template Reference	155
11.45	QuantLib::CashFlows::Coupon Class Reference	157
11.46	QuantLib::FiniteDifferences::CrankNicolson Class Template Reference	159
11.47	QuantLib::Math::CubicSpline Class Template Reference	160
11.48	QuantLib::CurrencyFormatter Class Reference	161
11.49	QuantLib::Date Class Reference	162
11.50	QuantLib::DateFormatter Class Reference	164
11.51	QuantLib::DayCounter Class Reference	165
11.52	QuantLib::DayCounter::DayCounterImpl Class Reference	167
11.53	QuantLib::TermStructures::DepositRateHelper Class Reference	168
11.54	QuantLib::DerivedMarketElement Class Template Reference	170
11.55	QuantLib::DiffusionProcess Class Reference	171
11.56	QuantLib::DiscountStructure Class Reference	172
11.57	QuantLib::Pricers::DiscreteGeometricAPO Class Reference	173
11.58	QuantLib::Pricers::DiscreteGeometricASO Class Reference	174
11.59	QuantLib::FiniteDifferences::DMinus Class Reference	175
11.60	QuantLib::DoubleFormatter Class Reference	176
11.61	QuantLib::FiniteDifferences::DPlus Class Reference	177
11.62	QuantLib::FiniteDifferences::DPlusDMinus Class Reference	178
11.63	QuantLib::FiniteDifferences::DZero Class Reference	179
11.64	QuantLib::Error Class Reference	180

11.65	QuantLib::Indexes::Euribor Class Reference	181
11.66	QuantLib::EuroFormatter Class Reference	182
11.67	QuantLib::Pricers::EuropeanEngine Class Reference	183
11.68	QuantLib::Pricers::EuropeanOption Class Reference	184
11.69	QuantLib::MonteCarlo::EuropeanPathPricer Class Reference	185
11.70	QuantLib::MonteCarlo::EverestPathPricer Class Reference	186
11.71	QuantLib::Exercise Class Reference	187
11.72	QuantLib::FiniteDifferences::ExplicitEuler Class Template Reference	188
11.73	QuantLib::InterestRateModelling::ExtendedVasicek Class Reference	189
11.74	QuantLib::Solvers1D::FalsePosition Class Reference	191
11.75	QuantLib::Pricers::FdAmericanOption Class Reference	192
11.76	QuantLib::Pricers::FdBermudanOption Class Reference	193
11.77	QuantLib::Pricers::FdBsmOption Class Reference	194
11.78	QuantLib::Pricers::FdDividendEuropeanOption Class Reference	196
11.79	QuantLib::Pricers::FdDividendShoutOption Class Reference	197
11.80	QuantLib::Pricers::FdEuropean Class Reference	198
11.81	QuantLib::Pricers::FdStepConditionOption Class Reference	199
11.82	QuantLib::Utilities::filtering_iterator Class Template Reference	200
11.83	QuantLib::FiniteDifferences::FiniteDifferenceModel Class Template Reference	201
11.84	QuantLib::CashFlows::FixedRateCoupon Class Reference	202
11.85	QuantLib::CashFlows::FixedRateCouponVector Class Reference	203
11.86	QuantLib::CashFlows::FloatingRateCoupon Class Reference	204
11.87	QuantLib::CashFlows::FloatingRateCouponVector Class Reference	206
11.88	QuantLib::ForwardRateStructure Class Reference	207
11.89	QuantLib::ForwardSpreadedTermStructure Class Reference	209
11.90	QuantLib::Calendars::Frankfurt Class Reference	211
11.91	QuantLib::TermStructures::FraRateHelper Class Reference	212
11.92	QuantLib::TermStructures::FuturesRateHelper Class Reference	214
11.93	QuantLib::Indexes::GBPLibor Class Reference	215
11.94	QuantLib::InterestRateModelling::GeneralBlackKarasinski Class Reference	216
11.95	QuantLib::InterestRateModelling::GeneralCoxIngersollRoss Class Reference	218
11.96	QuantLib::MonteCarlo::GeometricAOPPathPricer Class Reference	220
11.97	QuantLib::MonteCarlo::GeometricASOPPathPricer Class Reference	221
11.98	QuantLib::Grid Class Reference	222
11.99	QuantLib::Handle Class Template Reference	223
11.100	QuantLib::Calendars::Helsinki Class Reference	225

11.10QuantLib::MonteCarlo::HimalayaPathPricer Class Reference	226
11.10QuantLib::History Class Reference	227
11.10QuantLib::History::const_iterator Class Reference	230
11.10QuantLib::History::Entry Class Reference	231
11.10QuantLib::InterestRateModelling::HullWhite Class Reference	232
11.10QuantLib::RandomNumbers::ICGaussianRng Class Template Reference	234
11.10QuantLib::IllegalArgumentError Class Reference	235
11.10QuantLib::IllegalResultError Class Reference	236
11.10QuantLib::FiniteDifferences::ImplicitEuler Class Template Reference	237
11.11QuantLib::ImpliedTermStructure Class Reference	238
11.11QuantLib::Index Class Reference	240
11.11QuantLib::IndexError Class Reference	241
11.11QuantLib::Instrument Class Reference	242
11.11QuantLib::IntegerFormatter Class Reference	245
11.11QuantLib::Math::Interpolation Class Template Reference	246
11.11QuantLib::Math::Interpolation2D Class Template Reference	248
11.11QuantLib::Pricers::JamshidianSwaption Class Reference	250
11.11QuantLib::Calendars::Johannesburg Class Reference	251
11.11QuantLib::Indexes::JPYLibor Class Reference	252
11.12QuantLib::RandomNumbers::KnuthUniformRng Class Reference	253
11.12QuantLib::Optimization::LeastSquareFunction Class Reference	254
11.12QuantLib::RandomNumbers::LecuyerUniformRng Class Reference	255
11.12QuantLib::Math::LexicographicalView Class Template Reference	256
11.12QuantLib::Math::LinearInterpolation Class Template Reference	258
11.12QuantLib::Link Class Template Reference	259
11.12QuantLib::Calendars::London Class Reference	261
11.12QuantLib::Utilities::lowest_category_iterator Struct Template Reference	262
11.12QuantLib::MarketElement Class Reference	263
11.12QuantLib::Math::Matrix Class Reference	264
11.13QuantLib::MonteCarlo::MaxBasketPathPricer Class Reference	267
11.13QuantLib::Pricers::McBasket Class Reference	268
11.13QuantLib::Pricers::McDiscreteArithmeticAPO Class Reference	269
11.13QuantLib::Pricers::McDiscreteArithmeticASO Class Reference	270
11.13QuantLib::Pricers::McEuropean Class Reference	271
11.13QuantLib::Pricers::McEverest Class Reference	272
11.13QuantLib::Pricers::McHimalaya Class Reference	273

11.13	QuantLib::Pricers::McMaxBasket Class Reference	274
11.13	QuantLib::Pricers::McPagoda Class Reference	275
11.13	QuantLib::Pricers::McPricer Class Template Reference	276
11.14	QuantLib::Calendars::Milan Class Reference	277
11.14	QuantLib::FiniteDifferences::MixedScheme Class Template Reference	278
11.14	QuantLib::InterestRateModelling::Model Class Reference	280
11.14	QuantLib::InterestRateModelling::ModelTermStructure Class Reference	282
11.14	QuantLib::MonteCarlo::MonteCarloModel Class Template Reference	283
11.14	QuantLib::MonteCarlo::MultiPath Class Reference	284
11.14	QuantLib::MonteCarlo::MultiPathGenerator Class Template Reference	285
11.14	QuantLib::Math::MultivariateAccumulator Class Reference	286
11.14	QuantLib::Solvers1D::Newton Class Reference	288
11.14	QuantLib::Solvers1D::NewtonSafe Class Reference	289
11.15	QuantLib::Calendars::NewYork Class Reference	290
11.15	QuantLib::Optimization::NonLinearLeastSquare Class Reference	291
11.15	QuantLib::Null Class Template Reference	293
11.15	QuantLib::ObjectiveFunction Class Reference	294
11.15	QuantLib::Patterns::Observable Class Reference	295
11.15	QuantLib::Patterns::Observer Class Reference	297
11.15	QuantLib::InterestRateModelling::OneFactorModel Class Reference	299
11.15	QuantLib::FiniteDifferences::OneFactorOperator Class Reference	300
11.15	QuantLib::Optimization::OptimizationMethod Class Reference	301
11.15	QuantLib::Optimization::OptimizationProblem Class Reference	303
11.16	QuantLib::Option Class Reference	304
11.16	QuantLib::OptionGreeks Class Reference	306
11.16	QuantLib::OptionPricingEngine Class Reference	307
11.16	QuantLib::OptionValue Class Reference	308
11.16	QuantLib::OrnsteinUhlenbeckProcess Class Reference	309
11.16	QuantLib::OutOfMemoryError Class Reference	310
11.16	QuantLib::MonteCarlo::PagodaPathPricer Class Reference	311
11.16	QuantLib::MonteCarlo::Path Class Reference	312
11.16	QuantLib::MonteCarlo::PathGenerator Class Template Reference	313
11.16	QuantLib::MonteCarlo::PathPricer Class Template Reference	314
11.17	QuantLib::Period Class Reference	315
11.17	QuantLib::TermStructures::PiecewiseFlatForward Class Reference	316
11.17	QuantLib::Instruments::PlainOption Class Reference	319

11.17	QuantLib::Pricers::PlainOptionEngine Class Reference	321
11.17	QuantLib::Instruments::PlainOptionParameters Class Reference	322
11.17	QuantLib::Instruments::PlainOptionResults Class Reference	323
11.17	QuantLib::PostconditionNotSatisfiedError Class Reference	324
11.17	QuantLib::PreconditionNotSatisfiedError Class Reference	325
11.17	QuantLib::Utilities::processing_iterator Class Template Reference	326
11.17	QuantLib::RandomNumbers::RandomArrayGenerator Class Template Reference	328
11.18	QuantLib::RateFormatter Class Reference	329
11.18	QuantLib::TermStructures::RateHelper Class Reference	330
11.18	QuantLib::RelinkableHandle Class Template Reference	332
11.18	QuantLib::Results Class Reference	334
11.18	QuantLib::Solvers1D::Ridder Class Reference	335
11.18	QuantLib::Math::RiskMeasures Class Reference	336
11.18	QuantLib::RiskStatistics Class Reference	337
11.18	QuantLib::MonteCarlo::Sample Struct Template Reference	339
11.18	QuantLib::Scheduler Class Reference	340
11.18	QuantLib::Solvers1D::Secant Class Reference	341
11.19	QuantLib::Math::SegmentIntegral Class Reference	342
11.19	QuantLib::CashFlows::ShortFloatingRateCoupon Class Reference	343
11.19	QuantLib::CashFlows::SimpleCashFlow Class Reference	345
11.19	QuantLib::SimpleMarketElement Class Reference	346
11.19	QuantLib::Instruments::SimpleSwap Class Reference	347
11.19	QuantLib::Optimization::Simplex Class Reference	348
11.19	QuantLib::Pricers::SingleAssetOption Class Reference	349
11.19	QuantLib::Solver1D Class Reference	351
11.19	QuantLib::Math::Statistics Class Reference	353
11.19	QuantLib::Optimization::SteepestDescent Class Reference	356
11.20	QuantLib::FiniteDifferences::StepCondition Class Template Reference	357
11.20	QuantLib::Utilities::stepping_iterator Class Template Reference	358
11.20	QuantLib::Instruments::Stock Class Reference	360
11.20	QuantLib::StringFormatter Class Reference	361
11.20	QuantLib::Instruments::Swap Class Reference	362
11.20	QuantLib::TermStructures::SwapRateHelper Class Reference	364
11.20	QuantLib::Instruments::Swaption Class Reference	366
11.20	QuantLib::Instruments::SwaptionParameters Class Reference	367
11.20	QuantLib::Pricers::SwaptionPricingEngine Class Template Reference	368

11.20	QuantLib::Instruments::SwaptionResults Class Reference	369
11.21	QuantLib::Volatilities::SwaptionVolatilityMatrix Class Reference	370
11.21	QuantLib::SwaptionVolatilityStructure Class Reference	371
11.21	QuantLib::Calendars::Sydney Class Reference	372
11.21	QuantLib::Math::SymmetricSchurDecomposition Class Reference	373
11.21	QuantLib::Calendars::TARGET Class Reference	374
11.21	QuantLib::TermStructure Class Reference	375
11.21	QuantLib::DayCounters::Thirty360 Class Reference	377
11.21	QuantLib::TimeGrid Class Reference	378
11.21	QuantLib::Calendars::Tokyo Class Reference	379
11.21	QuantLib::Calendars::Toronto Class Reference	380
11.22	QuantLib::Pricers::TreeCapFloor Class Reference	381
11.22	QuantLib::Pricers::TreeSwaption Class Reference	382
11.22	QuantLib::FiniteDifferences::TridiagonalOperator Class Reference	383
11.22	QuantLib::FiniteDifferences::TridiagonalOperator::TimeSetter Class Reference	385
11.22	QuantLib::Indexes::USDLibor Class Reference	386
11.22	QuantLib::Calendars::Wellington Class Reference	387
11.22	QuantLib::Indexes::Xibor Class Reference	388
11.22	QuantLib::Indexes::XiborManager Class Reference	390
11.22	QuantLib::Indexes::ZARLibor Class Reference	391
11.22	QuantLib::ZeroSpreadedTermStructure Class Reference	392
11.23	QuantLib::ZeroYieldStructure Class Reference	394
11.23	QuantLib::Calendars::Zurich Class Reference	396
12	QuantLib File Documentation	397
12.1	actual360.hpp File Reference	397
12.2	actual365.hpp File Reference	398
12.3	actualactual.cpp File Reference	399
12.4	actualactual.hpp File Reference	400
12.5	americancondition.hpp File Reference	401
12.6	analyticalcapfloor.cpp File Reference	402
12.7	analyticalcapfloor.hpp File Reference	403
12.8	argsandresults.hpp File Reference	404
12.9	arithmeticapopathpricer.cpp File Reference	405
12.10	arithmeticapopathpricer.hpp File Reference	406
12.11	arithmeticasopathpricer.cpp File Reference	407
12.12	arithmeticasopathpricer.hpp File Reference	408

12.13armijo.cpp File Reference	409
12.14armijo.hpp File Reference	410
12.15array.hpp File Reference	411
12.16audlibor.hpp File Reference	412
12.17barrieroption.cpp File Reference	413
12.18barrieroption.hpp File Reference	414
12.19basketpathpricer.cpp File Reference	415
12.20basketpathpricer.hpp File Reference	416
12.21bilinearinterpolation.hpp File Reference	417
12.22binaryoption.cpp File Reference	418
12.23binaryoption.hpp File Reference	419
12.24binomialtree.cpp File Reference	420
12.25binomialtree.hpp File Reference	421
12.26bisection.cpp File Reference	422
12.27bisection.hpp File Reference	423
12.28blackcapfloor.cpp File Reference	424
12.29blackcapfloor.hpp File Reference	425
12.30blackkarasinski.cpp File Reference	426
12.31blackkarasinski.hpp File Reference	427
12.32blackmodel.hpp File Reference	428
12.33blackswaption.cpp File Reference	429
12.34blackswaption.hpp File Reference	430
12.35boundarycondition.hpp File Reference	431
12.36boxmullergaussianrng.hpp File Reference	432
12.37brent.cpp File Reference	433
12.38brent.hpp File Reference	434
12.39bsmoperator.cpp File Reference	435
12.40bsmoperator.hpp File Reference	436
12.41cadlibor.hpp File Reference	437
12.42calendar.cpp File Reference	438
12.43calendar.hpp File Reference	439
12.44calibrationhelper.cpp File Reference	440
12.45calibrationhelper.hpp File Reference	441
12.46capflatvolvector.hpp File Reference	442
12.47capfloor.cpp File Reference	443
12.48capfloor.hpp File Reference	444

12.49caphelper.cpp File Reference	445
12.50caphelper.hpp File Reference	446
12.51capvolstructures.hpp File Reference	447
12.52cashflow.hpp File Reference	448
12.53cashflowvectors.cpp File Reference	449
12.54cashflowvectors.hpp File Reference	450
12.55centrallimitgaussianrng.hpp File Reference	451
12.56chflibor.hpp File Reference	452
12.57cliquetoption.cpp File Reference	453
12.58cliquetoption.hpp File Reference	454
12.59combiningiterator.hpp File Reference	455
12.60conjugategradient.cpp File Reference	456
12.61conjugategradient.hpp File Reference	457
12.62constraint.hpp File Reference	458
12.63continuousgeometricapo.hpp File Reference	459
12.64costfunction.hpp File Reference	460
12.65couplingiterator.hpp File Reference	461
12.66coupon.hpp File Reference	462
12.67coxingersollross.cpp File Reference	463
12.68coxingersollross.hpp File Reference	464
12.69cranknicolson.hpp File Reference	465
12.70criteria.hpp File Reference	466
12.71cubicspline.hpp File Reference	467
12.72currency.hpp File Reference	468
12.73dataformatters.cpp File Reference	469
12.74dataformatters.hpp File Reference	470
12.75date.cpp File Reference	471
12.76date.hpp File Reference	472
12.77daycounter.hpp File Reference	473
12.78daycounters.cpp File Reference	474
12.79daycounters.hpp File Reference	475
12.80diffusionprocess.hpp File Reference	476
12.81discretegeometricapo.cpp File Reference	477
12.82discretegeometricapo.hpp File Reference	478
12.83discretegeometricaso.cpp File Reference	479
12.84discretegeometricaso.hpp File Reference	480

12.85dminus.hpp File Reference	481
12.86dplus.hpp File Reference	482
12.87dplusdminus.hpp File Reference	483
12.88dzero.hpp File Reference	484
12.89errors.hpp File Reference	485
12.90euribor.hpp File Reference	487
12.91europeanengine.cpp File Reference	488
12.92europeanengine.hpp File Reference	489
12.93europeanoption.cpp File Reference	490
12.94europeanoption.hpp File Reference	491
12.95europeanpathpricer.cpp File Reference	492
12.96europeanpathpricer.hpp File Reference	493
12.97everestpathpricer.cpp File Reference	494
12.98everestpathpricer.hpp File Reference	495
12.99expliciteuler.hpp File Reference	496
12.100expressiontemplates.hpp File Reference	497
12.101falseposition.cpp File Reference	498
12.102falseposition.hpp File Reference	499
12.103fdtypedefs.hpp File Reference	500
12.104filteringiterator.hpp File Reference	501
12.105fnitedifferencemodel.hpp File Reference	502
12.106fixedratecoupon.hpp File Reference	503
12.107flatforward.hpp File Reference	504
12.108floatingratecoupon.cpp File Reference	505
12.109floatingratecoupon.hpp File Reference	506
12.110frankfurt.cpp File Reference	507
12.111frankfurt.hpp File Reference	508
12.112g2.hpp File Reference	509
12.113gbplibor.hpp File Reference	510
12.114geometricapopathpricer.cpp File Reference	511
12.115geometricapopathpricer.hpp File Reference	512
12.116geometricasopathpricer.cpp File Reference	513
12.117geometricasopathpricer.hpp File Reference	514
12.118getcovariance.cpp File Reference	515
12.119grid.hpp File Reference	516
12.120handle.hpp File Reference	517

12.12helsinki.cpp File Reference	518
12.12helsinki.hpp File Reference	519
12.12himalayapathpricer.cpp File Reference	520
12.12himalayapathpricer.hpp File Reference	521
12.12history.hpp File Reference	522
12.12hullwhite.cpp File Reference	523
12.12hullwhite.hpp File Reference	524
12.12impliciteuler.hpp File Reference	525
12.12index.hpp File Reference	526
12.13instrument.hpp File Reference	527
12.13interpolation.hpp File Reference	528
12.13interpolation2D.hpp File Reference	529
12.13inversedgaussianrng.hpp File Reference	530
12.13iteratorcategories.hpp File Reference	531
12.13jamshidianwaption.cpp File Reference	532
12.13jamshidianwaption.hpp File Reference	533
12.13johannesburg.cpp File Reference	534
12.13johannesburg.hpp File Reference	535
12.13pylibor.hpp File Reference	536
12.14knuthuniformrng.cpp File Reference	537
12.14knuthuniformrng.hpp File Reference	538
12.14leastsquare.hpp File Reference	539
12.14lecuyeruniformrng.cpp File Reference	540
12.14lecuyeruniformrng.hpp File Reference	541
12.14lexicographicalview.hpp File Reference	542
12.14linearinterpolation.hpp File Reference	543
12.14linsearch.hpp File Reference	544
12.14london.cpp File Reference	545
12.14london.hpp File Reference	546
12.15marketelement.hpp File Reference	547
12.15mathf.cpp File Reference	548
12.15mathf.hpp File Reference	549
12.15matrix.cpp File Reference	550
12.15matrix.hpp File Reference	551
12.15maxbasketpathpricer.cpp File Reference	552
12.15maxbasketpathpricer.hpp File Reference	553

12.157	mcbasket.cpp File Reference	554
12.158	mcdiscretearithmeticapo.cpp File Reference	555
12.159	mcdiscretearithmeticapo.hpp File Reference	556
12.160	mcdiscretearithmeticaso.cpp File Reference	557
12.161	mcdiscretearithmeticaso.hpp File Reference	558
12.162	mceuropan.cpp File Reference	559
12.163	mceuropan.hpp File Reference	560
12.164	mceverest.cpp File Reference	561
12.165	mceverest.hpp File Reference	562
12.166	mcchimalaya.cpp File Reference	563
12.167	mcmaxbasket.cpp File Reference	564
12.168	mcpagoda.cpp File Reference	565
12.169	mcpriker.hpp File Reference	566
12.170	mctypedefs.hpp File Reference	567
12.171	milan.cpp File Reference	568
12.172	milan.hpp File Reference	569
12.173	mixedscheme.hpp File Reference	570
12.174	model.cpp File Reference	571
12.175	model.hpp File Reference	572
12.176	multipathgenerator.hpp File Reference	573
12.177	multivariateaccumulator.cpp File Reference	574
12.178	multivariateaccumulator.hpp File Reference	575
12.179	newton.cpp File Reference	576
12.180	newton.hpp File Reference	577
12.181	newtonsafe.cpp File Reference	578
12.182	newtonsafe.hpp File Reference	579
12.183	newyork.cpp File Reference	580
12.184	newyork.hpp File Reference	581
12.185	node.hpp File Reference	582
12.186	normaldistribution.cpp File Reference	583
12.187	normaldistribution.hpp File Reference	584
12.188	null.hpp File Reference	585
12.189	numericalmethod.hpp File Reference	586
12.190	observable.hpp File Reference	587
12.191	onefactormodel.cpp File Reference	588
12.192	onefactormodel.hpp File Reference	589

12.193	defactoroperator.cpp File Reference	590
12.194	defactoroperator.hpp File Reference	591
12.195	optimizer.hpp File Reference	592
12.196	option.cpp File Reference	593
12.197	option.hpp File Reference	594
12.198	pagodapathpricer.cpp File Reference	595
12.199	pagodapathpricer.hpp File Reference	596
12.200	parameter.hpp File Reference	597
12.201	pathpricer.hpp File Reference	598
12.202	piecewiseflatforward.cpp File Reference	599
12.203	piecewiseflatforward.hpp File Reference	600
12.204	plainoption.cpp File Reference	601
12.205	plainoption.hpp File Reference	602
12.206	processingiterator.hpp File Reference	603
12.207	undefines.hpp File Reference	604
12.208	randomarraygenerator.hpp File Reference	606
12.209	ratehelpers.cpp File Reference	607
12.210	ratehelpers.hpp File Reference	608
12.211	relinkablehandle.hpp File Reference	609
12.212	bidder.cpp File Reference	610
12.213	bidder.hpp File Reference	611
12.214	riskmeasures.hpp File Reference	612
12.215	riskstatistics.hpp File Reference	613
12.216	rngtypedefs.hpp File Reference	614
12.217	sample.hpp File Reference	615
12.218	scheduler.cpp File Reference	616
12.219	scheduler.hpp File Reference	617
12.220	secant.cpp File Reference	618
12.221	secant.hpp File Reference	619
12.222	segmentintegral.cpp File Reference	620
12.223	segmentintegral.hpp File Reference	621
12.224	shortfloatingcoupon.cpp File Reference	622
12.225	shortfloatingcoupon.hpp File Reference	623
12.226	shortrateprocess.hpp File Reference	624
12.227	shoutcondition.hpp File Reference	625
12.228	simplecashflow.hpp File Reference	626

12.229	implswap.cpp File Reference	627
12.230	implswap.hpp File Reference	628
12.231	implex.cpp File Reference	629
12.232	implex.hpp File Reference	630
12.233	ingleassetoption.cpp File Reference	631
12.234	ingleassetoption.hpp File Reference	632
12.235	olver1d.cpp File Reference	633
12.236	olver1d.hpp File Reference	634
12.237	tistics.cpp File Reference	635
12.238	tistics.hpp File Reference	636
12.239	tepestdescent.cpp File Reference	637
12.240	tepestdescent.hpp File Reference	638
12.241	tepcondition.hpp File Reference	639
12.242	teppingiterator.hpp File Reference	640
12.243	ttock.cpp File Reference	641
12.244	ttock.hpp File Reference	642
12.245	wap.cpp File Reference	643
12.246	wap.hpp File Reference	644
12.247	waption.cpp File Reference	645
12.248	waption.hpp File Reference	646
12.249	waptionhelper.cpp File Reference	647
12.250	waptionhelper.hpp File Reference	648
12.251	waptionvolmatrix.hpp File Reference	649
12.252	waptionvolstructure.hpp File Reference	650
12.253	ydney.cpp File Reference	651
12.254	ydney.hpp File Reference	652
12.255	ymmetriceigenvalues.hpp File Reference	653
12.256	ymmetricsturdecomposition.cpp File Reference	654
12.257	ymmetricsturdecomposition.hpp File Reference	655
12.258	arget.cpp File Reference	656
12.259	arget.hpp File Reference	657
12.260	ermstructure.hpp File Reference	658
12.261	hirty360.cpp File Reference	659
12.262	hirty360.hpp File Reference	660
12.263	okyo.cpp File Reference	661
12.264	okyo.hpp File Reference	662

12.265	toronto.cpp File Reference	663
12.266	toronto.hpp File Reference	664
12.267	tree.cpp File Reference	665
12.268	tree.hpp File Reference	666
12.269	treecapfloor.cpp File Reference	667
12.270	treecapfloor.hpp File Reference	668
12.271	treewaption.cpp File Reference	669
12.272	treewaption.hpp File Reference	670
12.273	tridiagonaloperator.cpp File Reference	671
12.274	tridiagonaloperator.hpp File Reference	672
12.275	trinomialtree.cpp File Reference	673
12.276	trinomialtree.hpp File Reference	674
12.277	twofactormodel.hpp File Reference	675
12.278	types.hpp File Reference	676
12.279	tsdlibor.hpp File Reference	677
12.280	valueatcenter.cpp File Reference	678
12.281	valueatcenter.hpp File Reference	679
12.282	wellington.cpp File Reference	680
12.283	wellington.hpp File Reference	681
12.284	wibor.cpp File Reference	682
12.285	wibor.hpp File Reference	683
12.286	wibormanager.cpp File Reference	684
12.287	wibormanager.hpp File Reference	685
12.288	warlibor.hpp File Reference	686
12.289	zurich.cpp File Reference	687
12.290	zurich.hpp File Reference	688
13	QuantLib Example Documentation	689
13.1	DiscreteHedging.cpp	689
13.2	EuropeanOption.cpp	695
13.3	history_iterators.cpp	699
13.4	swapvaluation.cpp	700

Part I

User Manual

Chapter 1

An introduction to QuantLib

1.1 Introduction

QuantLib (<http://quantlib.org/>) is a C++ library for financial quantitative analysts and developers.

QuantLib is Non-Copylefted Free Software released under the modified BSD License. It is also OSI Certified Open Source Software. OSI Certified is a certification mark of the Open Source Initiative.

QuantLib is free software and you are allowed to use, copy, modify, merge, publish, distribute, and/or sell copies of it under the conditions stated in the QuantLib [Copyright and License](#).

QuantLib and its documentation are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the [Copyright and License](#) for more details.

1.1.1 Disclaimer

At this time, this documentation is widely incomplete and must be regarded as a work in progress. Eventually, all QuantLib classes will be documented. However, no date is currently set for this goal. Join the mailing lists (<http://quantlib.org/maillinglists.html>) for the latest updates.

1.2 Project overview

The QuantLib project is at this time in *beta* status.

The following list is an overview of the goals set for release 1.0. It is open for suggestions, corrections, and additions which can be submitted through the [QuantLib-users](#) mailing list. Every proposal will be seriously considered.

1.2.1 Project goals

Interfaces

- yield term structure—done;
- financial instrument—done;
- deal—to do;
- folder/portfolio/book levels of deal aggregation (strict vertical inclusion)—to do;
- trade-group deal aggregation (horizontal cross section)—to do;
- volatility term structure for cap/floor and swaption—to do;

Generic Tools

- date/time module—done;
- one-dimensional solver—done;
- one-dimensional optimizer—to do;
- multi-dimensional solver—to do;
- multi-dimensional optimizer—in progress;
- linear algebra module—in progress;
- statistical module—in progress;
- PDE framework—in progress;
- Monte Carlo framework—in progress;

Financial Tools

- Black-Scholes analytic implementation—done;
- Black-Scholes PDE implementation (for American and exotic options)—done;
- Deposits, FRA, futures, swaps—partly done;
- Deposit/FRA/futures/swap yield term structure—partly done;
- Swaptions—in progress;
- Caps/floors—in progress;
- Bonds (IRR/duration)—to do;
- Treasury yield term structure (Nelson-Siegel/Vasicek-Fong)—to do;
- Short-rate models (BDT, BK, HW, etc.)—in progress.

1.3 Where to get QuantLib

1.3.1 QuantLib releases

Source code, documentation, modules, etc. of current and previous QuantLib releases can be downloaded from <http://quantlib.org/download.html>

Debian package is available from <http://ftp.us.debian.org/debian/pool/main/q/quantlib> (the .us can be replaced by .jp, .uk, .de, .it, etc.)

1.3.2 Current CVS snapshot

Instructions for anonymous CVS access are available at <http://quantlib.org/cvs.html>

Access to the CVS repository is intended mainly for developers and is not recommended to end users which should download the latest stable release instead.

1.4 Installation

1.4.1 Linux/Unix

The simplest way to compile and install QuantLib is:

1. get the sources from
<http://quantlib.org/download.html> and untar them.
2. 'cd' to the QuantLib directory and type './configure' to configure the package for your system. If you're using 'csh' on an old version of System V, you might need to type 'sh ./configure' instead to prevent 'csh' from trying to execute 'configure' itself. Running 'configure' takes awhile. While running, it prints some messages telling which features it is checking for.
3. Type 'make' to compile the package.
4. Type 'make install' to install the programs and any data files and documentation.
5. You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you will need some GNU tools that usually only developers use, and which are not required to build QuantLib from tarballs. These are automake, autoconf, libtool, GNU m4, GNU make, and maybe others. They all come with recent GNU/Linux distributions. To begin the build process after 'make distclean' start with 'sh ./bootstrap' which will prepare the package for compilation. You can then use 'configure' and 'make' in the usual way.

The file INSTALL.txt in the QuantLib source distribution contains more detailed instructions.

1.4.2 Win32

Binary installers are available at

<http://quantlib.org/download.html>

Visual C++ 6.0 projects files and Borland C++ makefiles are supplied in case one wants to rebuild the library.

The free Borland C++ compiler is available at

<http://www.borland.com/bcppbuilder/freecompiler/>

1.4.3 Macintosh

Codewarrior support is currently broken. QuantLib should compile under Mac OS X as outlined under [Linux/Unix](#).

1.5 Usage

To use QuantLib classes in your own code just add

```
#include <ql/quantlib.hpp>
```

at the beginning of your source/header files.

Under the Examples folder you can find examples of QuantLib usage, including makefiles for gcc, Borland Free compiler, and Microsoft Visual C++. For the latter project files are also available.

1.5.1 Microsoft Visual C++

A few suggestions for Visual C++ users wanting to use QuantLib into their own application:

1. As long as you include `ql/quantlib.hpp`, you won't have to explicitly link your application to `QuantLib.lib` (or `QuantLib_d.lib`). This is automatically done by `quantlib.hpp` using the `pragma` statement:

```
#ifdef _DEBUG
    #pragma comment(lib, "QuantLib_d.lib")
#else
    #pragma comment(lib, "QuantLib.lib")
#endif
```

2. Your `main()` must be compiled with the same options that were used in compiling the QuantLib library - namely, you'll have to set the run-time library to "Multithreaded DLL" or "Debug Multithreaded DLL" depending on whether you're linking to `QuantLib.lib` or `QuantLib_d.lib`, respectively. This setting is in the project settings, "C/C++" tab, "Code Generation" category. You'll have to check the "Use RTTI" option under the "C++ Language" category, too. Finally, you have to define `NOMINMAX`
3. To create your own project, add it to a new or existing Workspace. Under `File | New | Projects` select "Win32 Console Application" (or equivalent). Under `Project | Settings` select your project. Select settings for: "All configuration". On the "C/C++" tab select "Preprocessor" and under "Additional include directory" add "`$(QL_DIR)`". This will add your current QuantLib installation directory to the include path. On the "Link" tab select "Input" and under "Additional library path" add "`$(QL_DIR)\lib\Win32\VisualStudio\`". This will add your current QuantLib installation directory to the library path.
4. To compile in "Win32 Debug" configuration you will need the QuantLib debug libraries, available for download from <http://quantlib.org/download.html>
5. The "Win32 OnTheEdgeRelease" and "Win32 OnTheEdgeDebug" configurations are equivalent to "Win32 Release" and "Win32 Debug" respectively, except that they do not use the installed QuantLib libraries, but look for the `hpp` and `lib` QuantLib files in a relative `..\QuantLib` path. This is mainly for QuantLib developers, to allow them to check the examples without having to install their unstable working version of QuantLib. The "OnTheEdge" approach may be adopted in projects using QuantLib, allowing for a quick and easy way to check how the projects work with a new QuantLib release without having to install it.

1.6 Supported platforms

The following is a list of configurations and compilers supported by QuantLib. All configuration names may have up to four parts: `cpu-manufacturer-kernel-operating_system`.

There are 3 officially supported configuration/compiler couples:

CONFIGURATION	COMPILER	COORDINATOR
i686-pc-linux-debian2.2	gcc 2.95 and 3.0.1	Enrico Sirola
i686-pc-WinNT4	MS Visual C++ 6	Ferdinando Ametrano
i686-pc-WinNT4	Borland free compiler	Luigi Ballabio

Dirk Eddebuettel is the maintainer of the QuantLib Debian packages. QuantLib was reported to compile also under:

CONFIGURATION	COMPILER
i686-pc-cygwin	gcc 2.95.2
alpha-RedHat6.2	Compaq 6.3.9.3
FreeBSD	gcc 2.95.2
sparc-sun-solaris	gcc 2.95.2
sparc-sun-linux-debian	gcc 2.95.2
sgi-linux	gcc 2.95
ppc-linux	gcc 2.95.2
arm	
ia64	
m68k	
s390	

For all these platforms Debian packages are available. If you compiled (or failed to compile) QuantLib on different configurations and/or compilers please report your feedback to: quantlib-users@lists.sourceforge.net

1.7 Version history

Release 0.2.1 - December 3rd, 2001

MONTE CARLO FRAMEWORK

- Path and MultiPath are now classes on their own
- PathPricer now handles both Path and MultiPath
- MonteCarloModel now handles both single factor and multi factors simulations.
- McPricer now handles both single factor and multi factors pricing. New pricing interface
- antithetic variance-reduction technique made possible in Monte Carlo for both single factor and multi factors
- Control Variate specific class removed: control variation technique is now handled by the general MC model
- average price and average strike asian option refactored
- Sample as a (value,weight) struct
- random number generators moved under RandomNumbers folder and namespace

FINITE DIFFERENCE FRAMEWORK

- BackwardEuler and ForwardEuler renamed ImplicitEuler and ExplicitEuler, respectively
- refactoring of TridiagonalOperator and derived classes

YIELD TERM STRUCTURE AND FIXED INCOME

- Added some useful methods to term structure classes
- Allowed passing a quote to RateHelpers as double
- added FuturesRateHelpers (no convexity adjustment yet)
- PiecewiseFlatForward now observer of rates passed as MarketElements
- Unified Date and Time interface in TermStructure
- Added BPS to generic swap legs
- added term_structure+swap example
- Fixing days introduced for floating-coupon bond

PATTERNS

- Added factory pattern
- Calendar and DayCounter now use the Strategy pattern

VARIOUS

- used do-while-false idiom in QL_REQUIRE-like macros
- now using size_t where appropriate
- dividendYield is now a Spread instead of a Rate (that is: cost of carry is allowed)
- RelinkableHandle initialized with an optional Handle
- Worked around VC++ problems in History constructor
- added QL_VERSION and QL_HEX_VERSION
- generic bug fixes
- removed classes deprecated in 0.2.0

INSTALLATION FACILITIES

- improved and smoother Win32 binary installer

DOCUMENTATION

- general re-hauling
- improved and extended Monte Carlo documentation
- improved and extended examples
- Upgraded to Doxygen 1.2.11.1
- Added man pages for installed executables
- added docs in Windows Help format
- added info on "Win32 OnTheEdgeRelease" and "Win32 OnTheEdgeDebug" MS VC++ configurations
- additional information on how to create a MS VC++ project based on [QuantLib](#)

Release 0.2.0 - September 18th, 2001

- Library:
 - source code moved under ql, better GNU standards
 - gcc build dir can now be separated from source tree
 - gcc 3.0.1 port
 - clean compilation (no warnings)
 - bootstrap script on cygwin
 - Fixed automatic choice of seed for random number generators
 - Actual/actual classes
 - extended platform support (see table in documentation)
 - antithetic variance-reduction technique made possible in Monte Carlo
 - added dividend-Rho greek
 - First implementation of segment integral (to be redesigned)
 - Knuth random generator
 - Cash flows, scheduler, and swap (both generic and simple) added
 - added ICGaussian random generator
 - generic bug fixes
- Installation facilities:
 - improved and smoother Win32 binary installer
 - better distribution
 - debian packages available
- Documentation:
 - general re-hauling
 - added examples of using QuantLib and of projects based on QL

Release 0.1.9 - May 31st, 2001

- Library:
 - Style guidelines introduced (see <http://quantlib.org/style.html>) and partially enforced
 - full support for Microsoft Visual Studio
 - full support for Linux/gcc
 - momentarily broken support for Metrowerks CodeWarrior
 - autoconfiscation (with specialized config*.hpp files for platforms without automake/autoconf support)
 - Include files moved under Include/ql folder and referenced as "ql/header.hpp"

- Implemented expression templates techniques for array algebra optimization
- Added custom iterators
- Improved term structure
- Added Asian, Bermudan, Shout, Cliquet, Himalaya, and Barrier options (all with greeks calculation, control variated where possible)
- Added Helsinki and Wellington calendars
- Improved Normal distribution related functions: cumulative, inverse cumulative, etc.
- Added uniform and Gaussian random number generators
- Added Statistics class (mean, variance, skewness, downside variance, etc.)
- Added RiskMeasures class: VAR, average shortfall, expected shortfall, etc.
- Added RiskStatistics class combining Statistics and RiskMeasures
- Added sample accumulator for multivariate analysis
- Added Monte Carlo tools
- Added matrix-related functions (square root, symmetric Schur decomposition)
- Added interpolation framework (linear and cubic spline interpolation implemented).
- Installation facilities:
 - Added Win32 GUI installer for binaries
- Documentation:
 - support for Doxygen 1.2.7
 - Added man documentation

Release 0.1.1 - November 21st, 2000

Initial release.

1.8 Todo List

- Class `QuantLib::Volatilities::CapFlatVolatilityVector`** Either add correct copy behavior or inhibit copy. Right now, a copied instance would end up with its own copy of the length vector but an interpolation pointing to the original ones.
- Class `QuantLib::Pricers::ContinuousGeometricAPO`** add Average Strike version and make it backward starting
- Class `QuantLib::Pricers::DiscreteGeometricAPO`** add analytical greeks
- Class `QuantLib::Pricers::DiscreteGeometricASO`** add analytical greeks
- Class `QuantLib::FiniteDifferences::ExplicitEuler`** add Richardson extrapolation
- Class `QuantLib::Pricers::FdAmericanOption`** make american call with no dividends = european
- Class `QuantLib::CashFlows::FloatingRateCouponVector`** A suitable algorithm should be implemented for the calculation of the interpolated index fixing for a short/long first coupon.
- Class `QuantLib::TermStructures::FraRateHelper`** convexity adjustment should be implemented.
- Class `QuantLib::Math::Interpolation2D`** Bicubic interpolation and bicubic spline
- Class `QuantLib::Pricers::McDiscreteArithmeticAPO`** Continuous Averaging version
- Class `QuantLib::Pricers::McDiscreteArithmeticASO`** Continuous Averaging version
- Class `QuantLib::FiniteDifferences::MixedScheme`** add Douglas Scheme
- Class `QuantLib::MonteCarlo::MultiPath`** make it time-aware
- Class `QuantLib::MonteCarlo::Path`** make it time-aware
- Class `QuantLib::MonteCarlo::PathGenerator`** add more general path generator with drift(S,t) and variance(S,t)
- Class `QuantLib::Math::SegmentIntegral`** Redesign as a template function.
- Class `QuantLib::Volatilities::SwaptionVolatilityMatrix`** Either add correct copy behavior or inhibit copy. Right now, a copied instance would end up with its own copy of the exercise date and length vector but an interpolation pointing to the original ones.
- Class `QuantLib::Calendars::Tokyo`** insert actual rules for Equinoxes

1.9 Additional resources

The main QuantLib resource is the QuantLib site (<http://quantlib.org>).

Additional resources available from the above site include:

- current news;
- the QuantLib mailing lists (<http://quantlib.org/maillinglists.html>);
- the QuantLib FAQ (<http://quantlib.org/FAQ.html>);
- the QuantLib programming style guidelines (<http://quantlib.org/style.html>);
- a link to the QuantLib project page on Sourceforge.net;
- links to pages for bug reports, patch submissions and feature requests;
- a page (<http://quantlib.org/extensions.html>) about how to use QuantLib in other languages/platforms;
- as well as links to additional quantitative finance resources.

1.10 The QuantLib Group

1.10.1 Authors

The QuantLib Group members are:

- Mario Aleppo (mario.aleppo@riskmap.net)
- Ferdinando Ametrano (ferdinando@ametrano.net), project manager
- Luigi Ballabio (luigi.ballabio@riskmap.net), library designer
- Adolfo Benin (adolfo.benin@libero.it)
- Nicolas Di Césaré, spline and optimizer
- Marco Marchioro (marco.marchioro@riskmap.net)
- Sadruddin Rejeb (sad.rejeb@riskmap.it), single factor interest rate models
- Enrico Sirola (enrico.sirola@riskmap.net)

1.10.2 Contributors

- Dirk Eddelbuettel, Debian maintainer
- Christopher Baus, inspiration for custom iterators
- Thomas Becker, inspiration for custom iterators
- Toyin Akin
- James Battle
- David Binderman
- Matteo Gallivanoni (matteo.gallivanoni@riskmap.net)
- Gilbert Peffer
- Peter Schmitteckert (quantlib@schmitteckert.com)
- Maxim Sokolov (maxim.sokolov@rambler.ru), examples
- Bernd Johannes Wuebben

1.11 Copyright and License

Copyright ©2002 Ferdinando Ametrano

Copyright ©2001, 2002 Nicolas Di Césaré

Copyright ©2001, 2002 Sadruddin Rejeb

Copyright ©2000, 2001, 2002 RiskMap srl

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the copyright holders nor the names of the QuantLib Group and its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.11.1 Comments on Copyright and License

QuantLib is Non-Copylefted Free Software [1] released under the modified BSD License [2] (also known as XFree86-style license).

QuantLib is Open Source [3] because of its license: it is OSI Certified Open Source Software [4]. OSI Certified is a certification mark of the Open Source Initiative [5].

The modified BSD License is GPL compatible as confirmed by the Free Software Foundation [6].

This license has been adopted to allow free use of QuantLib and its source, to make QuantLib flourish as a free-software/open-source project. It allows proprietary extensions to be commercialized.

[1] <http://www.gnu.org/philosophy/categories.html#Non-CopyleftedFree-Software>

[2] <http://www.opensource.org/licenses/bsd-license.html>

[3] <http://www.opensource.org/docs/definition.html>

[4] <http://www.opensource.org/docs/certification.mark.html>

[5] <http://www.opensource.org>

[6] <http://www.gnu.org/philosophy/bsd.html>

Chapter 2

QuantLib components

2.1 Core classes

Documentation for this part of the library is in progress.

The root namespace [QuantLib](#) contains what can be considered the QuantLib foundation, i.e., the core of abstract classes upon which the rest of the library is built.

The folder structure of the sources reflects the organization of classes and functions into namespaces. These are in turn chosen to represent definite *packages* or *class families*: the former being groups of classes forming a framework for particular applications, e.g., the PDE package for partial differential equations; and the latter being sets of classes which inherit from a common base class or otherwise share a common functionality and interface, e.g., the TermStructures or Operators families, respectively.

In the [QuantLib](#) namespace:

- A number of financial types is provided, such as [QuantLib::Rate](#), [QuantLib::Spread](#), or [QuantLib::DiscountFactor](#). All these types act as, and indeed are, doubles.
 - The Date, Calendar and DayCounter classes are defined (see [Date and time calculations](#)).
 - The abstract classes [QuantLib::ForwardVolatilitySurface](#) and [QuantLib::SwaptionVolatilitySurface](#) provide the common interface to concrete volatility models. No examples are available at this time, and the classes themselves will probably need some reworking.
 - The abstract class [QuantLib::Solver1D](#) provides a common interface for the one-dimensional solvers implemented in the [QuantLib::Solvers1D](#) namespace.
 - The abstract class [QuantLib::Index](#) provides the common interface of the classes implemented in the [QuantLib::Indexes](#) namespace.
 - The abstract class [QuantLib::Instrument](#) provides a common interface and a calculation framework for the classes implemented in the [QuantLib::Instruments](#) namespace.
 - The abstract class [QuantLib::TermStructure](#) provides a common interface for the classes implemented in the [QuantLib::TermStructures](#) namespace.
-

2.2 Date and time calculations

Documentation for this part of the library is in progress.

2.2.1 Dates

The concrete class [QuantLib::Date](#) implements the concept of date. Its functionalities include:

- providing basic information such as weekday, day of the month, day of the year, month, and year;
- comparing two dates to determine whether they are equal, or which one is the earlier or later, or the difference between them expressed in days;
- incrementing or decrementing a date of a given number of days, or of a given period expressed in weeks, months, or years.

2.2.2 Calendars

The abstract class [QuantLib::Calendar](#) provides the interface for determining whether a date is a business day or a holiday for a given market, or incrementing/decrementing a date of a given number of business days. A number of calendars is contained in the [QuantLib::Calendars](#) namespace, namely,

- Frankfurt
- Helsinki
- London
- Milan
- New York
- TARGET
- Wellington
- Zurich

2.2.3 Day counters

The abstract class [QuantLib::DayCounter](#) provides more advanced means of measuring the distance between two dates according to a given market convention, both as number of days or fraction of year. A number of such conventions is contained in the [QuantLib::DayCounters](#) namespace, namely,

- Actual/360
- Actual/365
- Actual/Actual, which can be calculated according to:
 - ISMA and US Treasury convention, also known as "Actual/Actual (Bond)";
 - ISDA, also known as "Actual/Actual (Historical)";
 - AFB, also known as "Actual/Actual (Euro)";
- 30/360, which can be calculated according to:
 - USA convention;
 - European convention;
 - Italian convention.

2.3 The finite differences framework

This framework (corresponding to the `QuantLib::FiniteDifferences` namespace) contains basic building blocks for the numerical solution of a generic differential equation

$$\frac{\partial f}{\partial t} = Lf$$

where L is a differential operator in “space”, i.e., one which does not contain partial derivatives in t but can otherwise contain any derivative in any other variable of the problem.

Writing the equation in the above form allows us to implement separately the discretization of the differential operator L and the time scheme used for the evolution of the solution. The `QuantLib::FiniteDifferences::FiniteDifferenceModel` class acts as a glue for such two steps—which are outlined in the following subsections—and provides the interface of the resulting finite difference model for the end user. Furthermore, it provides the possibility of checking and operating on the solution array at each step—which is typically used to apply an exercise condition for an option. This is also outlined in a subsection below.

2.3.1 Differential operators

The discretization of the differential operator L depends on the discretization chosen for the solution f of the given equation.

Such choice is obvious in the 1-D case where the domain $[a, b]$ of the equation is discretized as a series of points $x_i, i = 0 \dots N-1$ (note that the index is zero based) where $x_i = a + hi$ and $h = (b-a)/(N-1)$. In turn, the solution f of the equation is discretized as an array $u_i, i = 0 \dots N-1$ whose elements are defined as $u_i = f(x_i)$. The discretization of the differential operator follows by substituting the derivatives with the corresponding incremental ratios defined in terms of the f_i . A number of basic operators are defined in the framework which can be composed to form more complex operators, namely:

the identity operator I is implemented in class `Identity`;

the first derivative $\partial/\partial x$ is discretized as the operator D_+ , defined as

$$D_+ u_i = \frac{u_{i+1} - u_i}{h}$$

and implemented in class `QuantLib::FiniteDifferences::DPlus`; the operator D_- , defined as

$$D_- u_i = \frac{u_i - u_{i-1}}{h}$$

and implemented in class `QuantLib::FiniteDifferences::DMinus`; and the operator D_0 , defined as

$$D_0 u_i = \frac{u_{i+1} - u_{i-1}}{2h}$$

and implemented in class `QuantLib::FiniteDifferences::DZero`. The discretization error of the above operators is $O(h)$ for D_+ and D_- and $O(h^2)$ for D_0 ;

the second derivative $\partial^2/\partial x^2$ is discretized as the operator $D_+ D_-$, defined as

$$D_+ D_- u_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

and implemented in class `QuantLib::FiniteDifferences::DPlusDMinus`. Its discretization error is $O(h^2)$.

The boundary condition for the above operators is by default linear extrapolation. Methods are currently provided for setting other kinds of boundary conditions to a tridiagonal operator which these operators inherit, namely, Dirichlet—i.e., constant value—and Neumann—i.e., constant derivative—boundary conditions. This might change in the future as boundary conditions could be abstracted and passed as an additional argument to the model.

A programmer can also implement its own operator. However, in order to fit into this framework it will have to implement a required interface depending on the chosen evolver (see below). Also, it is currently required to manage itself any boundary conditions. Again, this could change in the future.

On the other hand, there is no obvious choice in the 2-D case. While it is immediate to discretize the domain into a series of points (x_i, y_j) and the solution into a matrix $f_{ij} = f(x_i, y_j)$, there is a number of ways into which the f_{ij} can be arranged into an array—each of them determining a different discretization of the differential operators. One of such ways was implemented in the `LexicographicalView` class, while others will be implemented in the future. No 2-D operator is currently implemented apart from the identity operator.

2.3.2 Time schemes

Once the differential operator L has been discretized, a number of choices are available for discretizing the time derivative at the left-hand side of the equation.

In this framework, such choice is encapsulated in so-called evolvers which, given L and the solution $u^{(k)}$ at time t_k , yield the solution $u^{(k-1)}$ at the previous time step.

A number of evolvers are currently provided in the library which implement well-known schemes, namely, the forward Euler explicit scheme in which the equation is discretized as

$$\frac{u^{(k)} - u^{(k-1)}}{\Delta t} = Lu^{(k)}$$

hence

$$u^{(k-1)} = (I - \Delta t L) u^{(k)}$$

from which $u^{(k-1)}$ can be obtained directly;

the backward Euler implicit scheme in which the equation is discretized as

$$\frac{u^{(k)} - u^{(k-1)}}{\Delta t} = Lu^{(k-1)}$$

hence

$$(I + \Delta t L) u^{(k-1)} = u^{(k)}$$

from which $u^{(k-1)}$ can be obtained by solving a linear system;

the Crank-Nicolson scheme in which the equation is discretized as

$$\frac{u^{(k)} - u^{(k-1)}}{\Delta t} = L \frac{u^{(k)} + u^{(k-1)}}{2}$$

hence

$$\left(I + \frac{\Delta t}{2} L\right) u^{(k-1)} = \left(I - \frac{\Delta t}{2} L\right) u^{(k)}$$

from which $u^{(k-1)}$ can be obtained by solving a linear system.

Each of the above evolvers forces a set of interface requirements upon the differential operator which are detailed in the documentation of the corresponding class, namely, [QuantLib::FiniteDifferences::ExplicitEuler](#), [QuantLib::FiniteDifferences::ImplicitEuler](#), and [QuantLib::FiniteDifferences::CrankNicolson](#), respectively.

A programmer could implement its own evolver, which does not need to inherit from any base class.

However, it must implement the following interface:


```

class Evolver {
public:
    typedef ... arrayType;
    typedef ... operatorType;
    // constructors
    Evolver(const operatorType& D);
    // member functions
    void step(arrayType& a, Time t) const;
    void setStep(Time dt);
};

```

Finally, we note again that the pricing of an option requires the finite difference model to solve the corresponding equation *backwards* in time. Therefore, given a discretization u of the solution at a given time t , the call

```
evolver.step(u,t)
```

must calculate the discrete solution at the *previous* time, $t - dt$.

2.3.3 Step conditions

A finite difference model can be passed a step condition to be applied at each step during the rollback of the solution (e.g. the early exercise American condition). Such condition must be embodied in a class derived from [QuantLib::FiniteDifferences::StepCondition](#) and must implement the interface of the latter, namely,

```

class MyCondition : public StepCondition<arrayType> {
public:
    void applyTo(arrayType& a, Time t) const;
};

```

2.3.4 An example of finite difference model

The Black-Scholes equation can be written in the above form as

$$\frac{\partial f}{\partial t} = -\frac{\sigma^2}{2} \frac{\partial^2 f}{\partial x^2} - \nu \frac{\partial f}{\partial x} + rf.$$

It can be seen that the operator L_{BS} is

$$L_{BS} = -\frac{\sigma^2}{2} \frac{\partial^2}{\partial x^2} - \nu \frac{\partial}{\partial x} + rI$$

and can be built from the basic operators provided in the library as

$$L_{BS} = -\frac{\sigma^2}{2} D_+ D_- - \nu D_0 + rI.$$

Its implementation closely reflects the above decomposition and can be written as

```

class BlackScholesOperator : public TridiagonalOperator {
public:
    BlackScholesOperator(
        double sigma, double nu,    // parameters of the
        Rate r,                    // Black-Scholes equation;
        unsigned int points,        // number of discretized points;
        double h)                  // grid spacing.

```

```

    : TridiagonalOperator(
      // build the operator by adding basic ones
      - (sigma*sigma/2.0) * DPlusDMinus(points,h)
      - nu * DZero(points,h)
      + r * TridiagonalOperator::identity(points)
    ) {}
};

```

taking as inputs the relevant parameters of the equation (σ , ν and r) as well as model parameters such as the number N of grid points and their spacing h .

As simple example cases, we will use the above operator to price both an European and an American option. The parameters of the two options will be the same, namely, they will be both call options with underlying price $u = 100$, strike $s = 95$, residual time $T = 1$ year, dividend yield $q = 3\%$ and volatility $\sigma = 10\%$. The risk-free rate will be $r = 5\%$. Such parameters are expressed using QuantLib types as

```

Option::Type type = Option::Call;
double underlying = 100.0, strike = 95.0;
Time residualTime = 1.0;
Rate dividendYield = 0.03, riskFreeRate = 0.05;
double volatility = 0.10;

```

The grid upon which the model will act will be a logarithmic grid of underlying prices, i.e., f will be defined in a range $[\ln u_{\min}, \ln u_{\max}]$ discretized as an array $x_i, i = 0 \dots N - 1$ with $x_i = \ln u_{\min} + ih$ and $h = (\ln u_{\max} - \ln u_{\min}) / (N - 1)$. Such a grid and the corresponding vector of actual prices can be built as shown in the code below. The domain of the model will be defined as $[\ln u - \Delta, \ln u + \Delta]$ where $\Delta = 4\sigma\sqrt{T}$. A number of grid points $N = 101$ will be used.

```

unsigned int gridPoints = 101;
Array grid(gridPoints), prices(gridPoints);
double x0 = QL_LOG(underlying);
double Delta = 4.0*volatility*QL_SQRT(residualTime);
double xMin = x0 - Delta, xMax = x0 + Delta;
double h = (xMax-xMin)/(gridPoints-1);
for (unsigned int i=0; i<gridPoints; i++) {
    grid[i] = xMin + i*h;
    prices[i] = QL_EXP(grid[i]);
}

```

The initial condition is determined by the values of the option at maturity, i.e., either the difference between underlying price and strike if such difference is positive, or 0 if that is not the case (the above will have to be suitably modified for a put option or a straddle.) Such “initial” condition will be rolled back in time by our model.

```

Array exercisingValue(gridPoints);
for (unsigned int i=0; i<gridPoints; i++)
    exercisingValue[i] = QL_MAX(prices[i]-strike,0.0);

```

Now the differential operator can be initialized. Also, Neumann initial conditions are set which correspond to the initial value of the derivatives at the boundaries (see the BoundaryCondition class documentation for details).

```

double nu = riskFreeRate - dividendYield - volatility*volatility/2.0;
TridiagonalOperator L = BlackScholesOperator(volatility, nu,
    riskFreeRate, gridPoints, h);
L.setLowerBC(BoundaryCondition(BoundaryCondition::Neumann,
    exercisingValue[1]-exercisingValue[0]));
L.setUpperBC(BoundaryCondition(BoundaryCondition::Neumann,
    exercisingValue[gridPoints-1]-exercisingValue[gridPoints-2]));

```

We are now already set for the pricing of the European option. Also, the exercise condition is the only thing still to be defined for the American option to be priced. Such condition is equivalent to the statement that at each time step, the value of the option is the maximum between the profit realized in exercising the option (which we already calculated and stored in `exercisingValue`) and the value of the option should we keep it (which corresponds to the solution rolled back to the current time step). This logic can be implemented as:

```
class ExerciseCondition : public StepCondition<Array> {
public:
    ExerciseCondition(const Array& exercisingValue)
        : exercisingValue_(exercisingValue) {}
    void applyTo(Array& a, Time) const {
        for (unsigned int i = 0; i < a.size(); i++)
            a[i] = QL_MAX(a[i], exercisingValue_[i]);
    }
private:
    Array exercisingValue_;
};
```

Everything is now ready. The model can be created gluing the piece together by means of the [QuantLib::FiniteDifferences::FiniteDifferenceModel](#) class. The current value of the option is calculated by rolling back the solution to the current time, i.e., $t = 0$, and by taking the value corresponding at the current underlying price—which by construction corresponds to the central value provided that the number of grid points is odd.

```
unsigned int timeSteps = 365;

// build the model - Crank-Nicolson scheme chosen
FiniteDifferenceModel<CrankNicolson<TridiagonalOperator> > model(L);

// European option
Array f = exercisingValue; // initial condition
model.rollback(f, residualTime, 0.0, timeSteps);
double europeanValue = valueAtCenter(f);

// American option
f = exercisingValue; // reset
Handle<StepCondition<Array> > condition(
    new ExerciseCondition(exercisingValue));
model.rollback(f, residualTime, 0.0, timeSteps, condition);
double americanValue = valueAtCenter(f);
```

2.4 The Monte Carlo framework

Anyone attempting to generate random numbers by deterministic means is, of course, living in a state of sin.
— John Von Neumann

This framework (corresponding to the [QuantLib::MonteCarlo](#) namespace) contains basic building blocks for the numerical calculation of the integral

$$\int_{\Omega} f(\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

where $p(\mathbf{x})$ is a normalized probability function. Monte Carlo methods solve the above integral by approximating it with the discrete sum

$$\frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i)w(\mathbf{x}_i)$$

where the \mathbf{x}_i are drawn from $p(\mathbf{x})$, possibly with a weight $w(\mathbf{x}_i)$ —which otherwise can be considered uniformly equal to 1.

The above sum has a straightforward interpretation in the case of a derivative product, namely, the \mathbf{x}_i are N generated random paths which the value of the underlying can possibly follow, while the $f(\mathbf{x}_i)$ are the values of the derivative on each of such paths. The sum above can therefore be taken as an estimate of the price of the derivative, namely, the average of its value on all possible paths—or rather all considered paths. Such a method enables the user to construct pricing classes for an unlimited range of derivatives, most notably path-dependent ones which cannot be priced by means of finite difference methods.

It must also be mentioned that for all such methods, the error e on the estimated value is proportional to the square root of the number of samples N . A number of so-called *variance-reduction* methods have been found which allows one to reduce the coefficient of proportionality between e and $1/\sqrt{N}$.

Separate implementations are provided in the library for the three components of the above average, namely, the random drawing of the \mathbf{x}_i , the evaluation of the $f(\mathbf{x}_i)$, and the averaging process itself. The [QuantLib::MonteCarlo::MonteCarloModel](#) class acts as a glue for such three steps—which are outlined in the following subsections—and provides the interface of the resulting Monte Carlo model to the end user.

2.4.1 Path generation

The Black-Scholes equation

$$\frac{\partial f}{\partial t} + \frac{\sigma^2}{2} \frac{\partial^2 f}{\partial x^2} + \nu \frac{\partial f}{\partial x} - rf = 0,$$

where r is the risk-free rate, σ is the volatility of the underlying, and $\nu = r - \sigma^2/2$, has the form of a diffusion process. According to this heuristic interpretation (1), paths followed by the logarithm of the underlying would be Brownian random walks with a constant drift ν per unit time and a standard deviation $\sigma\sqrt{T}$ over a time T .

Therefore, the paths to be generated for a Monte Carlo model of the Black-Scholes equation will be vectors of successive variations of the logarithm of the underlying price over N consecutive time intervals $\Delta t_i, i = 0 \dots N-1$. Each such variation will be drawn from a Gaussian distribution with average $\nu\Delta T_i$ and standard deviation $\sigma\sqrt{\Delta T_i}$ —or possibly $\nu_i\Delta T_i$ and $\sigma_i\sqrt{\Delta T_i}$ should ν and σ vary in time.

The [QuantLib::MonteCarlo::Path](#) class stores the variation vector decomposed in its drift (determined) and diffusion (random) components. As shown below, this allows the implementation of antithetic variance reduction techniques.

The [QuantLib::MonteCarlo::MultiPath](#) class is a straightforward extension which acts as a vector of Path objects.

Classes are provided which generate paths and multi-paths with the desired drift and diffusion components, namely, [QuantLib::MonteCarlo::PathGenerator](#) and [QuantLib::MonteCarlo::MultiPathGenerator](#).

For the time being, the path generator is initialized with a constant drift and variance. This requirement will most likely be relaxed in the next release. The multi-path generator is initialized with an array of constant drifts—one for each single path—and a covariance matrix which encapsulates the relations between the diffusion components of the single paths.

The time discretization of the (multi)paths can be specified either as a given number of equal time steps over a given time span, or as a vector of explicitly specified times at which the path will be sampled.

2.4.2 Pricing an instrument on a path

The [QuantLib::MonteCarlo::PathPricer](#) class is the base class from which path pricers must inherit. The only method which subclasses are required to implement is

```
double operator()(const P&) const;
```

where P can be Path or MultiPath depending on the derivative whose value must be calculated.

Similarly, the term *path* will be used in the following discussion as meaning either path or multi-path depending on the context. The term *single path* is not to be taken as opposite to multi-path, but rather as meaning “a single instance of a (multi)path” as opposed to the set of all generated (multi)paths.

The above method encapsulates the pricing of the derivative on a single path and must return its value had the evolution of the underlying(s) followed the path passed as argument. For this reason, control variate techniques (see below) must not be implemented at this level since they would cause the returned value to differ from the actual price of the derivative on the path.

Instead, antithetic variance-reduction techniques can be effectively implemented at this level and indeed are used in the pricers currently included in the library.

In short, such techniques consist in pricing an option on both the given path and its antithetic, the latter being a path with the same drift and the opposite diffusion component. The value of the sample is defined as the average of the prices on the two paths.

A generic implementation of antithetic techniques could consist of a path pricer class which takes a concrete path pricer upon construction and whose operator() simply proxies two calls to the contained pricer, passing the given path and its antithetic, and averages the result. However, this would not take full advantage of the technique.

In fact, it must be noted that using antithetic paths not only reduces the variance *per se* but also allows to factor out calculations common to a path and its antithetic, thus reducing greatly the computation time. Therefore, such techniques are best implemented inside the path pricer itself, whose algorithm can fully exploit such factorization.

A number of path pricers are available in the library and are listed in the [QuantLib::MonteCarlo](#) namespace reference.

2.4.3 Accumulating and averaging samples

The class [QuantLib::MonteCarlo::MonteCarloModel](#) encapsulates the general structure of a Monte Carlo calculations, namely, the generation of a number of paths, the pricing of the derivative on each path, and the averaging of the results to yield the actual derivative price.

As outlined above, the first two steps are delegated to a path generator and a path pricer. The third step is also delegated to an object which accumulates weighted values and returns the statistic properties of the set of such values. One such class provided by the library is [QuantLib::Math::Statistics](#).

The concern of the Monte Carlo model is therefore to act as a glue between such three components and can be expressed by the following pseudo-code:

```
given pathGenerator, pathPricer, accumulator;
for i in number of samples {
    path, weight = pathGenerator.next();
    price = pathPricer(path);
    accumulator.add(price, weight);
}
```

The Monte Carlo model also provides the user with the possibility to take advantage of control-variate techniques.

Such techniques consist in pricing a portfolio from which the price of the derivative can be deduced, but with a lower variance than the derivative alone.

In our current implementation, static-hedge control variate is used, namely, the formed portfolio is long of the derivative we need to price and short of a similar derivative whose price can be calculated analytically. The value of the portfolio on a given path will of course be given by the difference of the values of the two derivatives on such path. However, due to the similarity between the derivatives, the portfolio price will have a lower variance than either derivative alone since any variation in the price of the latter will be partly compensated by a similar variation in the price of the other. Lastly, given the portfolio price, the price of the derivative we are interested in can be deduced by adding the analytic value of the other.

In order to use such technique, the user must provide the model with a path pricer for the additional option and the value of the latter. The action of the Monte Carlo model is in this case expressed as:

```
given pathGenerator, pathPricer, cvPathPricer, cvPrice, accumulator;
for i in number of samples {
    path, weight = pathGenerator.next();
    portfolioPrice = pathPricer(path) - cvPathPricer(path);
    accumulator.add(portfolioPrice+cvPrice, weight);
}
```

Martingale (a.k.a. dynamic-hedge) control variate techniques are planned for future releases.

A [QuantLib::Pricers::McPricer](#) class is also available which wraps the typical usage of a Monte Carlo model.

Details on the Monte Carlo Pricer interface will be available in the pricers section.

2.4.4 Examples of Monte Carlo models

As a simple example, we will use the outlined tools to price an European option by means of Monte Carlo techniques.

Given a current underlying price u_0 and a path $p = [p_1, \dots, p_N]$ where every variation p_i is the sum of a drift term d_i and a random diffusion term r_i , the price of the underlying at maturity is

$$u = u_0 \prod_1^N e^{p_i} = u_0 \exp \left(\sum_1^N p_i \right) = u_0 \exp \left(\sum_1^N d_i + \sum_1^N r_i \right)$$

while the price on the antithetic path—i.e., same drift and opposite diffusion—is

$$u_0 \exp \left(\sum_1^N d_i - \sum_1^N r_i \right).$$

The corresponding path pricer can be implemented as:

```

class EuropeanPathPricer : public PathPricer<Path> {
public:
    EuropeanPathPricer(Option::Type type, double underlying,
                       double strike, DiscountFactor discount,
                       bool useAntithetic)
    // just store the needed parameters
    : type_(type), underlying_(underlying), strike_(strike),
      discount_(discount), useAntithetic_(useAntithetic) {}
    // here is the logic
    double operator()(const Path& path) const {

        size_t n = path.size();

        // factor out the sums in the formula above
        double sum_d = 0.0, sum_r = 0.0;
        for (size_t i = 0; i < n; i++) {
            sum_d += path.drift()[i];
            sum_r += path.diffusion()[i];
        }

        // calculate final underlying price on path
        double price = underlying_*QL_EXP(sum_d+sum_r);

        // calculate payoff
        double payoff;
        switch (type_) {
            case Option::Call;
                payoff = QL_MAX(price-strike,0.0);
                break;
            // other cases are left as an exercise to the reader
            ...
        }

        // current value of the option is the discounted payoff
        double optionValue = payoff*discount_;

        // stop here if not antithetic...
        if (!useAntithetic_)
            return optionValue;

        // ...otherwise calculate the value on the antithetic path
        double antiPrice = underlying_*QL_EXP(sum_d-sum_r);

        // calculate payoff and option value as above
        ...

        // return the average of the results on the two paths
        return (optionValue + antiOptionValue)/2.0;
    }
private:
    // stored parameters
    ...
};

```

The path pricer can now be used in a model. Let us assume the following parameters:

```

Option::Type type = Option::Call;
double underlying = 100.0, strike = 95.0;
Time residualTime = 1.0;
Rate dividendYield = 0.03, riskFreeRate = 0.05;
double volatility = 0.10;

```

The path generator can be instantiated as

```
// parameters of the Black-Scholes equation
double vol2 = volatility*volatility;
double nu = riskFreeRate - dividendYield - vol2/2.0;
// in this case we are only interested in the final underlying price.
// Therefore, we can cover all the residual time in one big time step.
int timeSteps = 1;

Handle<GaussianPathGenerator> pathGenerator(
    new GaussianPathGenerator(nu,vol2,residualTime,timeSteps));
```

where [QuantLib::MonteCarlo::GaussianPathGenerator](#) is a typedef to a path generator using the default choice for a Gaussian random number generator.

The path pricer is instantiated as

```
// discount at maturity
DiscountFactor discount = QL_EXP(-riskFreeRate*residualTime);
bool antithetic = true;

Handle<PathPricer<Path> > pathPricer(
    new EuropeanPathPricer(type,underlying,strike,discount,antithetic));
```

The model can now be created and used as following:

```
// number of samples to be generated
size_t samples = 1000000;

// pass the path generator and pricer we just created and a
// newly instantiated Statistics object
MonteCarloModel<Statistics,GaussianPathGenerator,PathPricer> model(
    pathGenerator,pathPricer,Statistics());

model.addSamples(samples);

// now get the results: the option price is given by value with
// a confidence level given by error
value = model.sampleAccumulator().mean();
error = model.sampleAccumulator().errorEstimate();
```

More examples of path pricers can be found in the [QuantLib::MonteCarlo](#) namespace, while examples of more sophisticated pricers which uses them in Monte Carlo models can be found in the [QuantLib::Pricers](#) namespace.

2.4.5 Notes

(1) A more rigorous approach would lead us to integrate the above equation by means of Green functions or Laplace transforms. Both such methods would show that the price at time $t = 0$ of an option with payoff $G(S(T))$ where $S(T)$ is the underlying price at expiry is given by the integral

$$\int_{-\infty}^{\infty} e^{-rT} G(S_0 e^{\xi}) \frac{1}{\sqrt{2\pi\sigma^2 T}} \exp\left(-\frac{(\xi - \nu T)^2}{2\sigma^2 T}\right) d\xi$$

where S_0 is the price of the underlying at $t = 0$. It can be seen that the above integral is of the form shown at the beginning of this section, namely, the pricing function is

$$f(x) = e^{-rT} G(S_0 e^x)$$

and can be interpreted as the option payoff discounted to the present time, while the probability distribution is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2T}} \exp\left(-\frac{(x - \nu T)^2}{2\sigma^2T}\right).$$

which again shows that the logarithms of the underlying prices at time T are distributed as a Gaussian with average νT and standard deviation $\sigma\sqrt{T}$.

2.5 The interest rate modelling framework

This framework (corresponding to the `InterestRateModelling` namespace) implements some basic interest rate modelling tools, i.e. single factor short rate models. The models implemented in this library are widely used by practitioners. For the moment, the model class basically define a driving stochastic equation of the type

$$dx = \mu(t, x)dt + \sigma(t, x)dW_t$$

where $r = f(t, x)$ and define, if possible, analytical formulas for discount bonds and discount bond options (useful for calibration). Numerical pricing can be done via recombining trinomial trees implemented in the `Lattices::TrinomialTree` class.

2.5.1 Single-factor models

The Hull & White model

$$dr_t = (\theta(t) - \alpha(t)r_t)dt + \sigma(t)dW_t$$

When α and σ are constants, this model has analytical formulas for discount bonds and discount bond options.

The Black-Karasinski model

$$d \ln r_t = (\theta(t) - \alpha \ln r_t)dt + \sigma dW_t$$

No analytical tractability here.

The Cox-Ingersoll-Ross model

$$dr_t = (\theta(t) - kr_t)dt + \sigma\sqrt{r_t}dW_t$$

There are analytical formulas for discount bonds and discount bond options.

2.5.2 Calibration

The class `CalibrationHelper` is a base class that facilitates the instantiation of market instruments used for calibration. It has a method `marketValue()` that gives the market price using a Black formula, and a `modelValue()` method that gives the price according to a model

The first derived class is `QuantLib::InterestRateModelling::CalibrationHelpers::CapHelper`. This helper creates a vanilla cap instrument, and prices it using discount bond option formulas. If no such analytical formula is available, the pricing is done using a trinomial recombining tree.

The second derived class is `QuantLib::InterestRateModelling::CalibrationHelpers::SwaptionHelper`. This helper creates a swaption instrument, and prices it using the Jamshidian decomposition. If no analytical formula for discount bonds is available, the pricing is done using a trinomial recombining tree.

For the calibration itself, you must choose an optimization method that will find constant parameters such that the value:

$$V = \sqrt{\sum_{i=1}^n \frac{T_i - M_i}{M_i}}^2$$

, where T_i is the price given by the model and M_i is the market price, is minimized. A few optimization methods are available in the `Optimization` namespace, and more are on the way...

2.5.3 Pricing using finite differences

If $x = x(t, r)$ is the state variable and follows this stochastic process:

$$dx_t = \mu(t, x)dt + \sigma(t, x)dW_t$$

any european-style instrument will follow the following PDE:

$$\frac{\partial P}{\partial t} + \mu \frac{\partial P}{\partial x} + \frac{1}{2} \sigma^2 \frac{\partial^2 P}{\partial x^2} = r(t, x)P$$

The adequate operator to feed a Finite Difference Model instance is defined in the [QuantLib::FiniteDifferences::OneFactorOperator](#) class.

2.5.4 Pricing using trees

Each model derived from the single-factor model class has the ability to return a trinomial tree. For yield-curve consistent models, the fitting parameter can be determined either analytically (when possible) or numerically. When a tree is built, it is then pretty straightforward to implement a pricer for any path-independent derivative. Just implement a class derived from `NumericalDerivative` (see [QuantLib::Pricers::NumericalSwaption](#) for example) and roll it back until the present time... Just look at [QuantLib::Pricers::TreeCapFloor](#) and [QuantLib::Pricers::TreeSwaption](#) for working pricers.

2.6 Currencies and FX rates

Documentation for this part of the library is in progress.

2.7 Instruments and pricers

Documentation for this part of the library is in progress.

This section will include documentation for classes such as [QuantLib::Instrument](#), its derived classes in the [QuantLib::Instruments](#) namespace, and the pricing engines in the [QuantLib::Pricers](#) namespace.

2.8 Math tools

Documentation for this part of the library is in progress.

Math facilities of the library include:

2.8.1 Random number and low discrepancy sequence generators

Documentation for this part of the library is in progress.

Implementations of random number and low discrepancy sequence generators. They share the [QuantLib::RandomNumbers](#) namespace.

2.8.2 One-dimensional solvers

Documentation for this part of the library is in progress.

The abstract class [QuantLib::Solver1D](#) provides the interface for one-dimensional solvers which can find the zeroes of a given function.

A number of such solvers is contained in the [QuantLib::Solvers1D](#) namespace.

The implementation of the algorithms was inspired by "Numerical Recipes in C", 2nd edition, Press, Teukolsky, Vetterling, Flannery - Chapter 9

Some work is needed to resolve the ambiguity of the root finding accuracy definition: for some algorithms it is the x-accuracy, for others it is f(x)-accuracy.

2.8.3 Optimizers

The optimization framework (corresponding to the Optimization namespace) implements some multi-dimensional minimizing methods. The function to be minimized is to be derived from the [QuantLib::Optimization::CostFunction](#) base class (if the gradient is not analytically implemented, it will be computed numerically).

The simplex method

see [QuantLib::Optimization::Simplex](#) .

The conjugate gradient method

see [QuantLib::Optimization::ConjugateGradient](#) .

2.9 Design patterns

Documentation for this part of the library is in progress.

2.10 Term structures

Documentation for this part of the library is in progress.

The abstract class [QuantLib::TermStructure](#) provides the common interface to concrete term structure models. Among others, methods are declared which return instantaneous forward rate, discount factor, and zero rate at a given date. Adapter classes are provided which already implement part of the required methods, thus allowing the programmer to define only the non-redundant part. The [PiecewiseConstantForwards](#) class is provided as an example in the [QuantLib::TermStructures](#) namespace.

2.11 Utilities

Documentation for this part of the library is in progress.

Iterators are meant to build a sequence on the fly from one or more other sequences, without having to allocate place for storing it. A couple of examples: suppose we have a function which calculates the average of a sequence, and that for genericity we have implemented it as a template function which takes the beginning and the end of the sequence, so that its declaration is:

```
template <class Iterator>
typename Iterator::value_type
average(const Iterator& begin, const Iterator& end)
```

This kind of genericity allows one to use the same function to calculate the average of a `std::vector`, a `std::list`, a [QuantLib::History](#), any other container, of a subset of any of the former.

Now let's say we have two sequences of numbers, and we want to calculate the average of their products. One approach could be to store the products in another sequence, and to calculate the average of the latter, as in:

```
// we have sequence1 and sequence2 and assume equal size:
// first we store their product in a vector...
std::vector<double> products;
std::transform(sequence1.begin(),sequence1.end(), // first sequence
               sequence2.begin(),               // second sequence
               std::back_inserter(products),     // output
               std::multiplies<double>());       // operation to perform
// ...then we calculate the average
double result = average(products.begin(),products.end());
```

The above works, however, it might be not particularly efficient since we have to allocate the product vector, quite possibly just to throw it away when the calculation is done.

[QuantLib::Utilities::coupling_iterator](#) allows us to do the same thing without allocating the extra vector: what we do is simply:

```
// we have sequence1 and sequence2 and assume equal size:
double result = average(
    make_coupling_iterator(sequence1.begin(),
                           sequence2.begin(),
                           std::multiplies<double>()),
    make_coupling_iterator(sequence1.end(),
                           sequence2.end(),
                           std::multiplies<double>()));
```

The call to `make_coupling_iterator` creates an iterator which is really a reference to the two iterators and the operation we passed. Dereferencing such iterator returns the result of applying such operation to the values pointed to by the two contained iterators. Advancing the coupling iterator advances the two underlying ones. You can see how iterating on such iterator is quite an alliteration but generates the products one by one so that they can be processed by `average()`, but does not need allocating memory for storing the results. The product sequence is generated on the fly.

The other iterators share the same principle but have different functionalities:

- `combining_iterator` is the same of `coupling_iterator`, but works on N sequences while the latter works on 2;
- `filtering_iterator` generates the elements of a given sequence which satisfy a given predicate, i.e., it takes a sequence `[x_0,x_1,...]` and a predicate `p` and generates the sequence of those `x_i` for which `p(x_i)` returns true;

- `processing_iterator` takes a sequence $[x_0, x_1, \dots]$ and a function f and generates the sequence $[f(x_0), f(x_1), \dots]$;
- `stepping_iterator` takes a sequence $[x_0, x_1, \dots]$ and a step m and generates the sequence $[x_0, x_m, x_{2m}, \dots]$

Chapter 3

Examples

More examples of using the QuantLib library can be found in chapter [13](#) of the reference manual. Should your viewer allow it, they can also be reached via the hyperlinks in the following list.

3.1 QuantLib Examples

Here is a list of all examples:

- [DiscreteHedging.cpp](#)
- [EuropeanOption.cpp](#)
- [history_iterators.cpp](#)
- [swapvaluation.cpp](#)

Part II

Reference Manual

Chapter 4

QuantLib Module Index

4.1 QuantLib Modules

Here is a list of all modules:

Deprecated classes	69
Global QuantLib macros	70
Math functions	71
Numeric limits	72
Time functions	73
Character functions	74
Input/output functions	75
Min and max functions	76
Template capabilities	77
Iterator support	78

Chapter 5

QuantLib Namespace Index

5.1 QuantLib Namespace List

Here is a list of all documented namespaces with brief descriptions:

QuantLib (A.k.a. the QuantLib Foundation)	79
QuantLib::Calendars (Specialized Calendar classes)	87
QuantLib::CashFlows (Concrete implementations of the CashFlow interface)	88
QuantLib::DayCounters (Specialized DayCounter classes)	89
QuantLib::FiniteDifferences (Finite difference framework)	90
QuantLib::Indexes (Concrete implementations of the Index interface)	92
QuantLib::Instruments (Concrete implementations of the Instrument interface)	93
QuantLib::Math (Mathematical functions and classes)	94
QuantLib::MonteCarlo (Monte Carlo framework)	96
QuantLib::Patterns (Implementations of design patterns)	98
QuantLib::Pricers (Pricing models for options)	99
QuantLib::RandomNumbers (Random Number Generators and Low Discrepancy Sequences) . .	102
QuantLib::Solvers1D (Concrete implementations of the Solver1D interface)	103
QuantLib::TermStructures (Concrete implementations of the TermStructure interface)	104
QuantLib::Utilities (Classes and functions of general utility)	105

Chapter 6

QuantLib Hierarchical Index

6.1 QuantLib Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Arguments	111
CapFloorParameters	141
PlainOptionParameters	322
SwaptionParameters	367
Array	114
Grid	222
BoundaryCondition	129
BoxMullerGaussianRng	130
Calendar	134
Frankfurt	211
Helsinki	225
Johannesburg	251
London	261
Milan	277
NewYork	290
Sydney	372
TARGET	374
Tokyo	379
Toronto	380
Wellington	387
Zurich	396
CalendarImpl	137
WesternCalendarImpl	138
CLGaussianRng	147
combining_iterator	149
CostFunction	154
LeastSquareFunction	254
coupling_iterator	155
CurrencyFormatter	161
Date	162
DateFormatter	164
DayCounter	165

Actual360	107
Actual365	108
ActualActual	109
Thirty360	377
DayCounterImpl	167
DiffusionProcess	171
BlackScholesProcess	127
OrnsteinUhlenbeckProcess	309
DoubleFormatter	176
Error	180
AssertionFailedError	117
IllegalArgumentError	235
IllegalResultError	236
IndexError	241
OutOfMemoryError	310
PostconditionNotSatisfiedError	324
PreconditionNotSatisfiedError	325
EuroFormatter	182
Exercise	187
FdBermudanOption	193
FdDividendShoutOption	197
filtering_iterator	200
FiniteDifferenceModel	201
FixedRateCouponVector	203
FloatingRateCouponVector	206
Handle	223
Handle< InterestRateModelling::AffineModel >	223
Handle< InterestRateModelling::BlackModel >	223
Handle< InterestRateModelling::OneFactorAffineModel >	223
Handle< InterestRateModelling::OneFactorModel >	223
Handle< Link< Type > >	223
RelinkableHandle	332
Handle< MonteCarlo::GaussianMultiPathGenerator >	223
Handle< MonteCarlo::GaussianPathGenerator >	223
Handle< MonteCarlo::MonteCarloModel< Math::Statistics, MonteCarlo::GaussianMultiPath- Generator, MonteCarlo::PathPricer< MonteCarlo::MultiPath > > >	223
Handle< MonteCarlo::MonteCarloModel< Math::Statistics, MonteCarlo::GaussianPath- Generator, MonteCarlo::PathPricer< MonteCarlo::Path > > >	223
Handle< MonteCarlo::PathPricer< MonteCarlo::MultiPath > >	223
Handle< MonteCarlo::PathPricer< MonteCarlo::Path > >	223
History	227
const_iterator	230
Entry	231
ICGaussianRng	234
Index	240
Xibor	388
AUDLibor	118
CADLibor	133
CHFLibor	146
Euribor	181
GBPLibor	215
JPYLibor	252

USDLibor	386
ZARLibor	391
IntegerFormatter	245
Interpolation	246
CubicSpline	160
LinearInterpolation	258
Interpolation2D	248
BilinearInterpolation	121
KnuthUniformRng	253
LecuyerUniformRng	255
LexicographicalView	256
lowest_category_iterator	262
Matrix	264
McPricer	276
McPricer< Math::Statistics, MonteCarlo::GaussianMultiPathGenerator, MonteCarlo::Path- Pricer< MonteCarlo::MultiPath > >	276
McBasket	268
McEverest	272
McHimalaya	273
McMaxBasket	274
McPagoda	275
McPricer< Math::Statistics, MonteCarlo::GaussianPathGenerator, MonteCarlo::PathPricer< MonteCarlo::Path > >	276
McDiscreteArithmeticAPO	269
McDiscreteArithmeticASO	270
McEuropean	271
MixedScheme	278
CrankNicolson	159
ExplicitEuler	188
ImplicitEuler	237
MonteCarloModel	283
MonteCarloModel< Math::Statistics, MonteCarlo::GaussianMultiPathGenerator, Monte- Carlo::PathPricer< MonteCarlo::MultiPath > >	283
MonteCarloModel< Math::Statistics, MonteCarlo::GaussianPathGenerator, MonteCarlo::Path- Pricer< MonteCarlo::Path > >	283
MultiPath	284
MultiPathGenerator	285
MultivariateAccumulator	286
NonLinearLeastSquare	291
Null	293
ObjectiveFunction	294
Observable	295
CapFlatVolatilityStructure	139
CapFlatVolatilityVector	140
CapletForwardVolatilityStructure	144
CashFlow	145
Coupon	157
FixedRateCoupon	202
FloatingRateCoupon	204
ShortFloatingRateCoupon	343
SimpleCashFlow	345

Instrument	242
Stock	360
Swap	362
SimpleSwap	347
Option	304
PlainOption	319
Swaption	366
BlackModel	126
Model	280
OneFactorModel	299
ExtendedVasicek	189
HullWhite	232
GeneralBlackKarasinski	216
BlackKarasinski	125
GeneralCoxIngersollRoss	218
Link	259
MarketElement	263
CompositeMarketElement	151
DerivedMarketElement	170
SimpleMarketElement	346
CapFloorPricingEngine	142
CapFloorPricingEngine< InterestRateModelling::AffineModel >	142
AnalyticalCapFloor	110
CapFloorPricingEngine< InterestRateModelling::BlackModel >	142
BlackCapFloor	124
CapFloorPricingEngine< InterestRateModelling::OneFactorModel >	142
TreeCapFloor	381
SwaptionPricingEngine	368
SwaptionPricingEngine< InterestRateModelling::BlackModel >	368
BlackSwaption	128
SwaptionPricingEngine< InterestRateModelling::OneFactorAffineModel >	368
JamshidianSwaption	250
SwaptionPricingEngine< InterestRateModelling::OneFactorModel >	368
TreeSwaption	382
SwaptionVolatilityStructure	371
SwaptionVolatilityMatrix	370
TermStructure	375
DiscountStructure	172
ImpliedTermStructure	238
ModelTermStructure	282
ForwardRateStructure	207
ForwardSpreadedTermStructure	209
PiecewiseFlatForward	316
ZeroYieldStructure	394
ZeroSpreadedTermStructure	392
RateHelper	330
DepositRateHelper	168
FraRateHelper	212
FuturesRateHelper	214
SwapRateHelper	364
Observer	297

FloatingRateCoupon	204
ShortFloatingRateCoupon	343
CompositeMarketElement	151
DerivedMarketElement	170
ForwardSpreadedTermStructure	209
ImpliedTermStructure	238
Instrument	242
BlackModel	126
Model	280
Link	259
CapFloorPricingEngine	142
CapFloorPricingEngine< InterestRateModelling::AffineModel >	142
CapFloorPricingEngine< InterestRateModelling::BlackModel >	142
CapFloorPricingEngine< InterestRateModelling::OneFactorModel >	142
SwaptionPricingEngine	368
SwaptionPricingEngine< InterestRateModelling::BlackModel >	368
SwaptionPricingEngine< InterestRateModelling::OneFactorAffineModel >	368
SwaptionPricingEngine< InterestRateModelling::OneFactorModel >	368
PiecewiseFlatForward	316
RateHelper	330
ZeroSpreadedTermStructure	392
OptimizationMethod	301
ConjugateGradient	152
Simplex	348
SteepestDescent	356
OptimizationProblem	303
OptionPricingEngine	307
CapFloorPricingEngine	142
CapFloorPricingEngine< InterestRateModelling::AffineModel >	142
CapFloorPricingEngine< InterestRateModelling::BlackModel >	142
CapFloorPricingEngine< InterestRateModelling::OneFactorModel >	142
PlainOptionEngine	321
EuropeanEngine	183
SwaptionPricingEngine	368
SwaptionPricingEngine< InterestRateModelling::BlackModel >	368
SwaptionPricingEngine< InterestRateModelling::OneFactorAffineModel >	368
SwaptionPricingEngine< InterestRateModelling::OneFactorModel >	368
Path	312
PathGenerator	313
PathPricer	314
PathPricer< MonteCarlo::MultiPath >	314
PathPricer< MonteCarlo::Path >	314
PathPricer< MultiPath >	314
BasketPathPricer	120
EverestPathPricer	186
HimalayaPathPricer	226
MaxBasketPathPricer	267
PagodaPathPricer	311
PathPricer< Path >	314
ArithmeticAPOPPathPricer	112
ArithmeticASOPPathPricer	113
EuropeanPathPricer	185

GeometricAOPathPricer	220
GeometricASOPathPricer	221
Period	315
processing_iterator	326
RandomArrayGenerator	328
RateFormatter	329
Results	334
OptionGreeks	306
PlainOptionResults	323
OptionValue	308
CapFloorResults	143
PlainOptionResults	323
SwaptionResults	369
RiskMeasures	336
RiskStatistics	337
Sample	339
Scheduler	340
SegmentIntegral	342
SingleAssetOption	349
BarrierOption	119
BinaryOption	122
CliquetOption	148
DiscreteGeometricAPO	173
DiscreteGeometricASO	174
EuropeanOption	184
ContinuousGeometricAPO	153
FdDividendEuropeanOption	196
FdBsmOption	194
FdEuropean	198
FdStepConditionOption	199
FdAmericanOption	192
Solver1D	351
Bisection	123
Brent	131
FalsePosition	191
Newton	288
NewtonSafe	289
Ridder	335
Secant	341
Statistics	353
StepCondition	357
StepCondition< Array >	357
stepping_iterator	358
StringFormatter	361
SymmetricSchurDecomposition	373
TimeGrid	378
TridiagonalOperator	383
BSMOperator	132
DMinus	175
DPlus	177
DPlusDMinus	178
DZero	179

OneFactorOperator	300
TimeSetter	385
XiborManager	390

Chapter 7

QuantLib Compound Index

7.1 QuantLib Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

Actual360 (Actual/360 day count convention)	107
Actual365 (Actual/365 day count convention)	108
ActualActual (Actual/Actual day count)	109
AnalyticalCapFloor (Analytical pricer for cap/floor)	110
Arguments (Base class for generic argument groups)	111
ArithmeticAPOPPathPricer (path pricer for arithmetic average price option)	112
ArithmeticASOPathPricer (path pricer for average strike Asian options)	113
Array (1-D array used in linear algebra)	114
AssertionFailedError (Specialized error)	117
AUDLibor (AUD Libor index (Also known as TIBOR, check settlement days))	118
BarrierOption (Barrier option)	119
BasketPathPricer (Multipath pricer for European-type basket option)	120
BilinearInterpolation (Bilinear interpolation between discrete points)	121
BinaryOption (Binary (digital) option)	122
Bisection (bisection 1-D solver)	123
BlackCapFloor (CapFloor priced by the Black formula)	124
BlackKarasinski (Standard Black-Karasinski model class)	125
BlackModel (Abstract short-rate model class)	126
BlackScholesProcess (Black-Scholes diffusion process class)	127
BlackSwaption (Swaption priced by the Black formula)	128
BoundaryCondition (Boundary condition for finite difference problems)	129
BoxMullerGaussianRng (Gaussian random number generator)	130
Brent (Brent 1-D solver)	131
BSMOperator (Black-Scholes-Merton differential operator)	132
CADLibor (CAD Libor index (Also known as CDOR, check settlement days))	133
Calendar (calendar class)	134
CalendarImpl (Abstract base class for calendar implementations)	137
WesternCalendarImpl	138
CapFlatVolatilityStructure (Cap/floor flat volatility structure)	139
CapFlatVolatilityVector (Cap/floor at-the-money flat volatility vector)	140
CapFloorParameters (Parameters for cap/floor calculation)	141
CapFloorPricingEngine (Base class for cap/floor pricing engines)	142
CapFloorResults (results from cap/floor calculation)	143

CapletForwardVolatilityStructure (Caplet/floorlet forward volatility structure)	144
CashFlow (Base class for cash flows)	145
CHFLibor (CHF Libor index (Also known as ZIBOR, check settlement days and day-count)) . .	146
CLGaussianRng (Gaussian random number generator)	147
CliquetOption (Cliquet (Ratchet) option)	148
combining_iterator (Iterator mapping a function to a set of underlying sequences)	149
CompositeMarketElement (Market element whose value depends on two other market element) .	151
ConjugateGradient (Multi-dimensionnal Conjugate Gradient class)	152
ContinuousGeometricAPO (Continuous Geometric Average Price Option (European exercise)) .	153
CostFunction (Cost function abstract class for optimization problem)	154
coupling_iterator (Iterator mapping a function to a pair of underlying sequences)	155
Coupon (coupon accruing over a fixed period)	157
CrankNicolson (Crank-Nicolson scheme for finite difference methods)	159
CubicSpline (Cubic spline interpolation between discrete points)	160
CurrencyFormatter (Formats currencies for output)	161
Date (Concrete date class)	162
DateFormatter (Formats dates for output)	164
DayCounter (Day counter class)	165
DayCounterImpl (Abstract base class for day counter implementations)	167
DepositRateHelper (Deposit rate)	168
DerivedMarketElement (Market element whose value depends on another market element) . . .	170
DiffusionProcess (Diffusion process class)	171
DiscountStructure (Discount factor term structure)	172
DiscreteGeometricAPO (Discrete Geometric Average Price Asian Option (European style)) . . .	173
DiscreteGeometricASO (Discrete Geometric Average Strike Asian Option (European style)) . .	174
DMinus (D_- matricial representation)	175
DoubleFormatter (Formats doubles for output)	176
DPlus (D_+ matricial representation)	177
DPlusDMinus ($D_+ D_-$ matricial representation)	178
DZero (D_0 matricial representation)	179
Error (Base error class)	180
Euribor (Euribor index)	181
EuroFormatter (Formats amounts in Euro for output)	182
EuropeanEngine (Analytic pricing engine for European options)	183
EuropeanOption (Black-Scholes-Merton European option)	184
EuropeanPathPricer (path pricer for European options)	185
EverestPathPricer (path pricer for European-type Everest option)	186
Exercise (Exercise class (American, Bermudan or European))	187
ExplicitEuler (Forward Euler scheme for finite difference methods)	188
ExtendedVasicek (Extended Vasicek model class)	189
FalsePosition (False position 1-D solver)	191
FdAmericanOption (American option)	192
FdBermudanOption (Bermudan option)	193
FdBsmOption (Black-Scholes-Merton option priced numerically)	194
FdDividendEuropeanOption (European option with dividends)	196
FdDividendShoutOption (Shout option with dividends)	197
FdEuropean (Example of European option calculated using finite differences)	198
FdStepConditionOption (option executing additional code at each time step)	199
filtering_iterator (Iterator filtering undesired data)	200
FiniteDifferenceModel (Generic finite difference model)	201
FixedRateCoupon (coupon paying a fixed interest rate)	202
FixedRateCouponVector (Helper class building a sequence of fixed rate coupons)	203
FloatingRateCoupon (coupon at par on a term structure)	204
FloatingRateCouponVector (Helper class building a sequence of floating rate coupons)	206

ForwardRateStructure (Forward rate term structure)	207
ForwardSpreadedTermStructure (Term structure with an added spread on the instantaneous forward rate)	209
Frankfurt (Frankfurt calendar)	211
FraRateHelper (Forward rate agreement)	212
FuturesRateHelper (Interest Rate Futures)	214
GBPLibor (GBP Libor index)	215
GeneralBlackKarasinski (General Black-Karasinski model class)	216
GeneralCoxIngersollRoss (General single-factor extended Cox-Ingersoll-Ross model class)	218
GeometricAPOPPathPricer (path pricer for geometric average price option)	220
GeometricASOPathPricer (path pricer for geometric average price option)	221
Grid (Spatial grid class)	222
Handle (Reference-counted pointer)	223
Helsinki (Helsinki calendar)	225
HimalayaPathPricer (Multipath pricer for European-type Himalaya option)	226
History (Container for historical data)	227
const_iterator (Random access iterator on history entries)	230
Entry (Single datum in history)	231
HullWhite (Analytically tractable single-factor Hull-White model class)	232
ICGaussianRng (Inverse cumulative Gaussian random number generator)	234
IllegalArgumentError (Specialized error)	235
IllegalResultError (Specialized error)	236
ImplicitEuler (Backward Euler scheme for finite difference methods)	237
ImpliedTermStructure (Implied term structure at a given date in the future)	238
Index (Purely virtual base class for indexes)	240
IndexError (Specialized error)	241
Instrument (Abstract instrument class)	242
IntegerFormatter (Formats integers for output)	245
Interpolation (Abstract base class for 1-D interpolations)	246
Interpolation2D (Abstract base class for 2-D interpolations)	248
JamshidianSwaption (Jamshidian swaption pricer)	250
Johannesburg (Johannesburg calendar)	251
JPYLibor (JPY Libor index (Also known as TIBOR, check settlement days))	252
KnuthUniformRng (Uniform random number generator)	253
LeastSquareFunction	254
LecuyerUniformRng (Uniform random number generator)	255
LexicographicalView (Lexicographical 2-D view of a contiguous set of data)	256
LinearInterpolation (Linear interpolation between discrete points)	258
Link (Relinkable access to a Handle)	259
London (London calendar)	261
lowest_category_iterator (Most generic of two given iterator categories)	262
MarketElement (Purely virtual base class for market observables)	263
Matrix (matrix used in linear algebra)	264
MaxBasketPathPricer (Multipath pricer for European-type basket option)	267
McBasket (Simple example of multi-factor Monte Carlo pricer)	268
McDiscreteArithmeticAPO (Example of Monte Carlo pricer using a control variate)	269
McDiscreteArithmeticASO (Example of Monte Carlo pricer using a control variate)	270
McEuropean (Simple example of Monte Carlo pricer)	271
McEverest (Everest-type option pricer)	272
McHimalaya (Himalayan-type option pricer)	273
McMaxBasket (Simple example of multi-factor Monte Carlo pricer)	274
McPagoda (Roofed Asian option)	275
McPricer (Base class for Monte Carlo pricers)	276
Milan (Milan calendar)	277

MixedScheme (Mixed (explicit/implicit) scheme for finite difference methods)	278
Model (Abstract short-rate model class)	280
ModelTermStructure (Term structure implied by a model)	282
MonteCarloModel (General purpose Monte Carlo model for path samples)	283
MultiPath (Single random walk)	284
MultiPathGenerator (Generates a multipath from a random number generator)	285
MultivariateAccumulator (A sample accumulator for multivariate analysis)	286
Newton (Newton 1-D solver)	288
NewtonSafe (Safe Newton 1-D solver)	289
NewYork (New York calendar)	290
NonLinearLeastSquare	291
Null (Template class providing a null value for a given type)	293
ObjectiveFunction (Objective function for 1-D solvers)	294
Observable (Object that notifies its changes to a set of observables)	295
Observer (Object that gets notified when a given observable changes)	297
OneFactorModel (Single-factor short-rate model abstract class)	299
OneFactorOperator (Interest-rate single factor model differential operator)	300
OptimizationMethod (Optimization Method abstract class for unconstrained optimization pb)	301
OptimizationProblem (Unconstrained optimization pb)	303
Option (Base option class)	304
OptionGreeks (option pricing results)	306
OptionPricingEngine (Base class for option pricing engines)	307
OptionValue (option pricing results)	308
OrnsteinUhlenbeckProcess (Ornstein-Uhlenbeck process class)	309
OutOfMemoryError (Specialized error)	310
PagodaPathPricer (multipath pricer for pagoda options)	311
Path (Single factor random walk)	312
PathGenerator (Generates random paths from a random number generator)	313
PathPricer (Base class for path pricers)	314
Period (Time period described by a number of a given time unit)	315
PiecewiseFlatForward (Piecewise flat forward term structure)	316
PlainOption (Plain (no dividends, no barriers) option on a single asset)	319
PlainOptionEngine (Base class for plain option pricing engines)	321
PlainOptionParameters (Parameters for plain option calculation)	322
PlainOptionResults (results from plain option calculation)	323
PostconditionNotSatisfiedError (Specialized error)	324
PreconditionNotSatisfiedError (Specialized error)	325
processing_iterator (Iterator mapping a unary function to an underlying sequence)	326
RandomArrayGenerator (Generates random arrays from a random number generator)	328
RateFormatter (Formats rates for output)	329
RateHelper (Base class for rate helpers)	330
RelinkableHandle (Globally accessible relinkable pointer)	332
Results (Base class for generic result groups)	334
Ridder (Ridder 1-D solver)	335
RiskMeasures (Interface for risk functions)	336
RiskStatistics (Risk analysis tool)	337
Sample (Weighted sample)	339
Scheduler (Date scheduler)	340
Secant (secant 1-D solver)	341
SegmentIntegral (Integral of a one-dimensional function)	342
ShortFloatingRateCoupon (Short coupon at par on a term structure)	343
SimpleCashFlow (Predetermined cash flow)	345
SimpleMarketElement (Market element returning a stored value)	346
SimpleSwap (Simple fixed-rate vs Libor swap)	347

Simplex (Multi-dimensionnal Simplex class)	348
SingleAssetOption (Black-Scholes-Merton option)	349
Solver1D (Abstract base class for 1-D solvers)	351
Statistics (Statistic tool)	353
SteepestDescent (Multi-dimensionnal Steepest Descend class)	356
StepCondition (Condition to be applied at every time step)	357
stepping_iterator (Iterator advancing in constant steps)	358
Stock (Simple stock class)	360
StringFormatter (Formats strings as lower- or uppercase)	361
Swap (Interest rate swap)	362
SwapRateHelper (Swap rate)	364
Swaption (Swaption class)	366
SwaptionParameters (Parameters for swaption calculation)	367
SwaptionPricingEngine (Base class for swaption pricing engines)	368
SwaptionResults (results from swaption calculation)	369
SwaptionVolatilityMatrix (Swaption at-the-money volatility matrix)	370
SwaptionVolatilityStructure (Swaption volatility structure)	371
Sydney (Sydney, calendar (New South Wales, Australia))	372
SymmetricSchurDecomposition (Symmetric threshold Jacobi algorithm)	373
TARGET (TARGET calendar)	374
TermStructure (Term structure)	375
Thirty360 (30/360 day count convention)	377
TimeGrid (Time grid class)	378
Tokyo (Tokyo calendar)	379
Toronto (Toronto calendar)	380
TreeCapFloor (Cap/Floor priced in a tree)	381
TreeSwaption (Swaption priced in a tree)	382
TridiagonalOperator (Base implementation for tridiagonal operator)	383
TimeSetter (Encapsulation of time-setting logic)	385
USDLibor (USD Libor index)	386
Wellington (Wellington calendar)	387
Xibor (Base class for libor indexes)	388
XiborManager (Global repository for libor histories)	390
ZARLibor (ZAR Libor index (also known as JIBAR, check settlement days))	391
ZeroSpreadedTermStructure (Term structure with an added spread on the zero yield rate)	392
ZeroYieldStructure (Zero yield term structure)	394
Zurich (Zurich calendar)	396

Chapter 8

QuantLib File Index

8.1 QuantLib File List

Here is a list of all documented files with brief descriptions:

actual360.hpp (Act/360 day counter)	397
actual365.hpp (Act/365 day counter)	398
actualactual.cpp (Act/act day counters)	399
actualactual.hpp (Act/act day counters)	400
americancondition.hpp (American option exercise condition)	401
analyticalcapfloor.cpp (Analytical pricer for caps/floors)	402
analyticalcapfloor.hpp (Analytical pricer for caps/floors)	403
argsandresults.hpp (Base classes for generic arguments and results)	404
arithmeticapopathpricer.cpp (Arithmetic average price option path pricer)	405
arithmeticapopathpricer.hpp (Arithmetic average price option path pricer)	406
arithmeticasopathpricer.cpp (Arithmetic average strike option path pricer)	407
arithmeticasopathpricer.hpp (Arithmetic average strike option path pricer)	408
armijo.cpp (Armijo line-search class)	409
armijo.hpp (Armijo line-search class)	410
array.hpp (1-D array used in linear algebra)	411
audlibor.hpp (AUD Libor index (check settlement days))	412
barrieroption.cpp (Barrier option)	413
barrieroption.hpp (Barrier option)	414
basketpathpricer.cpp (Multipath pricer for European-type basket option)	415
basketpathpricer.hpp (Multipath pricer for European-type basket option)	416
bilinearinterpolation.hpp (Bilinear interpolation between discrete points)	417
binaryoption.cpp (European style cash-or-nothing option)	418
binaryoption.hpp (European style cash-or-nothing option)	419
binomialtree.cpp (Binomial tree class)	420
binomialtree.hpp (Binomial tree class)	421
bisection.cpp (Bisection 1-D solver)	422
bisection.hpp (Bisection 1-D solver)	423
blackcapfloor.cpp (European capfloor calculated using Black formula)	424
blackcapfloor.hpp (CapFloor calculated using the Black formula)	425
blackkarasinski.cpp (Black-Karasinski model)	426
blackkarasinski.hpp (Black-Karasinski model)	427
blackmodel.hpp (Abstract class for Black-type models (market models))	428
blackswaption.cpp (European swaption calculated using Black formula)	429

blackswaption.hpp (Swaption calculated using the Black formula)	430
boundarycondition.hpp (Boundary conditions for differential operators)	431
boxmullergaussianrng.hpp (Box-Muller Gaussian random-number generator)	432
brent.cpp (Brent 1-D solver)	433
brent.hpp (Brent 1-D solver)	434
bsmoperator.cpp (Differential operator for Black-Scholes-Merton equation)	435
bsmoperator.hpp (Differential operator for Black-Scholes-Merton equation)	436
cadlibor.hpp (CAD Libor index (Also known as CDOR, check settlement days))	437
calendar.cpp (Abstract calendar class)	438
calendar.hpp (calendar class)	439
calibrationhelper.cpp (Calibration helper class)	440
calibrationhelper.hpp (Calibration helper class)	441
capflatvolvector.hpp (Cap/floor at-the-money flat volatility vector)	442
capfloor.cpp (European cap and floor class)	443
capfloor.hpp (Cap and Floor class)	444
caphelper.cpp (Cap calibration helper)	445
caphelper.hpp (CapHelper calibration helper)	446
capvolstructures.hpp (Cap/Floor volatility structures)	447
cashflow.hpp (Base class for cash flows)	448
cashflowvectors.cpp (Cash flow vector builders)	449
cashflowvectors.hpp (Cash flow vector builders)	450
centrallimitgaussianrng.hpp (Central limit Gaussian random-number generator)	451
chflibor.hpp (CHF Libor index (Also known as ZIBOR, check settlement days and day-count))	452
cliquetoption.cpp (Textbook example of european-style multi-period option)	453
cliquetoption.hpp (Textbook example of european-style multi-period option)	454
combiningiterator.hpp (Iterator mapping a function to a set of underlying sequences)	455
config.ansi.hpp	??
conjugategradient.cpp (Conjugate gradient optimization method)	456
conjugategradient.hpp (Conjugate gradient optimization method)	457
constraint.hpp (Abstract constraint class)	458
continuousgeometricapocap.hpp (Continuous Geometric Average Price Option (European exercise))	459
costfunction.hpp (Optimization cost function class)	460
couplingiterator.hpp (Iterator mapping a function to a pair of underlying sequences)	461
coupon.hpp (Coupon accruing over a fixed period)	462
coxingersollross.cpp (Cox-Ingersoll-Ross model)	463
coxingersollross.hpp (Cox-Ingersoll-Ross model)	464
cranknicolson.hpp (Crank-Nicolson scheme for finite difference methods)	465
criteria.hpp (Optimization criteria class)	466
cubicspline.hpp (Cubic spline interpolation between discrete points)	467
currency.hpp (Known currencies)	468
dataformatters.cpp (Classes used to format data for output)	469
dataformatters.hpp (Classes used to format data for output)	470
date.cpp (Date- and time-related classes, typedefs and enumerations)	471
date.hpp (Date- and time-related classes, typedefs and enumerations)	472
daycounter.hpp (Day counter class)	473
daycounters.cpp (Day counters functions)	474
daycounters.hpp (Day counters functions)	475
diffusionprocess.hpp (Diffusion process)	476
discretegeometricapocap.hpp (Discrete Geometric Average Price Option)	477
discretegeometricapocap.hpp (Discrete Geometric Average Price Option)	478
discretegeometriccaso.hpp (Discrete Geometric Average Strike Option)	479
discretegeometriccaso.hpp (Discrete Geometric Average Strike Option)	480
dminus.hpp (D_- matricial representation)	481
dplus.hpp (D_+ matricial representation)	482

dplusplusminus.hpp (D_+D_- matricial representation)	483
dzero.hpp (D_0 matricial representation)	484
errors.hpp (Classes and functions for error handling)	485
euribor.hpp (Euribor index)	487
europeanengine.cpp (Analytic pricing engine for European options)	488
europeanengine.hpp (Analytic pricing engine for European options)	489
europeanoption.cpp (European option)	490
europeanoption.hpp (European option)	491
europeanpathpricer.cpp (Path pricer for European options)	492
europeanpathpricer.hpp (Path pricer for European options)	493
everestpathpricer.cpp (Path pricer for European-type Everest option)	494
everestpathpricer.hpp (Path pricer for European-type Everest option)	495
exercise.hpp	??
expliciteuler.hpp (Explicit Euler scheme for finite difference methods)	496
expressiontemplates.hpp (Expression template implementation)	497
falseposition.cpp (False-position 1-D solver)	498
falseposition.hpp (False-position 1-D solver)	499
fdamericanoption.hpp	??
fdbermudanoption.hpp	??
fdbsmoption.hpp	??
fddividendamericanoption.hpp	??
fddividendeuropeanoption.hpp	??
fddividendoption.hpp	??
fddividendshoutoption.hpp	??
fd european.hpp	??
fdmultiperiodoption.hpp	??
fdshoutoption.hpp	??
fdstepconditionoption.hpp	??
fdtypedefs.hpp (Default choices for template instantiations)	500
filteringiterator.hpp (Iterator filtering undesired data)	501
finitedifferencemodel.hpp (Generic finite difference model)	502
fixedratecoupon.hpp (Coupon paying a fixed annual rate)	503
flatforward.hpp (Flat forward rate term structure)	504
floatingratecoupon.cpp (Coupon at par on a term structure)	505
floatingratecoupon.hpp (Coupon at par on a term structure)	506
frankfurt.cpp (Frankfurt calendar)	507
frankfurt.hpp (Frankfurt calendar)	508
g2.hpp (Two-additive-factor Gaussian Model G2++)	509
gbplibor.hpp (GBP Libor index)	510
geometricapopathpricer.cpp (Path pricer for geometric average price option)	511
geometricapopathpricer.hpp (Path pricer for geometric average price option)	512
geometricasopathpricer.cpp (Path pricer for geometric average strike option)	513
geometricasopathpricer.hpp (Path pricer for geometric average strike option)	514
getcovariance.cpp (Covariance matrix calculation)	515
getcovariance.hpp	??
grid.hpp (Grid classes with useful constructors for trees and finite diffs)	516
handle.hpp (Reference-counted pointer)	517
helsinki.cpp (Helsinki calendar)	518
helsinki.hpp (Helsinki calendar)	519
himalayapathpricer.cpp (Multipath pricer for European-type Himalaya option)	520
himalayapathpricer.hpp (Multipath pricer for European-type Himalaya option)	521
history.hpp (History class)	522
hullwhite.cpp (Hull & White model)	523
hullwhite.hpp (Hull & White (HW) model)	524

impliciteuler.hpp (Implicit Euler scheme for finite difference methods)	525
index.hpp (Purely virtual base class for indexes)	526
instrument.hpp (Abstract instrument class)	527
interpolation.hpp (Abstract base classes for interpolations)	528
interpolation2D.hpp (Abstract base classes for 2-D interpolations)	529
inversecumgaussianrng.hpp (Inverse cumulative Gaussian random-number generator)	530
iteratorcategories.hpp (Lowest common denominator between two iterator categories)	531
jamshidianswaption.cpp (Swaption pricer using Jamshidian's decomposition)	532
jamshidianswaption.hpp (Swaption pricer using Jamshidian's decomposition)	533
johannesburg.cpp (Johannesburg calendar)	534
johannesburg.hpp (Johannesburg calendar)	535
jpylibor.hpp (JPY Libor index (Also known as TIBOR, check settlement days))	536
knuthuniformrng.cpp (Knuth uniform random number generator)	537
knuthuniformrng.hpp (Knuth uniform random number generator)	538
leastsquare.hpp (Least square cost function)	539
lecuyeruniformrng.cpp (L'Ecuyer uniform random number generator)	540
lecuyeruniformrng.hpp (L'Ecuyer uniform random number generator)	541
lexicographicalview.hpp (Lexicographical 2-D view of a contiguous set of data)	542
linearinterpolation.hpp (Linear interpolation between discrete points)	543
linesearch.hpp (Line search abstract class)	544
london.cpp (London calendar)	545
london.hpp (London calendar)	546
marketelement.hpp (Purely virtual base class for market observables)	547
mathf.cpp (Math functions)	548
mathf.hpp (Math functions)	549
matrix.cpp (Matrix used in linear algebra)	550
matrix.hpp (Matrix used in linear algebra)	551
maxbasketpathpricer.cpp (Multipath pricer for max basket option)	552
maxbasketpathpricer.hpp (Multipath pricer for max basket option)	553
mcbasket.cpp (Simple example of multi-factor Monte Carlo pricer)	554
mcbasket.hpp	??
mcdiscretearithmeticapo.cpp (Discrete Arithmetic Average Price Option)	555
mcdiscretearithmeticapo.hpp (Discrete Arithmetic Average Price Option)	556
mcdiscretearithmeticcaso.cpp (Discrete Arithmetic Average Strike Option)	557
mcdiscretearithmeticcaso.hpp (Discrete Arithmetic Average Strike Option)	558
mceuropean.cpp (Simple example of Monte Carlo pricer)	559
mceuropean.hpp (Simple example of Monte Carlo pricer)	560
mceverest.cpp (Everest-type option pricer)	561
mceverest.hpp (Everest-type option pricer)	562
mchimalaya.cpp (Himalayan-type option pricer)	563
mchimalaya.hpp	??
mcmaxbasket.cpp (Max Basket Monte Carlo pricer)	564
mcmaxbasket.hpp	??
mcpagoda.cpp (Roofed multi asset Asian option)	565
mcpagoda.hpp	??
mcpricer.hpp (Base class for Monte Carlo pricers)	566
mctypedefs.hpp (Default choices for template instantiations)	567
milan.cpp (Milan calendar)	568
milan.hpp (Milan calendar)	569
mixedscheme.hpp (Mixed (explicit/implicit) scheme for finite difference methods)	570
model.cpp (Abstract interest rate model class)	571
model.hpp (Abstract interest rate model class)	572
montecarlomodel.hpp	??
multipath.hpp	??

multipathgenerator.hpp (Generates a multi path from a random-array generator)	573
multivariateaccumulator.cpp (A simple accumulator for vector-type samples)	574
multivariateaccumulator.hpp (A simple accumulator for vector-type samples)	575
newton.cpp (Newton 1-D solver)	576
newton.hpp (Newton 1-D solver)	577
newtonsafe.cpp (Safe (bracketed) Newton 1-D solver)	578
newtonsafe.hpp (Safe (bracketed) Newton 1-D solver)	579
newyork.cpp (New York calendar)	580
newyork.hpp (New York calendar)	581
node.hpp (Node class)	582
normaldistribution.cpp (Normal, cumulative and inverse cumulative distributions)	583
normaldistribution.hpp (Normal, cumulative and inverse cumulative distributions)	584
null.hpp (Null values)	585
numericalmethod.hpp (Numerical method class)	586
observable.hpp (Observer/observable pattern)	587
onefactormodel.cpp (Abstract one-factor interest rate model class)	588
onefactormodel.hpp (Abstract one-factor interest rate model class)	589
onefactoroperator.cpp (Differential operator for one-factor interest rate models)	590
onefactoroperator.hpp (General differential operator for one-factor interest rate models)	591
optimizer.hpp (Abstract optimization class)	592
option.cpp (Base option class)	593
option.hpp (Base option class)	594
pagodapathpricer.cpp (Path pricer for pagoda options)	595
pagodapathpricer.hpp (Path pricer for pagoda options)	596
parameter.hpp (Model parameter classes)	597
path.hpp	??
pathgenerator.hpp	??
pathpricer.hpp (Base class for single-path pricers)	598
piecewiseflatforward.cpp (Piecewise flat forward term structure)	599
piecewiseflatforward.hpp (Piecewise flat forward term structure)	600
plainoption.cpp (Plain (no dividends, no barriers) option on a single asset)	601
plainoption.hpp (Plain (no dividends, no barriers) option on a single asset)	602
processingiterator.hpp (Iterator mapping a unary function to an underlying sequence)	603
qldefines.hpp (Global definitions and compiler switches)	604
randomarraygenerator.hpp (Generates random arrays from a random number generator)	606
ratehelpers.cpp (Rate helpers base class)	607
ratehelpers.hpp (Rate helpers base class)	608
relinkablehandle.hpp (Globally accessible relinkable pointer)	609
ridder.cpp (Ridder 1-D solver)	610
ridder.hpp (Ridder 1-D solver)	611
riskmeasures.hpp (Risk functions)	612
riskstatistics.hpp (Normal distribution risk analysis tool: VAR, (average) shortfall)	613
rngtypedefs.hpp (Default choices for template instantiations)	614
sample.hpp (Weighted sample)	615
scheduler.cpp (Date scheduler)	616
scheduler.hpp (Date scheduler)	617
secant.cpp (Secant 1-D solver)	618
secant.hpp (Secant 1-D solver)	619
segmentintegral.cpp (Integral of a function over a segment)	620
segmentintegral.hpp (Integral of a one-dimensional function)	621
shortfloatingcoupon.cpp (Short coupon at par on a term structure)	622
shortfloatingcoupon.hpp (Short (or long) coupon at par on a term structure)	623
shortrateprocess.hpp (Short rate process)	624
shoutcondition.hpp (Shout option exercise condition)	625

simplecashflow.hpp (Predetermined cash flow)	626
simpleswap.cpp (Simple fixed-rate vs Libor swap)	627
simpleswap.hpp (Simple fixed-rate vs Libor swap)	628
simplex.cpp (Simplex optimization method)	629
simplex.hpp (Simplex optimization method)	630
singleassetoption.cpp (Common code for option evaluation)	631
singleassetoption.hpp (Common code for option evaluation)	632
solver1d.cpp (Abstract 1-D solver class)	633
solver1d.hpp (Abstract 1-D solver class)	634
statistics.cpp (Statistic tools)	635
statistics.hpp (Statistic tools)	636
steepestdescent.cpp (Steepest descent optimization method)	637
steepestdescent.hpp (Steepest descent optimization method)	638
stepcondition.hpp (Conditions to be applied at every time step)	639
steppingiterator.hpp (Iterator advancing in constant steps)	640
stock.cpp (Concrete stock class)	641
stock.hpp (Concrete stock class)	642
swap.cpp (Interest rate swap)	643
swap.hpp (Interest rate swap)	644
swaption.cpp (Swaption)	645
swaption.hpp (Swaption class)	646
swaptionhelper.cpp (Swaption calibration helper)	647
swaptionhelper.hpp (Swaption calibration helper)	648
swaptionvolmatrix.hpp (Swaption at-the-money volatility matrix)	649
swaptionvolstructure.hpp (Swaption volatility structure)	650
sydney.cpp (Sydney calendar)	651
sydney.hpp (Sydney calendar)	652
symmetriceigenvalues.hpp (Eigenvalues / eigenvectors of a real symmetric matrix)	653
symmetricschurdecomposition.cpp (Eigenvalues / eigenvectors of a real symmetric matrix)	654
symmetricschurdecomposition.hpp (Eigenvalues / eigenvectors of a real symmetric matrix)	655
target.cpp (TARGET calendar)	656
target.hpp (TARGET calendar)	657
termstructure.hpp (Term structure)	658
thirty360.cpp (30/360 day counters)	659
thirty360.hpp (30/360 day counters)	660
tokyo.cpp (Tokyo calendar)	661
tokyo.hpp (Tokyo calendar)	662
toronto.cpp (Toronto calendar)	663
toronto.hpp (Toronto calendar)	664
tree.cpp (Tree class)	665
tree.hpp (Tree class)	666
treecapfloor.cpp (Cap/Floor calculated using a tree)	667
treecapfloor.hpp (Cap/Floor calculated using a tree)	668
treeswaption.cpp (European swaption calculated using finite differences)	669
treeswaption.hpp (Swaption calculated using a tree)	670
tridiagonaloperator.cpp (Tridiagonal operator)	671
tridiagonaloperator.hpp (Tridiagonal operator)	672
trinomialtree.cpp (Trinomial tree class)	673
trinomialtree.hpp (Trinomial tree class)	674
twofactormodel.hpp (Abstract two-factor interest rate model class)	675
types.hpp (Custom types)	676
usdlibor.hpp (USD Libor index)	677
valueatcenter.cpp (Compute value, first, and second derivatives at grid center)	678
valueatcenter.hpp (Compute value, first, and second derivatives at grid center)	679

wellington.cpp (Wellington calendar)	680
wellington.hpp (Wellington calendar)	681
xibor.cpp (Purely virtual base class for libor indexes)	682
xibor.hpp (Purely virtual base class for libor indexes)	683
xibormanager.cpp (Global repository for Xibor histories)	684
xibormanager.hpp (Global repository for Xibor histories)	685
zarlibor.hpp (ZAR Libor index (also known as JIBAR, check settlement days))	686
zurich.cpp (Zurich calendar)	687
zurich.hpp (Zurich calendar)	688

8.2 QuantLib Related Pages

Here is a list of all related documentation pages:

The QuantLib Group	14
Core classes	17
Currencies and FX rates	32
Date and time calculations	18
The finite differences framework	19
The interest rate modelling framework	30
Version history	9
Introduction	3
Installation	6
Instruments and pricers	33
Copyright and License	15
Math tools	34
The Monte Carlo framework	24
Project overview	4
Design patterns	35
Supported platforms	8
Additional resources	13
Term structures	36
Usage	7
Utilities	37
Where to get QuantLib	5
Todo List	12

Chapter 9

QuantLib Module Documentation

9.1 Deprecated classes

The use of classes and functions in the above list is deprecated. They will be removed in future releases.

9.2 Global QuantLib macros

Modules

- [Math functions](#)
- [Numeric limits](#)
- [Time functions](#)
- [Character functions](#)
- [Input/output functions](#)
- [Min and max functions](#)
- [Template capabilities](#)
- [Iterator support](#)

Defines

- `#define QL_HEX_VERSION 0x000300b1`
version hexadecimal number.
- `#define QL_VERSION "0.3.0b1-20020322"`
version string.
- `#define QL_TRACE_LEVEL 0`
global trace level (may be superseded locally by a greater value).
- `#define QL_DUMMY_RETURN(x)`
Is a dummy return statement required?

9.2.1 Detailed Description

Global definitions and quite a few macros which help porting the code to different compilers

9.2.2 Define Documentation

9.2.2.1 `#define QL_DUMMY_RETURN(x)`

Some compilers will issue a warning if it is missing even though it could never be reached during execution, e.g., after a block like

```
if (condition)
    return validResult;
else
    throw HideousError();
```

On the other hand, other compilers will issue a warning if it is present because it cannot be reached. For the code to be portable this macro should be used after the block.

9.3 Math functions

Defines

- `#define QL_SQRT std::sqrt`
square root.
- `#define QL_FABS std::fabs`
absolute value.
- `#define QL_EXP std::exp`
exponential.
- `#define QL_LOG std::log`
logarithm.
- `#define QL_SIN std::sin`
sine.
- `#define QL_COS std::cos`
cosine.
- `#define QL_POW std::pow`
power.
- `#define QL_MODF std::modf`
floating-point module.

9.3.1 Detailed Description

Some compilers still define math functions them in the global namespace. For the code to be portable these macros should be used instead of the actual functions.

9.4 Numeric limits

Defines

- `#define QL_MIN_INT ((std::numeric_limits<int>::min)())`
- `#define QL_MAX_INT ((std::numeric_limits<int>::max)())`
- `#define QL_MIN_DOUBLE -((std::numeric_limits<double>::max)())`
- `#define QL_MAX_DOUBLE ((std::numeric_limits<double>::max)())`
- `#define QL_EPSILON ((std::numeric_limits<double>::epsilon)())`
- `#define QL_MIN_POSITIVE_DOUBLE ((std::numeric_limits<double>::min)())`

9.4.1 Detailed Description

Some compilers do not give an implementation of `<limits>` yet. For the code to be portable these macros should be used instead of the corresponding method of `std::numeric_limits` or the corresponding macro defined in `<limits.h>`.

9.4.2 Define Documentation

9.4.2.1 `#define QL_MIN_INT ((std::numeric_limits<int>::min)())`

Defines the value of the maximum representable negative integer value

9.4.2.2 `#define QL_MAX_INT ((std::numeric_limits<int>::max)())`

Defines the value of the maximum representable integer value

9.4.2.3 `#define QL_MIN_DOUBLE -((std::numeric_limits<double>::max)())`

Defines the value of the maximum representable negative double value

9.4.2.4 `#define QL_MAX_DOUBLE ((std::numeric_limits<double>::max)())`

Defines the value of the maximum representable double value

9.4.2.5 `#define QL_EPSILON ((std::numeric_limits<double>::epsilon)())`

Defines the machine precision for operations over doubles

9.5 Time functions

Defines

- #define `QL_CLOCK` `std::clock`
clock value.
- #define `QL_TIME` `std::time`
time value.

9.5.1 Detailed Description

Some compilers still define time functions in the global namespace. For the code to be portable these macros should be used instead of the actual functions.

9.6 Character functions

Defines

- #define [QL_STRLEN](#) std::strlen
string length.
- #define [QL_TOUPPER](#) std::toupper
convert to uppercase.
- #define [QL_TOLOWER](#) std::tolower
convert to lowercase.

9.6.1 Detailed Description

Some compilers still define character functions in the global namespace. For the code to be portable these macros should be used instead of the actual functions.

9.7 Input/output functions

Defines

- #define `QL_SPRINTF` `std::sprintf`
print to string.

9.7.1 Detailed Description

Some compilers still define i/o functions in the global namespace. For the code to be portable these macros should be used instead of the actual functions.

9.8 Min and max functions

Defines

- `#define QL_MIN std::min`
minimum between two elements.
- `#define QL_MAX std::max`
maximum between two elements.

9.8.1 Detailed Description

Some compilers still do not define `std::min` and `std::max`. Moreover, Visual C++ defines them but for unfathomable reasons garble their names. For the code to be portable these macros should be used instead of the actual functions.

9.9 Template capabilities

Defines

- `#define QL_DECLARE_TEMPLATE_SPECIALIZATIONS`
Blame Microsoft for this one...
- `#define QL_ALLOW_TEMPLATE_METHOD_CALLS 1`
Blame Microsoft for this one...
- `#define QL_EXPRESSION_TEMPLATES_WORK 1`
- `#define QL_TEMPLATE_METAPROGRAMMING_WORKS 1`

9.9.1 Detailed Description

Some compilers still do not fully implement the template syntax. These macros can be used to select between alternate implementations of blocks of code, namely, one that takes advantage of template programming techniques and a less efficient one which is compatible with all compilers.

9.9.2 Define Documentation

9.9.2.1 `#define QL_DECLARE_TEMPLATE_SPECIALIZATIONS`

They decided that a declaration and a definition of a specialized template function amount to a redefinition and should issue a linker error. For the code to be portable, template specializations should be declared (as opposed to defined) only if this macro is defined.

9.9.2.2 `#define QL_ALLOW_TEMPLATE_METHOD_CALLS 1`

Their compiler cannot cope with method calls such as

```
Handle<Type1> h1(whatever);
h2 = h1.downcast<Type2>();
```

For compatibility, a workaround should be implemented (which of course will be less solid or more complex - as I said, blame Microsoft...)

9.9.2.3 `#define QL_EXPRESSION_TEMPLATES_WORK 1`

Expression templates techniques (see T. L. Veldhuizen, *Expression templates*, C++ Report, 7(5):26-31, June 1995, available at <http://extreme.indiana.edu/~tveldhui/papers>) are sometimes too advanced for the template implementation of current compilers.

9.9.2.4 `#define QL_TEMPLATE_METAPROGRAMMING_WORKS 1`

Template metaprogramming techniques (see T. L. Veldhuizen, *Using C++ Template Metaprograms*, C++ Report, Vol 7 No. 4, May 1995, available at <http://extreme.indiana.edu/~tveldhui/papers>) are sometimes too advanced for the template implementation of current compilers.

9.10 Iterator support

Defines

- `#define QL_ITERATOR std::iterator`
- `#define QL_ITERATOR_TRAITS std::iterator_traits`
- `#define QL_SPECIALIZE_ITERATOR_TRAITS(T)`
- `#define QL_REVERSE_ITERATOR(iterator, type) std::reverse_iterator< iterator >`
Blame Microsoft for this one...
- `#define QL_FULL_ITERATOR_SUPPORT`

9.10.1 Detailed Description

Some compilers still define the iterator struct outside the std namespace, only partially implement it, or do not implement it at all. For the code to be portable these macros should be used instead of the actual functions.

9.10.2 Define Documentation

9.10.2.1 `#define QL_ITERATOR std::iterator`

Custom iterators should be derived from this struct for the code to be portable.

9.10.2.2 `#define QL_ITERATOR_TRAITS std::iterator_traits`

For the code to be portable this macro should be used instead of the actual struct.

9.10.2.3 `#define QL_SPECIALIZE_ITERATOR_TRAITS(T)`

When using the QuantLib implementation of `iterator_traits`, this macro might be needed to specialize `QL_ITERATOR_TRAITS` for a pointer to a user-defined type.

9.10.2.4 `#define QL_REVERSE_ITERATOR(iterator, type) std::reverse_iterator< iterator >`

They decided that `std::reverse_iterator<iterator>` needed an extra template argument. For the code to be portable this macro should be used instead of the actual class.

Chapter 10

QuantLib Namespace Documentation

10.1 QuantLib Namespace Reference

a.k.a. the QuantLib Foundation.

Compounds

- class [QuantLib::Arguments](#)
base class for generic argument groups.
 - class [QuantLib::Results](#)
base class for generic result groups.
 - class [QuantLib::Array](#)
1-D array used in linear algebra.
 - class [QuantLib::Calendar](#)
calendar class.
 - class [QuantLib::Calendar::CalendarImpl](#)
abstract base class for calendar implementations.
 - class [QuantLib::Calendar::WesternCalendarImpl](#)
 - class [QuantLib::CapFlatVolatilityStructure](#)
Cap/floor flat volatility structure.
 - class [QuantLib::CapletForwardVolatilityStructure](#)
Caplet/floorlet forward volatility structure.
 - class [QuantLib::CashFlow](#)
Base class for cash flows.
 - class [QuantLib::IntegerFormatter](#)
Formats integers for output.
-

- class [QuantLib::DoubleFormatter](#)
Formats doubles for output.
- class [QuantLib::EuroFormatter](#)
Formats amounts in Euro for output.
- class [QuantLib::RateFormatter](#)
Formats rates for output.
- class [QuantLib::DateFormatter](#)
Formats dates for output.
- class [QuantLib::CurrencyFormatter](#)
Formats currencies for output.
- class [QuantLib::StringFormatter](#)
Formats strings as lower- or uppercase.
- class [QuantLib::Period](#)
Time period described by a number of a given time unit.
- class [QuantLib::Date](#)
Concrete date class.
- class [QuantLib::DayCounter](#)
day counter class.
- class [QuantLib::DayCounter::DayCounterImpl](#)
abstract base class for day counter implementations.
- class [QuantLib::DiffusionProcess](#)
Diffusion process class.
- class [QuantLib::BlackScholesProcess](#)
Black-Scholes diffusion process class.
- class [QuantLib::OrnsteinUhlenbeckProcess](#)
Ornstein-Uhlenbeck process class.
- class [QuantLib::Error](#)
Base error class.
- class [QuantLib::AssertionFailedError](#)
Specialized error.
- class [QuantLib::PreconditionNotSatisfiedError](#)
Specialized error.
- class [QuantLib::PostconditionNotSatisfiedError](#)
Specialized error.

- class [QuantLib::IndexError](#)
Specialized error.
- class [QuantLib::IllegalArgumentError](#)
Specialized error.
- class [QuantLib::IllegalResultError](#)
Specialized error.
- class [QuantLib::OutOfMemoryError](#)
Specialized error.
- class [QuantLib::Exercise](#)
Exercise class (American, Bermudan or European).
- class [QuantLib::Grid](#)
spatial grid class.
- class [QuantLib::TimeGrid](#)
time grid class.
- class [QuantLib::Handle](#)
Reference-counted pointer.
- class [QuantLib::History](#)
Container for historical data.
- class [QuantLib::History::Entry](#)
single datum in history.
- class [QuantLib::History::const_iterator](#)
random access iterator on history entries.
- class [QuantLib::Index](#)
purely virtual base class for indexes.
- class [QuantLib::Instrument](#)
Abstract instrument class.
- class [QuantLib::MarketElement](#)
purely virtual base class for market observables.
- class [QuantLib::SimpleMarketElement](#)
market element returning a stored value.
- class [QuantLib::DerivedMarketElement](#)
market element whose value depends on another market element.
- class [QuantLib::CompositeMarketElement](#)

market element whose value depends on two other market element.

- class [QuantLib::Null](#)
template class providing a null value for a given type.
- class [QuantLib::Option](#)
base option class.
- class [QuantLib::OptionValue](#)
option pricing results.
- class [QuantLib::OptionGreeks](#)
option pricing results.
- class [QuantLib::OptionPricingEngine](#)
base class for option pricing engines.
- class [QuantLib::Link](#)
Relinkable access to a [Handle](#).
- class [QuantLib::RelinkableHandle](#)
Globally accessible relinkable pointer.
- class [QuantLib::RiskStatistics](#)
Risk analysis tool.
- class [QuantLib::Scheduler](#)
Date scheduler.
- class [QuantLib::ObjectiveFunction](#)
Objective function for 1-D solvers.
- class [QuantLib::Solver1D](#)
Abstract base class for 1-D solvers.
- class [QuantLib::SwaptionVolatilityStructure](#)
Swaption volatility structure.
- class [QuantLib::TermStructure](#)
Term structure.
- class [QuantLib::ZeroYieldStructure](#)
Zero yield term structure.
- class [QuantLib::DiscountStructure](#)
Discount factor term structure.
- class [QuantLib::ForwardRateStructure](#)
Forward rate term structure.

- class [QuantLib::ImpliedTermStructure](#)
Implied term structure at a given date in the future.
- class [QuantLib::ZeroSpreadedTermStructure](#)
Term structure with an added spread on the zero yield rate.
- class [QuantLib::ForwardSpreadedTermStructure](#)
Term structure with an added spread on the instantaneous forward rate.

Typedefs

- typedef int [Day](#)
Day number.
- typedef int [Year](#)
Year number.
- typedef int [Integer](#)
integer number.
- typedef double [Real](#)
real number.
- typedef QL_SIZE_T [Size](#)
size of a container.
- typedef double [Time](#)
continuous quantity with 1-year units.
- typedef double [DiscountFactor](#)
used to describe discount factors between dates.
- typedef double [Rate](#)
used to describe interest rates.
- typedef double [Spread](#)
used to describe spreads on interest rates.

Enumerations

- enum [RollingConvention](#) { [Preceding](#), [ModifiedPreceding](#), [Following](#), [ModifiedFollowing](#) }
Rolling conventions.
- enum [Currency](#) { [AUD](#), [BGL](#), [CAD](#), [CHF](#), [CYP](#), [CZK](#), [DEM](#), [DKK](#), [EEK](#), [EUR](#), [GBP](#), [GRD](#), [HKD](#), [HUF](#), [ISK](#), [ITL](#), [JPY](#), [KRW](#), [LTL](#), [LVL](#), [MTL](#), [NOK](#), [NZD](#), [PLZ](#), [ROL](#), [SEK](#), [SGD](#), [SIT](#), [SKK](#), [TRL](#), [USD](#), [ZAR](#) }
Known currencies.

- enum [Weekday](#) { **Sunday** = 1, **Monday** = 2, **Tuesday** = 3, **Wednesday** = 4, **Thursday** = 5, **Friday** = 6, **Saturday** = 7 }
- enum [Month](#) { **January** = 1, **February** = 2, **March** = 3, **April** = 4, **May** = 5, **June** = 6, **July** = 7, **August** = 8, **September** = 9, **October** = 10, **November** = 11, **December** = 12 }
Month names.
- enum [TimeUnit](#) { **Days** = 0, **Weeks** = 1, **Months** = 2, **Years** = 3 }
Units used to describe time periods.

Functions

- double **DotProduct** (const [Array](#) &v1, const [Array](#) &v2)
- [Array](#) **operator+** (const [Array](#) &v)
- [Array](#) **operator-** (const [Array](#) &v)
- [Array](#) **operator+** (const [Array](#) &v1, const [Array](#) &v2)
- [Array](#) **operator+** (const [Array](#) &v1, double a)
- [Array](#) **operator+** (double a, const [Array](#) &v2)
- [Array](#) **operator-** (const [Array](#) &v1, const [Array](#) &v2)
- [Array](#) **operator-** (const [Array](#) &v1, double a)
- [Array](#) **operator-** (double a, const [Array](#) &v2)
- [Array](#) **operator** * (const [Array](#) &v1, const [Array](#) &v2)
- [Array](#) **operator** * (const [Array](#) &v1, double a)
- [Array](#) **operator** * (double a, const [Array](#) &v2)
- [Array](#) **operator** / (const [Array](#) &v1, const [Array](#) &v2)
- [Array](#) **operator** / (const [Array](#) &v1, double a)
- [Array](#) **operator** / (double a, const [Array](#) &v2)
- [Array](#) **Abs** (const [Array](#) &v)
- [Array](#) **Sqrt** (const [Array](#) &v)
- [Array](#) **Log** (const [Array](#) &v)
- [Array](#) **Exp** (const [Array](#) &v)
- std::ostream & **operator**<< (std::ostream &stream, const [Array](#) &array)
- bool **operator**== (const [Calendar](#) &h1, const [Calendar](#) &h2)
- bool **operator**!= (const [Calendar](#) &h1, const [Calendar](#) &h2)
- std::ostream & **operator**<< (std::ostream &stream, const [Date](#) &date)
- int **operator-** (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**== (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**!= (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**< (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**<= (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**> (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**>= (const [Date](#) &d1, const [Date](#) &d2)
- bool **operator**== (const [DayCounter](#) &h1, const [DayCounter](#) &h2)
- bool **operator**!= (const [DayCounter](#) &h1, const [DayCounter](#) &h2)

Variables

- const double **growthFactor** = 1.6

10.1.1 Detailed Description

Armijo linesearch.

Let α and β be 2 scalars in $[0,1]$. Let x be the current value of the unknown, d the search direction and t the step. Let f be the function to minimize. The line search stop when t verifies $f(x+t*d) - f(x) \leq -\alpha*t*f'(x+t*d)$ and $f(x+t/\beta*d) - f(x) > -\alpha*t*f'(x+t*d)/\beta$

(see Polak. Algorithms and consistent approximations, Optimization, volume 124 of Applied Mathematical Sciences. Springer-Verlag, N-Y, 1997)

10.1.2 Enumeration Type Documentation

10.1.2.1 enum RollingConvention

These conventions specify the algorithm used to find the business day which is "closest" to a given holiday.

Enumeration values:

Preceding Choose the first business day before the given holiday.

ModifiedPreceding Choose the first business day before the given holiday unless it belongs to a different month, in which case choose the first business day after the holiday.

Following Choose the first business day after the given holiday.

ModifiedFollowing Choose the first business day after the given holiday unless it belongs to a different month, in which case choose the first business day before the holiday.

10.1.2.2 enum Currency

Enumeration values:

AUD Australian Dollar.

BGL Bulgarian Lev.

CAD Canadian Dollar.

CHF Swiss Franc.

CYP Cyprus Pound.

CZK Czech Koruna.

DEM German Mark.

DKK Danish Krone.

EEK Estonian Kroon.

EUR Euro.

GBP British Pound.

GRD Greek Drachma.

HKD Hong Kong Dollar.

HUF Hungarian Forint.

ISK Iceland Krona.

ITL Italian Lira.

JPY Japanese Yen.

KRW South-Korean Won.
LTL Lithuanian Litas.
LVL Latvian Lats.
MTL Maltese Lira.
NOK Norwegian Kroner.
NZD New Zealand Dollar.
PLZ Polish Zloty.
ROL Romanian Leu.
SEK Swedish Krona.
SGD Singapore Dollar.
SIT Slovenian Tolar.
SKK Slovak Koruna.
TRL Turkish Lira.
USD US Dollar.
ZAR South African Rand.

10.1.2.3 enum Weekday

Day's serial number MOD 7; WEEKDAY Excel function is the same except for Sunday = 7

10.2 QuantLib::Calendars Namespace Reference

Specialized [Calendar](#) classes.

Compounds

- class [QuantLib::Calendars::Frankfurt](#)
Frankfurt calendar.
- class [QuantLib::Calendars::Helsinki](#)
Helsinki calendar.
- class [QuantLib::Calendars::Johannesburg](#)
Johannesburg calendar.
- class [QuantLib::Calendars::London](#)
London calendar.
- class [QuantLib::Calendars::Milan](#)
Milan calendar.
- class [QuantLib::Calendars::NewYork](#)
New York calendar.
- class [QuantLib::Calendars::Sydney](#)
Sydney, calendar (New South Wales, Australia).
- class [QuantLib::Calendars::TARGET](#)
TARGET calendar.
- class [QuantLib::Calendars::Tokyo](#)
Tokyo calendar.
- class [QuantLib::Calendars::Toronto](#)
Toronto calendar.
- class [QuantLib::Calendars::Wellington](#)
Wellington calendar.
- class [QuantLib::Calendars::Zurich](#)
Zurich calendar.

10.2.1 Detailed Description

See sect. [Calendars](#)

10.3 QuantLib::CashFlows Namespace Reference

Concrete implementations of the [CashFlow](#) interface.

Compounds

- class [QuantLib::CashFlows::FixedRateCouponVector](#)
helper class building a sequence of fixed rate coupons.
- class [QuantLib::CashFlows::FloatingRateCouponVector](#)
helper class building a sequence of floating rate coupons.
- class [QuantLib::CashFlows::Coupon](#)
coupon accruing over a fixed period.
- class [QuantLib::CashFlows::FixedRateCoupon](#)
coupon paying a fixed interest rate.
- class [QuantLib::CashFlows::FloatingRateCoupon](#)
coupon at par on a term structure.
- class [QuantLib::CashFlows::ShortFloatingRateCoupon](#)
short coupon at par on a term structure.
- class [QuantLib::CashFlows::SimpleCashFlow](#)
Predetermined cash flow.

10.3.1 Detailed Description

See sect. cashflows

10.4 QuantLib::DayCounters Namespace Reference

Specialized [DayCounter](#) classes.

Compounds

- class [QuantLib::DayCounters::Actual360](#)
Actual/360 day count convention.
- class [QuantLib::DayCounters::Actual365](#)
Actual/365 day count convention.
- class [QuantLib::DayCounters::ActualActual](#)
Actual/Actual day count.
- class [QuantLib::DayCounters::Thirty360](#)
30/360 day count convention.

10.4.1 Detailed Description

See sect. [daycounters](#)

10.5 QuantLib::FiniteDifferences Namespace Reference

Finite difference framework.

Compounds

- class [QuantLib::FiniteDifferences::BoundaryCondition](#)
Boundary condition for finite difference problems.
- class [QuantLib::FiniteDifferences::BSMOperator](#)
Black-Scholes-Merton differential operator.
- class [QuantLib::FiniteDifferences::CrankNicolson](#)
Crank-Nicolson scheme for finite difference methods.
- class [QuantLib::FiniteDifferences::DMinus](#)
 D_- matricial representation.
- class [QuantLib::FiniteDifferences::DPlus](#)
 D_+ matricial representation.
- class [QuantLib::FiniteDifferences::DPlusDMinus](#)
 D_+D_- matricial representation.
- class [QuantLib::FiniteDifferences::DZero](#)
 D_0 matricial representation.
- class [QuantLib::FiniteDifferences::ExplicitEuler](#)
Forward Euler scheme for finite difference methods.
- class [QuantLib::FiniteDifferences::FiniteDifferenceModel](#)
Generic finite difference model.
- class [QuantLib::FiniteDifferences::ImplicitEuler](#)
Backward Euler scheme for finite difference methods.
- class [QuantLib::FiniteDifferences::MixedScheme](#)
Mixed (explicit/implicit) scheme for finite difference methods.
- class [QuantLib::FiniteDifferences::OneFactorOperator](#)
Interest-rate single factor model differential operator.
- class [QuantLib::FiniteDifferences::StepCondition](#)
condition to be applied at every time step.
- class [QuantLib::FiniteDifferences::TridiagonalOperator](#)
Base implementation for tridiagonal operator.
- class [QuantLib::FiniteDifferences::TridiagonalOperator::TimeSetter](#)
encapsulation of time-setting logic.

Typedefs

- typedef [FiniteDifferenceModel](#)< [CrankNicolson](#)< [TridiagonalOperator](#) > > [StandardFiniteDifferenceModel](#)
default choice for finite-difference model.
- typedef [StepCondition](#)< [Array](#) > [StandardStepCondition](#)
default choice for step condition.

Functions

- [TridiagonalOperator](#) **operator+** (const [TridiagonalOperator](#) &D)
- [TridiagonalOperator](#) **operator-** (const [TridiagonalOperator](#) &D)
- [TridiagonalOperator](#) **operator+** (const [TridiagonalOperator](#) &D1, const [TridiagonalOperator](#) &D2)
- [TridiagonalOperator](#) **operator-** (const [TridiagonalOperator](#) &D1, const [TridiagonalOperator](#) &D2)
- [TridiagonalOperator](#) **operator** * (double a, const [TridiagonalOperator](#) &D)
- [TridiagonalOperator](#) **operator** * (const [TridiagonalOperator](#) &D, double a)
- [TridiagonalOperator](#) **operator** / (const [TridiagonalOperator](#) &D, double a)
- double [valueAtCenter](#) (const [Array](#) &a)
mid-point value.
- double [firstDerivativeAtCenter](#) (const [Array](#) &a, const [Array](#) &g)
mid-point first derivative.
- double [secondDerivativeAtCenter](#) (const [Array](#) &a, const [Array](#) &g)
mid-point second derivative.

10.5.1 Detailed Description

See sect. [The finite differences framework](#)

10.6 QuantLib::Indexes Namespace Reference

Concrete implementations of the [Index](#) interface.

Compounds

- class [QuantLib::Indexes::AUDLibor](#)
AUD Libor index (Also known as TIBOR, check settlement days).
- class [QuantLib::Indexes::CADLibor](#)
CAD Libor index (Also known as CDOR, check settlement days).
- class [QuantLib::Indexes::CHFLibor](#)
CHF Libor index (Also known as ZIBOR, check settlement days and day-count).
- class [QuantLib::Indexes::Euribor](#)
Euribor index.
- class [QuantLib::Indexes::GBPLibor](#)
GBP Libor index.
- class [QuantLib::Indexes::JPYLibor](#)
JPY Libor index (Also known as TIBOR, check settlement days).
- class [QuantLib::Indexes::USDLibor](#)
USD Libor index.
- class [QuantLib::Indexes::Xibor](#)
base class for libor indexes.
- class [QuantLib::Indexes::XiborManager](#)
global repository for libor histories.
- class [QuantLib::Indexes::ZARLibor](#)
ZAR Libor index (also known as JIBAR, check settlement days).

10.6.1 Detailed Description

See sect. indexes

10.7 QuantLib::Instruments Namespace Reference

Concrete implementations of the [Instrument](#) interface.

Compounds

- class [QuantLib::Instruments::CapFloorParameters](#)
parameters for cap/floor calculation.
- class [QuantLib::Instruments::CapFloorResults](#)
results from cap/floor calculation.
- class [QuantLib::Instruments::PlainOption](#)
Plain (no dividends, no barriers) option on a single asset.
- class [QuantLib::Instruments::PlainOptionParameters](#)
parameters for plain option calculation.
- class [QuantLib::Instruments::PlainOptionResults](#)
results from plain option calculation.
- class [QuantLib::Instruments::SimpleSwap](#)
Simple fixed-rate vs Libor swap.
- class [QuantLib::Instruments::Stock](#)
Simple stock class.
- class [QuantLib::Instruments::Swap](#)
Interest rate swap.
- class [QuantLib::Instruments::Swaption](#)
Swaption class.
- class [QuantLib::Instruments::SwaptionParameters](#)
parameters for swaption calculation.
- class [QuantLib::Instruments::SwaptionResults](#)
results from swaption calculation.

10.7.1 Detailed Description

See sect. [Instruments and pricers](#)

10.8 QuantLib::Math Namespace Reference

Mathematical functions and classes.

Compounds

- class [QuantLib::Math::BilinearInterpolation](#)
bilinear interpolation between discrete points.
- class [QuantLib::Math::CubicSpline](#)
cubic spline interpolation between discrete points.
- class [QuantLib::Math::Interpolation](#)
abstract base class for 1-D interpolations.
- class [QuantLib::Math::Interpolation2D](#)
abstract base class for 2-D interpolations.
- class [QuantLib::Math::LexicographicalView](#)
Lexicographical 2-D view of a contiguous set of data.
- class [QuantLib::Math::LinearInterpolation](#)
linear interpolation between discrete points.
- class [QuantLib::Math::Matrix](#)
matrix used in linear algebra.
- class [QuantLib::Math::MultivariateAccumulator](#)
A sample accumulator for multivariate analysis.
- class [QuantLib::Math::RiskMeasures](#)
Interface for risk functions.
- class [QuantLib::Math::SegmentIntegral](#)
Integral of a one-dimensional function.
- class [QuantLib::Math::Statistics](#)
Statistic tool.
- class [QuantLib::Math::SymmetricSchurDecomposition](#)
symmetric threshold Jacobi algorithm.

Typedefs

- typedef NormalDistribution **GaussianDistribution**

Functions

- [Matrix](#) **matrixSqrt** (const [Matrix](#) &realSymmMatrix)
- [Matrix](#) **operator+** (const [Matrix](#) &m1, const [Matrix](#) &m2)
- [Matrix](#) **operator-** (const [Matrix](#) &m1, const [Matrix](#) &m2)
- [Matrix](#) **operator *** (const [Matrix](#) &m, double x)
- [Matrix](#) **operator *** (double x, const [Matrix](#) &m)
- [Matrix](#) **operator/** (const [Matrix](#) &m, double x)
- [Array](#) **operator *** (const [Array](#) &v, const [Matrix](#) &m)
- [Array](#) **operator *** (const [Matrix](#) &m, const [Array](#) &v)
- [Matrix](#) **operator *** (const [Matrix](#) &m1, const [Matrix](#) &m2)
- [Matrix](#) **transpose** (const [Matrix](#) &m)
- [Matrix](#) **outerProduct** (const [Array](#) &v1, const [Array](#) &v2)
- [Array](#) **SymmetricEigenvalues** ([Matrix](#) &s)
- [Matrix](#) **SymmetricEigenvectors** ([Matrix](#) &s)

10.8.1 Detailed Description

See sect. [Math tools](#)

10.9 QuantLib::MonteCarlo Namespace Reference

Monte Carlo framework.

Compounds

- class [QuantLib::MonteCarlo::ArithmeticAOPPathPricer](#)
path pricer for arithmetic average price option.
- class [QuantLib::MonteCarlo::ArithmeticASOPathPricer](#)
path pricer for average strike Asian options.
- class [QuantLib::MonteCarlo::BasketPathPricer](#)
multipath pricer for European-type basket option.
- class [QuantLib::MonteCarlo::EuropeanPathPricer](#)
path pricer for European options.
- class [QuantLib::MonteCarlo::EverestPathPricer](#)
path pricer for European-type Everest option.
- class [QuantLib::MonteCarlo::GeometricAOPPathPricer](#)
path pricer for geometric average price option.
- class [QuantLib::MonteCarlo::GeometricASOPathPricer](#)
path pricer for geometric average price option.
- class [QuantLib::MonteCarlo::HimalayaPathPricer](#)
multipath pricer for European-type Himalaya option.
- class [QuantLib::MonteCarlo::MaxBasketPathPricer](#)
multipath pricer for European-type basket option.
- class [QuantLib::MonteCarlo::MonteCarloModel](#)
General purpose Monte Carlo model for path samples.
- class [QuantLib::MonteCarlo::MultiPath](#)
single random walk.
- class [QuantLib::MonteCarlo::MultiPathGenerator](#)
Generates a multipath from a random number generator.
- class [QuantLib::MonteCarlo::PagodaPathPricer](#)
multipath pricer for pagoda options.
- class [QuantLib::MonteCarlo::Path](#)
single factor random walk.
- class [QuantLib::MonteCarlo::PathGenerator](#)

Generates random paths from a random number generator.

- class [QuantLib::MonteCarlo::PathPricer](#)
base class for path pricers.
- struct [QuantLib::MonteCarlo::Sample](#)
weighted sample.

Typedefs

- typedef [PathGenerator](#)< [RandomNumbers::GaussianRandomGenerator](#) > [GaussianPathGenerator](#)
default choice for Gaussian path generator.
- typedef [MultiPathGenerator](#)< [RandomNumbers::RandomArrayGenerator](#)< [RandomNumbers::GaussianRandomGenerator](#) > > [GaussianMultiPathGenerator](#)
default choice for Gaussian multi-path generator.
- typedef [MonteCarloModel](#)< [Math::Statistics](#), [GaussianPathGenerator](#), [PathPricer](#)< [Path](#) > > [OneFactorMonteCarloOption](#)
default choice for one-factor Monte Carlo model.
- typedef [MonteCarloModel](#)< [Math::Statistics](#), [GaussianMultiPathGenerator](#), [PathPricer](#)< [MultiPath](#) > > [MultiFactorMonteCarloOption](#)
default choice for multi-factor Monte Carlo model.

Functions

- Matrix [getCovariance](#) (const [Array](#) &volatilities, const Matrix &correlations)
- [Math::Matrix](#) [getCovariance](#) (const [Array](#) &volatilities, const [Math::Matrix](#) &correlations)

10.9.1 Detailed Description

See sect. [The Monte Carlo framework](#)

10.9.2 Function Documentation

10.9.2.1 [Math::Matrix](#) [getCovariance](#) (const [Array](#) & *volatilities*, const [Math::Matrix](#) & *correlations*)

Combines the correlation matrix and the vector of volatilities to return the covariance matrix. Note that only the symmetric part of the correlation matrix is used. Also it is assumed that the diagonal member of the correlation matrix equals one.

10.10 QuantLib::Patterns Namespace Reference

Implementations of design patterns.

Compounds

- class [QuantLib::Patterns::Observable](#)
Object that notifies its changes to a set of observables.
- class [QuantLib::Patterns::Observer](#)
Object that gets notified when a given observable changes.

10.10.1 Detailed Description

See sect. [Design patterns](#)

10.11 QuantLib::Pricers Namespace Reference

Pricing models for options.

Compounds

- class [QuantLib::Pricers::CapFloorPricingEngine](#)
base class for cap/floor pricing engines.
- class [QuantLib::Pricers::PlainOptionEngine](#)
base class for plain option pricing engines.
- class [QuantLib::Pricers::SwaptionPricingEngine](#)
base class for swaption pricing engines.
- class [QuantLib::Pricers::AnalyticalCapFloor](#)
Analytical pricer for cap/floor.
- class [QuantLib::Pricers::BarrierOption](#)
Barrier option.
- class [QuantLib::Pricers::BinaryOption](#)
Binary (digital) option.
- class [QuantLib::Pricers::BlackCapFloor](#)
CapFloor priced by the Black formula.
- class [QuantLib::Pricers::BlackSwaption](#)
Swaption priced by the Black formula.
- class [QuantLib::Pricers::CliquetOption](#)
cliquet (Ratchet) option.
- class [QuantLib::Pricers::ContinuousGeometricAPO](#)
Continuous Geometric Average Price Option (European exercise).
- class [QuantLib::Pricers::DiscreteGeometricAPO](#)
Discrete Geometric Average Price Asian Option (European style).
- class [QuantLib::Pricers::DiscreteGeometricASO](#)
Discrete Geometric Average Strike Asian Option (European style).
- class [QuantLib::Pricers::EuropeanEngine](#)
analytic pricing engine for European options.
- class [QuantLib::Pricers::EuropeanOption](#)
Black-Scholes-Merton European option.
- class [QuantLib::Pricers::FdAmericanOption](#)

American option.

- class [QuantLib::Pricers::FdBermudanOption](#)
Bermudan option.
- class [QuantLib::Pricers::FdBsmOption](#)
Black-Scholes-Merton option priced numerically.
- class [QuantLib::Pricers::FdDividendEuropeanOption](#)
European option with dividends.
- class [QuantLib::Pricers::FdDividendShoutOption](#)
Shout option with dividends.
- class [QuantLib::Pricers::FdEuropean](#)
Example of European option calculated using finite differences.
- class [QuantLib::Pricers::FdStepConditionOption](#)
option executing additional code at each time step.
- class [QuantLib::Pricers::JamshidianSwaption](#)
Jamshidian swaption pricer.
- class [QuantLib::Pricers::McBasket](#)
simple example of multi-factor Monte Carlo pricer.
- class [QuantLib::Pricers::McDiscreteArithmeticAPO](#)
example of Monte Carlo pricer using a control variate.
- class [QuantLib::Pricers::McDiscreteArithmeticASO](#)
example of Monte Carlo pricer using a control variate.
- class [QuantLib::Pricers::McEuropean](#)
simple example of Monte Carlo pricer.
- class [QuantLib::Pricers::McEverest](#)
Everest-type option pricer.
- class [QuantLib::Pricers::McHimalaya](#)
Himalayan-type option pricer.
- class [QuantLib::Pricers::McMaxBasket](#)
simple example of multi-factor Monte Carlo pricer.
- class [QuantLib::Pricers::McPagoda](#)
roofed Asian option.
- class [QuantLib::Pricers::McPricer](#)
base class for Monte Carlo pricers.

- class [QuantLib::Pricers::SingleAssetOption](#)
Black-Scholes-Merton option.
- class [QuantLib::Pricers::TreeCapFloor](#)
Cap/Floor priced in a tree.
- class [QuantLib::Pricers::TreeSwaption](#)
Swaption priced in a tree.

Functions

- double **ExercisePayoff** (Option::Type type, double price, double strike)

10.11.1 Detailed Description

See sect. pricers

10.12 QuantLib::RandomNumbers Namespace Reference

Random Number Generators and Low Discrepancy Sequences.

Compounds

- class [QuantLib::RandomNumbers::BoxMullerGaussianRng](#)
Gaussian random number generator.
- class [QuantLib::RandomNumbers::CLGaussianRng](#)
Gaussian random number generator.
- class [QuantLib::RandomNumbers::ICGaussianRng](#)
Inverse cumulative Gaussian random number generator.
- class [QuantLib::RandomNumbers::KnuthUniformRng](#)
Uniform random number generator.
- class [QuantLib::RandomNumbers::LecuyerUniformRng](#)
Uniform random number generator.
- class [QuantLib::RandomNumbers::RandomArrayGenerator](#)
Generates random arrays from a random number generator.

Typedefs

- typedef [LecuyerUniformRng](#) [UniformRandomGenerator](#)
default choice for uniform random number generator.
- typedef [BoxMullerGaussianRng](#)< [UniformRandomGenerator](#) > [GaussianRandomGenerator](#)
default choice for Gaussian random number generator.
- typedef [RandomArrayGenerator](#)< [GaussianRandomGenerator](#) > [GaussianArrayGenerator](#)
default choice for Gaussian array generator.

10.12.1 Detailed Description

See sect. [Random number and low discrepancy sequence generators](#)

10.13 QuantLib::Solvers1D Namespace Reference

Concrete implementations of the [Solver1D](#) interface.

Compounds

- class [QuantLib::Solvers1D::Bisection](#)
bisection 1-D solver.
- class [QuantLib::Solvers1D::Brent](#)
Brent 1-D solver.
- class [QuantLib::Solvers1D::FalsePosition](#)
False position 1-D solver.
- class [QuantLib::Solvers1D::Newton](#)
Newton 1-D solver.
- class [QuantLib::Solvers1D::NewtonSafe](#)
safe Newton 1-D solver.
- class [QuantLib::Solvers1D::Ridder](#)
Ridder 1-D solver.
- class [QuantLib::Solvers1D::Secant](#)
secant 1-D solver.

10.13.1 Detailed Description

See sect. [One-dimensional solvers](#)

10.14 QuantLib::TermStructures Namespace Reference

Concrete implementations of the [TermStructure](#) interface.

Compounds

- class [QuantLib::TermStructures::PiecewiseFlatForward](#)
Piecewise flat forward term structure.
- class [QuantLib::TermStructures::RateHelper](#)
base class for rate helpers.
- class [QuantLib::TermStructures::DepositRateHelper](#)
Deposit rate.
- class [QuantLib::TermStructures::FraRateHelper](#)
Forward rate agreement.
- class [QuantLib::TermStructures::FuturesRateHelper](#)
Interest Rate Futures.
- class [QuantLib::TermStructures::SwapRateHelper](#)
swap rate.

10.14.1 Detailed Description

See sect. [Term structures](#)

10.15 QuantLib::Utilities Namespace Reference

Classes and functions of general utility.

Compounds

- class [QuantLib::Utilities::combining_iterator](#)
Iterator mapping a function to a set of underlying sequences.
- class [QuantLib::Utilities::coupling_iterator](#)
Iterator mapping a function to a pair of underlying sequences.
- class [QuantLib::Utilities::filtering_iterator](#)
Iterator filtering undesired data.
- struct [QuantLib::Utilities::lowest_category_iterator](#)
most generic of two given iterator categories.
- class [QuantLib::Utilities::processing_iterator](#)
Iterator mapping a unary function to an underlying sequence.
- class [QuantLib::Utilities::stepping_iterator](#)
Iterator advancing in constant steps.

10.15.1 Detailed Description

See sect. [Utilities](#)

Chapter 11

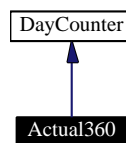
QuantLib Class Documentation

11.1 QuantLib::DayCounters::Actual360 Class Reference

Actual/360 day count convention.

```
#include <actual360.hpp>
```

Inheritance diagram for QuantLib::DayCounters::Actual360:



Public Methods

- `Actual360()`

The documentation for this class was generated from the following file:

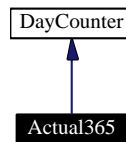
- [actual360.hpp](#)

11.2 QuantLib::DayCounters::Actual365 Class Reference

Actual/365 day count convention.

```
#include <actual365.hpp>
```

Inheritance diagram for QuantLib::DayCounters::Actual365:



Public Methods

- **Actual365** ()

The documentation for this class was generated from the following file:

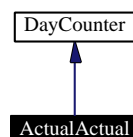
- [actual365.hpp](#)

11.3 QuantLib::DayCounters::ActualActual Class Reference

Actual/Actual day count.

```
#include <actualactual.hpp>
```

Inheritance diagram for QuantLib::DayCounters::ActualActual:



Public Types

- enum **Convention** { **ISMA**, **Bond**, **ISDA**, **Historical**, **AFB**, **Euro** }

Public Methods

- **ActualActual** (Convention c=ActualActual::ISMA)

11.3.1 Detailed Description

The day count can be calculated according to ISMA and US Treasury convention, also known as "Actual/Actual (Bond)"; to ISDA, also known as "Actual/Actual (Historical)"; or to AFB, also known as "Actual/Actual (Euro)".

For more details, refer to http://www.isda.org/c_and_a/pdf/mktc1198.pdf

The documentation for this class was generated from the following files:

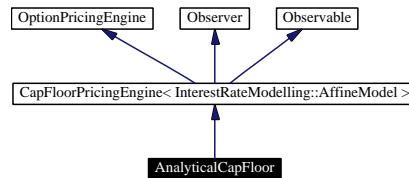
- [actualactual.hpp](#)
- [actualactual.cpp](#)

11.4 QuantLib::Pricers::AnalyticalCapFloor Class Reference

Analytical pricer for cap/floor.

```
#include <analyticalcapfloor.hpp>
```

Inheritance diagram for QuantLib::Pricers::AnalyticalCapFloor:



Public Methods

- **AnalyticalCapFloor** (const [Handle](#)< `InterestRateModelling::AffineModel` > &model)
- void **calculate** () const

The documentation for this class was generated from the following files:

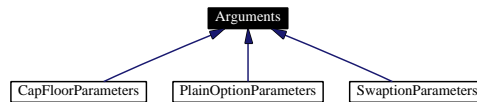
- [analyticalcapfloor.hpp](#)
- [analyticalcapfloor.cpp](#)

11.5 QuantLib::Arguments Class Reference

base class for generic argument groups.

```
#include <argsandresults.hpp>
```

Inheritance diagram for QuantLib::Arguments:



Public Methods

- virtual `~Arguments()`

The documentation for this class was generated from the following file:

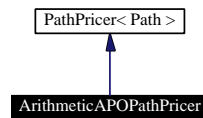
- [argsandresults.hpp](#)

11.6 QuantLib::MonteCarlo::ArithmeticAOPathPricer Class Reference

path pricer for arithmetic average price option.

```
#include <arithmeticapopathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::ArithmeticAOPathPricer:



Public Methods

- **ArithmeticAOPathPricer** (Option::Type type, double underlying, double strike, [DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [Path](#) &path) const

The documentation for this class was generated from the following files:

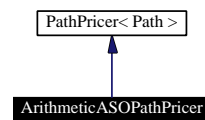
- [arithmeticapopathpricer.hpp](#)
- [arithmeticapopathpricer.cpp](#)

11.7 QuantLib::MonteCarlo::ArithmeticASOPathPricer Class Reference

path pricer for average strike Asian options.

```
#include <arithmeticsopathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::ArithmeticASOPathPricer:



Public Methods

- **ArithmeticASOPathPricer** (Option::Type type, double underlying, [DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [Path](#) &path) const

The documentation for this class was generated from the following files:

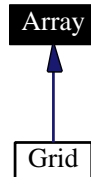
- [arithmeticsopathpricer.hpp](#)
- [arithmeticsopathpricer.cpp](#)

11.8 QuantLib::Array Class Reference

1-D array used in linear algebra.

```
#include <array.hpp>
```

Inheritance diagram for QuantLib::Array:



Public Types

- typedef double * **iterator**
- typedef const double * **const_iterator**

Public Methods

- typedef **QL_REVERSE_ITERATOR** (iterator, double) **reverse_iterator**
- typedef **QL_REVERSE_ITERATOR** (const_iterator, double) **const_reverse_iterator**

Constructors, destructor, and assignment

- **Array** (**Size** size=0)
creates the array with the given dimension.
- **Array** (**Size** size, double value)
creates the array and fills it with value.
- **Array** (**Size** size, double value, double increment)
creates the array and fills it according to $a_0 = \text{value}$, $a_i = a_{i-1} + \text{increment}$.
- **Array** (const **Array** &from)
- **~Array** ()
- **Array** & **operator=** (const **Array** &from)

Vector algebra

$v \ += \ x$ and similar operation involving a scalar value are shortcuts for $\forall i : v_i = v_i + x$

$v \ *= \ w$ and similar operation involving two vectors are shortcuts for $\forall i : v_i = v_i \times w_i$

This implementation was inspired by T. L. Veldhuizen, Expression templates, *C++ Report*, 7(5):26-31, June 1995 available at <http://extreme.indiana.edu/~tveldhui/papers/>

Precondition:

all arrays involved in an algebraic expression must have the same size.

- **Array** & **operator+=** (const **Array** &)
- **Array** & **operator+=** (double)
- **Array** & **operator-=** (const **Array** &)

- Array & **operator**-= (double)
- Array & **operator** *= (const Array &)
- Array & **operator** /= (double)
- Array & **operator** /= (const Array &)
- Array & **operator** /= (double)

Element access

- double **operator**[] (Size) const
read-only.
- double & **operator**[] (Size)
read-write.

Inspectors

- Size **size** () const
dimension of the array.

Iterator access

- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **end** () const
- iterator **end** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rend** () const
- reverse_iterator **rend** ()

Related Functions

(Note that these are not member functions.)

- double **DotProduct** (const Array &, const Array &)
- Array **operator**+ (const Array &v)
- Array **operator**- (const Array &v)
- Array **operator**+ (const Array &, const Array &)
- Array **operator**+ (const Array &, double)
- Array **operator**+ (double, const Array &)
- Array **operator**- (const Array &, const Array &)
- Array **operator**- (const Array &, double)
- Array **operator**- (double, const Array &)
- Array **operator** * (const Array &, const Array &)
- Array **operator** * (const Array &, double)
- Array **operator** * (double, const Array &)
- Array **operator**/ (const Array &, const Array &)
- Array **operator**/ (const Array &, double)
- Array **operator**/ (double, const Array &)
- Array **Abs** (const Array &)
- Array **Sqrt** (const Array &)
- Array **Log** (const Array &)
- Array **Exp** (const Array &)

11.8.1 Detailed Description

This class implements the concept of vector as used in linear algebra. As such, it is **not** meant to be used as a container - `std::vector` should be used instead.

The documentation for this class was generated from the following file:

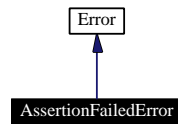
- [array.hpp](#)

11.9 QuantLib::AssertionFailedError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::AssertionFailedError:



Public Methods

- **AssertionFailedError** (const std::string &what=“”)

11.9.1 Detailed Description

Thrown upon a failed assertion.

The documentation for this class was generated from the following file:

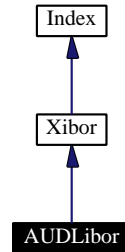
- [errors.hpp](#)

11.10 QuantLib::Indexes::AUDLiber Class Reference

AUD Libor index (Also known as TIBOR, check settlement days).

```
#include <audlibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::AUDLiber:



Public Methods

- **AUDLiber** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

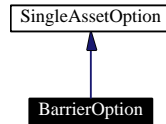
- [audlibor.hpp](#)

11.11 QuantLib::Pricers::BarrierOption Class Reference

Barrier option.

```
#include <ql/Pricers/barrieroption.hpp>
```

Inheritance diagram for QuantLib::Pricers::BarrierOption:



Public Types

- enum **BarrierType** { **DownIn**, **UpIn**, **DownOut**, **UpOut** }

Public Methods

- **BarrierOption** (BarrierType barrType, Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, double barrier, double rebate=0.0)
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

Protected Methods

- void **calculate_** () const

Protected Attributes

- double **greeksCalculated_**
- double **delta_**
- double **gamma_**
- double **theta_**

11.11.1 Detailed Description

The analytical calculation are taken from "Option pricing formulas", E.G. Haug, McGraw-Hill, p.69 and following.

The documentation for this class was generated from the following files:

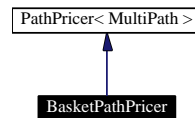
- [barrieroption.hpp](#)
- [barrieroption.cpp](#)

11.12 QuantLib::MonteCarlo::BasketPathPricer Class Reference

multipath pricer for European-type basket option.

```
#include <basketpathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::BasketPathPricer:



Public Methods

- **BasketPathPricer** (Option::Type type, const [Array](#) &underlying, double strike, [DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [MultiPath](#) &multiPath) const

11.12.1 Detailed Description

The value of the option at expiration is given by the value of the underlying which has best performed.

The documentation for this class was generated from the following files:

- [basketpathpricer.hpp](#)
- [basketpathpricer.cpp](#)

11.13 QuantLib::Math::BilinearInterpolation Class Template Reference

bilinear interpolation between discrete points.

```
#include <bilinearinterpolation.hpp>
```

Inheritance diagram for QuantLib::Math::BilinearInterpolation:



Public Types

- typedef double **result_type**

Public Methods

- **BilinearInterpolation** (const RandomAccessIteratorX &xBegin, const RandomAccessIteratorX &xEnd, const RandomAccessIteratorY &yBegin, const RandomAccessIteratorY &yEnd, const MatricialData &data, bool allowExtrapolation)
- double [operator\(\)](#) (const first_argument_type &x, const second_argument_type &y) const

Public Attributes

- typedef< RandomAccessIteratorX >::value_type **first_argument_type**
- typedef< RandomAccessIteratorY >::value_type **second_argument_type**

```
template<class RandomAccessIteratorX, class RandomAccessIteratorY, class MatricialData>
class QuantLib::Math::BilinearInterpolation< RandomAccessIteratorX, RandomAccessIteratorY,
MatricialData >
```

11.13.1 Member Function Documentation

11.13.1.1 double [operator\(\)](#) (const first_argument_type &x, const second_argument_type &y)
const [virtual]

This operator must be overridden to provide an implementation of the actual interpolation.

Precondition:

The sequence of values for x must have been sorted for the result to make sense.

Implements [QuantLib::Math::Interpolation2D](#).

The documentation for this class was generated from the following file:

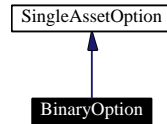
- [bilinearinterpolation.hpp](#)

11.14 QuantLib::Pricers::BinaryOption Class Reference

Binary (digital) option.

```
#include <binaryoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::BinaryOption:



Public Methods

- **BinaryOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, double cashPayoff=1)
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- double **vega** () const
- double **rho** () const
- double **dividendRho** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

The documentation for this class was generated from the following files:

- [binaryoption.hpp](#)
- [binaryoption.cpp](#)

11.15 QuantLib::Solvers1D::Bisection Class Reference

bisection 1-D solver.

```
#include <bisection.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::Bisection:



Protected Methods

- double [solve_](#)(const [ObjectiveFunction](#) &*f*, double *xAccuracy*) const

11.15.1 Member Function Documentation

11.15.1.1 double [solve_](#)(const [ObjectiveFunction](#) &*f*, double *xAccuracy*) const [protected, virtual]

This method must be implemented in derived classes and contains the actual code which searches for the zeroes of the [ObjectiveFunction](#). It assumes that:

- **xMin_** and **xMax_** form a valid bracket;
- **fxMin_** and **fxMax_** contain the values of the function in **xMin_** and **xMax_**;
- **root_** was initialized to a valid initial guess.

Implements [QuantLib::Solver1D](#).

The documentation for this class was generated from the following files:

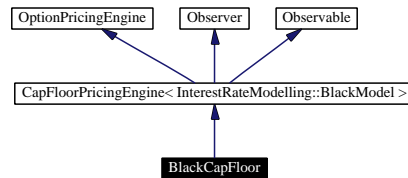
- [bisection.hpp](#)
- [bisection.cpp](#)

11.16 QuantLib::Pricers::BlackCapFloor Class Reference

CapFloor priced by the Black formula.

```
#include <blackcapfloor.hpp>
```

Inheritance diagram for QuantLib::Pricers::BlackCapFloor:



Public Methods

- **BlackCapFloor** (const [Handle](#)< [InterestRateModelling::BlackModel](#) > &mod)
- void **calculate** () const

The documentation for this class was generated from the following files:

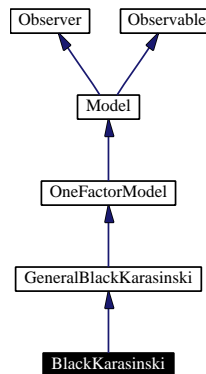
- [blackcapfloor.hpp](#)
- [blackcapfloor.cpp](#)

11.17 QuantLib::InterestRateModelling::BlackKarasinski Class Reference

Standard Black-Karasinski model class.

```
#include <blackkarasinski.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::BlackKarasinski:



Public Methods

- **BlackKarasinski** (const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- virtual ~**BlackKarasinski** ()

11.17.1 Detailed Description

This class implements the standard Black-Karasinski model defined by $d \ln r_t = (\theta(t) - \alpha \ln r_t)dt + \sigma dW_t$, where *alpha* and *sigma* are constants.

The documentation for this class was generated from the following file:

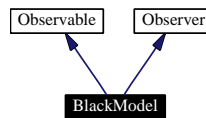
- [blackkarasinski.hpp](#)

11.18 QuantLib::InterestRateModelling::BlackModel Class Reference

Abstract short-rate model class.

```
#include <blackmodel.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::BlackModel:



Public Methods

- **BlackModel** (const [RelinkableHandle](#)< [MarketElement](#) > &volatility, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- double **volatility** () const
- const [RelinkableHandle](#)< [TermStructure](#) > & **termStructure** () const
- void **update** ()

Static Public Methods

- double **formula** (double k, double f, double v, double w)

11.18.1 Member Function Documentation

11.18.1.1 void update () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

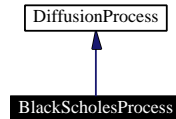
- [blackmodel.hpp](#)

11.19 QuantLib::BlackScholesProcess Class Reference

Black-Scholes diffusion process class.

```
#include <diffusionprocess.hpp>
```

Inheritance diagram for QuantLib::BlackScholesProcess:



Public Methods

- **BlackScholesProcess** ([Rate](#) rate, double volatility, double s0=0.0)
- virtual double **drift** ([Time](#) t, double x) const
- virtual double **diffusion** ([Time](#) t, double x) const
- virtual double **expectation** ([Time](#) t0, double x0, [Time](#) dt) const
Euler approximation of the expectation.
- virtual double **variance** ([Time](#) t0, double x0, [Time](#) dt) const
Euler approximation of the variance.

11.19.1 Detailed Description

This class describes the stochastic process governed by

$$dS = rdt + \sigma dW_t$$

.

The documentation for this class was generated from the following file:

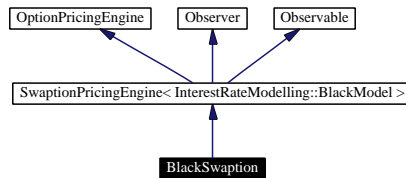
- [diffusionprocess.hpp](#)

11.20 QuantLib::Pricers::BlackSwaption Class Reference

Swaption priced by the Black formula.

```
#include <blackswaption.hpp>
```

Inheritance diagram for QuantLib::Pricers::BlackSwaption:



Public Methods

- **BlackSwaption** (const [Handle](#)< [InterestRateModelling::BlackModel](#) > &mod)
- void **calculate** () const

The documentation for this class was generated from the following files:

- [blackswaption.hpp](#)
- [blackswaption.cpp](#)

11.21 QuantLib::FiniteDifferences::BoundaryCondition Class Reference

Boundary condition for finite difference problems.

```
#include <boundarycondition.hpp>
```

Public Types

- enum **Type** { **None**, **Neumann**, **Dirichlet** }

Public Methods

- **BoundaryCondition** (Type type=None, double value=[Null](#)< double >())
- Type **type** () const
- double **value** () const

11.21.1 Detailed Description

Three possibilities are given for setting boundary conditions, namely, no boundary condition, Dirichlet boundary condition (i.e., constant value), and Neumann boundary condition (i.e., constant derivative).

Warning:

For Neumann conditions. the value passed must not be the value of the derivative. Instead, it must be comprehensive of the grid step between the first two points—i.e., it must be the difference between $f[0]$ and $f[1]$.

The documentation for this class was generated from the following file:

- [boundarycondition.hpp](#)

11.22 QuantLib::RandomNumbers::BoxMullerGaussianRng Class Template Reference

Gaussian random number generator.

```
#include <boxmullergaussianrng.hpp>
```

Public Types

- typedef [MonteCarlo::Sample](#)< double > **sample_type**

Public Methods

- **BoxMullerGaussianRng** (long seed=0)
- sample_type [next](#) () const
returns next sample from the Gaussian distribution.

11.22.1 Detailed Description

```
template<class U> class QuantLib::RandomNumbers::BoxMullerGaussianRng< U >
```

It uses the well-known Box-Muller transformation to return a normal distributed Gaussian deviate with average 0.0 and standard deviation of 1.0, from a uniform deviate in (0,1) supplied by U.

Class U must implement the following interface:

```
U::U(long seed);  
U::sample_type U::next() const;
```

The documentation for this class was generated from the following file:

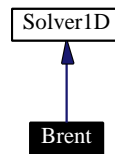
- [boxmullergaussianrng.hpp](#)

11.23 QuantLib::Solvers1D::Brent Class Reference

Brent 1-D solver.

```
#include <brent.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::Brent:



The documentation for this class was generated from the following files:

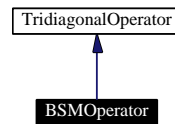
- [brent.hpp](#)
- [brent.cpp](#)

11.24 QuantLib::FiniteDifferences::BSMOperator Class Reference

Black-Scholes-Merton differential operator.

```
#include <bsmoperator.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::BSMOperator:



Public Methods

- **BSMOperator** ()
- **BSMOperator** ([Size](#) size, double dx, double r, double q, double sigma)

The documentation for this class was generated from the following files:

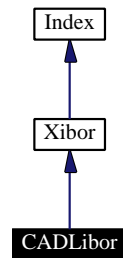
- [bsmoperator.hpp](#)
- [bsmoperator.cpp](#)

11.25 QuantLib::Indexes::CADLiber Class Reference

CAD Liber index (Also known as CDOR, check settlement days).

```
#include <cadlibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::CADLiber:



Public Methods

- **CADLiber** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

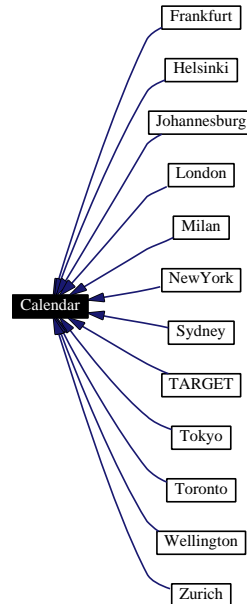
- [cadlibor.hpp](#)

11.26 QuantLib::Calendar Class Reference

calendar class.

```
#include <calendar.hpp>
```

Inheritance diagram for QuantLib::Calendar:



Public Methods

Calendar interface

- `std::string name () const`
Returns the name of the calendar.
- `bool isBusinessDay (const Date &d) const`
- `bool isHoliday (const Date &d) const`
- `Date roll (const Date &, RollingConvention convention=Following) const`
- `Date advance (const Date &, int n, TimeUnit unit, RollingConvention convention=Following) const`

Protected Methods

- `Calendar (const Handle< CalendarImpl > &impl)`

Related Functions

(Note that these are not member functions.)

- `bool operator== (const Calendar &, const Calendar &)`
- `bool operator!= (const Calendar &, const Calendar &)`

11.26.1 Detailed Description

This class provides methods for determining whether a date is a business day or a holiday for a given market, and for incrementing/decrementing a date of a given number of business days.

The Strategy pattern is used to provide the base behavior of the calendar, namely, to determine whether a date is a business day.

11.26.2 Constructor & Destructor Documentation

11.26.2.1 `Calendar (const Handle< CalendarImpl > &impl) [inline, protected]`

this protected constructor will only be invoked by derived classes which define a given [Calendar](#) implementation

11.26.3 Member Function Documentation

11.26.3.1 `std::string name () const [inline]`

Warning:

This method is used for output and comparison between calendars. It is **not** meant to be used for writing switch-on-type code.

11.26.3.2 `bool isBusinessDay (const Date &d) const [inline]`

Returns `true` iff the date is a business day for the given market.

11.26.3.3 `bool isHoliday (const Date &d) const [inline]`

Returns `true` iff the date is a holiday for the given market.

11.26.3.4 `Date roll (const Date &, RollingConvention convention = Following) const`

Returns the next business day on the given market with respect to the given date and convention.

11.26.3.5 `Date advance (const Date &, int n, TimeUnit unit, RollingConvention convention = Following) const`

Advances the given date of the given number of business days and returns the result.

Note:

The input date is not modified.

11.26.4 Friends And Related Function Documentation

11.26.4.1 `bool operator== (const Calendar &, const Calendar &) [related]`

Returns `true` iff the two calendars belong to the same derived class.

The documentation for this class was generated from the following files:

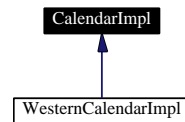
- [calendar.hpp](#)
- [calendar.cpp](#)

11.27 QuantLib::Calendar::CalendarImpl Class Reference

abstract base class for calendar implementations.

```
#include <calendar.hpp>
```

Inheritance diagram for QuantLib::Calendar::CalendarImpl:



Public Methods

- virtual std::string **name** () const=0
- virtual bool **isBusinessDay** (const [Date](#) &) const=0

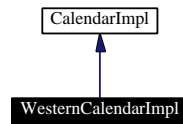
The documentation for this class was generated from the following file:

- [calendar.hpp](#)

11.28 QuantLib::Calendar::WesternCalendarImpl Class Reference

```
#include <calendar.hpp>
```

Inheritance diagram for QuantLib::Calendar::WesternCalendarImpl:



Static Protected Methods

- [Day easterMonday](#) ([Year](#) y)
expressed relative to first day of year.

11.28.1 Detailed Description

partial implementation providing the means of determining the Easter Monday for a given year.

The documentation for this class was generated from the following files:

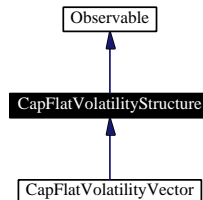
- [calendar.hpp](#)
- [calendar.cpp](#)

11.29 QuantLib::CapFlatVolatilityStructure Class Reference

Cap/floor flat volatility structure.

```
#include <capvolstructures.hpp>
```

Inheritance diagram for QuantLib::CapFlatVolatilityStructure:



Public Methods

- virtual `~CapFlatVolatilityStructure ()`
- virtual `Date todaysDate () const=0`
returns today's date.
- virtual `Date settlementDate () const=0`
returns the settlement date.
- virtual `DayCounter dayCounter () const=0`
returns the day counter used for internal date/time conversions.
- double `volatility (const Date &end, Rate strike) const`
returns the volatility for a given end date and strike rate.
- double `volatility (const Period &length, Rate strike) const`
returns the volatility for a given cap/floor length and strike rate.
- double `volatility (Time t, Rate strike) const`
returns the volatility for a given end time and strike rate.

Protected Methods

- virtual double `volatilityImpl (Time length, Rate strike) const=0`
implements the actual volatility calculation in derived classes.

11.29.1 Detailed Description

This class is purely abstract and defines the interface of concrete structures which will be derived from this one.

The documentation for this class was generated from the following file:

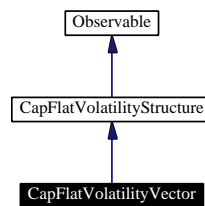
- [capvolstructures.hpp](#)

11.30 QuantLib::Volatilities::CapFlatVolatilityVector Class Reference

Cap/floor at-the-money flat volatility vector.

```
#include <capflatvolvector.hpp>
```

Inheritance diagram for QuantLib::Volatilities::CapFlatVolatilityVector:



Public Methods

- **CapFlatVolatilityVector** (const [Date](#) &todayDate, const [Calendar](#) &calendar, int settlementDays, const std::vector< [Period](#) > &lengths, const std::vector< double > &volatilities, const [DayCounter](#) &dayCounter=[DayCounters::Thirty360](#)())
- [Date](#) **todayDate** () const
returns today's date.
- [Date](#) **settlementDate** () const
returns the settlement date.
- [DayCounter](#) **dayCounter** () const
returns the day counter used for internal date/time conversions.

11.30.1 Detailed Description

This class provides the at-the-money volatility for a given cap by interpolating a volatility vector whose elements are the market volatilities of a set of caps/floors with given length.

Todo:

Either add correct copy behavior or inhibit copy. Right now, a copied instance would end up with its own copy of the length vector but an interpolation pointing to the original ones.

The documentation for this class was generated from the following file:

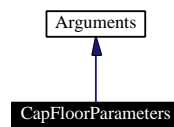
- [capflatvolvector.hpp](#)

11.31 QuantLib::Instruments::CapFloorParameters Class Reference

parameters for cap/floor calculation.

```
#include <capfloor.hpp>
```

Inheritance diagram for QuantLib::Instruments::CapFloorParameters:



Public Methods

- **CapFloorParameters** ()
- void **validate** () const

Public Attributes

- VanillaCapFloor::Type **type**
- std::vector< [Time](#) > **startTimes**
- std::vector< [Time](#) > **endTimes**
- std::vector< [Time](#) > **accrualTimes**
- std::vector< [Rate](#) > **capRates**
- std::vector< [Rate](#) > **floorRates**
- std::vector< [Rate](#) > **forwards**
- std::vector< double > **nominals**

The documentation for this class was generated from the following files:

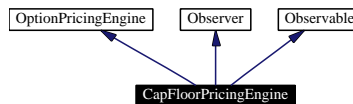
- [capfloor.hpp](#)
- [capfloor.cpp](#)

11.32 QuantLib::Pricers::CapFloorPricingEngine Class Template Reference

base class for cap/floor pricing engines.

```
#include <capfloor.hpp>
```

Inheritance diagram for QuantLib::Pricers::CapFloorPricingEngine:



Public Methods

- **CapFloorPricingEngine** ()
- **CapFloorPricingEngine** (const [Handle](#)< ModelType > &model)
- [Arguments](#) * **parameters** ()
- const [Results](#) * **results** () const
- void **validateParameters** () const
- void **setModel** (const [Handle](#)< ModelType > &model)
- void [update](#) ()

Protected Attributes

- [Instruments::CapFloorParameters](#) **parameters_**
- [Instruments::CapFloorResults](#) **results_**
- [Handle](#)< ModelType > **model_**

11.32.1 Detailed Description

```
template<class ModelType> class QuantLib::Pricers::CapFloorPricingEngine< ModelType >
```

Derived engines only need to implement the `calculate()` method

11.32.2 Member Function Documentation

11.32.2.1 void update() [inline, virtual]

This method must be implemented in derived classes. An instance of `Observer` does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

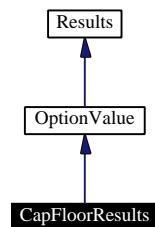
- [capfloor.hpp](#)

11.33 QuantLib::Instruments::CapFloorResults Class Reference

results from cap/floor calculation.

```
#include <capfloor.hpp>
```

Inheritance diagram for QuantLib::Instruments::CapFloorResults:



The documentation for this class was generated from the following file:

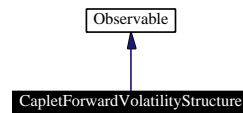
- [capfloor.hpp](#)

11.34 QuantLib::CapletForwardVolatilityStructure Class Reference

Caplet/floorlet forward volatility structure.

```
#include <capvolstructures.hpp>
```

Inheritance diagram for QuantLib::CapletForwardVolatilityStructure:



Public Methods

- virtual `~CapletForwardVolatilityStructure ()`
- virtual `Date todaysDate () const=0`
returns today's date.
- virtual `Date settlementDate () const=0`
returns the settlement date.
- virtual `DayCounter dayCounter () const=0`
returns the day counter used for internal date/time conversions.
- double `volatility (const Date &start, Rate strike) const`
returns the volatility for a given start date and strike rate.
- double `volatility (Time t, Rate strike) const`
returns the volatility for a given start time and strike rate.

Protected Methods

- virtual double `volatilityImpl (Time length, Rate strike) const=0`
implements the actual volatility calculation in derived classes.

11.34.1 Detailed Description

This class is purely abstract and defines the interface of concrete structures which will be derived from this one.

The documentation for this class was generated from the following file:

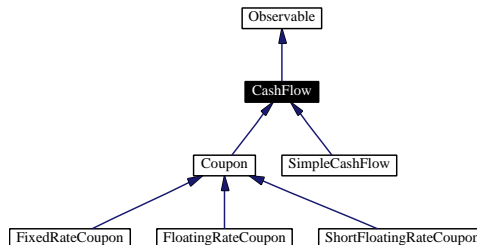
- [capvolstructures.hpp](#)

11.35 QuantLib::CashFlow Class Reference

Base class for cash flows.

```
#include <cashflow.hpp>
```

Inheritance diagram for QuantLib::CashFlow:



Public Methods

- virtual `~CashFlow()`
- virtual double `amount()` const=0
returns the amount of the cash flow.
- virtual `Date date()` const=0
returns the date at which the cash flow is settled.

11.35.1 Detailed Description

This class is purely virtual and acts as a base class for the actual cash flow implementations.

11.35.2 Member Function Documentation

11.35.2.1 virtual double amount() const [pure virtual]

Note:

The amount is not discounted, i.e., it is the actual amount paid at the cash flow date.

Implemented in [QuantLib::CashFlows::FixedRateCoupon](#).

The documentation for this class was generated from the following file:

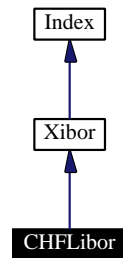
- [cashflow.hpp](#)

11.36 QuantLib::Indexes::CHFLibor Class Reference

CHF Libor index (Also known as ZIBOR, check settlement days and day-count).

```
#include <chflibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::CHFLibor:



Public Methods

- **CHFLibor** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

- [chflibor.hpp](#)

11.37 QuantLib::RandomNumbers::CLGaussianRng Class Template Reference

Gaussian random number generator.

```
#include <centrallimitgaussianrng.hpp>
```

Public Types

- typedef [MonteCarlo::Sample](#)< double > **sample_type**

Public Methods

- **CLGaussianRng** (long seed=0)
- sample_type [next](#) () const
returns next sample from the Gaussian distribution.

11.37.1 Detailed Description

```
template<class U> class QuantLib::RandomNumbers::CLGaussianRng< U >
```

It uses the well-known fact that the sum of 12 uniform deviate in (-.5,.5) is approximately a Gaussian deviate with average 0 and standard deviation 1. The uniform deviate is supplied by U.

Class U must implement the following interface:

```
U::U(long seed);  
U::sample_type U::next() const;
```

The documentation for this class was generated from the following file:

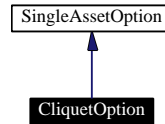
- [centrallimitgaussianrng.hpp](#)

11.38 QuantLib::Pricers::CliquetOption Class Reference

cliquet (Ratchet) option.

```
#include <cliquetoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::CliquetOption:



Public Methods

- **CliquetOption** (Option::Type type, double underlying, [Spread](#) dividendYield, [Rate](#) riskFreeRate, const std::vector< [Time](#) > &dates, double volatility)
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- double **vega** () const
- double **rho** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

11.38.1 Detailed Description

A cliquet option, also known as Ratchet option, is a series of forward-starting options where the strike for the next exercise date is set to the spot price at the beginning of each period. In the particular case in which only two dates are given the price of the option is the same as that of a forward-starting option starting at the first date and expiring at the second date.

The documentation for this class was generated from the following files:

- [cliquetoption.hpp](#)
- [cliquetoption.cpp](#)

11.39 QuantLib::Utilities::combining_iterator Class Template Reference

Iterator mapping a function to a set of underlying sequences.

```
#include <combiningiterator.hpp>
```

Public Types

- typedef Function::result_type **value_type**
- typedef const Function::result_type * **pointer**
- typedef const Function::result_type & **reference**

Public Methods

- template<class IteratorCollectionIterator> **combining_iterator** (IteratorCollectionIterator it1, IteratorCollectionIterator it2, Function f)

Dereferencing

- reference **operator** * () const
- pointer **operator** → () const

Random access

- value_type **operator**[] (difference_type n) const

Increment and decrement

- combining_iterator & **operator**++ ()
- combining_iterator **operator**++ (int)
- combining_iterator & **operator**-- ()
- combining_iterator **operator**-- (int)
- combining_iterator & **operator**+= (difference_type n)
- combining_iterator & **operator**-= (difference_type n)
- combining_iterator **operator**+ (difference_type n) const
- combining_iterator **operator**- (difference_type n) const

Difference

- difference_type **operator**- (const combining_iterator< Iterator, Function > &rhs) const

Comparisons

- bool **operator**== (const combining_iterator< Iterator, Function > &rhs) const
- bool **operator**!= (const combining_iterator< Iterator, Function > &rhs) const

Public Attributes

- typedef< Iterator >::difference_type **difference_type**

Related Functions

(Note that these are not member functions.)

- `combining_iterator< typename 1< It >::value_type, Function >` [make_combining_iterator](#) (It it1, It it2, Function f)
helper function to create combining iterators.

11.39.1 Detailed Description

`template<class Iterator, class Function> class QuantLib::Utilities::combining_iterator< Iterator, Function >`

This iterator advances a set of underlying iterators and returns the values obtained by applying a function to the sets of values such iterators point to.

This class was implemented based on Christopher Baus and Thomas Becker, *Custom Iterators for the STL*, included in the proceedings of the First Workshop on C++ Template Programming, Erfurt, Germany, 2000 (<http://www.oonumerics.org/tmpw00/>)

The documentation for this class was generated from the following file:

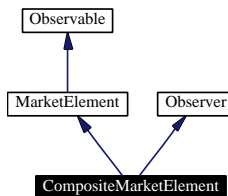
- [combiningiterator.hpp](#)

11.40 QuantLib::CompositeMarketElement Class Template Reference

market element whose value depends on two other market element.

```
#include <marketelement.hpp>
```

Inheritance diagram for QuantLib::CompositeMarketElement:



Public Methods

- **CompositeMarketElement** (const [RelinkableHandle](#)< [MarketElement](#) > &element1, const [RelinkableHandle](#)< [MarketElement](#) > &element2, const BinaryFunction &f)

Market element interface

- double [value](#) () const
returns the current value.

Observer interface

- void [update](#) ()

```
template<class BinaryFunction> class QuantLib::CompositeMarketElement< BinaryFunction >
```

11.40.1 Member Function Documentation

11.40.1.1 void update () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

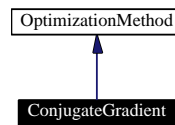
- [marketelement.hpp](#)

11.41 QuantLib::Optimization::ConjugateGradient Class Reference

Multi-dimensionnal Conjugate Gradient class.

```
#include <conjugategradient.hpp>
```

Inheritance diagram for QuantLib::Optimization::ConjugateGradient:



Public Methods

- [ConjugateGradient](#) ()
default constructor.
- [ConjugateGradient](#) (const [Handle](#)< [LineSearch](#) > &lineSearch)
- virtual [~ConjugateGradient](#) ()
destructor.
- virtual void [minimize](#) ([OptimizationProblem](#) &P)
minimize the optimization problem P.

11.41.1 Detailed Description

User has to provide line-search method and optimization end criteria

search direction $d_i = -f'(x_i) + c_i * d_{i-1}$ where $c_i = ||f'(x_i)||^2 / ||f'(x_{i-1})||^2$ and $d_1 = -f'(x_1)$

The documentation for this class was generated from the following files:

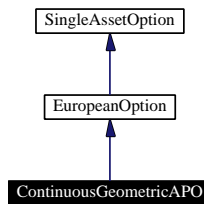
- [conjugategradient.hpp](#)
- [conjugategradient.cpp](#)

11.42 QuantLib::Pricers::ContinuousGeometricAPO Class Reference

Continuous Geometric Average Price [Option](#) (European exercise).

```
#include <continuousgeometricapo.hpp>
```

Inheritance diagram for QuantLib::Pricers::ContinuousGeometricAPO:



Public Methods

- **ContinuousGeometricAPO** (Option::Type type, double underlying, double strike, [Spread](#) dividend-Yield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility)
- double **vega** () const
- double **rho** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

11.42.1 Detailed Description

This class implements a continuous geometric average price asian option with european exercise. The formula is from "[Option Pricing Formulas](#)", E. G. Haug (1997) pag 96-97

Todo:

add Average Strike version and make it backward starting

The documentation for this class was generated from the following file:

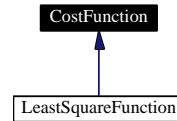
- [continuousgeometricapo.hpp](#)

11.43 QuantLib::Optimization::CostFunction Class Reference

Cost function abstract class for optimization problem.

```
#include <costfunction.hpp>
```

Inheritance diagram for QuantLib::Optimization::CostFunction:



Public Methods

- virtual double `value` (const `Array` &x)=0
method to overload to compute the cost function value in x.
- virtual void `gradient` (`Array` &grad, const `Array` &x)
method to overload to compute grad_f , the first derivative of.
- virtual double `valueAndGradient` (`Array` &grad, const `Array` &x)
method to overload to compute grad_f , the first derivative of.
- virtual double `finiteDifferenceEpsilon` ()
Default epsilon for finite difference method .:

The documentation for this class was generated from the following file:

- `costfunction.hpp`

11.44 QuantLib::Utilities::coupling_iterator Class Template Reference

Iterator mapping a function to a pair of underlying sequences.

```
#include <couplingiterator.hpp>
```

Public Types

- typedef Function::result_type **value_type**
- typedef const Function::result_type * **pointer**
- typedef const Function::result_type & **reference**

Public Methods

- **coupling_iterator** (Iterator1 it1, Iterator2 it2, Function f)

Dereferencing

- reference **operator** * () const
- pointer **operator** → () const

Random access

- value_type **operator**[] (difference_type n) const

Increment and decrement

- coupling_iterator & **operator**++ ()
- coupling_iterator **operator**++ (int)
- coupling_iterator & **operator**-- ()
- coupling_iterator **operator**-- (int)
- coupling_iterator & **operator**+= (difference_type n)
- coupling_iterator & **operator**-= (difference_type n)
- coupling_iterator **operator**+ (difference_type n) const
- coupling_iterator **operator**- (difference_type n) const

Difference

- difference_type **operator**- (const coupling_iterator &rhs) const

Comparisons

- bool **operator**== (const coupling_iterator &rhs) const
- bool **operator**!= (const coupling_iterator &rhs) const

Public Attributes

- typedef< Iterator1 >::difference_type **difference_type**

Related Functions

(Note that these are not member functions.)

- `coupling_iterator< It1, It2, Function >` [make_coupling_iterator](#) (It1 it1, It2 it2, Function f)
helper function to create combining iterators.

11.44.1 Detailed Description

`template<class Iterator1, class Iterator2, class Function> class QuantLib::Utilities::coupling_iterator< Iterator1, Iterator2, Function >`

This iterator advances two underlying iterators and returns the values obtained by applying a function to the two values such iterators point to.

The documentation for this class was generated from the following file:

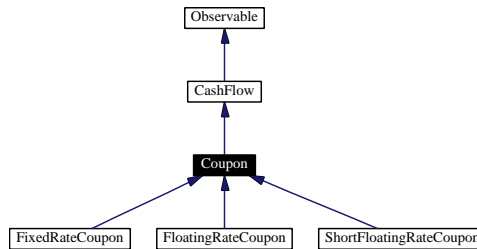
- [couplingiterator.hpp](#)

11.45 QuantLib::CashFlows::Coupon Class Reference

coupon accruing over a fixed period.

```
#include <coupon.hpp>
```

Inheritance diagram for QuantLib::CashFlows::Coupon:



Public Methods

- **Coupon** (double nominal, const [Calendar](#) &calendar, [RollingConvention](#) rollingConvention, const [DayCounter](#) &dayCounter, const [Date](#) &startDate, const [Date](#) &endDate, const [Date](#) &refPeriodStart=[Date](#)(), const [Date](#) &refPeriodEnd=[Date](#)())

Partial CashFlow interface

- [Date](#) **date** () const
returns the date at which the cash flow is settled.

Inspectors

- double **nominal** () const
- const [Date](#) & **accrualStartDate** () const
start of the accrual period.
- const [Date](#) & **accrualEndDate** () const
end of the accrual period.
- [Time](#) **accrualPeriod** () const
accrual period as fraction of year.
- int **accrualDays** () const
accrual period in days.
- virtual double **accruedAmount** (const [Date](#) &) const=0
accrued amount at the given date.

Protected Attributes

- double **nominal_**
- [Date](#) **startDate_**

- [Date](#) `endDate_`
- [Date](#) `refPeriodStart_`
- [Date](#) `refPeriodEnd_`
- [Calendar](#) `calendar_`
- [RollingConvention](#) `rollingConvention_`
- [DayCounter](#) `dayCounter_`

11.45.1 Detailed Description

This class implements part of the [CashFlow](#) interface but it is still abstract and provides derived classes with methods for accrual period calculations.

The documentation for this class was generated from the following file:

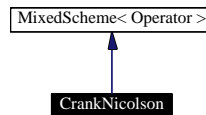
- [coupon.hpp](#)

11.46 QuantLib::FiniteDifferences::CrankNicolson Class Template Reference

Crank-Nicolson scheme for finite difference methods.

```
#include <cranknicolson.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::CrankNicolson:



Friends

- class `FiniteDifferenceModel< CrankNicolson< Operator > >`

11.46.1 Detailed Description

```
template<class Operator> class QuantLib::FiniteDifferences::CrankNicolson< Operator >
```

See sect. [The finite differences framework](#) for details on the method.

In this implementation, the passed operator must be derived from either `TimeConstantOperator` or `TimeDependentOperator`. Also, it must implement at least the following interface:

```
typedef ... arrayType;

// copy constructor/assignment
// (these will be provided by the compiler if none is defined)
Operator(const Operator&);
Operator& operator=(const Operator&);

// inspectors
Size size();

// modifiers
void setTime(Time t);

// operator interface
arrayType applyTo(const arrayType&);
arrayType solveFor(const arrayType&);
static Operator identity(Size size);

// operator algebra
Operator operator*(double, const Operator&);
Operator operator+(const Operator&, const Operator&);
Operator operator+(const Operator&, const Operator&);
```

Warning:

The differential operator must be linear for this evolver to work.

The documentation for this class was generated from the following file:

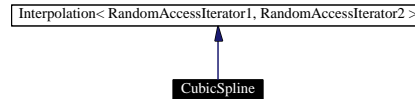
- [cranknicolson.hpp](#)

11.47 QuantLib::Math::CubicSpline Class Template Reference

cubic spline interpolation between discrete points.

```
#include <cubicspline.hpp>
```

Inheritance diagram for QuantLib::Math::CubicSpline:



Public Methods

- **CubicSpline** (const RandomAccessIterator1 &xBegin, const RandomAccessIterator1 &xEnd, const RandomAccessIterator2 &yBegin, bool allowExtrapolation)
- result_type **operator()** (const argument_type &x) const
- virtual ~**CubicSpline** ()

Public Attributes

- typedef< RandomAccessIterator1 >::value_type **argument_type**
- typedef< RandomAccessIterator2 >::value_type **result_type**

```
template<class RandomAccessIterator1, class RandomAccessIterator2> class QuantLib::Math::CubicSpline< RandomAccessIterator1, RandomAccessIterator2 >
```

11.47.1 Member Function Documentation

11.47.1.1 result_type operator() (const argument_type &x) const [virtual]

This operator must be overridden to provide an implementation of the actual interpolation.

Precondition:

The sequence of values for x must have been sorted for the result to make sense.

Implements [QuantLib::Math::Interpolation](#).

The documentation for this class was generated from the following file:

- [cubicspline.hpp](#)

11.48 QuantLib::CurrencyFormatter Class Reference

Formats currencies for output.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toString (Currency c)`

The documentation for this class was generated from the following files:

- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.49 QuantLib::Date Class Reference

Concrete date class.

```
#include <date.hpp>
```

Public Methods

constructors

- [Date](#) ()
Default constructor returning a null date.
- [Date](#) (int serialNumber)
Constructor taking a serial number as given by Applix or Excel.
- [Date](#) (Day d, Month m, Year y)
More traditional constructor.

inspectors

- [Weekday](#) [weekday](#) () const
- [Day](#) [dayOfMonth](#) () const
- [Day](#) [dayOfYear](#) () const
One-based (Jan 1st = 1).
- [Month](#) [month](#) () const
- [Year](#) [year](#) () const
- int [serialNumber](#) () const

date algebra

- Date & [operator+=](#) (int days)
increments date in place.
- Date & [operator-=](#) (int days)
decrement date in place.
- Date & [operator++](#) ()
1-day pre-increment.
- Date [operator++](#) (int)
1-day post-increment.
- Date & [operator--](#) ()
1-day pre-decrement.
- Date [operator--](#) (int)
1-day post-decrement.
- Date [operator+](#) (int days) const
returns a new incremented date.

- Date [operator-](#) (int days) const
returns a new decremented date.

other methods to increment/decrement dates

- Date **plusDays** (int days) const
- Date **plusWeeks** (int weeks) const
- Date **plusMonths** (int months) const
- Date **plusYears** (int years) const
- Date **plus** (int units, [TimeUnit](#)) const
- Date **plus** (const [Period](#) &) const

Static Public Methods

static methods

- bool **isLeap** ([Year](#) y)
- Date [minDate](#) ()
earliest allowed date.
- Date [maxDate](#) ()
latest allowed date.

Related Functions

(Note that these are not member functions.)

- int [operator-](#) (const Date &, const Date &)
Difference in days between dates.
- bool **operator==** (const Date &, const Date &)
- bool **operator!=** (const Date &, const Date &)
- bool **operator<** (const Date &, const Date &)
- bool **operator<=** (const Date &, const Date &)
- bool **operator>** (const Date &, const Date &)
- bool **operator>=** (const Date &, const Date &)
- std::ostream & **operator<<** (std::ostream &stream, const Date &result)

11.49.1 Detailed Description

This class provides methods to inspect dates as well as methods and operators which implement a limited date algebra (increasing and decreasing dates, and calculating their difference).

The documentation for this class was generated from the following files:

- [date.hpp](#)
- [date.cpp](#)

11.50 QuantLib::DateFormatter Class Reference

Formats dates for output.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toString (const Date &d, bool shortFormat=false)`

11.50.1 Detailed Description

Formatting can be in short (mm/dd/yyyy) or long (Month ddth, yyyy) form.

The documentation for this class was generated from the following files:

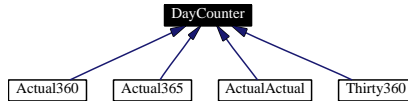
- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.51 QuantLib::DayCounter Class Reference

day counter class.

```
#include <daycounter.hpp>
```

Inheritance diagram for QuantLib::DayCounter:



Public Methods

DayCounter interface

- `std::string name () const`
Returns the name of the day counter.
- `int dayCount (const Date &, const Date &) const`
Returns the number of days between two dates.
- `Time yearFraction (const Date &, const Date &, const Date &refPeriodStart=Date(), const Date &refPeriodEnd=Date()) const`
Returns the period between two dates as a fraction of year.

Protected Methods

- `DayCounter (const Handle< DayCounterImpl > &impl)`

Related Functions

(Note that these are not member functions.)

- `bool operator== (const DayCounter &, const DayCounter &)`
- `bool operator!= (const DayCounter &, const DayCounter &)`

11.51.1 Detailed Description

This class provides methods for determining the length of a time period according to given market convention, both as a number of days and as a year fraction.

The Strategy pattern is used to provide the base behavior of the day counter.

11.51.2 Constructor & Destructor Documentation

11.51.2.1 DayCounter (const Handle< DayCounterImpl > &impl) [inline, protected]

this protected constructor will only be invoked by derived classes which define a given `Calendar` implementation

11.51.3 Member Function Documentation

11.51.3.1 `std::string name () const` [inline]

Warning:

This method is used for output and comparison between day counters. It is **not** meant to be used for writing switch-on-type code.

11.51.4 Friends And Related Function Documentation

11.51.4.1 `bool operator==(const DayCounter &, const DayCounter &)` [related]

Returns `true` iff the two day counters belong to the same derived class.

The documentation for this class was generated from the following file:

- [daycounter.hpp](#)

11.52 QuantLib::DayCounter::DayCounterImpl Class Reference

abstract base class for day counter implementations.

```
#include <daycounter.hpp>
```

Public Methods

- virtual std::string **name** () const=0
- virtual int **dayCount** (const [Date](#) &, const [Date](#) &) const=0
- virtual [Time](#) **yearFraction** (const [Date](#) &, const [Date](#) &, const [Date](#) &refPeriodStart, const [Date](#) &refPeriodEnd) const=0

The documentation for this class was generated from the following file:

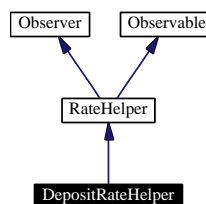
- [daycounter.hpp](#)

11.53 QuantLib::TermStructures::DepositRateHelper Class Reference

Deposit rate.

```
#include <ratehelpers.hpp>
```

Inheritance diagram for QuantLib::TermStructures::DepositRateHelper:



Public Methods

- **DepositRateHelper** (const [RelinkableHandle](#)< [MarketElement](#) > &rate, int settlementDays, int n, [TimeUnit](#) units, const [Calendar](#) &calendar, [RollingConvention](#) convention, const [DayCounter](#) &dayCounter)
- **DepositRateHelper** (double rate, int settlementDays, int n, [TimeUnit](#) units, const [Calendar](#) &calendar, [RollingConvention](#) convention, const [DayCounter](#) &dayCounter)
- double **impliedQuote** () const
- [DiscountFactor](#) **discountGuess** () const
- void **setTermStructure** ([TermStructure](#) *)
sets the term structure to be used for pricing.
- [Date](#) **maturity** () const
maturity date.

11.53.1 Detailed Description

Warning:

This class assumes that today's date does not change between calls of [setTermStructure](#)().

11.53.2 Member Function Documentation

11.53.2.1 void setTermStructure ([TermStructure](#) *) [virtual]

Warning:

Being a pointer and not a [Handle](#), the term structure is not guaranteed to remain allocated for the whole life of the rate helper. It is responsibility of the programmer to ensure that the pointer remains valid. It is advised that rate helpers be used only in term structure constructors, setting the term structure to **this**, i.e., the one being constructed.

Reimplemented from [QuantLib::TermStructures::RateHelper](#).

The documentation for this class was generated from the following files:

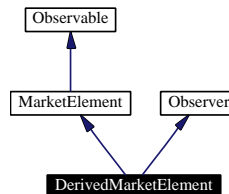
- [ratehelpers.hpp](#)
- [ratehelpers.cpp](#)

11.54 QuantLib::DerivedMarketElement Class Template Reference

market element whose value depends on another market element.

```
#include <marketelement.hpp>
```

Inheritance diagram for QuantLib::DerivedMarketElement:



Public Methods

- **DerivedMarketElement** (const [RelinkableHandle](#)< [MarketElement](#) > &element, const Unary-Function &f)

Market element interface

- double [value](#) () const
returns the current value.

Observer interface

- void [update](#) ()

```
template<class UnaryFunction> class QuantLib::DerivedMarketElement< UnaryFunction >
```

11.54.1 Member Function Documentation

11.54.1.1 void update () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

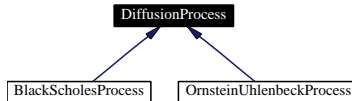
- [marketelement.hpp](#)

11.55 QuantLib::DiffusionProcess Class Reference

Diffusion process class.

```
#include <diffusionprocess.hpp>
```

Inheritance diagram for QuantLib::DiffusionProcess:



Public Methods

- **DiffusionProcess** (double x0)
- virtual \sim **DiffusionProcess** ()
- double **x0** () const
- virtual double **drift** (**Time** t, double x) const=0
- virtual double **diffusion** (**Time** t, double x) const=0
- virtual double **expectation** (**Time** t0, double x0, **Time** dt) const
Euler approximation of the expectation.
- virtual double **variance** (**Time** t0, double x0, **Time** dt) const
Euler approximation of the variance.

11.55.1 Detailed Description

This class describes a stochastic process goverved by

$$dx = \mu(t, x)dt + \sigma(t, x)dW_t$$

.

The documentation for this class was generated from the following file:

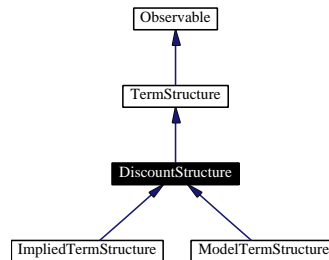
- [diffusionprocess.hpp](#)

11.56 QuantLib::DiscountStructure Class Reference

Discount factor term structure.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::DiscountStructure:



Public Methods

- virtual `~DiscountStructure()`

Protected Methods

- `Rate zeroYieldImpl (Time, bool extrapolate=false) const`
- `Rate forwardImpl (Time, bool extrapolate=false) const`

11.56.1 Detailed Description

This abstract class acts as an adapter to [TermStructure](#) allowing the programmer to implement only the `discount(const Date&)` method in derived classes.

11.56.2 Member Function Documentation

11.56.2.1 `Rate zeroYieldImpl (Time, bool extrapolate = false) const` [inline, protected, virtual]

Returns the zero yield rate for the given date calculating it from the discount.

Implements [QuantLib::TermStructure](#).

11.56.2.2 `Rate forwardImpl (Time, bool extrapolate = false) const` [inline, protected, virtual]

Returns the instantaneous forward rate for the given date calculating it from the discount.

Implements [QuantLib::TermStructure](#).

The documentation for this class was generated from the following file:

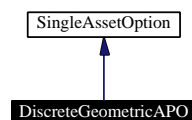
- [termstructure.hpp](#)

11.57 QuantLib::Pricers::DiscreteGeometricAPO Class Reference

Discrete Geometric Average Price Asian [Option](#) (European style).

```
#include <discretegeometricapo.hpp>
```

Inheritance diagram for QuantLib::Pricers::DiscreteGeometricAPO:



Public Methods

- **DiscreteGeometricAPO** (Option::Type type, double underlying, double strike, [Spread](#) dividend-Yield, [Rate](#) riskFreeRate, const std::vector< [Time](#) > ×, double volatility)
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

11.57.1 Detailed Description

This class implements a discrete geometric average price asian option, with european exercise. The formula is from "Asian [Option](#)", E. Levy (1997) in "Exotic Options: The State of the Art", edited by L. Clewlow, C. Strickland, pag65-97

Todo:

add analytical greeks

The documentation for this class was generated from the following files:

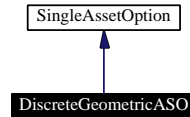
- [discretegeometricapo.hpp](#)
- [discretegeometricapo.cpp](#)

11.58 QuantLib::Pricers::DiscreteGeometricASO Class Reference

Discrete Geometric Average Strike Asian [Option](#) (European style).

```
#include <discretegeometricaso.hpp>
```

Inheritance diagram for QuantLib::Pricers::DiscreteGeometricASO:



Public Methods

- **DiscreteGeometricASO** (Option::Type type, double underlying, [Spread](#) dividendYield, [Rate](#) risk-FreeRate, const std::vector< [Time](#) > ×, double volatility)
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

11.58.1 Detailed Description

This class implements a discrete geometric average strike asian option, with european exercise. The formula is from "Asian [Option](#)", E. Levy (1997) in "Exotic Options: The State of the Art", edited by L. Clewlow, C. Strickland, pag65-97

Todo:

add analytical greeks

The documentation for this class was generated from the following files:

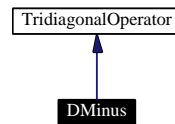
- [discretegeometricaso.hpp](#)
- [discretegeometricaso.cpp](#)

11.59 QuantLib::FiniteDifferences::DMinus Class Reference

D_- matricial representation.

```
#include <dminus.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::DMinus:



Public Methods

- **DMinus** ([Size](#) gridPoints, double h)

11.59.1 Detailed Description

The differential operator D_- discretizes the first derivative with the first-order formula

$$\frac{\partial u_i}{\partial x} \approx \frac{u_i - u_{i-1}}{h} = D_- u_i$$

The documentation for this class was generated from the following file:

- [dminus.hpp](#)

11.60 QuantLib::DoubleFormatter Class Reference

Formats doubles for output.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toString` (double x, int precision=6, int digits=0)

11.60.1 Detailed Description

Examples:

[DiscreteHedging.cpp](#).

The documentation for this class was generated from the following files:

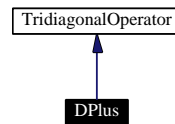
- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.61 QuantLib::FiniteDifferences::DPlus Class Reference

D_+ matricial representation.

```
#include <dplus.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::DPlus:



Public Methods

- **DPlus** ([Size](#) gridPoints, double h)

11.61.1 Detailed Description

The differential operator D_+ discretizes the first derivative with the first-order formula

$$\frac{\partial u_i}{\partial x} \approx \frac{u_{i+1} - u_i}{h} = D_+ u_i$$

The documentation for this class was generated from the following file:

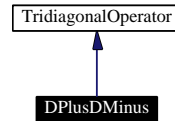
- [dplus.hpp](#)

11.62 QuantLib::FiniteDifferences::DPlusDMinus Class Reference

D_+D_- matricial representation.

```
#include <dplusdminus.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::DPlusDMinus:



Public Methods

- **DPlusDMinus** ([Size](#) gridPoints, double h)

11.62.1 Detailed Description

The differential operator D_+D_- discretizes the second derivative with the second-order formula

$$\frac{\partial^2 u_i}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = D_+D_-u_i$$

The documentation for this class was generated from the following file:

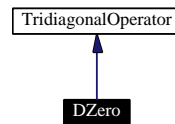
- [dplusdminus.hpp](#)

11.63 QuantLib::FiniteDifferences::DZero Class Reference

D_0 matricial representation.

```
#include <dzero.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::DZero:



Public Methods

- **DZero** ([Size](#) gridPoints, double h)

11.63.1 Detailed Description

The differential operator D_0 discretizes the first derivative with the second-order formula

$$\frac{\partial u_i}{\partial x} \approx \frac{u_{i+1} - u_{i-1}}{2h} = D_0 u_i$$

The documentation for this class was generated from the following file:

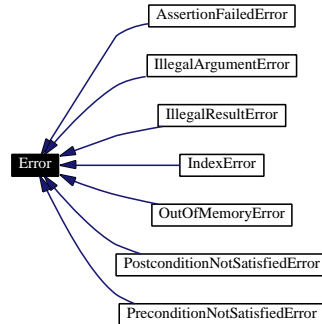
- [dzero.hpp](#)

11.64 QuantLib::Error Class Reference

Base error class.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::Error:



Public Methods

- **Error** (const std::string &what="")
- **~Error** () throw ()
- const char * **what** () const throw ()
returns the error message.

The documentation for this class was generated from the following file:

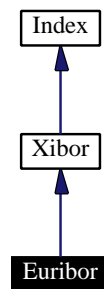
- [errors.hpp](#)

11.65 QuantLib::Indexes::Euribor Class Reference

Euribor index.

```
#include <euribor.hpp>
```

Inheritance diagram for QuantLib::Indexes::Euribor:



Public Methods

- **Euribor** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

- [euribor.hpp](#)

11.66 QuantLib::EuroFormatter Class Reference

Formats amounts in Euro for output.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toString` (double amount)

11.66.1 Detailed Description

Formatting follows Euro convention (x,xxx,xxx.xx)

The documentation for this class was generated from the following files:

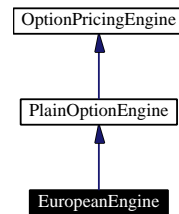
- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.67 QuantLib::Pricers::EuropeanEngine Class Reference

analytic pricing engine for European options.

```
#include <europeanengine.hpp>
```

Inheritance diagram for QuantLib::Pricers::EuropeanEngine:



Public Methods

- void **calculate** () const

The documentation for this class was generated from the following files:

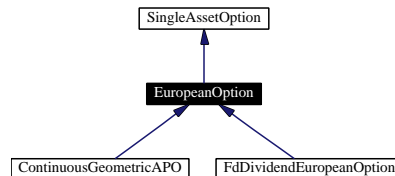
- [europeanengine.hpp](#)
- [europeanengine.cpp](#)

11.68 QuantLib::Pricers::EuropeanOption Class Reference

Black-Scholes-Merton European option.

```
#include <europeanoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::EuropeanOption:



Public Methods

- **EuropeanOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility)
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- double **vega** () const
- double **rho** () const
- double **dividendRho** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const
- void **setVolatility** (double newVolatility)
- void **setRiskFreeRate** ([Rate](#) newRate)
- void **setDividendYield** ([Rate](#) newDividendYield)

11.68.1 Detailed Description

Examples:

[DiscreteHedging.cpp](#), and [EuropeanOption.cpp](#).

The documentation for this class was generated from the following files:

- [europeanoption.hpp](#)
- [europeanoption.cpp](#)

11.69 QuantLib::MonteCarlo::EuropeanPathPricer Class Reference

path pricer for European options.

```
#include <europeanpathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::EuropeanPathPricer:



Public Methods

- **EuropeanPathPricer** (Option::Type type, double underlying, double strike, [DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [Path](#) &path) const

The documentation for this class was generated from the following files:

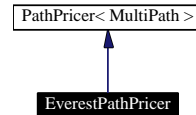
- [europeanpathpricer.hpp](#)
- [europeanpathpricer.cpp](#)

11.70 QuantLib::MonteCarlo::EverestPathPricer Class Reference

path pricer for European-type Everest option.

```
#include <everestpathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::EverestPathPricer:



Public Methods

- **EverestPathPricer** ([DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [MultiPath](#) &multiPath) const

11.70.1 Detailed Description

The payoff of an Everest option is given by the final-price initial-price ratio of the worst performer.

The documentation for this class was generated from the following files:

- [everestpathpricer.hpp](#)
- [everestpathpricer.cpp](#)

11.71 QuantLib::Exercise Class Reference

[Exercise](#) class (American, Bermudan or European).

```
#include <exercise.hpp>
```

Public Types

- enum **Type** { **American**, **Bermudan**, **European** }

Public Methods

- **Exercise** (Type type, const std::vector< [Date](#) > &dates)
- Type **type** () const
- [Date](#) **date** ([Size](#) index=0) const
- const std::vector< [Date](#) > & **dates** () const
- [RollingConvention](#) **rollingConvention** () const
- [Calendar](#) **calendar** () const
- int **settlementDays** () const

Protected Attributes

- Type **type_**
- std::vector< [Date](#) > **dates_**
- [Calendar](#) **calendar_**
- [RollingConvention](#) **convention_**
- int **settlementDays_**

The documentation for this class was generated from the following file:

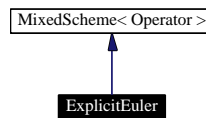
- [exercise.hpp](#)

11.72 QuantLib::FiniteDifferences::ExplicitEuler Class Template Reference

Forward Euler scheme for finite difference methods.

```
#include <expliciteuler.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::ExplicitEuler:



Friends

- class `FiniteDifferenceModel< ExplicitEuler< Operator > >`

11.72.1 Detailed Description

```
template<class Operator> class QuantLib::FiniteDifferences::ExplicitEuler< Operator >
```

See sect. [The finite differences framework](#) for details on the method.

In this implementation, the passed operator must be derived from either `TimeConstantOperator` or `TimeDependentOperator`. Also, it must implement at least the following interface:

```

typedef ... arrayType;

// copy constructor/assignment
// (these will be provided by the compiler if none is defined)
Operator(const Operator&);
Operator& operator=(const Operator&);

// inspectors
Size size();

// modifiers
void setTime(Time t);

// operator interface
arrayType applyTo(const arrayType&);
static Operator identity(Size size);

// operator algebra
Operator operator*(double, const Operator&);
Operator operator-(const Operator&, const Operator&);

```

Todo:

add Richardson extrapolation

The documentation for this class was generated from the following file:

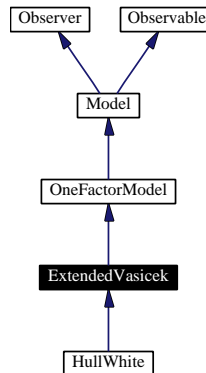
- [expliciteuler.hpp](#)

11.73 QuantLib::InterestRateModelling::ExtendedVasicek Class Reference

Extended Vasicek model class.

```
#include <hullwhite.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::ExtendedVasicek:



Public Methods

- **ExtendedVasicek** (const Parameter &a, const Parameter &sigma, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- virtual ~**ExtendedVasicek** ()
- virtual [Handle](#)< Lattices::Tree > **tree** (const [TimeGrid](#) &grid) const
Return a recombining tree.
- virtual [Handle](#)< ShortRateProcess > **process** () const
returns the driving stochastic equation.

Protected Methods

- virtual void **generateParameters** ()

Protected Attributes

- Parameter & **a_**
- Parameter & **sigma_**
- Parameter & **phi_**

11.73.1 Detailed Description

This class implements the extended Vasicek model defined by

$$dr_t = (\theta(t) - \alpha(t)r_t)dt + \sigma(t)dW_t.$$

It is actually implemented as $r_t = x_t + \varphi(t)$ where x_t is defined by

$$dx_t = -\alpha(t)r_t dt + \sigma(t)dW_t$$

The documentation for this class was generated from the following file:

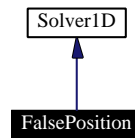
- [hullwhite.hpp](#)

11.74 QuantLib::Solvers1D::FalsePosition Class Reference

False position 1-D solver.

```
#include <falseposition.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::FalsePosition:



The documentation for this class was generated from the following files:

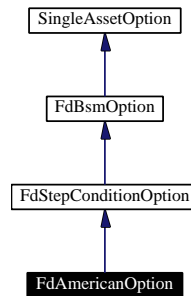
- [falseposition.hpp](#)
- [falseposition.cpp](#)

11.75 QuantLib::Pricers::FdAmericanOption Class Reference

American option.

```
#include <fdamericanoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::FdAmericanOption:



Public Methods

- **FdAmericanOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, int timeSteps, int gridPoints)
- void **initializeStepCondition** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

11.75.1 Detailed Description

Todo:

make american call with no dividends = european

The documentation for this class was generated from the following file:

- [fdamericanoption.hpp](#)

11.76 QuantLib::Pricers::FdBermudanOption Class Reference

Bermudan option.

```
#include <fdbermudanoption.hpp>
```

Public Methods

- **FdBermudanOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, const std::vector< [Time](#) > &dates=std::vector< [Time](#) >(), int timeSteps=100, int gridPoints=100)
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

Protected Methods

- void **initializeStepCondition** () const
- void **executeIntermediateStep** (int) const

Protected Attributes

- double **extraTermInBermudan**

The documentation for this class was generated from the following files:

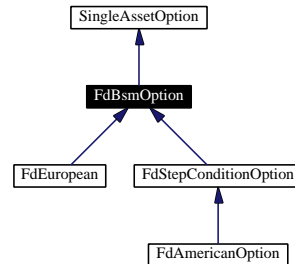
- [fdbermudanoption.hpp](#)
- [fdbermudanoption.cpp](#)

11.77 QuantLib::Pricers::FdBsmOption Class Reference

Black-Scholes-Merton option priced numerically.

```
#include <fdbsmoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::FdBsmOption:



Public Methods

- **FdBsmOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, [Size](#) gridPoints)
- virtual void **calculate** () const=0
- double **value** () const
- double **delta** () const
- double **gamma** () const
- double **theta** () const
- [Array](#) **getGrid** () const

Protected Methods

- virtual void **setGridLimits** (double center, double timeDelay) const
- virtual void **initializeGrid** () const
- virtual void **initializeInitialCondition** () const
- virtual void **initializeOperator** () const

Protected Attributes

- [Size](#) **gridPoints_**
- double **value_**
- double **delta_**
- double **gamma_**
- double **theta_**
- [Array](#) **grid_**
- [FiniteDifferences::BSMOperator](#) **finiteDifferenceOperator_**
- [Array](#) **initialPrices_**
- double **sMin_**
- double **center_**
- double **sMax_**

The documentation for this class was generated from the following files:

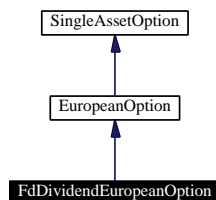
- [fdbsmoption.hpp](#)
- [fdbsmoption.cpp](#)

11.78 QuantLib::Pricers::FdDividendEuropeanOption Class Reference

European option with dividends.

```
#include <fddividendeuropeanoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::FdDividendEuropeanOption:



Public Methods

- **FdDividendEuropeanOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, const std::vector< double > ÷nds, const std::vector< [Time](#) > &exdivdates)
- double **theta** () const
- double **rho** () const
- double **dividendRho** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

The documentation for this class was generated from the following files:

- [fddividendeuropeanoption.hpp](#)
- [fddividendeuropeanoption.cpp](#)

11.79 QuantLib::Pricers::FdDividendShoutOption Class Reference

Shout option with dividends.

```
#include <fddividendshoutoption.hpp>
```

Public Methods

- **FdDividendShoutOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, const std::vector< double > ÷nds=std::vector< double >(), const std::vector< [Time](#) > &exdivdates=std::vector< [Time](#) >(), int timeSteps=100, int gridPoints=100)
- [Handle](#)< [SingleAssetOption](#) > **clone** () const
- double **dividendRho** () const

Protected Methods

- void **initializeStepCondition** () const

The documentation for this class was generated from the following files:

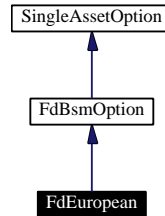
- [fddividendshoutoption.hpp](#)
- [fddividendshoutoption.cpp](#)

11.80 QuantLib::Pricers::FdEuropean Class Reference

Example of European option calculated using finite differences.

```
#include <fdEuropean.hpp>
```

Inheritance diagram for QuantLib::Pricers::FdEuropean:



Public Methods

- **FdEuropean** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) risk-FreeRate, [Time](#) residualTime, double volatility, [Size](#) timeSteps=200, [Size](#) gridPoints=800)
- [Array](#) **getPrices** () const
- [Handle](#)< [SingleAssetOption](#) > **clone** () const

Protected Methods

- void **calculate** () const

11.80.1 Detailed Description

Examples:

[EuropeanOption.cpp](#).

The documentation for this class was generated from the following files:

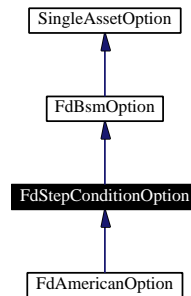
- [fdEuropean.hpp](#)
- [fdEuropean.cpp](#)

11.81 QuantLib::Pricers::FdStepConditionOption Class Reference

option executing additional code at each time step.

```
#include <fdstepconditionoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::FdStepConditionOption:



Protected Methods

- **FdStepConditionOption** (Option::Type type, double underlying, double strike, [Spread](#) dividend-Yield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility, int timeSteps, int gridPoints)
- void **calculate** () const
- virtual void **initializeStepCondition** () const=0

Protected Attributes

- [Handle](#)< [FiniteDifferences::StandardStepCondition](#) > **stepCondition_**
- int **timeSteps_**

The documentation for this class was generated from the following files:

- [fdstepconditionoption.hpp](#)
- [fdstepconditionoption.cpp](#)

11.82 QuantLib::Utilities::filtering_iterator Class Template Reference

Iterator filtering undesired data.

```
#include <filteringiterator.hpp>
```

Public Methods

- **filtering_iterator** (const Iterator &, const UnaryPredicate &, const Iterator &beforeBegin, const Iterator &end)

Dereferencing

- reference **operator** * () const
- pointer **operator** → () const

Increment and decrement

- filtering_iterator & **operator**++ ()
- filtering_iterator **operator**++ (int)
- filtering_iterator & **operator**-- ()
- filtering_iterator **operator**-- (int)

Comparisons

- bool **operator**== (const filtering_iterator< Iterator, UnaryPredicate > &)
- bool **operator**!= (const filtering_iterator< Iterator, UnaryPredicate > &)

Public Attributes

- typedef< Iterator >::pointer **pointer**
- typedef< Iterator >::reference **reference**

Related Functions

(Note that these are not member functions.)

- filtering_iterator< Iterator, UnaryPredicate > [make_filtering_iterator](#) (Iterator it, UnaryPredicate p, Iterator beforeBegin, Iterator end)
helper function to create filtering iterators.

11.82.1 Detailed Description

```
template<class Iterator, class UnaryPredicate> class QuantLib::Utilities::filtering_iterator< Iterator, UnaryPredicate >
```

This iterator advances an underlying iterator returning only those data satisfying a given condition.

The documentation for this class was generated from the following file:

- [filteringiterator.hpp](#)

11.83 QuantLib::FiniteDifferences::FiniteDifferenceModel Class Template Reference

Generic finite difference model.

```
#include <finitedifferencemodel.hpp>
```

Public Types

- typedef Evolver::arrayType **arrayType**
- typedef Evolver::operatorType **operatorType**

Public Methods

- **FiniteDifferenceModel** (const operatorType &L)
- void [rollback](#) (arrayType &a, [Time](#) from, [Time](#) to, [Size](#) steps, [Handle](#)< [StepCondition](#)< arrayType > > condition=[Handle](#)< [StepCondition](#)< arrayType > >())

11.83.1 Detailed Description

```
template<class Evolver> class QuantLib::FiniteDifferences::FiniteDifferenceModel< Evolver >
```

See sect. [The finite differences framework](#)

11.83.2 Member Function Documentation

11.83.2.1 void [rollback](#) (arrayType &a, [Time](#) from, [Time](#) to, [Size](#) steps, [Handle](#)< [StepCondition](#)< arrayType > > condition = [Handle](#)< [StepCondition](#)< arrayType > >())

solves the problem between the given times, possibly applying a condition at every step.

Warning:

being this a rollback, from must be a later time than to.

The documentation for this class was generated from the following file:

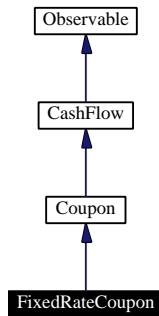
- [finitedifferencemodel.hpp](#)

11.84 QuantLib::CashFlows::FixedRateCoupon Class Reference

coupon paying a fixed interest rate.

```
#include <fixedratecoupon.hpp>
```

Inheritance diagram for QuantLib::CashFlows::FixedRateCoupon:



Public Methods

- **FixedRateCoupon** (double nominal, [Rate](#) rate, const [Calendar](#) &calendar, [RollingConvention](#) rollingConvention, const [DayCounter](#) &dayCounter, const [Date](#) &startDate, const [Date](#) &endDate, const [Date](#) &refPeriodStart=[Date](#)(), const [Date](#) &refPeriodEnd=[Date](#)())

CashFlow interface

- double [amount](#) () const
returns the amount of the cash flow.

Coupon interface

- double [accruedAmount](#) (const [Date](#) &) const
accrued amount at the given date.

Inspectors

- [Rate](#) [rate](#) () const

11.84.1 Member Function Documentation

11.84.1.1 double amount () const [inline, virtual]

Note:

The amount is not discounted, i.e., it is the actual amount paid at the cash flow date.

Implements [QuantLib::CashFlow](#).

The documentation for this class was generated from the following file:

- [fixedratecoupon.hpp](#)

11.85 QuantLib::CashFlows::FixedRateCouponVector Class Reference

helper class building a sequence of fixed rate coupons.

```
#include <cashflowvectors.hpp>
```

Public Methods

- **FixedRateCouponVector** (const std::vector< double > &nominals, const std::vector< [Rate](#) > &couponRates, const [Date](#) &startDate, const [Date](#) &endDate, int frequency, const [Calendar](#) &calendar, [RollingConvention](#) rollingConvention, bool isAdjusted, const [DayCounter](#) &dayCount, const [DayCounter](#) &firstPeriodDayCount, const [Date](#) &stubDate=[Date](#)())

The documentation for this class was generated from the following files:

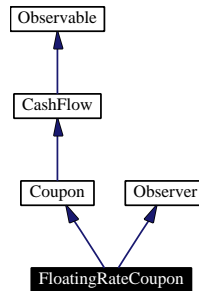
- [cashflowvectors.hpp](#)
- [cashflowvectors.cpp](#)

11.86 QuantLib::CashFlows::FloatingRateCoupon Class Reference

coupon at par on a term structure.

```
#include <floatingratecoupon.hpp>
```

Inheritance diagram for QuantLib::CashFlows::FloatingRateCoupon:



Public Methods

- **FloatingRateCoupon** (double nominal, const [Handle](#)< [Indexes::Xibor](#) > &index, const [Relinkable-Handle](#)< [TermStructure](#) > &termStructure, const [Date](#) &startDate, const [Date](#) &endDate, int fixing-Days, [Spread](#) spread=0.0, const [Date](#) &refPeriodStart=[Date](#)(), const [Date](#) &refPeriodEnd=[Date](#)())

CashFlow interface

- double [amount](#) () const
returns the amount of the cash flow.

Coupon interface

- double [accruedAmount](#) (const [Date](#) &) const
accrued amount at the given date.

Inspectors

- const [Handle](#)< [Indexes::Xibor](#) > & [index](#) () const
- int [fixingDays](#) () const
- [Rate](#) [fixing](#) () const
- [Spread](#) [spread](#) () const

Observer interface

- void [update](#) ()

11.86.1 Detailed Description

Warning:

This class does not perform any date adjustment, i.e., the start and end date passed upon construction should be already rolled to a business day.

11.86.2 Member Function Documentation

11.86.2.1 `double amount () const` [virtual]

Note:

The amount is not discounted, i.e., it is the actual amount paid at the cash flow date.

Implements [QuantLib::CashFlow](#).

11.86.2.2 `void update ()` [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following files:

- [floatingratecoupon.hpp](#)
- [floatingratecoupon.cpp](#)

11.87 QuantLib::CashFlows::FloatingRateCouponVector Class Reference

helper class building a sequence of floating rate coupons.

```
#include <cashflowvectors.hpp>
```

Public Methods

- **FloatingRateCouponVector** (const std::vector< double > &nominals, const [Date](#) &startDate, const [Date](#) &endDate, int frequency, const [Calendar](#) &calendar, [RollingConvention](#) rollingConvention, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure, const [Handle](#)< [Indexes::Xibor](#) > &index, int fixingDays, const std::vector< [Spread](#) > &spreads=std::vector< [Spread](#) >(), const [Date](#) &stubDate=[Date](#)())

11.87.1 Detailed Description

Warning:

The passing of a non-null stub date - i.e., the creation of a short/long first coupon - is currently disabled.

Todo:

A suitable algorithm should be implemented for the calculation of the interpolated index fixing for a short/long first coupon.

The documentation for this class was generated from the following file:

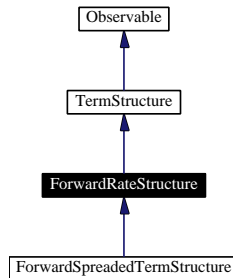
- [cashflowvectors.hpp](#)

11.88 QuantLib::ForwardRateStructure Class Reference

Forward rate term structure.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::ForwardRateStructure:



Public Methods

- virtual `~ForwardRateStructure()`

Protected Methods

- `Rate zeroYieldImpl (Time, bool extrapolate=false) const`
- `DiscountFactor discountImpl (Time, bool extrapolate=false) const`

11.88.1 Detailed Description

This abstract class acts as an adapter to [TermStructure](#) allowing the programmer to implement only the `forward(const Date&)` method in derived classes.

11.88.2 Member Function Documentation

11.88.2.1 `Rate zeroYieldImpl (Time, bool extrapolate = false) const` [inline, protected, virtual]

Returns the zero yield rate for the given date calculating it from the instantaneous forward rate.

Warning:

This is just a default, highly inefficient implementation. Derived classes should implement their own `zeroYield` method.

Implements [QuantLib::TermStructure](#).

Reimplemented in [QuantLib::ForwardSpreadedTermStructure](#).

11.88.2.2 `DiscountFactor discountImpl (Time, bool extrapolate = false) const` [inline, protected, virtual]

Returns the discount factor for the given date calculating it from the instantaneous forward rate.

Implements [QuantLib::TermStructure](#).

The documentation for this class was generated from the following file:

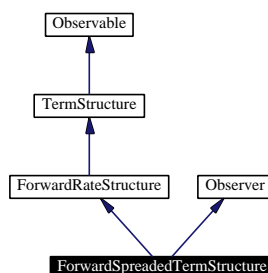
- [termstructure.hpp](#)

11.89 QuantLib::ForwardSpreadedTermStructure Class Reference

Term structure with an added spread on the instantaneous forward rate.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::ForwardSpreadedTermStructure:



Public Methods

- **ForwardSpreadedTermStructure** (const [RelinkableHandle](#)< [TermStructure](#) > &, const [RelinkableHandle](#)< [MarketElement](#) > &spread)

TermStructure interface

- [Currency](#) **currency** () const
returns the currency upon which the term structure is defined.
- [Date](#) **todayDate** () const
returns today's date.
- [int](#) **settlementDays** () const
returns the number of settlement days.
- [Calendar](#) **calendar** () const
returns the calendar for settlement calculation.
- [DayCounter](#) **dayCounter** () const
returns the day counter.
- [Date](#) **settlementDate** () const
returns the settlement date.
- [Date](#) **maxDate** () const
returns the latest date for which the curve can return rates.
- [Date](#) **minDate** () const
returns the earliest date for which the curve can return rates.
- [Time](#) **maxTime** () const
returns the latest date for which the curve can return rates.
- [Time](#) **minTime** () const

returns the earliest time for which the curve can return rates.

Observer interface

- void [update](#) ()

Protected Methods

- [Rate forwardImpl](#) ([Time](#), bool *extrapolate*=false) const
returns the spreaded forward rate.
- [Rate zeroYieldImpl](#) ([Time](#), bool *extrapolate*=false) const
returns the spreaded zero yield rate.

11.89.1 Detailed Description

Note:

This term structure will remain linked to the original structure, i.e., any changes in the latter will be reflected in this structure as well.

11.89.2 Member Function Documentation

11.89.2.1 void [update](#) () [inline, virtual]

This method must be implemented in derived classes. An instance of [Observer](#) does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

11.89.2.2 [Rate zeroYieldImpl](#) ([Time](#), bool *extrapolate* = false) const [inline, protected, virtual]

Warning:

This method must disappear should the spread become a curve

Reimplemented from [QuantLib::ForwardRateStructure](#).

The documentation for this class was generated from the following file:

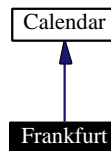
- [termstructure.hpp](#)

11.90 QuantLib::Calendars::Frankfurt Class Reference

Frankfurt calendar.

```
#include <frankfurt.hpp>
```

Inheritance diagram for QuantLib::Calendars::Frankfurt:



Public Methods

- **Frankfurt ()**

11.90.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st
- Good Friday
- Easter Monday
- Ascension Thursday
- Whit Monday
- Corpus Christi
- Labour Day, May 1st
- National Day, October 3rd
- Christmas Eve, December 24th
- Christmas, December 25th
- Boxing Day, December 26th
- New Year's Eve, December 31st

The documentation for this class was generated from the following file:

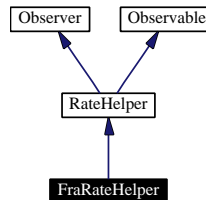
- [frankfurt.hpp](#)

11.91 QuantLib::TermStructures::FraRateHelper Class Reference

Forward rate agreement.

```
#include <ratehelpers.hpp>
```

Inheritance diagram for QuantLib::TermStructures::FraRateHelper:



Public Methods

- **FraRateHelper** (const [RelinkableHandle](#)< [MarketElement](#) > &rate, int settlementDays, int monthsToStart, int monthsToEnd, const [Calendar](#) &calendar, [RollingConvention](#) convention, const [DayCounter](#) &dayCounter)
- **FraRateHelper** (double rate, int settlementDays, int monthsToStart, int monthsToEnd, const [Calendar](#) &calendar, [RollingConvention](#) convention, const [DayCounter](#) &dayCounter)
- double **impliedQuote** () const
- [DiscountFactor](#) **discountGuess** () const
- void **setTermStructure** ([TermStructure](#) *)
sets the term structure to be used for pricing.
- [Date](#) **maturity** () const
maturity date.

11.91.1 Detailed Description

Warning:

This class assumes that today's date does not change between calls of [setTermStructure](#)().

Todo:

convexity adjustment should be implemented.

11.91.2 Member Function Documentation

11.91.2.1 void setTermStructure ([TermStructure](#) *) [virtual]

Warning:

Being a pointer and not a [Handle](#), the term structure is not guaranteed to remain allocated for the whole life of the rate helper. It is responsibility of the programmer to ensure that the pointer remains valid. It is advised that rate helpers be used only in term structure constructors, setting the term structure to **this**, i.e., the one being constructed.

Reimplemented from [QuantLib::TermStructures::RateHelper](#).

The documentation for this class was generated from the following files:

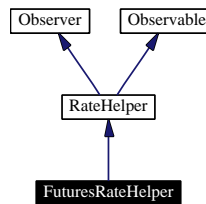
- [ratehelpers.hpp](#)
- [ratehelpers.cpp](#)

11.92 QuantLib::TermStructures::FuturesRateHelper Class Reference

Interest Rate Futures.

```
#include <ratehelpers.hpp>
```

Inheritance diagram for QuantLib::TermStructures::FuturesRateHelper:



Public Methods

- **FuturesRateHelper** (const [RelinkableHandle](#)< [MarketElement](#) > &price, const [Date](#) &ImmDate, int settlementDays, int nMonths, const [Calendar](#) &calendar, [RollingConvention](#) convention, const [DayCounter](#) &dayCounter)
- **FuturesRateHelper** (double price, const [Date](#) &ImmDate, int settlementDays, int nMonths, const [Calendar](#) &calendar, [RollingConvention](#) convention, const [DayCounter](#) &dayCounter)
- double **impliedQuote** () const
- [DiscountFactor](#) **discountGuess** () const
- [Date](#) **maturity** () const
maturity date.

11.92.1 Detailed Description

Warning:

This class assumes that today's date does not change between calls of [setTermStructure\(\)](#).

The documentation for this class was generated from the following files:

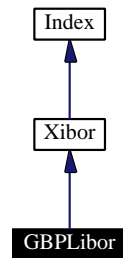
- [ratehelpers.hpp](#)
- [ratehelpers.cpp](#)

11.93 QuantLib::Indexes::GBPLibor Class Reference

GBP Libor index.

```
#include <gbplibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::GBPLibor:



Public Methods

- **GBPLibor** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

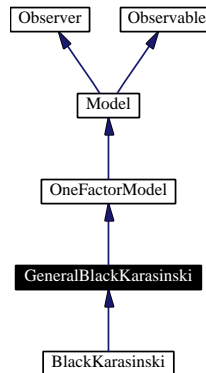
- [gbplibor.hpp](#)

11.94 QuantLib::InterestRateModelling::GeneralBlackKarasinski Class Reference

General Black-Karasinski model class.

```
#include <blackkarasinski.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::GeneralBlackKarasinski:



Public Methods

- **GeneralBlackKarasinski** (const Parameter &a, const Parameter &sigma, const [RelinkableHandle](#)<[TermStructure](#)> &termStructure)
- virtual ~**GeneralBlackKarasinski** ()
- virtual [Handle](#)< ShortRateProcess > [process](#) () const
returns the driving stochastic equation.
- virtual [Handle](#)< Lattices::Tree > [tree](#) (const [TimeGrid](#) &grid) const
Return a recombining tree.

Protected Methods

- virtual void **generateParameters** ()

Protected Attributes

- Parameter & **a_**
- Parameter & **sigma_**
- Parameter & **f_**

11.94.1 Detailed Description

This class implements the general Black-Karasinski model defined by

$$d \ln r_t = (\theta(t) - \alpha(t) \ln r_t) dt + \sigma(t) dW_t.$$

It is actually implemented as $r_t = e^{x_t + \varphi(t)}$ where x_t is defined by

$$dx_t = -\alpha(t)r_t dt + \sigma(t)dW_t.$$

The documentation for this class was generated from the following file:

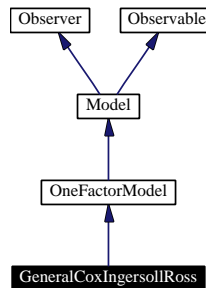
- [blackkarasinski.hpp](#)

11.95 QuantLib::InterestRateModelling::GeneralCoxIngersollRoss Class Reference

General single-factor extended Cox-Ingersoll-Ross model class.

```
#include <coxingersollross.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::GeneralCoxIngersollRoss:



Public Methods

- **GeneralCoxIngersollRoss** (const Parameter &theta, const Parameter &k, const Parameter &sigma, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- virtual ~**GeneralCoxIngersollRoss** ()
- virtual [Handle](#)< Lattices::Tree > **tree** (const [TimeGrid](#) &grid) const
Return a recombining tree.
- virtual [Handle](#)< ShortRateProcess > **process** () const
returns the driving stochastic equation.

Protected Methods

- virtual void **generateParameters** ()

Protected Attributes

- Parameter & **theta_**
- Parameter & **k_**
- Parameter & **sigma_**
- Parameter & **phi_**
- double **x0_**

11.95.1 Detailed Description

This class implements the extended Cox-Ingersoll-Ross model defined by

$$dr_t = (\theta(t) - \alpha(t)r_t)dt + \sqrt{r_t}\sigma(t)dW_t.$$

It is actually implemented as $r_t = \varphi(t) + y_t^2$ where y_t is defined by

$$dy_t = \left[\left(\frac{k\theta}{2} + \frac{\sigma^2}{8} \right) \frac{1}{y_t} - \frac{k}{2} y_t \right] dt + \frac{\sigma}{2} dW_t$$

The documentation for this class was generated from the following file:

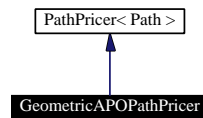
- [coxingersollross.hpp](#)

11.96 QuantLib::MonteCarlo::GeometricAOPathPricer Class Reference

path pricer for geometric average price option.

```
#include <geometricapopathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::GeometricAOPathPricer:



Public Methods

- **GeometricAOPathPricer** (Option::Type type, double underlying, double strike, [DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [Path](#) &path) const

The documentation for this class was generated from the following files:

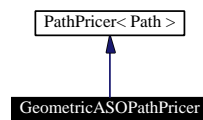
- [geometricapopathpricer.hpp](#)
- [geometricapopathpricer.cpp](#)

11.97 QuantLib::MonteCarlo::GeometricASOPathPricer Class Reference

path pricer for geometric average price option.

```
#include <geometricasopathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::GeometricASOPathPricer:



Public Methods

- **GeometricASOPathPricer** (Option::Type type, double underlying, [DiscountFactor](#) discount, bool useAntitheticVariance)
- double **operator()** (const [Path](#) &path) const

The documentation for this class was generated from the following files:

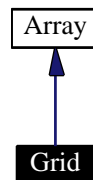
- [geometricasopathpricer.hpp](#)
- [geometricasopathpricer.cpp](#)

11.98 QuantLib::Grid Class Reference

spatial grid class.

```
#include <grid.hpp>
```

Inheritance diagram for QuantLib::Grid:



Public Methods

- **Grid** (double center, double dx, [Size](#) steps)

The documentation for this class was generated from the following file:

- [grid.hpp](#)

11.99 QuantLib::Handle Class Template Reference

Reference-counted pointer.

```
#include <handle.hpp>
```

Public Methods

constructors, destructor, and assignment

- [Handle](#) (T *ptr=0, bool owns=true)
Constructor taking a pointer.
- `template<class U> Handle (const Handle< U > &from)`
- `Handle (const Handle &from)`
- `~Handle ()`
- `template<class U> Handle & operator= (const Handle< U > &from)`
- `Handle & operator= (const Handle &from)`

Dereferencing

- `T & operator * () const`
- `T * operator → () const`

Inspectors

- `bool isNull () const`
Checks if the contained pointer is actually allocated.
- `bool shareSameObject (const Handle< T > &) const`
Checks if the two handles point to the same object.

Friends

- class `HandleCopier`

11.99.1 Detailed Description

```
template<class T> class QuantLib::Handle< T >
```

This class acts as a proxy to a pointer contained in it. Such pointer is owned by the handle, i.e., the handle will be responsible for its deletion, unless explicitly stated by the programmer.

A count of the references to the contained pointer is incremented every time a handle is copied, and decremented every time a handle is deleted or goes out of scope. When the handle owns the pointer, this mechanism ensures on one hand, that the pointer will not be deallocated as long as a handle refers to it, and on the other hand, that it will be deallocated when no more handles do.

Note:

The implementation of this class was originally taken from "The C++ Programming Language", 3rd ed., B.Stroustrup, Addison-Wesley, 1997.

Warning:

This mechanism will break and result in untimely deallocation of the pointer (and very possible death of your executable) if two handles are explicitly initialized with the same pointer, as in

```
SomeObj* so = new SomeObj;
Handle<SomeObj> h1(so);
Handle<SomeObj> h2 = h1;    // this is safe.
Handle<SomeObj> h3(so);    // this is definitely not.
```

It is good practice to create the pointer and immediately pass it to the handle, as in

```
Handle<SomeObj> h1(new SomeObj);    // this is as safe as can be.
```

Warning:

When the programmer keeps the ownership of the pointer, as explicitly declared in

```
SomeObj so;
Handle<SomeObj> h(&so, false);
```

it is responsibility of the programmer to make sure that the object remain in scope as long as there are handles pointing to it. Also, the programmer must explicitly delete the object if required.

Examples:

[DiscreteHedging.cpp](#).

11.99.2 Constructor & Destructor Documentation

11.99.2.1 Handle (T **ptr* = 0, bool *owns* = true) [inline, explicit]

If **owns** is set to **true** (the default), the handle will be responsible for the deletion of the pointer. If it is set to **false**, the programmer must make sure that the pointed object remains in scope for the lifetime of the handle and its copies. Destruction of the object is also responsibility of the programmer.

It is advised that handles be used with **owns = false** only in a controlled and self-contained environment. Such a case happens when an object needs to pass a handle to itself to inner classes or bootstrappers - i.e., contained or temporary objects whose lifetime is guaranteed not to last more than the lifetime of the object.

The documentation for this class was generated from the following file:

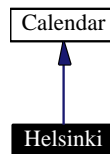
- [handle.hpp](#)

11.100 QuantLib::Calendars::Helsinki Class Reference

Helsinki calendar.

```
#include <helsinki.hpp>
```

Inheritance diagram for QuantLib::Calendars::Helsinki:



Public Methods

- **Helsinki ()**

11.100.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st
- Epiphany, January 6th
- Good Friday
- Easter Monday
- Ascension Thursday
- Labour Day, May 1st
- Midsummer Eve, June 21st
- Independence Day, December 6th
- Christmas Eve, December 24th
- Christmas, December 25th
- Boxing Day, December 26th

Note:

The holiday rules for [Wellington](http://www.jrefinery.com/ibd/) were documented by Veli-Pekka Mattila for IDB (<http://www.jrefinery.com/ibd/>)

The documentation for this class was generated from the following file:

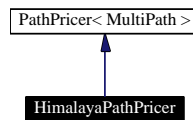
- [helsinki.hpp](#)

11.101 QuantLib::MonteCarlo::HimalayaPathPricer Class Reference

multipath pricer for European-type Himalaya option.

```
#include <himalayapathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::HimalayaPathPricer:



Public Methods

- **HimalayaPathPricer** (const [Array](#) &underlying, double strike, [DiscountFactor](#) discount, bool use-AntitheticVariance)
- double **operator()** (const [MultiPath](#) &multiPath) const

11.101.1 Detailed Description

The payoff of an himalaya option is computed in the following way: given a basket of N assets, and M time periods, at end of each period the option who performed the best is added to the average and then discarded from the basket. At the end of the M periods the option pays the max between the strike and the average of the best performers.

The documentation for this class was generated from the following files:

- [himalayapathpricer.hpp](#)
- [himalayapathpricer.cpp](#)

11.102 QuantLib::History Class Reference

Container for historical data.

```
#include <history.hpp>
```

Public Types

- typedef [Utilities::filtering_iterator](#)< [const_iterator](#), DataValidator > [const_valid_iterator](#)
bidirectional iterator on non-null history entries.
- typedef std::vector< double >::[const_iterator](#) [const_data_iterator](#)
random access iterator on historical data.
- typedef [Utilities::filtering_iterator](#)< [const_data_iterator](#), DataValidator > [const_valid_data_iterator](#)
bidirectional iterator on non-null historical data.

Public Methods

- [History](#) ()
- template<class Iterator> [History](#) (const [Date](#) &firstDate, const [Date](#) &lastDate, Iterator begin, Iterator end)
- [History](#) (const [Date](#) &firstDate, const std::vector< double > &values)
- [History](#) (const [Date](#) &firstDate, const [Date](#) &lastDate, const std::vector< double > &values)
- [History](#) (const std::vector< [Date](#) > &dates, const std::vector< double > &values)

Inspectors

- const [Date](#) & [firstDate](#) () const
returns the first date for which a historical datum exists.
- const [Date](#) & [lastDate](#) () const
returns the last date for which a historical datum exists.
- int [size](#) () const
returns the number of historical data including null ones.

Historical data access

- double [operator\[\]](#) (const [Date](#) &) const
returns the (possibly null) datum corresponding to the given date.

Iterator access

Four different types of iterators are provided, namely, [const_iterator](#), [const_valid_iterator](#), [const_data_iterator](#), and [const_valid_data_iterator](#).

[const_iterator](#) and [const_valid_iterator](#) point to an [Entry](#) structure, the difference being that the latter only iterates over valid entries - i.e., entries whose data are not null. The same difference exists between [const_data_iterator](#) and [const_valid_data_iterator](#) which point directly to historical values without reference to the date they are associated to.

- `const_iterator begin () const`
- `const_iterator end () const`
- `const_iterator iterator (const Date &d) const`
- `const_valid_iterator vbegin () const`
- `const_valid_iterator vend () const`
- `const_valid_iterator valid_iterator (const Date &d) const`
- `const_data_iterator dbegin () const`
- `const_data_iterator dend () const`
- `const_data_iterator data_iterator (const Date &d) const`
- `const_valid_data_iterator vdbegin () const`
- `const_valid_data_iterator vdend () const`
- `const_valid_data_iterator valid_data_iterator (const Date &d) const`

11.102.1 Detailed Description

This class acts as a generic repository for a set of historical data. Single data can be accessed through their date, while sets of consecutive data can be accessed through iterators.

A history can contain null data, which can either be returned or skipped according to the chosen iterator type.

Example: [uses of history iterators](#)

11.102.2 Constructor & Destructor Documentation

11.102.2.1 History () [inline]

Default constructor

11.102.2.2 History (const Date &firstDate, const Date &lastDate, Iterator begin, Iterator end) [inline]

This constructor initializes the history with the given set of values, corresponding to the date range between *firstDate* and *lastDate* included.

Precondition:

begin-end must equal the number of days from *firstDate* to *lastDate* included.

11.102.2.3 History (const Date &firstDate, const Date &lastDate, const std::vector< double > &values) [inline]

This constructor initializes the history with the given set of values, corresponding to the date range between *firstDate* and *lastDate* included.

Precondition:

The size of *values* must equal the number of days from *firstDate* to *lastDate* included.

11.102.2.4 History (const std::vector< [Date](#) > & *dates*, const std::vector< double > & *values*) [inline]

This constructor initializes the history with the given set of values, corresponding each to the element with the same index in the given set of dates. The whole date range between *dates*[0] and *dates*[N-1] will be automatically filled by inserting null values where a date is missing from the given set.

Precondition:

dates must be sorted.

There can be no pairs (*dates*[i],*values*[i]) and (*dates*[j],*values*[j]) such that *dates*[i] == *dates*[j] && *values*[i] != *values*[j]. Pairs with *dates*[i] == *dates*[j] && *values*[i] == *values*[j] are allowed; the duplicated entries will be discarded.

The size of *values* must equal the number of days from *firstDate* to *lastDate* included.

The documentation for this class was generated from the following file:

- [history.hpp](#)

11.103 QuantLib::History::const_iterator Class Reference

random access iterator on history entries.

```
#include <history.hpp>
```

Public Types

- typedef [Entry](#) **value_type**
- typedef int **difference_type**
- typedef const [Entry](#) * **pointer**
- typedef const [Entry](#) & **reference**

Public Methods

Dereferencing

- reference **operator** * () const
- pointer **operator** → () const

Random access

- value_type **operator**[] (difference_type i) const

Increment and decrement

- const_iterator & **operator**++ ()
- const_iterator **operator**++ (int)
- const_iterator & **operator**-- ()
- const_iterator **operator**-- (int)
- const_iterator & **operator**+= (difference_type i)
- const_iterator & **operator**-= (difference_type i)
- const_iterator **operator**+ (difference_type i)
- const_iterator **operator**- (difference_type i)

Difference

- difference_type **operator**- (const const_iterator &i)

Comparisons

- bool **operator**== (const const_iterator &i)
- bool **operator**!= (const const_iterator &i)
- bool **operator**< (const const_iterator &i)
- bool **operator**> (const const_iterator &i)
- bool **operator**<= (const const_iterator &i)
- bool **operator**>= (const const_iterator &i)

Friends

- class **History**

The documentation for this class was generated from the following file:

- [history.hpp](#)

11.104 QuantLib::History::Entry Class Reference

single datum in history.

```
#include <history.hpp>
```

Public Methods

- const [Date](#) & **date** () const
- double **value** () const

Friends

- class **const_iterator**

The documentation for this class was generated from the following file:

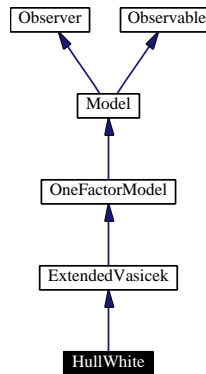
- [history.hpp](#)

11.105 QuantLib::InterestRateModelling::HullWhite Class Reference

Analytically tractable single-factor Hull-White model class.

```
#include <hullwhite.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::HullWhite:



Public Methods

- **HullWhite** (const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- virtual \sim **HullWhite** ()
- virtual [Handle](#)< Lattices::Tree > **tree** (const [TimeGrid](#) &grid) const
Return a recombining tree.
- double **discountBond** ([Time](#) T, [Time](#) s, [Rate](#) r) const
- double **discountBondOption** (Option::Type type, double strike, [Time](#) maturity, [Time](#) bondMaturity) const

Protected Methods

- virtual void **generateParameters** ()

11.105.1 Detailed Description

This class implements the standard single-factor Hull-White model defined by

$$dr_t = (\theta(t) - \alpha r_t)dt + \sigma dW_t$$

where α and σ are constants. $\varphi(t)$ is analytically defined by

$$\varphi(t) = f(t) + \frac{1}{2} \left[\frac{\sigma(1 - e^{-at})}{a} \right]^2$$

There is an analytical formula for discount bonds:

$$P(t, T, r_t) =$$

The documentation for this class was generated from the following files:

- [hullwhite.hpp](#)
- [hullwhite.cpp](#)

11.106 QuantLib::RandomNumbers::ICGaussianRng Class Template Reference

Inverse cumulative Gaussian random number generator.

```
#include <inversecumgaussianrng.hpp>
```

Public Types

- typedef [MonteCarlo::Sample](#)< double > **sample_type**

Public Methods

- **ICGaussianRng** (long seed=0)
- **sample_type next** () const
returns next sample from the Gaussian distribution.

11.106.1 Detailed Description

```
template<class U> class QuantLib::RandomNumbers::ICGaussianRng< U >
```

It uses a uniform deviate in (0, 1) as the source of cumulative normal distribution values. Then an inverse cumulative normal distribution is used as it is approximately a Gaussian deviate with average 0.0 and standard deviation 1.0.

The uniform deviate is supplied by U.

Class U must implement the following interface:

```
U::U(long seed);  
U::sample_type U::next() const;
```

The documentation for this class was generated from the following file:

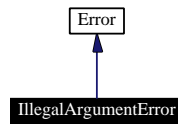
- [inversecumgaussianrng.hpp](#)

11.107 QuantLib::IllegalArgumentError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::IllegalArgumentError:



Public Methods

- `IllegalArgumentError` (const std::string &what=“”)

11.107.1 Detailed Description

Thrown upon passing an argument with an illegal value.

The documentation for this class was generated from the following file:

- [errors.hpp](#)

11.108 QuantLib::IllegalResultError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::IllegalResultError:



Public Methods

- `IllegalResultError` (const std::string &what=“”)

11.108.1 Detailed Description

Thrown upon obtaining a result outside the allowed range.

The documentation for this class was generated from the following file:

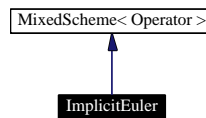
- [errors.hpp](#)

11.109 QuantLib::FiniteDifferences::ImplicitEuler Class Template Reference

Backward Euler scheme for finite difference methods.

```
#include <impliciteuler.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::ImplicitEuler:



Friends

- class `FiniteDifferenceModel< ImplicitEuler< Operator > >`

11.109.1 Detailed Description

```
template<class Operator> class QuantLib::FiniteDifferences::ImplicitEuler< Operator >
```

See sect. [The finite differences framework](#) for details on the method.

In this implementation, the passed operator must be derived from either `TimeConstantOperator` or `TimeDependentOperator`. Also, it must implement at least the following interface:

```

typedef ... arrayType;

// copy constructor/assignment
// (these will be provided by the compiler if none is defined)
Operator(const Operator&);
Operator& operator=(const Operator&);

// inspectors
Size size();

// modifiers
void setTime(Time t);

// operator interface
arrayType solveFor(const arrayType&);
static Operator identity(Size size);

// operator algebra
Operator operator*(double, const Operator&);
Operator operator+(const Operator&, const Operator&);

```

The documentation for this class was generated from the following file:

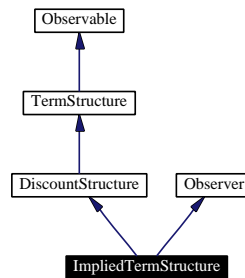
- [impliciteuler.hpp](#)

11.110 QuantLib::ImpliedTermStructure Class Reference

Implied term structure at a given date in the future.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::ImpliedTermStructure:



Public Methods

- **ImpliedTermStructure** (const [RelinkableHandle](#)< [TermStructure](#) > &, const [Date](#) &todayDate)

TermStructure interface

- [Currency](#) **currency** () const
returns the currency upon which the term structure is defined.
- [Date](#) **todayDate** () const
returns today's date.
- int **settlementDays** () const
returns the number of settlement days.
- [Calendar](#) **calendar** () const
returns the calendar for settlement calculation.
- [DayCounter](#) **dayCounter** () const
returns the day counter.
- [Date](#) **settlementDate** () const
returns the settlement date.
- [Date](#) **maxDate** () const
returns the latest date for which the curve can return rates.
- [Date](#) **minDate** () const
returns the earliest date for which the curve can return rates.
- [Time](#) **maxTime** () const
returns the latest date for which the curve can return rates.
- [Time](#) **minTime** () const

returns the earliest time for which the curve can return rates.

Observer interface

- void [update](#) ()

Protected Methods

- [DiscountFactor discountImpl](#) ([Time](#), bool extrapolate=false) const
returns the discount factor as seen from the evaluation date.

11.110.1 Detailed Description

The given date will be the implied today's date.

Note:

This term structure will remain linked to the original structure, i.e., any changes in the latter will be reflected in this structure as well.

11.110.2 Member Function Documentation

11.110.2.1 void [update](#) () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

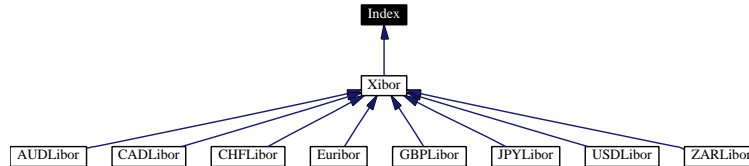
- [termstructure.hpp](#)

11.111 QuantLib::Index Class Reference

purely virtual base class for indexes.

```
#include <index.hpp>
```

Inheritance diagram for QuantLib::Index:



Public Methods

- virtual `~Index ()`
- virtual `std::string name () const=0`
Returns the name of the index.
- virtual `Rate fixing (const Date &fixingDate) const=0`
returns the fixing at the given date.

11.111.1 Member Function Documentation

11.111.1.1 virtual `std::string name () const` [pure virtual]

Warning:

This method is used for output and comparison between indexes. It is **not** meant to be used for writing switch-on-type code.

Implemented in [QuantLib::Indexes::Xibor](#).

11.111.1.2 virtual `Rate fixing (const Date &fixingDate) const` [pure virtual]

Note:

any date passed as arguments must be a value date, i.e., the real calendar date advanced by a number of settlement days.

Implemented in [QuantLib::Indexes::Xibor](#).

The documentation for this class was generated from the following file:

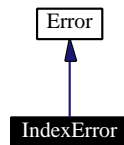
- [index.hpp](#)

11.112 QuantLib::IndexError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::IndexError:



Public Methods

- **IndexError** (const std::string &what=“”)

11.112.1 Detailed Description

Thrown upon accessing an array or container outside its range.

The documentation for this class was generated from the following file:

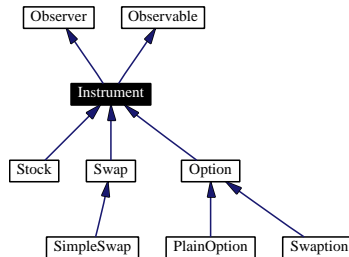
- [errors.hpp](#)

11.113 QuantLib::Instrument Class Reference

Abstract instrument class.

```
#include <instrument.hpp>
```

Inheritance diagram for QuantLib::Instrument:



Calculations

These methods do not modify the structure of the instrument and are therefore declared as `const`. Temporary variables will be declared as mutable.

- void `recalculate` ()
- void `calculate` () const
- virtual void `performCalculations` () const=0

Public Methods

- **Instrument** (const std::string &isinCode="", const std::string &description="")
- virtual ~**Instrument** ()

Inspectors

- std::string `isinCode` () const
returns the ISIN code of the instrument, when given.
- std::string `description` () const
returns a brief textual description of the instrument.
- double `NPV` () const
returns the net present value of the instrument.
- bool `isExpired` () const
returns whether the instrument is still tradable.

Observer interface

- void `update` ()

Protected Attributes

Results

*The value of these attributes must be set in the body of the **performCalculations** method.*

- double **NPV_**
- bool **isExpired_**

11.113.1 Detailed Description

This class is purely abstract and defines the interface of concrete instruments which will be derived from this one.

11.113.2 Member Function Documentation

11.113.2.1 **void update ()** [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

11.113.2.2 **void recalculate ()** [inline]

This method force the recalculation of the instrument value and other results which would otherwise be cached. It is not declared as const since it needs to call the non-const **notifyObservers** method.

Note:

Explicit invocation of this method is **not** necessary if the instrument registered itself as observer with the structures on which such results depend. It is strongly advised to follow this policy when possible.

11.113.2.3 **void calculate () const** [inline, protected]

This method performs all needed calculations by calling the **performCalculations** method.

Warning:

[Instruments](#) cache the results of the previous calculation. Such results will be returned upon later invocations of **calculate**. When the results depend on parameters such as term structures which could change between invocations, the instrument must register itself as observer of such objects for the calculations to be performed again when they change.

Warning:

This method should **not** be redefined in derived classes.

11.113.2.4 **virtual void performCalculations () const** [protected, pure virtual]

This method must implement any calculations which must be (re)done in order to calculate the NPV of the instrument.

Implemented in [QuantLib::Instruments::PlainOption](#).

The documentation for this class was generated from the following file:

- [instrument.hpp](#)

11.114 QuantLib::IntegerFormatter Class Reference

Formats integers for output.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toString (int i, int digits=0)`

The documentation for this class was generated from the following files:

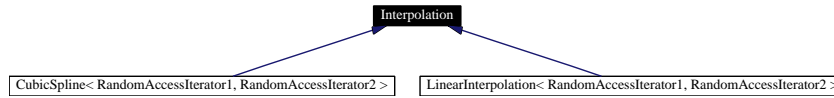
- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.115 QuantLib::Math::Interpolation Class Template Reference

abstract base class for 1-D interpolations.

```
#include <interpolation.hpp>
```

Inheritance diagram for QuantLib::Math::Interpolation:



Public Methods

- **Interpolation** (const RandomAccessIterator1 &xBegin, const RandomAccessIterator1 &xEnd, const RandomAccessIterator2 &yBegin, bool allowExtrapolation)
- virtual result_type [operator\(\)](#) (const argument_type &x) const=0

Public Attributes

- typedef< RandomAccessIterator1 >::value_type **argument_type**
- typedef< RandomAccessIterator2 >::value_type **result_type**

Protected Attributes

- RandomAccessIterator1 **xBegin_**
- RandomAccessIterator1 **xEnd_**
- RandomAccessIterator2 **yBegin_**
- bool **allowExtrapolation_**

11.115.1 Detailed Description

```
template<class RandomAccessIterator1, class RandomAccessIterator2> class QuantLib::Math::Interpolation< RandomAccessIterator1, RandomAccessIterator2 >
```

Classes derived from this class will override operator() to provide interpolated values from two sequences of equal length, representing discretized values of a variable and a function of the former, respectively.

11.115.2 Member Function Documentation

11.115.2.1 virtual result_type operator() (const argument_type &x) const [pure virtual]

This operator must be overridden to provide an implementation of the actual interpolation.

Precondition:

The sequence of values for x must have been sorted for the result to make sense.

Implemented in [QuantLib::Math::CubicSpline](#).

The documentation for this class was generated from the following file:

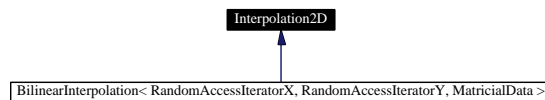
- [interpolation.hpp](#)

11.116 QuantLib::Math::Interpolation2D Class Template Reference

abstract base class for 2-D interpolations.

```
#include <interpolation2D.hpp>
```

Inheritance diagram for QuantLib::Math::Interpolation2D:



Public Types

- typedef double **result_type**

Public Methods

- **Interpolation2D** (const RandomAccessIteratorX &xBegin, const RandomAccessIteratorX &xEnd, const RandomAccessIteratorY &yBegin, const RandomAccessIteratorY &yEnd, const MatricialData &data, bool allowExtrapolation)
- virtual ~**Interpolation2D** ()
- virtual double **operator()** (const first_argument_type &x, const second_argument_type &y) const=0

Public Attributes

- typedef< RandomAccessIteratorX >::value_type **first_argument_type**
- typedef< RandomAccessIteratorY >::value_type **second_argument_type**

Protected Attributes

- RandomAccessIteratorX **xBegin_**
- RandomAccessIteratorX **xEnd_**
- RandomAccessIteratorY **yBegin_**
- RandomAccessIteratorY **yEnd_**
- const MatricialData & **data_**
- bool **allowExtrapolation_**

11.116.1 Detailed Description

```
template<class RandomAccessIteratorX, class RandomAccessIteratorY, class MatricialData> class
QuantLib::Math::Interpolation2D< RandomAccessIteratorX, RandomAccessIteratorY, Matricial-
Data >
```

Classes derived from this class will override operator() to provide interpolated values from two sequences of length N and M, representing the discretized values of the x,y variables, and a NxM matrix representing the function tabulated z values.

Todo:

Bicubic interpolation and bicubic spline

11.116.2 Member Function Documentation**11.116.2.1 virtual double operator() (const first_argument_type & x, const second_argument_type & y) const** [pure virtual]

This operator must be overridden to provide an implementation of the actual interpolation.

Precondition:

The sequence of values for x must have been sorted for the result to make sense.

Implemented in [QuantLib::Math::BilinearInterpolation](#).

The documentation for this class was generated from the following file:

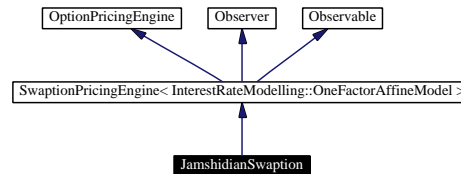
- [interpolation2D.hpp](#)

11.117 QuantLib::Pricers::JamshidianSwaption Class Reference

Jamshidian swaption pricer.

```
#include <jamshidianswaption.hpp>
```

Inheritance diagram for QuantLib::Pricers::JamshidianSwaption:



Public Methods

- **JamshidianSwaption** (const [Handle](#)< InterestRateModelling::OneFactorAffineModel > &mdl)
- void **calculate** () const

Friends

- class **rStarFinder**

The documentation for this class was generated from the following files:

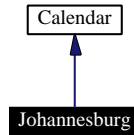
- [jamshidianswaption.hpp](#)
- [jamshidianswaption.cpp](#)

11.118 QuantLib::Calendars::Johannesburg Class Reference

Johannesburg calendar.

```
#include <johannesburg.hpp>
```

Inheritance diagram for QuantLib::Calendars::Johannesburg:



Public Methods

- **Johannesburg ()**

11.118.1 Detailed Description

Holidays:

- Sundays
- New Year's Day, January 1st (possibly moved to Monday)
- Good Friday
- Family Day, Easter Monday
- Human Rights Day, March 21st (possibly moved to Monday)
- Freedom Day, April 27th (possibly moved to Monday)
- Workers Day, May 1st (possibly moved to Monday)
- Youth Day, June 16th (possibly moved to Monday)
- National Women's Day, August 9th (possibly moved to Monday)
- Heritage Day, September 24th (possibly moved to Monday)
- Day of Reconciliation, December 16th (possibly moved to Monday)
- Christmas December 25th
- Day of Goodwill December 26th (possibly moved to Monday)

The documentation for this class was generated from the following file:

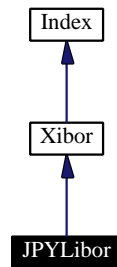
- [johannesburg.hpp](#)

11.119 QuantLib::Indexes::JPYLibor Class Reference

JPY Libor index (Also known as TIBOR, check settlement days).

```
#include <jpylibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::JPYLibor:



Public Methods

- **JPYLibor** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

- [jpylibor.hpp](#)

11.120 QuantLib::RandomNumbers::KnuthUniformRng Class Reference

Uniform random number generator.

```
#include <knuthuniformrng.hpp>
```

Public Types

- typedef [MonteCarlo::Sample](#)< double > **sample_type**

Public Methods

- [KnuthUniformRng](#) (long seed=0)
- **sample_type** [next](#) () const

11.120.1 Detailed Description

Random number generator by Knuth. For more details see Knuth, Seminumerical Algorithms, 3rd edition, Section 3.6.

Note:

This is **not** Knuth's original implementation which is available at <http://www-cs-faculty.stanford.edu/~knuth/programs.html>, but rather a slightly modified version wrapped in a C++ class. Such modifications did not affect the code but only the data structures used, which were converted in their C++/STL equivalents.

11.120.2 Constructor & Destructor Documentation

11.120.2.1 KnuthUniformRng (long *seed* = 0) [explicit]

if the given seed is 0, a random seed will be chosen based on clock()

11.120.3 Member Function Documentation

11.120.3.1 KnuthUniformRng::sample_type next () const [inline]

returns a sample with weight 1.0 containing a random number uniformly chosen from (0.0,1.0)

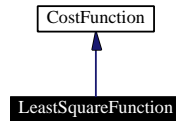
The documentation for this class was generated from the following files:

- [knuthuniformrng.hpp](#)
- [knuthuniformrng.cpp](#)

11.121 QuantLib::Optimization::LeastSquareFunction Class Reference

```
#include <leastsquare.hpp>
```

Inheritance diagram for QuantLib::Optimization::LeastSquareFunction:



Public Methods

- [LeastSquareFunction](#) (LeastSquareProblem &lsp)
Default constructor.
- virtual [~LeastSquareFunction](#) ()
Destructor.
- virtual double [value](#) (const [Array](#) &x)
compute value of the least square function.
- virtual void [gradient](#) ([Array](#) &grad.f, const [Array](#) &x)
compute vector of derivatives of the least square function.
- virtual double [valueAndGradient](#) ([Array](#) &grad.f, const [Array](#) &x)
compute value and vector of derivatives of the least square function.

Protected Attributes

- LeastSquareProblem & [lsp_](#)
least square problem.

11.121.1 Detailed Description

Design a least square function as a cost function using the interface provided by LeastSquareProblem class. [Array](#) vector class requires function DotProduct() that computes dot product and - operator M matrix class requires function transpose() that computes transpose and * operator with vector class

The documentation for this class was generated from the following file:

- [leastsquare.hpp](#)

11.122 QuantLib::RandomNumbers::LecuyerUniformRng Class Reference

Uniform random number generator.

```
#include <lecuyeruniformrng.hpp>
```

Public Types

- typedef [MonteCarlo::Sample](#)< double > **sample_type**

Public Methods

- [LecuyerUniformRng](#) (long seed=0)
- sample_type [next](#) () const

11.122.1 Detailed Description

Random number generator of L'Ecuyer with added Bays-Durham shuffle. For more details see Section 7.1 of Numerical Recipes in C, 2nd Edition, Cambridge University Press (available at <http://www.nr.com/>)

11.122.2 Constructor & Destructor Documentation

11.122.2.1 LecuyerUniformRng (long seed = 0) [inline, explicit]

if the given seed is 0, a random seed will be chosen based on clock()

11.122.3 Member Function Documentation

11.122.3.1 LecuyerUniformRng::sample_type next () const [inline]

returns a sample with weight 1.0 containing a random number uniformly chosen from (0.0,1.0)

The documentation for this class was generated from the following files:

- [lecuyeruniformrng.hpp](#)
- [lecuyeruniformrng.cpp](#)

11.123 QuantLib::Math::LexicographicalView Class Template Reference

Lexicographical 2-D view of a contiguous set of data.

```
#include <lexicographicalview.hpp>
```

Public Types

- typedef RandomAccessIterator [x_iterator](#)
iterates over v_{ij} with j fixed.
- typedef [Utilities::stepping_iterator](#)< RandomAccessIterator > [y_iterator](#)
iterates over v_{ij} with i fixed.

Public Methods

- [LexicographicalView](#) (const RandomAccessIterator &begin, const RandomAccessIterator &end, int xSize)
attaches the view with the given dimension to a sequence.
- typedef [QL_REVERSE_ITERATOR](#) (RandomAccessIterator, typename 1< RandomAccessIterator >::value_type) reverse_x_iterator
iterates backwards over v_{ij} with j fixed.
- typedef [QL_REVERSE_ITERATOR](#) ([y_iterator](#), typename 1< RandomAccessIterator >::value_type) reverse_y_iterator
iterates backwards over v_{ij} with i fixed.

Element access

- [y_iterator](#) operator[] (int i)

Iterator access

- [x_iterator](#) **xbegin** (int j)
- [x_iterator](#) **xend** (int j)
- reverse_x_iterator **rxbegin** (int j)
- reverse_x_iterator **rxend** (int j)
- [y_iterator](#) **ybegin** (int i)
- [y_iterator](#) **yend** (int i)
- reverse_y_iterator **rybegin** (int i)
- reverse_y_iterator **ryend** (int i)

Inspectors

- int [xSize](#) () const
dimension of the array along x.
- int [ySize](#) () const
dimension of the array along y.

11.123.1 Detailed Description

template<class RandomAccessIterator> class QuantLib::Math::LexicographicalView< RandomAccessIterator >

This view can be used to easily store a discretized 2-D function in an array to be used in a finite differences calculation.

The documentation for this class was generated from the following file:

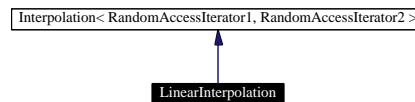
- [lexicographicalview.hpp](#)

11.124 QuantLib::Math::LinearInterpolation Class Template Reference

linear interpolation between discrete points.

```
#include <linearinterpolation.hpp>
```

Inheritance diagram for QuantLib::Math::LinearInterpolation:



Public Methods

- **LinearInterpolation** (const RandomAccessIterator1 &xBegin, const RandomAccessIterator1 &xEnd, const RandomAccessIterator2 &yBegin, bool allowExtrapolation)
- result_type **operator()** (const argument_type &x) const

Public Attributes

- typedef< RandomAccessIterator1 >::value_type **argument_type**
- typedef< RandomAccessIterator2 >::value_type **result_type**

```
template<class RandomAccessIterator1, class RandomAccessIterator2> class QuantLib::Math::LinearInterpolation< RandomAccessIterator1, RandomAccessIterator2 >
```

11.124.1 Member Function Documentation

11.124.1.1 result_type operator() (const argument_type &x) const [virtual]

This operator must be overridden to provide an implementation of the actual interpolation.

Precondition:

The sequence of values for x must have been sorted for the result to make sense.

Implements [QuantLib::Math::Interpolation](#).

The documentation for this class was generated from the following file:

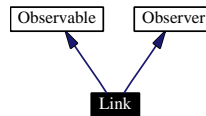
- [linearinterpolation.hpp](#)

11.125 QuantLib::Link Class Template Reference

Relinkable access to a [Handle](#).

```
#include <relinkablehandle.hpp>
```

Inheritance diagram for QuantLib::Link:



Public Methods

- [Link](#) (const [Handle](#)< Type > &h=[Handle](#)< Type >(), bool registerAsObserver=true)
- void [linkTo](#) (const [Handle](#)< Type > &h, bool registerAsObserver=true)
- bool [isNull](#) () const
Checks if the contained handle points to anything.
- const [Handle](#)< Type > & [currentLink](#) () const
Returns the contained handle.
- void [update](#) ()
Observer interface.

11.125.1 Detailed Description

```
template<class Type> class QuantLib::Link< Type >
```

Precondition:

Class "Type" must inherit from Observable

11.125.2 Constructor & Destructor Documentation

11.125.2.1 [Link](#) (const [Handle](#)< Type > &h = [Handle](#)< Type >(), bool *registerAsObserver* = true) [inline]

Warning:

see the documentation of the `linkTo` method for issues relatives to `registerAsObserver`.

11.125.3 Member Function Documentation

11.125.3.1 void [linkTo](#) (const [Handle](#)< Type > &h, bool *registerAsObserver* = true) [inline]

Warning:

`registerAsObserver` is left as a backdoor in case the programmer cannot guarantee that the object pointed to will remain alive for the whole lifetime of the handle—namely, it should be set to `false` when the passed handle was created with `owns = false` (the latter should only happen in a controlled environment, so that the programmer is aware of it). Failure to do so can very likely result in

a program crash. If the programmer does want the relinkable handle to register as observer of such a handle, it is his responsibility to ensure that the relinkable handle gets destroyed before the pointed object does.

The documentation for this class was generated from the following file:

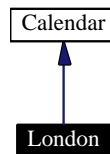
- [relinkablehandle.hpp](#)

11.126 QuantLib::Calendars::London Class Reference

London calendar.

```
#include <london.hpp>
```

Inheritance diagram for QuantLib::Calendars::London:



Public Methods

- **London** ()

11.126.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st (possibly moved to Monday)
- Good Friday
- Easter Monday
- Bank Holiday, first Monday of May
- Bank Holiday, last Monday of May
- Bank Holiday, last Monday of August
- Christmas, December 25th (possibly moved to Monday or Tuesday)
- Boxing Day, December 26th (possibly moved to Monday or Tuesday)

The documentation for this class was generated from the following file:

- [london.hpp](#)

11.127 QuantLib::Utilities::lowest_category_iterator Struct Template Reference

most generic of two given iterator categories.

```
#include <iteratorcategories.hpp>
```

11.127.1 Detailed Description

```
template<class Category1, class Category2> struct QuantLib::Utilities::lowest_category_iterator<
Category1, Category2 >
```

Specializations of this struct define a typedef `iterator_category` which corresponds to the most generic of the two input categories, e.g., [lowest_category_iterator<std::random_access_iterator_tag, std::forward_iterator_tag>](#) `iterator_category` corresponds to `std::forward_iterator_tag`.

The documentation for this struct was generated from the following file:

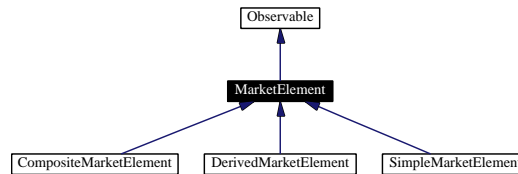
- [iteratorcategories.hpp](#)

11.128 QuantLib::MarketElement Class Reference

purely virtual base class for market observables.

```
#include <marketelement.hpp>
```

Inheritance diagram for QuantLib::MarketElement:



Public Methods

- virtual `~MarketElement()`
- virtual double `value()` const=0
returns the current value.

The documentation for this class was generated from the following file:

- [marketelement.hpp](#)

11.129 QuantLib::Math::Matrix Class Reference

matrix used in linear algebra.

```
#include <matrix.hpp>
```

Public Types

- typedef double * **iterator**
- typedef const double * **const_iterator**
- typedef double * **row_iterator**
- typedef const double * **const_row_iterator**
- typedef [Utilities::stepping_iterator](#)< double * > **column_iterator**
- typedef [Utilities::stepping_iterator](#)< const double * > **const_column_iterator**

Public Methods

- typedef **QL_REVERSE_ITERATOR** (iterator, double) **reverse_iterator**
- typedef **QL_REVERSE_ITERATOR** (const_iterator, double) **const_reverse_iterator**
- typedef **QL_REVERSE_ITERATOR** (row_iterator, double) **reverse_row_iterator**
- typedef **QL_REVERSE_ITERATOR** (const_row_iterator, double) **const_reverse_row_iterator**
- typedef **QL_REVERSE_ITERATOR** (column_iterator, double) **reverse_column_iterator**
- typedef **QL_REVERSE_ITERATOR** (const_column_iterator, double) **const_reverse_column_iterator**

Constructors, destructor, and assignment

- [Matrix](#) ()
creates a null matrix.
- [Matrix](#) (Size rows, Size columns)
creates a matrix with the given dimensions.
- [Matrix](#) (Size rows, Size columns, double value)
creates the matrix and fills it with value.
- **Matrix** (const Matrix &)
- **~Matrix** ()
- Matrix & **operator=** (const Matrix &)

Algebraic operators

- Matrix & [operator+=](#) (const Matrix &)
- Matrix & **operator-=** (const Matrix &)
- Matrix & **operator*=** (double)
- Matrix & **operator/=** (double)

Iterator access

- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **end** () const

- iterator **end** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rend** () const
- reverse_iterator **rend** ()
- const_row_iterator **row_begin** (Size i) const
- row_iterator **row_begin** (Size i)
- const_row_iterator **row_end** (Size i) const
- row_iterator **row_end** (Size i)
- const_reverse_row_iterator **row_rbegin** (Size i) const
- reverse_row_iterator **row_rbegin** (Size i)
- const_reverse_row_iterator **row_rend** (Size i) const
- reverse_row_iterator **row_rend** (Size i)
- const_column_iterator **column_begin** (Size i) const
- column_iterator **column_begin** (Size i)
- const_column_iterator **column_end** (Size i) const
- column_iterator **column_end** (Size i)
- const_reverse_column_iterator **column_rbegin** (Size i) const
- reverse_column_iterator **column_rbegin** (Size i)
- const_reverse_column_iterator **column_rend** (Size i) const
- reverse_column_iterator **column_rend** (Size i)

Element access

- const_row_iterator **operator[]** (Size) const
- row_iterator **operator[]** (Size)
- [Array diagonal](#) (void) const

Inspectors

- [Size rows](#) () const
- [Size columns](#) () const

Related Functions

(Note that these are not member functions.)

- Matrix **operator+** (const Matrix &, const Matrix &)
- Matrix **operator-** (const Matrix &, const Matrix &)
- Matrix **operator*** (const Matrix &, double)
- Matrix **operator*** (double, const Matrix &)
- Matrix **operator/** (const Matrix &, double)
- [Array operator*](#) (const [Array](#) &, const Matrix &)
- [Array operator*](#) (const Matrix &, const [Array](#) &)
- Matrix **operator*** (const Matrix &, const Matrix &)
- Matrix **transpose** (const Matrix &)
- Matrix **outerProduct** (const [Array](#) &v1, const [Array](#) &v2)
- Matrix [matrixSqrt](#) (const Matrix &realSymmetricMatrix)

returns the square root of a real symmetric matrix.

11.129.1 Detailed Description

This class implements the concept of vector as used in linear algebra. As such, it is **not** meant to be used as a container.

11.129.2 Member Function Documentation

11.129.2.1 `Matrix & operator+=(const Matrix &) [inline]`

Precondition:

all matrices involved in an algebraic expression must have the same size.

The documentation for this class was generated from the following file:

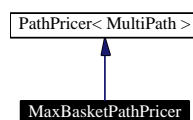
- [matrix.hpp](#)

11.130 QuantLib::MonteCarlo::MaxBasketPathPricer Class Reference

multipath pricer for European-type basket option.

```
#include <maxbasketpathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::MaxBasketPathPricer:



Public Methods

- **MaxBasketPathPricer** (const [Array](#) &underlying, [DiscountFactor](#) discount, bool useAntithetic-Variance)
- double **operator()** (const [MultiPath](#) &multiPath) const

11.130.1 Detailed Description

The value of the option at expiration is given by the value of the underlying which has best performed.

The documentation for this class was generated from the following files:

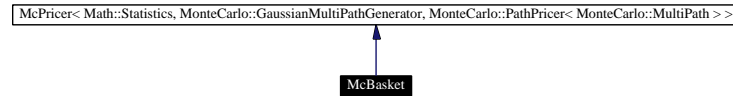
- [maxbasketpathpricer.hpp](#)
- [maxbasketpathpricer.cpp](#)

11.131 QuantLib::Pricers::McBasket Class Reference

simple example of multi-factor Monte Carlo pricer.

```
#include <mcbasket.hpp>
```

Inheritance diagram for QuantLib::Pricers::McBasket:



Public Methods

- **McBasket** (Option::Type type, const [Array](#) &underlying, double strike, const [Array](#) ÷ndYield, const [Math::Matrix](#) &covariance, [Rate](#) riskFreeRate, double residualTime, bool antitheticVariance, long seed=0)

The documentation for this class was generated from the following files:

- [mcbasket.hpp](#)
- [mcbasket.cpp](#)

11.132 QuantLib::Pricers::McDiscreteArithmeticAPO Class Reference

example of Monte Carlo pricer using a control variate.

```
#include <mcdiscretearithmeticapo.hpp>
```

Inheritance diagram for QuantLib::Pricers::McDiscreteArithmeticAPO:



Public Methods

- **McDiscreteArithmeticAPO** (Option::Type type, double underlying, double strike, [Spread](#) dividend-Yield, [Rate](#) riskFreeRate, const std::vector< [Time](#) > ×, double volatility, bool antithetic-Variance, bool controlVariate, long seed=0)

11.132.1 Detailed Description

Todo:

Continuous Averaging version

The documentation for this class was generated from the following files:

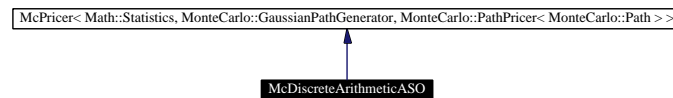
- [mcdiscretearithmeticapo.hpp](#)
- [mcdiscretearithmeticapo.cpp](#)

11.133 QuantLib::Pricers::McDiscreteArithmeticASO Class Reference

example of Monte Carlo pricer using a control variate.

```
#include <mcdiscretearithmeticaso.hpp>
```

Inheritance diagram for QuantLib::Pricers::McDiscreteArithmeticASO:



Public Methods

- **McDiscreteArithmeticASO** (Option::Type type, double underlying, [Spread](#) dividendYield, [Rate](#) riskFreeRate, const std::vector< [Time](#) > ×, double volatility, bool antitheticVariance, bool controlVariate, long seed=0)

11.133.1 Detailed Description

Todo:

Continuous Averaging version

The documentation for this class was generated from the following files:

- [mcdiscretearithmeticaso.hpp](#)
- [mcdiscretearithmeticaso.cpp](#)

11.134 QuantLib::Pricers::McEuropean Class Reference

simple example of Monte Carlo pricer.

```
#include <mceuropean.hpp>
```

Inheritance diagram for QuantLib::Pricers::McEuropean:



Public Methods

- **McEuropean** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, double residualTime, double volatility, bool antitheticVariance, long seed=0)

11.134.1 Detailed Description

Examples:

[EuropeanOption.cpp](#).

The documentation for this class was generated from the following files:

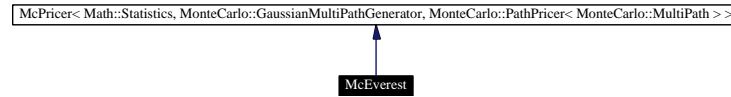
- [mceuropean.hpp](#)
- [mceuropean.cpp](#)

11.135 QuantLib::Pricers::McEverest Class Reference

Everest-type option pricer.

```
#include <mceverest.hpp>
```

Inheritance diagram for QuantLib::Pricers::McEverest:



Public Methods

- **McEverest** (const [Array](#) ÷ndYield, const [Math::Matrix](#) &covariance, [Rate](#) riskFreeRate, [Time](#) residualTime, bool antitheticVariance, long seed=0)

11.135.1 Detailed Description

The payoff of an Everest option is simply given by the final price / initial price ratio of the worst performer

The documentation for this class was generated from the following file:

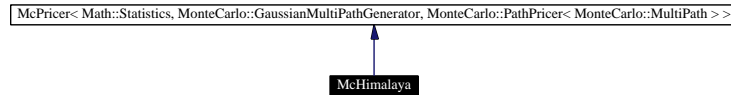
- [mceverest.hpp](#)

11.136 QuantLib::Pricers::McHimalaya Class Reference

Himalayan-type option pricer.

```
#include <mchimalaya.hpp>
```

Inheritance diagram for QuantLib::Pricers::McHimalaya:



Public Methods

- **McHimalaya** (const [Array](#) &underlying, const [Array](#) ÷ndYield, const [Math::Matrix](#) &covariance, [Rate](#) riskFreeRate, double strike, const std::vector< [Time](#) > ×, bool antitheticVariance, long seed=0)

11.136.1 Detailed Description

The payoff of a Himalaya option is computed in the following way: Given a basket of N assets, and N time periods, at end of each period the option who performed the best is added to the average and then discarded from the basket. At the end of the N periods the option pays the max between the strike and the average of the best performers.

The documentation for this class was generated from the following files:

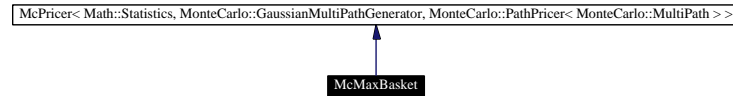
- [mchimalaya.hpp](#)
- [mchimalaya.cpp](#)

11.137 QuantLib::Pricers::McMaxBasket Class Reference

simple example of multi-factor Monte Carlo pricer.

```
#include <mcmxbasket.hpp>
```

Inheritance diagram for QuantLib::Pricers::McMaxBasket:



Public Methods

- **McMaxBasket** (const [Array](#) &underlying, const [Array](#) ÷ndYield, const [Math::Matrix](#) &co-variance, [Rate](#) riskFreeRate, double residualTime, bool antitheticVariance, long seed=0)

The documentation for this class was generated from the following files:

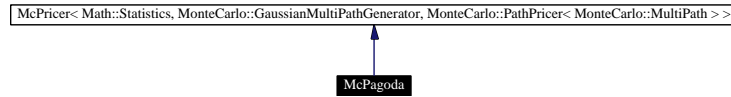
- [mcmxbasket.hpp](#)
- [mcmxbasket.cpp](#)

11.138 QuantLib::Pricers::McPagoda Class Reference

roofed Asian option.

```
#include <mcpagoda.hpp>
```

Inheritance diagram for QuantLib::Pricers::McPagoda:



Public Methods

- **McPagoda** (const [Array](#) &portfolio, double fraction, double roof, const [Array](#) ÷ndYield, const [Math::Matrix](#) &covariance, [Rate](#) riskFreeRate, const std::vector< [Time](#) > ×, bool antithetic, long seed=0)

11.138.1 Detailed Description

Given a certain portfolio of assets at the end of the period it is returned the minimum of a given roof and a certain fraction of the positive portfolio performance. If the performance of the portfolio is below then the payoff is null.

The documentation for this class was generated from the following file:

- [mcpagoda.hpp](#)

11.139 QuantLib::Pricers::McPricer Class Template Reference

base class for Monte Carlo pricers.

```
#include <mcpricer.hpp>
```

Public Methods

- virtual `~McPricer()`
- double `value` (double tolerance, [Size](#) maxSample=QL_MAX_INT) const
add samples until the required tolerance is reached.
- double `valueWithSamples` ([Size](#) samples) const
simulate a fixed number of samples.
- double `errorEstimate` () const
error Estimated of the samples simulated so far.
- const S & `sampleAccumulator` (void) const
access to the sample accumulator for more statistics.

Protected Methods

- `McPricer()`

Protected Attributes

- [Handle](#)< [MonteCarlo::MonteCarloModel](#)< S, PG, PP > > `mcModel_`

Static Protected Attributes

- const [Size](#) `minSample_` = 100

11.139.1 Detailed Description

```
template<class S, class PG, class PP> class QuantLib::Pricers::McPricer< S, PG, PP >
```

Eventually this class might be linked to the general tree of pricers, in order to have tools like implied-Volatility available. Also, it could, eventually, offer greeks methods. Deriving a class from [McPricer](#) gives an easy way to write a Monte Carlo Pricer. See [McEuropean](#) as example of one factor pricer, Basket as example of multi factor pricer.

The documentation for this class was generated from the following file:

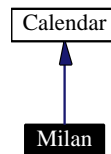
- [mcpricer.hpp](#)

11.140 QuantLib::Calendars::Milan Class Reference

Milan calendar.

```
#include <milan.hpp>
```

Inheritance diagram for QuantLib::Calendars::Milan:



Public Methods

- **Milan ()**

11.140.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st
- Epiphany, January 6th
- Easter Monday
- Liberation Day, April 25th
- Labour Day, May 1st
- Republic Day, June 2nd (since 2000)
- Assumption, August 15th
- All Saint's Day, November 1st
- Immaculate Conception, December 8th
- Christmas, December 25th
- St. Stephen, December 26th

The documentation for this class was generated from the following file:

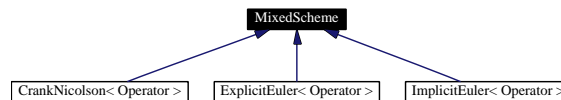
- [milan.hpp](#)

11.141 QuantLib::FiniteDifferences::MixedScheme Class Template Reference

Mixed (explicit/implicit) scheme for finite difference methods.

```
#include <mixedscheme.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::MixedScheme:



Protected Types

- typedef Operator::arrayType **arrayType**
- typedef Operator **operatorType**

Protected Methods

- **MixedScheme** (const Operator &L, double theta)
- void **step** (arrayType &a, [Time](#) t)
- void **setStep** ([Time](#) dt)

Protected Attributes

- Operator **L_**
- Operator **I_**
- Operator **explicitPart_**
- Operator **implicitPart_**
- [Time](#) **dt_**
- double **theta_**

Friends

- class **FiniteDifferenceModel**< **MixedScheme**< **Operator** > >

11.141.1 Detailed Description

```
template<class Operator> class QuantLib::FiniteDifferences::MixedScheme< Operator >
```

See sect. [The finite differences framework](#) for details on the method.

In this implementation, the passed operator must be derived from either TimeConstantOperator or Time-DependentOperator. Also, it must implement at least the following interface:

```
typedef ... arrayType;

// copy constructor/assignment
```

```
// (these will be provided by the compiler if none is defined)
Operator(const Operator&);
Operator& operator=(const Operator&);

// inspectors
size_t size();

// modifiers
void setTime(Time t);

// operator interface
arrayType applyTo(const arrayType&);
arrayType solveFor(const arrayType&);
static Operator identity(size_t size);

// operator algebra
Operator operator*(double, const Operator&);
Operator operator+(const Operator&, const Operator&);
Operator operator+(const Operator&, const Operator&);
```

Warning:

The differential operator must be linear for this evolver to work.

Todo:

add Douglas Scheme

The documentation for this class was generated from the following file:

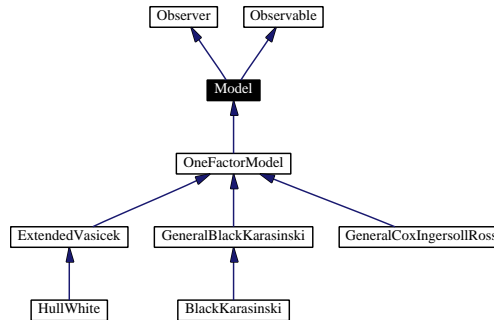
- [mixedscheme.hpp](#)

11.142 QuantLib::InterestRateModelling::Model Class Reference

Abstract short-rate model class.

```
#include <model.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::Model:



Public Methods

- **Model** (*Size* nParameters, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- virtual ~**Model** ()
- void [update](#) ()
- void **calibrate** (CalibrationSet &instruments, const [Handle](#)< [Optimization::OptimizationMethod](#) > &method)
- const [RelinkableHandle](#)< [TermStructure](#) > & [termStructure](#) () const

Returns the present term structure implied by the model.

Protected Methods

- virtual void **generateParameters** ()

Protected Attributes

- [Handle](#)< [Optimization::Constraint](#) > **constraint_**
- std::vector< [Parameter](#) > **parameters_**

Friends

- class **CalibrationFunction**

11.142.1 Member Function Documentation

11.142.1.1 void update () [inline, virtual]

This method must be implemented in derived classes. An instance of [Observer](#) does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

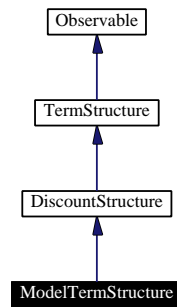
- [model.hpp](#)

11.143 QuantLib::InterestRateModelling::ModelTermStructure Class Reference

Term structure implied by a model.

```
#include <onefactormodel.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::ModelTermStructure:



Public Methods

- **ModelTermStructure** (const OneFactorAffineModel *model, [Time](#) t0, [Rate](#) r0)
- virtual [DiscountFactor](#) **discountImpl** ([Time](#) t, bool extrapolate)

The documentation for this class was generated from the following file:

- [onefactormodel.hpp](#)

11.144 QuantLib::MonteCarlo::MonteCarloModel Class Template Reference

General purpose Monte Carlo model for path samples.

```
#include <montecarlomodel.hpp>
```

Public Types

- typedef PG::sample_type **sample_type**
- typedef PP::result_type **result_type**

Public Methods

- **MonteCarloModel** (const [Handle](#)< PG > &pathGenerator, const [Handle](#)< PP > &pathPricer, const S &sampleAccumulator, const [Handle](#)< PP > &cvPathPricer=[Handle](#)< PP >(), result_type cvOptionValue=result_type())
- void **addSamples** ([Size](#) samples)
- const S & **sampleAccumulator** (void) const

11.144.1 Detailed Description

```
template<class S, class PG, class PP> class QuantLib::MonteCarlo::MonteCarloModel< S, PG, PP >
```

Any Monte Carlo which uses path samples has three main components, namely,

- S, a sample accumulator,
- PG, a path generator,
- PP, a path pricer. [MonteCarloModel](#)<S, PG, PP> puts together these three elements. The constructor accepts two safe references, i.e. two smart pointers, one to a path generator and the other to a path pricer. In case of control variate technique the user should provide the additional control option, namely the option path pricer and the option value.

The minimal interfaces for the classes S, PG, and PP are:

```
class S{
    void add(VALUE_TYPE sample, double weight) const;
};

class PG{
    Sample<PATH_TYPE> next() const;
};

class PP :: unary_function<PATH_TYPE, VALUE_TYPE> {
    VALUE_TYPE operator()(PATH_TYPE &) const;
};
```

The documentation for this class was generated from the following file:

- [montecarlomodel.hpp](#)

11.145 QuantLib::MonteCarlo::MultiPath Class Reference

single random walk.

```
#include <multipath.hpp>
```

Public Methods

- **MultiPath** ([Size](#) nAsset, [Size](#) pathSize)
- **MultiPath** (const std::vector< [Path](#) > &multiPath)

inspectors

- [Size](#) **assetNumber** () const
- [Size](#) **pathSize** () const

read/write access to components

- const [Path](#) & **operator[]** ([Size](#) j) const
- [Path](#) & **operator[]** ([Size](#) j)

11.145.1 Detailed Description

[MultiPath](#) contains the list of variations for each asset,

$$\log \frac{Y_{i+1}^j}{Y_i^j} \text{ for } i = 0, \dots, n-1 \quad \text{and} \quad j = 0, \dots, m-1$$

where Y_i^j is the value of the underlying j at discretized time t_i . The first index refers to the underlying, the second to the time position [MultiPath\[j,i\]](#)

Todo:

make it time-aware

The documentation for this class was generated from the following file:

- [multipath.hpp](#)

11.146 QuantLib::MonteCarlo::MultiPathGenerator Class Template Reference

Generates a multipath from a random number generator.

```
#include <multipathgenerator.hpp>
```

Public Types

- typedef [Sample](#)< [MultiPath](#) > **sample_type**

Public Methods

- **MultiPathGenerator** (const [Array](#) &drifts, const [Math::Matrix](#) &covariance, [Time](#) length, [Size](#) timeSteps, long seed)
- **MultiPathGenerator** (const [Array](#) &drifts, const [Math::Matrix](#) &covariance, const std::vector< [Time](#) > ×, long seed=0)
- const sample_type & **next** () const

11.146.1 Detailed Description

template<class RAG> class QuantLib::MonteCarlo::MultiPathGenerator< RAG >

[MultiPathGenerator](#)<RAG> is a class that returns a random multi path. RAG is a sample generator which returns a random array. It must have the minimal interface:

```
RAG{
    RAG();
    RAG(Matrix& covariance,
        long seed);
    Sample<Array> next();
};
```

The documentation for this class was generated from the following file:

- [multipathgenerator.hpp](#)

11.147 QuantLib::Math::MultivariateAccumulator Class Reference

A sample accumulator for multivariate analysis.

```
#include <multivariateaccumulator.hpp>
```

Public Methods

- **MultivariateAccumulator** ()
- **MultivariateAccumulator** ([Size](#) size)

Inspectors

- [Size](#) **size** () const
size of each sample.
- [Size](#) **samples** () const
number of samples collected.
- double [weightSum](#) () const
sum of data weights.
- [Array](#) **mean** () const
returns the mean as an [Array](#).
- std::vector< double > [meanVector](#) () const
returns the mean as a std::vector<double>.
- [Matrix](#) **covariance** () const
returns the covariance [Matrix](#).
- [Matrix](#) **correlation** () const
returns the correlation [Matrix](#).

Modifiers

- void [add](#) (const [Array](#) &arr, double weight=1.0)
adds an [Array](#) to the collection, possibly with a weight.
- void [add](#) (const std::vector< double > &vec, double weight=1.0)
adds a vector<double> to the collection, possibly with a weight.
- template<class DataIterator> void [addSequence](#) (DataIterator begin, DataIterator end)
adds a sequence of data to the collection.
- template<class DataIterator, class WeightIterator> void [addSequence](#) (DataIterator begin, DataIterator end, WeightIterator wbegin)
adds a sequence of data to the collection, each with its weight.
- void [reset](#) ()
resets the data to a null set.

11.147.1 Detailed Description

[MultivariateAccumulator](#) can accumulate vector-type samples and return the average vector, both in [Array](#) form and `std::vector<double>` form, and the covariance matrix

11.147.2 Member Function Documentation

11.147.2.1 [Matrix](#) covariance () const

Precondition:

weights must be positive or null

The documentation for this class was generated from the following files:

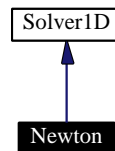
- [multivariateaccumulator.hpp](#)
- [multivariateaccumulator.cpp](#)

11.148 QuantLib::Solvers1D::Newton Class Reference

Newton 1-D solver.

```
#include <newton.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::Newton:



The documentation for this class was generated from the following files:

- [newton.hpp](#)
- [newton.cpp](#)

11.149 QuantLib::Solvers1D::NewtonSafe Class Reference

safe Newton 1-D solver.

```
#include <newtonsafe.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::NewtonSafe:



The documentation for this class was generated from the following files:

- [newtonsafe.hpp](#)
- [newtonsafe.cpp](#)

11.150 QuantLib::Calendars::NewYork Class Reference

New York calendar.

```
#include <newyork.hpp>
```

Inheritance diagram for QuantLib::Calendars::NewYork:



Public Methods

- **NewYork ()**

11.150.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st (possibly moved to Monday if actually on Sunday, or to Friday if on Saturday)
- Martin Luther King's birthday, third Monday in January
- Washington's birthday, third Monday in February
- Memorial Day, last Monday in May
- Independence Day, July 4th (moved to Monday if Sunday or Friday if Saturday)
- Labor Day, first Monday in September
- Columbus Day, second Monday in October
- Veteran's Day, November 11th (moved to Monday if Sunday or Friday if Saturday)
- Thanksgiving Day, fourth Thursday in November
- Christmas, December 25th (moved to Monday if Sunday or Friday if Saturday)

The documentation for this class was generated from the following file:

- [newyork.hpp](#)

11.151 QuantLib::Optimization::NonLinearLeastSquare Class Reference

```
#include <leastsquare.hpp>
```

Public Methods

- [NonLinearLeastSquare](#) (Constraint &c, double accuracy=1e-4, int maxiter=100)
Default constructor.
- [NonLinearLeastSquare](#) (Constraint &c, double accuracy, int maxiter, [Handle](#)< [OptimizationMethod](#) > om)
Default constructor.
- [~NonLinearLeastSquare](#) ()
Destructor.
- [Array](#) & [perform](#) (LeastSquareProblem &lsProblem)
Solve least square problem using numerix solver.
- void [setInitialValue](#) (const [Array](#) &initialValue)
- [Array](#) & [results](#) ()
return the results.
- double [residualNorm](#) ()
return the least square residual norm.
- double [lastValue](#) ()
return last function value.
- int [exitFlag](#) ()
return exit flag.
- int [iterationsNumber](#) ()
return the performed number of iterations.

11.151.1 Detailed Description

Default least square method using a given optimization algorithm (default is conjugate gradient).

$$\min \{ r(x) : x \in \mathbb{R}^n \}$$

where $r(x) = \|f(x)\|^2$ the euclidian norm of $f(x)$ for some vector-valued function f from \mathbb{R}^n to \mathbb{R}^m $f = (f_1, \dots, f_m)$ with $f_i(x) = b_i - \phi_i(x)$ where b_i is the vector of target data and ϕ_i is a scalar function.

Assuming the differentiability of f , the gradient of r is define by $\text{grad } r(x) = f'(x)^t \cdot f(x)$

[Array](#) vector class has the requirement of the previous class [Handle](#) class is need to manage pointer to optimization method

The documentation for this class was generated from the following file:

- [leastsquare.hpp](#)

11.152 QuantLib::Null Class Template Reference

template class providing a null value for a given type.

```
#include <null.hpp>
```

Public Methods

- **Null ()**
- **operator Type () const**

```
template<class Type> class QuantLib::Null< Type >
```

The documentation for this class was generated from the following file:

- [null.hpp](#)

11.153 QuantLib::ObjectiveFunction Class Reference

Objective function for 1-D solvers.

```
#include <solver1d.hpp>
```

Public Methods

- virtual `~ObjectiveFunction()`
- virtual double `operator()` (double x) const=0
returns $f(x)$.
- virtual double `derivative` (double x) const
returns $f'(x)$.

11.153.1 Detailed Description

This is the function whose zeroes must be found.

The documentation for this class was generated from the following file:

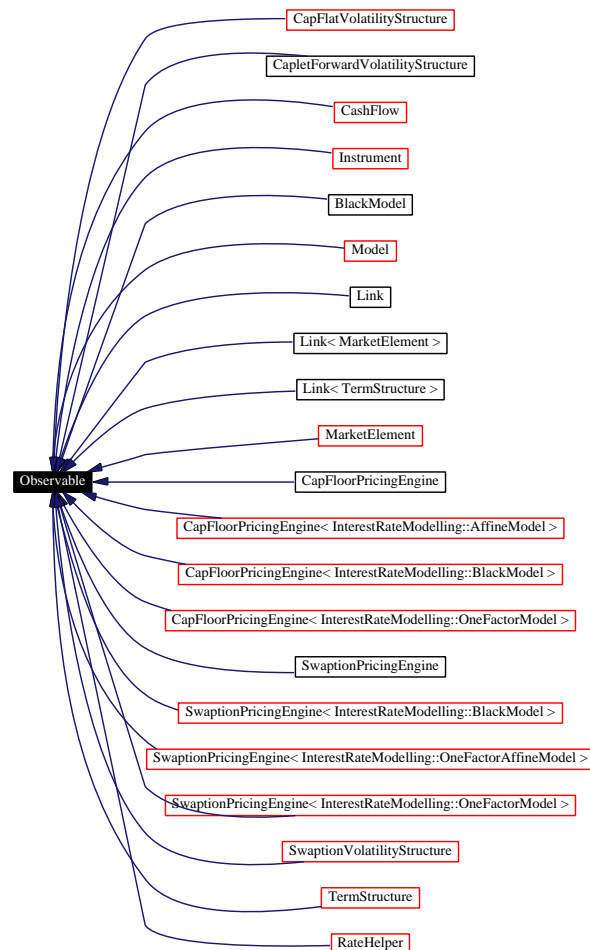
- [solver1d.hpp](#)

11.154 QuantLib::Patterns::Observable Class Reference

Object that notifies its changes to a set of observables.

```
#include <observable.hpp>
```

Inheritance diagram for QuantLib::Patterns::Observable:



Public Methods

- virtual `~Observable()`
- void `notifyObservers()`

Friends

- class `Observer`

11.154.1 Member Function Documentation

11.154.1.1 `void notifyObservers()` [inline]

This method should be called at the end of non-const methods or when the programmer desires to notify any changes.

The documentation for this class was generated from the following file:

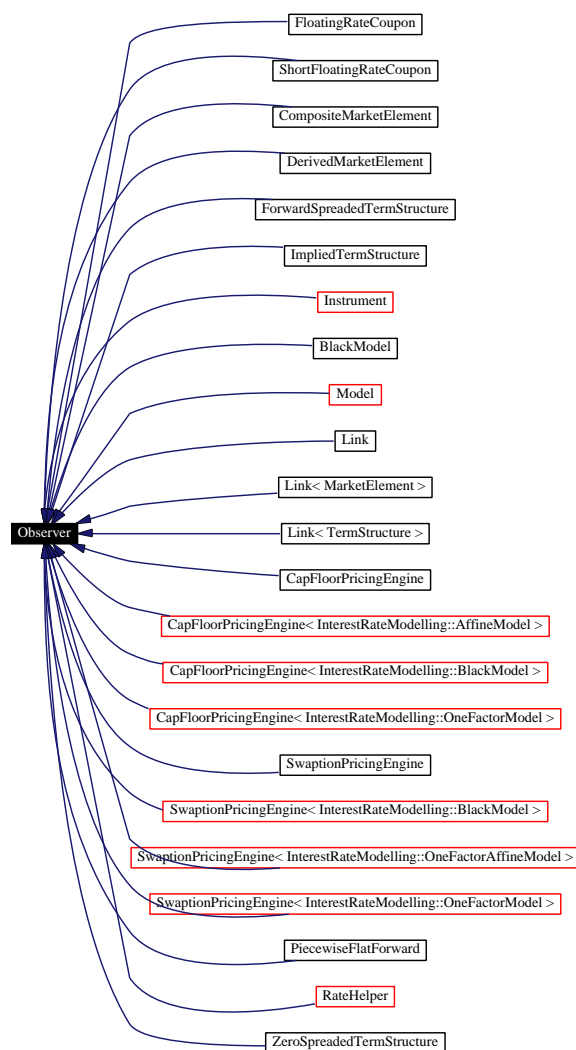
- [observable.hpp](#)

11.155 QuantLib::Patterns::Observer Class Reference

Object that gets notified when a given observable changes.

```
#include <observable.hpp>
```

Inheritance diagram for QuantLib::Patterns::Observer:



Public Methods

- **Observer** ()
- **Observer** (const Observer &)
- Observer & **operator=** (const Observer &)
- virtual ~**Observer** ()
- void **registerWith** (const [Handle](#)< [Observable](#) > &)
- void **unregisterWith** (const [Handle](#)< [Observable](#) > &)
- virtual void **update** ()=0

11.155.1 Member Function Documentation

11.155.1.1 `virtual void update ()` [pure virtual]

This method must be implemented in derived classes. An instance of `Observer` does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implemented in [QuantLib::CashFlows::FloatingRateCoupon](#).

The documentation for this class was generated from the following file:

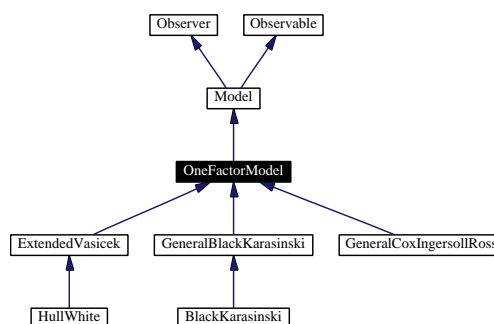
- [observable.hpp](#)

11.156 QuantLib::InterestRateModelling::OneFactorModel Class Reference

Single-factor short-rate model abstract class.

```
#include <onefactormodel.hpp>
```

Inheritance diagram for QuantLib::InterestRateModelling::OneFactorModel:



Public Methods

- **OneFactorModel** ([Size](#) nParameters, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure)
- virtual **~OneFactorModel** ()
- virtual [Handle](#)< [ShortRateProcess](#) > **process** () const=0
returns the driving stochastic equation.
- virtual [Handle](#)< [Lattices::Tree](#) > **tree** (const [TimeGrid](#) &grid) const
Return a recombining tree.

The documentation for this class was generated from the following files:

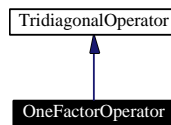
- [onefactormodel.hpp](#)
- [onefactormodel.cpp](#)

11.157 QuantLib::FiniteDifferences::OneFactorOperator Class Reference

Interest-rate single factor model differential operator.

```
#include <onefactoroperator.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::OneFactorOperator:



Public Methods

- **OneFactorOperator** ()
- **OneFactorOperator** (const [Array](#) &grid, const [Handle](#)< InterestRateModelling::ShortRateProcess > &process)
- virtual **~OneFactorOperator** ()

The documentation for this class was generated from the following file:

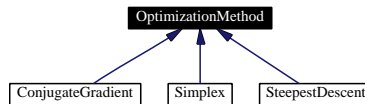
- [onefactoroperator.hpp](#)

11.158 QuantLib::Optimization::OptimizationMethod Class Reference

Optimization Method abstract class for unconstrained optimization pb.

```
#include <optimizer.hpp>
```

Inheritance diagram for QuantLib::Optimization::OptimizationMethod:



Public Methods

- **OptimizationMethod** ()
- virtual **~OptimizationMethod** ()
- void **setInitialValue** (const **Array** &initialValue)
Set initial value.
- void **setEndCriteria** (const OptimizationEndCriteria &endCriteria)
Set optimization end criteria.
- int & **iterationNumber** ()
current iteration number.
- OptimizationEndCriteria & **endCriteria** ()
optimization end criteria.
- int & **functionEvaluation** ()
number of evaluation of cost function.
- int & **gradientEvaluation** ()
number of evaluation of cost function gradient.
- double & **functionValue** ()
value of cost function.
- double & **gradientNormValue** ()
value of cost function gradient norm.
- **Array** & **x** ()
current value of the local minimum.
- **Array** & **searchDirection** ()
current value of the search direction.
- virtual void **minimize** (OptimizationProblem &P)=0
minimize the optimization problem P.

Protected Attributes

- [Array initialValue_](#)
initial value of unknowns.
- [int iterationNumber_](#)
current iteration step in the Optimization process.
- [OptimizationEndCriteria endCriteria_](#)
optimization end criteria.
- [int functionEvaluation_](#)
number of evaluation of cost function and its gradient.
- [int gradientEvaluation_](#)
number of evaluation of cost function and its gradient.
- [double functionValue_](#)
function and gradient norm values of the last step.
- [double squaredNorm_](#)
function and gradient norm values of the last step.
- [Array x_](#)
current values of the local minimum and the search direction.
- [Array searchDirection_](#)
current values of the local minimum and the search direction.

The documentation for this class was generated from the following file:

- [optimizer.hpp](#)

11.159 QuantLib::Optimization::OptimizationProblem Class Reference

Unconstrained optimization pb.

```
#include <optimizer.hpp>
```

Public Methods

- [OptimizationProblem](#) ([CostFunction](#) &f, [Constraint](#) &c, [OptimizationMethod](#) &meth)
default constructor.
- [~OptimizationProblem](#) ()
destructor.
- double [value](#) (const [Array](#) &x)
call cost function computation and increment evaluation counter.
- void [gradient](#) ([Array](#) &grad_f, const [Array](#) &x)
call cost function gradient computation and increment.
- double [valueAndGradient](#) ([Array](#) &grad_f, const [Array](#) &x)
call cost function computation and it gradient.
- [OptimizationMethod](#) & [optimisationMethod](#) ()
Unconstrained optimization method.
- [Constraint](#) & [constraint](#) ()
constraint.
- [CostFunction](#) & [costFunction](#) ()
- void [minimize](#) ()
Minimization.
- [Array](#) & [minimumValue](#) ()

Protected Attributes

- [CostFunction](#) & [costFunction_](#)
Unconstrained cost function.
- [Constraint](#) & [constraint_](#)
Constraint.
- [OptimizationMethod](#) & [method_](#)
Unconstrained optimization method.

The documentation for this class was generated from the following file:

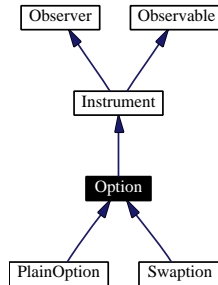
- [optimizer.hpp](#)

11.160 QuantLib::Option Class Reference

base option class.

```
#include <option.hpp>
```

Inheritance diagram for QuantLib::Option:



Public Types

- enum **Type** { **Call**, **Put**, **Straddle** }

Public Methods

- **Option** (const [Handle](#)< [OptionPricingEngine](#) > &engine, const std::string &isinCode="", const std::string &description="")
- virtual **~Option** ()
- void **setPricingEngine** (const [Handle](#)< [OptionPricingEngine](#) > &)

Protected Methods

- virtual void **setupEngine** () const=0
- virtual void **performCalculations** () const

Protected Attributes

- [Handle](#)< [OptionPricingEngine](#) > **engine_**

11.160.1 Detailed Description

Examples:

[DiscreteHedging.cpp](#).

11.160.2 Member Function Documentation

11.160.2.1 void performCalculations () const [protected, virtual]

Warning:

this method simply launches the engine and copies the returned value into NPV_. It does **not** set is-Expired_. This should be taken care of by redefining this method in derived classes and calling this

implementation after checking for validity and only if the check succeeded.

Implements [QuantLib::Instrument](#).

Reimplemented in [QuantLib::Instruments::PlainOption](#).

The documentation for this class was generated from the following files:

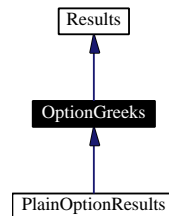
- [option.hpp](#)
- [option.cpp](#)

11.161 QuantLib::OptionGreeks Class Reference

option pricing results.

```
#include <option.hpp>
```

Inheritance diagram for QuantLib::OptionGreeks:



Public Methods

- **OptionGreeks ()**

Public Attributes

- double **delta**
- double **gamma**
- double **theta**
- double **vega**
- double **rho**
- double **dividendRho**

The documentation for this class was generated from the following file:

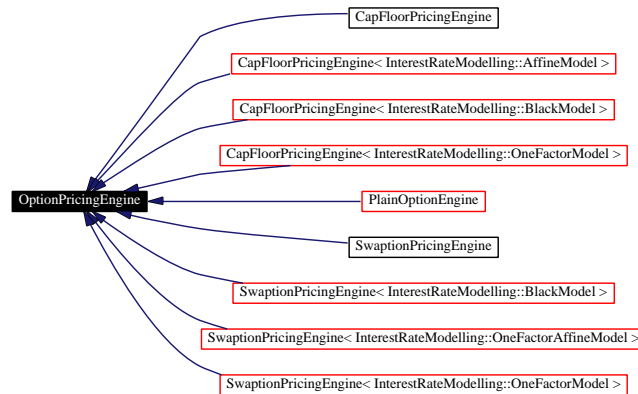
- [option.hpp](#)

11.162 QuantLib::OptionPricingEngine Class Reference

base class for option pricing engines.

```
#include <option.hpp>
```

Inheritance diagram for QuantLib::OptionPricingEngine:



Public Methods

- virtual `~OptionPricingEngine()`
- virtual `Arguments * parameters()=0`
- virtual void `validateParameters()` const=0
- virtual void `calculate()` const=0
- virtual const `Results * results()` const=0

The documentation for this class was generated from the following file:

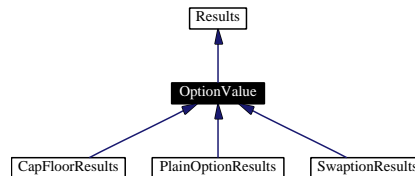
- [option.hpp](#)

11.163 QuantLib::OptionValue Class Reference

option pricing results.

```
#include <option.hpp>
```

Inheritance diagram for QuantLib::OptionValue:



Public Methods

- `OptionValue ()`

Public Attributes

- `double value`

11.163.1 Detailed Description

It must be noted that there's no result data specifying whether the option is expired. The expiry condition should be checked before calling the engine.

The documentation for this class was generated from the following file:

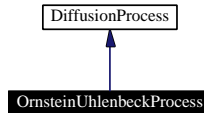
- [option.hpp](#)

11.164 QuantLib::OrnsteinUhlenbeckProcess Class Reference

Ornstein-Uhlenbeck process class.

```
#include <diffusionprocess.hpp>
```

Inheritance diagram for QuantLib::OrnsteinUhlenbeckProcess:



Public Methods

- **OrnsteinUhlenbeckProcess** (double speed, double vol, double x0=0.0)
- virtual double **drift** ([Time](#) t, double x) const
- virtual double **diffusion** ([Time](#) t, double x) const
- virtual double **expectation** ([Time](#) t0, double x0, [Time](#) dt) const

Euler approximation of the expectation.

- virtual double **variance** ([Time](#) t0, double x0, [Time](#) dt) const

Euler approximation of the variance.

11.164.1 Detailed Description

This class describes the Ornstein-Uhlenbeck process governed by

$$dx = -axdt + \sigma dW_t$$

.

The documentation for this class was generated from the following file:

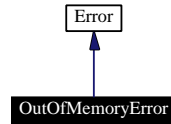
- [diffusionprocess.hpp](#)

11.165 QuantLib::OutOfMemoryError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::OutOfMemoryError:



Public Methods

- **OutOfMemoryError** (const std::string &whatClass="unknown class")

11.165.1 Detailed Description

Thrown upon failed allocation.

The documentation for this class was generated from the following file:

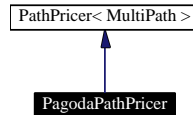
- [errors.hpp](#)

11.166 QuantLib::MonteCarlo::PagodaPathPricer Class Reference

multipath pricer for pagoda options.

```
#include <pagodapathpricer.hpp>
```

Inheritance diagram for QuantLib::MonteCarlo::PagodaPathPricer:



Public Methods

- **PagodaPathPricer** (const [Array](#) &underlying, double roof, [DiscountFactor](#) discount, bool use-AntitheticVariance)
- double **operator()** (const [MultiPath](#) &path) const

11.166.1 Detailed Description

A pagoda option is a multi-asset asian option with a cap (the pagoda "roof"). Given a portfolio of assets the payoff is the arithmetic average of the portfolio performance, with a maximum cap given by the roof.

The documentation for this class was generated from the following files:

- [pagodapathpricer.hpp](#)
- [pagodapathpricer.cpp](#)

11.167 QuantLib::MonteCarlo::Path Class Reference

single factor random walk.

```
#include <path.hpp>
```

Public Methods

- **Path** ([Size](#) size)
- **Path** (const std::vector< [Time](#) > ×, const [Array](#) &drift, const [Array](#) &diffusion)

inspectors

- double **operator[]** (int i) const
- [Size](#) **size** () const

read/write access to components

- const std::vector< [Time](#) > & **times** () const
- std::vector< [Time](#) > & **times** ()
- const [Array](#) & **drift** () const
- [Array](#) & **drift** ()
- const [Array](#) & **diffusion** () const
- [Array](#) & **diffusion** ()

11.167.1 Detailed Description

Todo:

make it time-aware

Examples:

[DiscreteHedging.cpp](#).

The documentation for this class was generated from the following file:

- [path.hpp](#)

11.168 QuantLib::MonteCarlo::PathGenerator Class Template Reference

Generates random paths from a random number generator.

```
#include <pathgenerator.hpp>
```

Public Types

- typedef [Sample](#)< [Path](#) > `sample_type`

Public Methods

- [PathGenerator](#) (double *drift*, double *variance*, [Time](#) *length*, [Size](#) *timeSteps*, long *seed*=0)
- [PathGenerator](#) (double *drift*, double *variance*, const std::vector< [Time](#) > &*times*, long *seed*=0)

inspectors

- const `sample_type` & `next` () const
- [Size](#) `size` () const

11.168.1 Detailed Description

```
template<class RNG> class QuantLib::MonteCarlo::PathGenerator< RNG >
```

[Todo:](#)

add more general path generator with `drift(S,t)` and `variance(S,t)`

11.168.2 Constructor & Destructor Documentation

11.168.2.1 [PathGenerator](#) (double *drift*, double *variance*, const std::vector< [Time](#) > &*times*, long *seed* = 0)

Warning:

the initial time is assumed to be zero and must **not** be included in the passed vector

The documentation for this class was generated from the following file:

- [pathgenerator.hpp](#)

11.169 QuantLib::MonteCarlo::PathPricer Class Template Reference

base class for path pricers.

```
#include <pathpricer.hpp>
```

Public Methods

- **PathPricer** ([DiscountFactor](#) discount, bool useAntitheticVariance)
- virtual **~PathPricer** ()
- virtual ValueType **operator()** (const PathType &path) const=0

Protected Attributes

- [DiscountFactor](#) discount_
- bool useAntitheticVariance_

11.169.1 Detailed Description

```
template<class PathType, class ValueType = double> class QuantLib::MonteCarlo::PathPricer<  
PathType, ValueType >
```

Given a path the value of an option is returned on that path.

Examples:

[DiscreteHedging.cpp](#).

The documentation for this class was generated from the following file:

- [pathpricer.hpp](#)

11.170 QuantLib::Period Class Reference

Time period described by a number of a given time unit.

```
#include <date.hpp>
```

Public Methods

- **Period** ()
- **Period** (int n, [TimeUnit](#) units)
- int **length** () const
- [TimeUnit](#) **units** () const

The documentation for this class was generated from the following file:

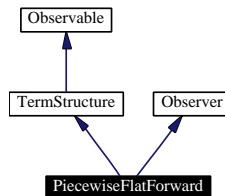
- [date.hpp](#)

11.171 QuantLib::TermStructures::PiecewiseFlatForward Class Reference

Piecewise flat forward term structure.

```
#include <piecewiseflatforward.hpp>
```

Inheritance diagram for QuantLib::TermStructures::PiecewiseFlatForward:



Public Methods

- **PiecewiseFlatForward** ([Currency](#) currency, const [DayCounter](#) &dayCounter, const [Date](#) &todayDate, const [Calendar](#) &calendar, int settlementDays, const std::vector< [Handle](#)< [RateHelper](#) > > &instruments, double accuracy=1.0e-12)

TermStructure interface

- [Currency](#) **currency** () const
returns the currency upon which the term structure is defined.
- [DayCounter](#) **dayCounter** () const
returns the day counter.
- [Date](#) **todayDate** () const
returns today's date.
- int **settlementDays** () const
returns the number of settlement days.
- [Calendar](#) **calendar** () const
returns the calendar for settlement calculation.
- [Date](#) **settlementDate** () const
returns the settlement date.
- const std::vector< [Date](#) > & **dates** () const
- [Date](#) **maxDate** () const
returns the latest date for which the curve can return rates.
- [Date](#) **minDate** () const
returns the earliest date for which the curve can return rates.
- const std::vector< [Time](#) > & **times** () const
- [Time](#) **maxTime** () const
returns the latest date for which the curve can return rates.

- [Time minTime](#) () const
returns the earliest time for which the curve can return rates.

Observer interface

- void [update](#) ()

Protected Methods

- [Rate zeroYieldImpl](#) (Time, bool extrapolate=false) const
implements the actual zero yield calculation in derived classes.
- [DiscountFactor discountImpl](#) (Time, bool extrapolate=false) const
implements the actual discount calculation in derived classes.
- [Rate forwardImpl](#) (Time, bool extrapolate=false) const
implements the actual forward rate calculation in derived classes.

Friends

- class [FFObjFunction](#)

11.171.1 Detailed Description

This term structure is bootstrapped on a number of interest rate instruments which are passed as a vector of handles to [RateHelper](#) instances. Their maturities mark the boundaries of the flat forward segments.

The values of the forward rates for each segment are determined sequentially starting from the earliest period to the latest.

The value for each segment is chosen so that the instrument whose maturity marks the end of such segment is correctly repriced on the curve.

Warning:

The bootstrapping algorithm will raise an exception if any two instruments have the same maturity date.

11.171.2 Member Function Documentation

11.171.2.1 void update () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following files:

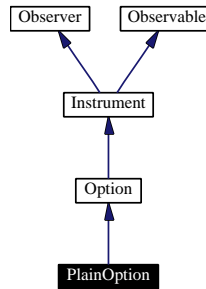
- [piecewiseflatforward.hpp](#)
- [piecewiseflatforward.cpp](#)

11.172 QuantLib::Instruments::PlainOption Class Reference

Plain (no dividends, no barriers) option on a single asset.

```
#include <plainoption.hpp>
```

Inheritance diagram for QuantLib::Instruments::PlainOption:



Public Methods

- **PlainOption** (Option::Type type, const RelinkableHandle< MarketElement > &underlying, double strike, const RelinkableHandle< TermStructure > ÷ndYield, const RelinkableHandle< TermStructure > &riskFreeRate, const Date &exerciseDate, const RelinkableHandle< MarketElement > &volatility, const Handle< OptionPricingEngine > &engine, const std::string &isinCode="", const std::string &description="")
- double **impliedVolatility** (double targetValue, double accuracy=1.0e-4, Size maxEvaluations=100, double minVol=1.0e-4, double maxVol=4.0) const

greeks

- double **delta** () const
- double **gamma** () const
- double **theta** () const
- double **vega** () const
- double **rho** () const
- double **dividendRho** () const

Protected Methods

- void **setupEngine** () const
- void **performCalculations** () const

11.172.1 Member Function Documentation

11.172.1.1 double **impliedVolatility** (double *targetValue*, double *accuracy* = 1.0e-4, Size *maxEvaluations* = 100, double *minVol* = 1.0e-4, double *maxVol* = 4.0) const

Warning:

Options with a gamma that changes sign have values that are **not** monotonic in the volatility, e.g. binary options. In these cases impliedVolatility can fail and in any case it is almost meaningless. Another possible source of failure is to have a targetValue that is not attainable with any volatility, e.g. a targetValue lower than the intrinsic value in the case of American options.

11.172.1.2 void performCalculations () const [protected, virtual]**Warning:**

this method simply launches the engine and copies the returned value into NPV_. It does **not** set is-Expired_. This should be taken care of by redefining this method in derived classes and calling this implementation after checking for validity and only if the check succeeded.

Reimplemented from [QuantLib::Option](#).

The documentation for this class was generated from the following files:

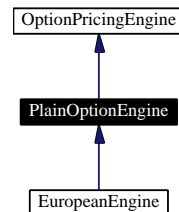
- [plainoption.hpp](#)
- [plainoption.cpp](#)

11.173 QuantLib::Pricers::PlainOptionEngine Class Reference

base class for plain option pricing engines.

```
#include <plainoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::PlainOptionEngine:



Public Methods

- [Arguments](#) * **parameters** ()
- void **validateParameters** () const
- const [Results](#) * **results** () const

Protected Attributes

- [Instruments::PlainOptionParameters](#) **parameters_**
- [Instruments::PlainOptionResults](#) **results_**

11.173.1 Detailed Description

Derived engines only need to implement the `calculate()` method

The documentation for this class was generated from the following files:

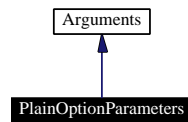
- [plainoption.hpp](#)
- [plainoption.cpp](#)

11.174 QuantLib::Instruments::PlainOptionParameters Class Reference

parameters for plain option calculation.

```
#include <plainoption.hpp>
```

Inheritance diagram for QuantLib::Instruments::PlainOptionParameters:



Public Methods

- **PlainOptionParameters** ()

Public Attributes

- Option::Type **type**
- double **underlying**
- double **strike**
- [Spread](#) **dividendYield**
- [Rate](#) **riskFreeRate**
- [Time](#) **residualTime**
- double **volatility**

The documentation for this class was generated from the following file:

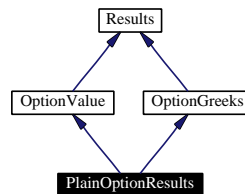
- [plainoption.hpp](#)

11.175 QuantLib::Instruments::PlainOptionResults Class Reference

results from plain option calculation.

```
#include <plainoption.hpp>
```

Inheritance diagram for QuantLib::Instruments::PlainOptionResults:



The documentation for this class was generated from the following file:

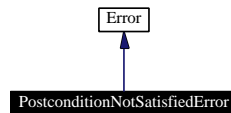
- [plainoption.hpp](#)

11.176 QuantLib::PostconditionNotSatisfiedError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::PostconditionNotSatisfiedError:



Public Methods

- **PostconditionNotSatisfiedError** (const std::string &what=’')

11.176.1 Detailed Description

Thrown upon an unsatisfied postcondition.

The documentation for this class was generated from the following file:

- [errors.hpp](#)

11.177 QuantLib::PreconditionNotSatisfiedError Class Reference

Specialized error.

```
#include <errors.hpp>
```

Inheritance diagram for QuantLib::PreconditionNotSatisfiedError:



Public Methods

- **PreconditionNotSatisfiedError** (const std::string &what=’')

11.177.1 Detailed Description

Thrown upon an unsatisfied precondition.

The documentation for this class was generated from the following file:

- [errors.hpp](#)

11.178 QuantLib::Utilities::processing_iterator Class Template Reference

Iterator mapping a unary function to an underlying sequence.

```
#include <processingiterator.hpp>
```

Public Types

- typedef UnaryFunction::result_type **value_type**
- typedef const value_type * **pointer**
- typedef const value_type & **reference**

Public Methods

- **processing_iterator** (const Iterator &, const UnaryFunction &)

Dereferencing

- reference **operator** * () const
- pointer **operator** → () const

Random access

- value_type **operator**[] (int) const

Increment and decrement

- processing_iterator & **operator**++ ()
- processing_iterator **operator**++ (int)
- processing_iterator & **operator**-- ()
- processing_iterator **operator**-- (int)
- processing_iterator & **operator**+= (difference_type)
- processing_iterator & **operator**-= (difference_type)
- processing_iterator **operator**+ (difference_type)
- processing_iterator **operator**- (difference_type)

Difference

- difference_type **operator**- (const processing_iterator< Iterator, UnaryFunction > &)

Comparisons

- bool **operator**== (const processing_iterator< Iterator, UnaryFunction > &)
- bool **operator**!= (const processing_iterator< Iterator, UnaryFunction > &)
- bool **operator**< (const processing_iterator< Iterator, UnaryFunction > &)
- bool **operator**> (const processing_iterator< Iterator, UnaryFunction > &)
- bool **operator**<= (const processing_iterator< Iterator, UnaryFunction > &)
- bool **operator**>= (const processing_iterator< Iterator, UnaryFunction > &)

Public Attributes

- typedef< Iterator >::difference_type **difference_type**

Related Functions

(Note that these are not member functions.)

- `processing_iterator< Iterator, UnaryFunction > make_processing_iterator` (`Iterator it`, `UnaryFunction p`)
helper function to create processing iterators.

11.178.1 Detailed Description

template<class Iterator, class UnaryFunction> class QuantLib::Utilities::processing_iterator< Iterator, UnaryFunction >

This iterator advances an underlying iterator and returns the values obtained by applying a unary function to the values such iterator points to.

This class was implemented based on Christopher Baus and Thomas Becker, *Custom Iterators for the STL*, included in the proceedings of the First Workshop on C++ Template Programming, Erfurt, Germany, 2000 (<http://www.oonumerics.org/tmpw00/>)

The documentation for this class was generated from the following file:

- [processingiterator.hpp](#)

11.179 QuantLib::RandomNumbers::RandomArrayGenerator Class Template Reference

Generates random arrays from a random number generator.

```
#include <randomarraygenerator.hpp>
```

Public Types

- typedef [MonteCarlo::Sample](#)< [Array](#) > `sample_type`

Public Methods

- **RandomArrayGenerator** (const [Array](#) &variance, long seed=0)
- **RandomArrayGenerator** (const [Math::Matrix](#) &covariance, long seed=0)
- const `sample_type` & **next** () const
- int **size** () const

```
template<class RNG> class QuantLib::RandomNumbers::RandomArrayGenerator< RNG >
```

The documentation for this class was generated from the following file:

- [randomarraygenerator.hpp](#)

11.180 QuantLib::RateFormatter Class Reference

Formats rates for output.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toString` (double rate, int precision=5)

11.180.1 Detailed Description

Formatting is in percentage form (xx.xxxxx%)

The documentation for this class was generated from the following files:

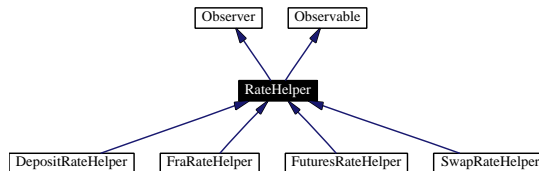
- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.181 QuantLib::TermStructures::RateHelper Class Reference

base class for rate helpers.

```
#include <ratehelpers.hpp>
```

Inheritance diagram for QuantLib::TermStructures::RateHelper:



Public Methods

- **RateHelper** (const [RelinkableHandle](#)< [MarketElement](#) > "e)
- **RateHelper** (double quote)
- virtual **~RateHelper** ()

RateHelper interface

- double **quoteError** () const
- virtual double **impliedQuote** () const=0
- virtual [DiscountFactor](#) **discountGuess** () const
- virtual void **setTermStructure** ([TermStructure](#) *)
sets the term structure to be used for pricing.
- virtual [Date](#) **maturity** () const=0
maturity date.

Observer interface

- void **update** ()

Protected Attributes

- [RelinkableHandle](#)< [MarketElement](#) > **quote_**
- [TermStructure](#) * **termStructure_**

11.181.1 Detailed Description

This class provides an abstraction for the instruments used to bootstrap a term structure. It is advised that a rate helper for an instrument contains an instance of the actual instrument class to ensure consistency between the algorithms used during bootstrapping and later instrument pricing. This is not yet fully enforced in the available rate helpers, though - only [SwapRateHelper](#) contains a Swap instrument for the time being.

11.181.2 Member Function Documentation

11.181.2.1 void setTermStructure ([TermStructure](#) *) [virtual]

Warning:

Being a pointer and not a [Handle](#), the term structure is not guaranteed to remain allocated for the whole life of the rate helper. It is responsibility of the programmer to ensure that the pointer remains valid. It is advised that rate helpers be used only in term structure constructors, setting the term structure to **this**, i.e., the one being constructed.

Reimplemented in [QuantLib::TermStructures::DepositRateHelper](#).

11.181.2.2 void update () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following files:

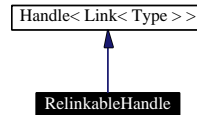
- [ratehelpers.hpp](#)
- [ratehelpers.cpp](#)

11.182 QuantLib::RelinkableHandle Class Template Reference

Globally accessible relinkable pointer.

```
#include <relinkablehandle.hpp>
```

Inheritance diagram for QuantLib::RelinkableHandle:



Public Methods

- **RelinkableHandle** (const **Handle**< Type > &h=**Handle**< Type >(), bool registerAsObserver=true)
- void **linkTo** (const **Handle**< Type > &h, bool registerAsObserver=true)
- const **Handle**< Type > & **operator →** () const
dereferencing.
- bool **isNull** () const
Checks if the contained handle points to anything.

11.182.1 Detailed Description

```
template<class Type> class QuantLib::RelinkableHandle< Type >
```

An instance of this class can be relinked to another **Handle**: such change will be propagated to all the copies of the instance.

Precondition:

Class "Type" must inherit from Observable

11.182.2 Constructor & Destructor Documentation

11.182.2.1 RelinkableHandle (const **Handle**< Type > &h = **Handle**< Type >(), bool *registerAsObserver* = true) [inline]

Warning:

see the documentation of **Link** for issues relatives to registerAsObserver.

11.182.3 Member Function Documentation

11.182.3.1 void linkTo (const **Handle**< Type > &h, bool *registerAsObserver* = true) [inline]

Warning:

see the documentation of **Link** for issues relatives to registerAsObserver.

The documentation for this class was generated from the following file:

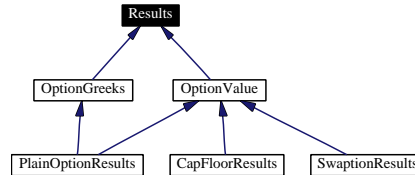
- [relinkablehandle.hpp](#)

11.183 QuantLib::Results Class Reference

base class for generic result groups.

```
#include <argsandresults.hpp>
```

Inheritance diagram for QuantLib::Results:



Public Methods

- virtual `~Results()`

The documentation for this class was generated from the following file:

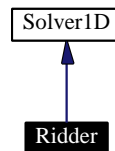
- [argsandresults.hpp](#)

11.184 QuantLib::Solvers1D::Ridder Class Reference

Ridder 1-D solver.

```
#include <ridder.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::Ridder:



The documentation for this class was generated from the following files:

- [ridder.hpp](#)
- [ridder.cpp](#)

11.185 QuantLib::Math::RiskMeasures Class Reference

Interface for risk functions.

```
#include <riskmeasures.hpp>
```

Public Methods

- **RiskMeasures** ()
- double **potentialUpside** (double *percentile*, double *mean*, double *std*) const
- double **valueAtRisk** (double *percentile*, double *mean*, double *std*) const
- double **expectedShortfall** (double *percentile*, double *mean*, double *std*) const
- double **shortfall** (double *target*, double *mean*, double *std*) const
- double **averageShortfall** (double *target*, double *mean*, double *std*) const

11.185.1 Member Function Documentation

11.185.1.1 double **potentialUpside** (double *percentile*, double *mean*, double *std*) const [inline]

Precondition:

percentile must be in range 90%-100%

11.185.1.2 double **valueAtRisk** (double *percentile*, double *mean*, double *std*) const [inline]

Precondition:

percentile must be in range 90%-100%

11.185.1.3 double **expectedShortfall** (double *percentile*, double *mean*, double *std*) const
[inline]

Precondition:

percentile must be in range 90%-100%

The documentation for this class was generated from the following file:

- [riskmeasures.hpp](#)

11.186 QuantLib::RiskStatistics Class Reference

Risk analysis tool.

```
#include <riskstatistics.hpp>
```

Public Methods

Inspectors

- [Size](#) **samples** () const
- double **weightSum** () const
- double **mean** () const
- double **variance** () const
- double **standardDeviation** () const
- double **errorEstimate** () const
- double **skewness** () const
- double **kurtosis** () const
- double **min** () const
- double **max** () const
- double [potentialUpside](#) (double percentile) const
returns the Potential-Up-Front at a given percentile.
- double [valueAtRisk](#) (double percentile) const
returns the Value-At-Risk at a given percentile.
- double [expectedShortfall](#) (double percentile) const
returns the Expected Shortfall at a given percentile.
- double [shortfall](#) (double target) const
returns the Shortfall (observations below target).
- double [averageShortfall](#) (double target) const
returns the Average Shortfall (averaged shortfallness).

Modifiers

- void [add](#) (double value, double weight=1.0)
- template<class DataIterator> void **addSequence** (DataIterator begin, DataIterator end)
- template<class DataIterator, class WeightIterator> void [addSequence](#) (DataIterator begin, DataIterator end, WeightIterator wbegin)
adds a sequence of data to the set, each with its weight.
- void **reset** ()

11.186.1 Detailed Description

It can accumulate a set of data and return risk quantities as Value-At-Risk, Expected Shortfall, Shortfall, Average Shortfall, plus statistic quantities as mean, variance, std. deviation, skewness, kurtosis.

11.186.2 Member Function Documentation

11.186.2.1 `void add (double value, double weight = 1.0) [inline]`

Precondition:

weights must be positive or null

The documentation for this class was generated from the following file:

- [riskstatistics.hpp](#)

11.187 QuantLib::MonteCarlo::Sample Struct Template Reference

weighted sample.

```
#include <sample.hpp>
```

Public Methods

- **Sample** (const T &value, double weight)

Public Attributes

- T **value**
- double **weight**

```
template<class T> struct QuantLib::MonteCarlo::Sample< T >
```

The documentation for this struct was generated from the following file:

- [sample.hpp](#)

11.188 QuantLib::Scheduler Class Reference

Date scheduler.

```
#include <scheduler.hpp>
```

Public Types

- typedef std::vector< [Date](#) >::const_iterator **const_iterator**

Public Methods

- **Scheduler** (const [Calendar](#) &calendar, const [Date](#) &startDate, const [Date](#) &endDate, int frequency, [RollingConvention](#) rollingConvention, bool isAdjusted, const [Date](#) &stubDate=[Date](#)())
- [Size](#) **size** () const
- const [Date](#) & **date** (int i) const
- bool **isRegular** ([Size](#) i) const
- const_iterator **begin** () const
- const_iterator **end** () const

The documentation for this class was generated from the following files:

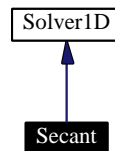
- [scheduler.hpp](#)
- [scheduler.cpp](#)

11.189 QuantLib::Solvers1D::Secant Class Reference

secant 1-D solver.

```
#include <secant.hpp>
```

Inheritance diagram for QuantLib::Solvers1D::Secant:



The documentation for this class was generated from the following files:

- [secant.hpp](#)
- [secant.cpp](#)

11.190 QuantLib::Math::SegmentIntegral Class Reference

Integral of a one-dimensional function.

```
#include <segmentintegral.hpp>
```

Public Methods

- **SegmentIntegral** ([Size](#) intervals)
- double **operator()** (const [ObjectiveFunction](#) &f, double a, double b) const

11.190.1 Detailed Description

Warning:

the use of this class is not recommended since it will be redesigned in one of the next minor releases.

Todo:

Redesign as a template function.

Examples:

[EuropeanOption.cpp](#).

The documentation for this class was generated from the following files:

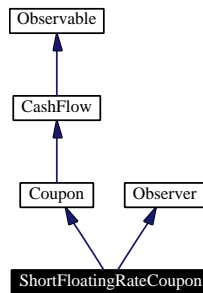
- [segmentintegral.hpp](#)
- [segmentintegral.cpp](#)

11.191 QuantLib::CashFlows::ShortFloatingRateCoupon Class Reference

short coupon at par on a term structure.

```
#include <shortfloatingcoupon.hpp>
```

Inheritance diagram for QuantLib::CashFlows::ShortFloatingRateCoupon:



Public Methods

- **ShortFloatingRateCoupon** (double nominal, const [Handle](#)< [Indexes::Xibor](#) > &index, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure, const [Date](#) &startDate, const [Date](#) &endDate, int fixingDays, [Spread](#) spread=0.0, const [Date](#) &refPeriodStart=[Date](#)(), const [Date](#) &refPeriodEnd=[Date](#)())

CashFlow interface

- double [amount](#) () const
returns the amount of the cash flow.

Coupon interface

- double [accruedAmount](#) (const [Date](#) &) const
accrued amount at the given date.

Inspectors

- const [Handle](#)< [Indexes::Xibor](#) > & **index** () const
- **Rate fixing** () const
- [Spread](#) **spread** () const

Observer interface

- void [update](#) ()

11.191.1 Detailed Description

Warning:

This class does not perform any date adjustment, i.e., the start and end date passed upon construction should be already rolled to a business day.

11.191.2 Member Function Documentation

11.191.2.1 `double amount () const` [virtual]

Note:

The amount is not discounted, i.e., it is the actual amount paid at the cash flow date.

Implements [QuantLib::CashFlow](#).

11.191.2.2 `void update ()` [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following files:

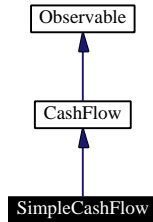
- [shortfloatingcoupon.hpp](#)
- [shortfloatingcoupon.cpp](#)

11.192 QuantLib::CashFlows::SimpleCashFlow Class Reference

Predetermined cash flow.

```
#include <simplecashflow.hpp>
```

Inheritance diagram for QuantLib::CashFlows::SimpleCashFlow:



Public Methods

- **SimpleCashFlow** (double amount, const [Date](#) &date)
- double [amount](#) () const
returns the amount of the cash flow.
- [Date](#) [date](#) () const
returns the date at which the cash flow is settled.

11.192.1 Detailed Description

This cash flow pays a predetermined amount at a given date.

11.192.2 Member Function Documentation

11.192.2.1 double amount () const [inline, virtual]

Note:

The amount is not discounted, i.e., it is the actual amount paid at the cash flow date.

Implements [QuantLib::CashFlow](#).

The documentation for this class was generated from the following file:

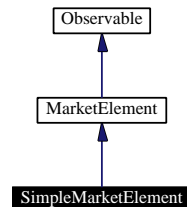
- [simplecashflow.hpp](#)

11.193 QuantLib::SimpleMarketElement Class Reference

market element returning a stored value.

```
#include <marketelement.hpp>
```

Inheritance diagram for QuantLib::SimpleMarketElement:



Public Methods

- **SimpleMarketElement** (double value)

Market element interface

- double **value** () const
returns the current value.

Modifiers

- void **setValue** (double value)

The documentation for this class was generated from the following file:

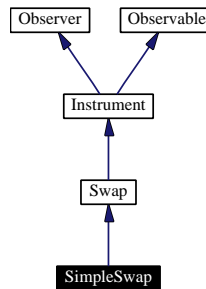
- [marketelement.hpp](#)

11.194 QuantLib::Instruments::SimpleSwap Class Reference

Simple fixed-rate vs Libor swap.

```
#include <swapswap.hpp>
```

Inheritance diagram for QuantLib::Instruments::SimpleSwap:



Public Methods

- **SimpleSwap** (bool payFixedRate, const [Date](#) &startDate, int n, [TimeUnit](#) units, const [Calendar](#) &calendar, [RollingConvention](#) rollingConvention, double nominal, int fixedFrequency, [Rate](#) fixedRate, bool fixedIsAdjusted, const [DayCounter](#) &fixedDayCount, int floatingFrequency, const [Handle](#)< [Indexes::Xibor](#) > &index, int indexFixingDays, [Spread](#) spread, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure, const std::string &isinCode="", const std::string &description="")
- [Rate](#) **fairRate** () const
- [Spread](#) **spread** () const
- double **fixedLegBPS** () const
- double **floatingLegBPS** () const
- [Rate](#) **fixedRate** () const
- double **nominal** () const
- const [Date](#) & **maturity** () const
- bool **payFixedRate** () const
- const std::vector< [Handle](#)< [CashFlow](#) > > & **fixedLeg** () const
- const std::vector< [Handle](#)< [CashFlow](#) > > & **floatingLeg** () const

The documentation for this class was generated from the following file:

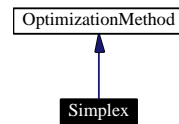
- [swapswap.hpp](#)

11.195 QuantLib::Optimization::Simplex Class Reference

Multi-dimensionnal [Simplex](#) class.

```
#include <simplex.hpp>
```

Inheritance diagram for QuantLib::Optimization::Simplex:



Public Methods

- [Simplex](#) (double *lambda*, double *tol*)
- virtual `~Simplex` ()
- double **extrapolate** ([OptimizationProblem](#) &P, [Size](#) iHighest, double factor)
- virtual void [minimize](#) ([OptimizationProblem](#) &P)
minimize the optimization problem P.

11.195.1 Constructor & Destructor Documentation

11.195.1.1 Simplex (double *lambda*, double *tol*) [inline]

Constructor taking as input λ as the characteristic length and *tol* as the precision

The documentation for this class was generated from the following files:

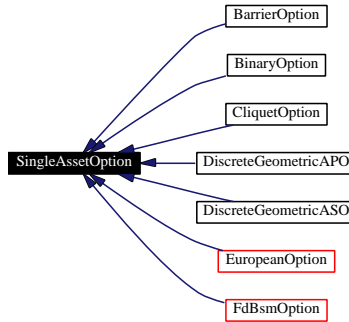
- [simplex.hpp](#)
- [simplex.cpp](#)

11.196 QuantLib::Pricers::SingleAssetOption Class Reference

Black-Scholes-Merton option.

```
#include <singleassetoption.hpp>
```

Inheritance diagram for QuantLib::Pricers::SingleAssetOption:



Public Methods

- **SingleAssetOption** (Option::Type type, double underlying, double strike, [Spread](#) dividendYield, [Rate](#) riskFreeRate, [Time](#) residualTime, double volatility)
- virtual \sim **SingleAssetOption** ()
- virtual void **setVolatility** (double newVolatility)
- virtual void **setRiskFreeRate** ([Rate](#) newRate)
- virtual void **setDividendYield** ([Rate](#) newDividendYield)
- virtual double **value** () const=0
- virtual double **delta** () const=0
- virtual double **gamma** () const=0
- virtual double **theta** () const=0
- virtual double **vega** () const
- virtual double **rho** () const
- virtual double **dividendRho** () const
- double **impliedVolatility** (double targetValue, double accuracy=1e-4, [Size](#) maxEvaluations=100, double minVol=0.0001, double maxVol=4.0) const
- virtual [Handle](#)< SingleAssetOption > **clone** () const=0

Protected Attributes

- Option::Type **type_**
- double **underlying_**
- double **strike_**
- [Spread](#) **dividendYield_**
- [Rate](#) **riskFreeRate_**
- [Time](#) **residualTime_**
- double **volatility_**
- bool **hasBeenCalculated_**
- double **rho_**
- double **dividendRho_**

- double **vega**_
- bool **rhoComputed**_
- bool **dividendRhoComputed**_
- bool **vegaComputed**_

Static Protected Attributes

- const double **dVolMultiplier**_ = 0.0001
- const double **dRMultiplier**_ = 0.0001

Friends

- class **VolatilityFunction**

11.196.1 Member Function Documentation

11.196.1.1 **double impliedVolatility** (double *targetValue*, double *accuracy* = 1e-4, [Size](#) *maxEvaluations* = 100, double *minVol* = 0.0001, double *maxVol* = 4.0) const

Warning:

Options with a gamma that changes sign have values that are **not** monotonic in the volatility, e.g. binary options. In these cases impliedVolatility can fail and in any case is meaningless. Another possible source of failure is to have a targetValue that is not attainable with any volatility, e.g. a targetValue lower than the intrinsic value in the case of American options.

The documentation for this class was generated from the following files:

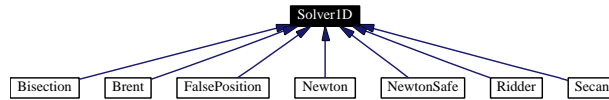
- [singleassetoption.hpp](#)
- [singleassetoption.cpp](#)

11.197 QuantLib::Solver1D Class Reference

Abstract base class for 1-D solvers.

```
#include <solver1d.hpp>
```

Inheritance diagram for QuantLib::Solver1D:



Public Methods

- **Solver1D** ()
- virtual \sim **Solver1D** ()

Modifiers

- double **solve** (const **ObjectiveFunction** &f, double xAccuracy, double guess, double step) const
- double **solve** (const **ObjectiveFunction** &f, double xAccuracy, double guess, double xMin, double xMax) const
- void **setMaxEvaluations** (int evaluations)
- void **setLowBound** (double lowBound)
sets the lower bound for the function domain.
- void **setHiBound** (double hiBound)
sets the upper bound for the function domain.

Protected Methods

- virtual double **solve_** (const **ObjectiveFunction** &f, double xAccuracy) const=0

Protected Attributes

- double **root_**
- double **xMin_**
- double **xMax_**
- double **fxMin_**
- double **fxMax_**
- int **maxEvaluations_**
- int **evaluationNumber_**

11.197.1 Member Function Documentation

11.197.1.1 double solve (const **ObjectiveFunction** &f, double xAccuracy, double guess, double step) const

This method returns the zero of the **ObjectiveFunction** f, determined with the given accuracy (i.e., x is considered a zero if $|f(x)| < accuracy$). This method contains a bracketing routine to which an initial guess must be supplied as well as a step used to scan the range of the possible bracketing values.

11.197.1.2 `double solve (const ObjectiveFunction &f, double xAccuracy, double guess, double xMin, double xMax) const`

This method returns the zero of the [ObjectiveFunction](#) `f`, determined with the given accuracy (i.e., x is considered a zero if $|f(x)| < accuracy$). An initial guess must be supplied, as well as two values which must bracket the zero (i.e., either $f(x_{min}) > 0$ && $f(x_{max}) < 0$, or $f(x_{min}) < 0$ && $f(x_{max}) > 0$ must be true).

11.197.1.3 `void setMaxEvaluations (int evaluations) [inline]`

This method sets the maximum number of function evaluations for the bracketing routine. An [Error](#) is thrown if a bracket is not found after this number of evaluations.

11.197.1.4 `virtual double solve_ (const ObjectiveFunction &f, double xAccuracy) const` `[protected, pure virtual]`

This method must be implemented in derived classes and contains the actual code which searches for the zeroes of the [ObjectiveFunction](#). It assumes that:

- `xMin_` and `xMax_` form a valid bracket;
- `fxMin_` and `fxMax_` contain the values of the function in `xMin_` and `xMax_`;
- `root_` was initialized to a valid initial guess.

Implemented in [QuantLib::Solvers1D::Bisection](#).

The documentation for this class was generated from the following files:

- [solver1d.hpp](#)
- [solver1d.cpp](#)

11.198 QuantLib::Math::Statistics Class Reference

Statistic tool.

```
#include <statistics.hpp>
```

Public Methods

- **Statistics ()**

Inspectors

- **Size samples ()** const
number of samples collected.
- **double weightSum ()** const
sum of data weights.
- **double mean ()** const
- **double variance ()** const
- **double standardDeviation ()** const
- **double downsideVariance ()** const
- **double downsideDeviation ()** const
- **double errorEstimate ()** const
- **double skewness ()** const
- **double kurtosis ()** const
- **double min ()** const
- **double max ()** const

Modifiers

- **void add (double value, double weight=1.0)**
adds a datum to the set, possibly with a weight.
- **template<class DataIterator> void addSequence (DataIterator begin, DataIterator end)**
adds a sequence of data to the set.
- **template<class DataIterator, class WeightIterator> void addSequence (DataIterator begin, DataIterator end, WeightIterator wbegin)**
adds a sequence of data to the set, each with its weight.
- **void reset ()**
resets the data to a null set.

11.198.1 Detailed Description

It can accumulate a set of data and return statistic quantities as mean, variance, std. deviation, skewness, and kurtosis.

Examples:

[DiscreteHedging.cpp](#).

11.198.2 Member Function Documentation

11.198.2.1 `double mean () const [inline]`

returns the mean, defined as

$$\langle x \rangle = \frac{\sum w_i x_i}{\sum w_i}.$$

11.198.2.2 `double variance () const [inline]`

returns the variance, defined as

$$\frac{N}{N-1} \left\langle (x - \langle x \rangle)^2 \right\rangle.$$

11.198.2.3 `double standardDeviation () const [inline]`

returns the standard deviation σ , defined as the square root of the variance.

11.198.2.4 `double downsideVariance () const [inline]`

returns the downside variance, defined as

$$\frac{N}{N-1} \times \frac{\sum_{i=1}^N \theta \times x_i^2}{\sum_{i=1}^N w_i}$$

, where $\theta = 0$ if $x > 0$ and $\theta = 1$ if $x < 0$

11.198.2.5 `double downsideDeviation () const [inline]`

returns the downside deviation, defined as the square root of the downside variance.

11.198.2.6 `double errorEstimate () const [inline]`

returns the error estimate ϵ , defined as the square root of the ratio of the variance to the number of samples.

11.198.2.7 `double skewness () const [inline]`

returns the skewness, defined as

$$\frac{N^2}{(N-1)(N-2)} \frac{\left\langle (x - \langle x \rangle)^3 \right\rangle}{\sigma^3}.$$

The above evaluates to 0 for a Gaussian distribution.

11.198.2.8 double kurtosis () const [inline]

returns the excess kurtosis, defined as

$$\frac{N(N+1)}{(N-1)(N-2)(N-3)} \frac{\langle (x - \langle x \rangle)^4 \rangle}{\sigma^4} - \frac{3(N-1)^2}{(N-2)(N-3)}.$$

The above evaluates to 0 for a Gaussian distribution.

11.198.2.9 double min () const [inline]

returns the minimum sample value

11.198.2.10 double max () const [inline]

returns the maximum sample value

11.198.2.11 void add (double *value*, double *weight* = 1.0) [inline]**Precondition:**

weights must be positive or null

The documentation for this class was generated from the following files:

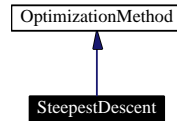
- [statistics.hpp](#)
- [statistics.cpp](#)

11.199 QuantLib::Optimization::SteepestDescent Class Reference

Multi-dimensionnal Steepest Descend class.

```
#include <steepestdescent.hpp>
```

Inheritance diagram for QuantLib::Optimization::SteepestDescent:



Public Methods

- [SteepestDescent](#) ()
default default constructor (msvc bug).
- [SteepestDescent](#) (const [Handle](#)< [LineSearch](#) > &lineSearch)
default constructor.
- virtual [~SteepestDescent](#) ()
destructor.
- virtual void [minimize](#) ([OptimizationProblem](#) &P)
minimize the optimization problem P.

11.199.1 Detailed Description

User has to provide line-search method and optimization end criteria

search direction = $-f'(x)$

The documentation for this class was generated from the following files:

- [steepestdescent.hpp](#)
- [steepestdescent.cpp](#)

11.200 QuantLib::FiniteDifferences::StepCondition Class Template Reference

condition to be applied at every time step.

```
#include <stepcondition.hpp>
```

Public Methods

- virtual `~StepCondition ()`
- virtual void `applyTo` (arrayType &a, [Time](#) t) const=0

```
template<class arrayType> class QuantLib::FiniteDifferences::StepCondition< arrayType >
```

The documentation for this class was generated from the following file:

- [stepcondition.hpp](#)

11.201 QuantLib::Utilities::stepping_iterator Class Template Reference

Iterator advancing in constant steps.

```
#include <steppingiterator.hpp>
```

Public Methods

- **stepping_iterator** (const RandomAccessIterator &, difference_type step)

Dereferencing

- reference **operator *** () const
- pointer **operator →** () const

Random access

- reference **operator[]** (int) const

Increment and decrement

- stepping_iterator & **operator++** ()
- stepping_iterator **operator++** (int)
- stepping_iterator & **operator--** ()
- stepping_iterator **operator--** (int)
- stepping_iterator & **operator+=** (difference_type)
- stepping_iterator & **operator-=** (difference_type)
- stepping_iterator< RandomAccessIterator > **operator+** (difference_type)
- stepping_iterator< RandomAccessIterator > **operator-** (difference_type)

Difference

- difference_type **operator-** (const stepping_iterator< RandomAccessIterator > &)

Comparisons

- bool **operator==** (const stepping_iterator< RandomAccessIterator > &)
- bool **operator!=** (const stepping_iterator< RandomAccessIterator > &)
- bool **operator<** (const stepping_iterator< RandomAccessIterator > &)
- bool **operator>** (const stepping_iterator< RandomAccessIterator > &)
- bool **operator<=** (const stepping_iterator< RandomAccessIterator > &)
- bool **operator>=** (const stepping_iterator< RandomAccessIterator > &)

Public Attributes

- typedef< RandomAccessIterator >::difference_type **difference_type**
- typedef< RandomAccessIterator >::pointer **pointer**
- typedef< RandomAccessIterator >::reference **reference**

Related Functions

(Note that these are not member functions.)

- `stepping_iterator< Iterator > make_stepping_iterator` (Iterator it, typename stepping_iterator< Iterator >::difference_type step)
helper function to create stepping iterators.

11.201.1 Detailed Description

`template<class RandomAccessIterator> class QuantLib::Utilities::stepping_iterator< RandomAccessIterator >`

This iterator advances an underlying random access iterator in steps of n positions, where n is an integer given upon construction.

The documentation for this class was generated from the following file:

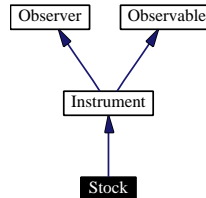
- [steppingiterator.hpp](#)

11.202 QuantLib::Instruments::Stock Class Reference

Simple stock class.

```
#include <stock.hpp>
```

Inheritance diagram for QuantLib::Instruments::Stock:



Public Methods

- **Stock** (const [RelinkableHandle](#)< [MarketElement](#) > "e, const std::string &isinCode, const std::string &description)

Protected Methods

- void [performCalculations](#) () const

11.202.1 Member Function Documentation

11.202.1.1 void performCalculations () const [protected, virtual]

This method must implement any calculations which must be (re)done in order to calculate the NPV of the instrument.

Implements [QuantLib::Instrument](#).

The documentation for this class was generated from the following files:

- [stock.hpp](#)
- [stock.cpp](#)

11.203 QuantLib::StringFormatter Class Reference

Formats strings as lower- or uppercase.

```
#include <dataformatters.hpp>
```

Static Public Methods

- `std::string toLowercase` (const std::string &s)
- `std::string toUppercase` (const std::string &s)

The documentation for this class was generated from the following files:

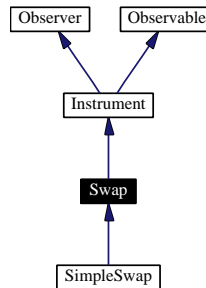
- [dataformatters.hpp](#)
- [dataformatters.cpp](#)

11.204 QuantLib::Instruments::Swap Class Reference

Interest rate swap.

```
#include <swap.hpp>
```

Inheritance diagram for QuantLib::Instruments::Swap:



Public Methods

- **Swap** (const std::vector< [Handle](#)< [CashFlow](#) > > &firstLeg, const std::vector< [Handle](#)< [CashFlow](#) > > &secondLeg, const [RelinkableHandle](#)< [TermStructure](#) > &termStructure, const std::string &isinCode="", const std::string &description="")
- double **firstLegBPS** () const
- double **secondLegBPS** () const

Protected Methods

- void [performCalculations](#) () const

Protected Attributes

- std::vector< [Handle](#)< [CashFlow](#) > > **firstLeg_**
- std::vector< [Handle](#)< [CashFlow](#) > > **secondLeg_**
- [RelinkableHandle](#)< [TermStructure](#) > **termStructure_**
- double **firstLegBPS_**
- double **secondLegBPS_**

11.204.1 Detailed Description

The cash flows belonging to the first leg are payed; the ones belonging to the first leg are received.

11.204.2 Member Function Documentation

11.204.2.1 void performCalculations () const [protected, virtual]

This method must implement any calculations which must be (re)done in order to calculate the NPV of the instrument.

Implements [QuantLib::Instrument](#).

The documentation for this class was generated from the following files:

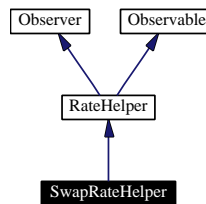
- [swap.hpp](#)
- [swap.cpp](#)

11.205 QuantLib::TermStructures::SwapRateHelper Class Reference

swap rate.

```
#include <ratehelpers.hpp>
```

Inheritance diagram for QuantLib::TermStructures::SwapRateHelper:



Public Methods

- **SwapRateHelper** (const [RelinkableHandle](#)< [MarketElement](#) > &rate, int settlementDays, int lengthInYears, const [Calendar](#) &calendar, [RollingConvention](#) convention, int fixedFrequency, bool fixedIsAdjusted, const [DayCounter](#) &fixedDayCount, int floatingFrequency)
- **SwapRateHelper** (double rate, int settlementDays, int lengthInYears, const [Calendar](#) &calendar, [RollingConvention](#) convention, int fixedFrequency, bool fixedIsAdjusted, const [DayCounter](#) &fixedDayCount, int floatingFrequency)
- double **impliedQuote** () const
- [Date](#) **maturity** () const
maturity date.
- void **setTermStructure** ([TermStructure](#) *)
sets the term structure to be used for pricing.

Protected Attributes

- int **settlementDays**_
- int **lengthInYears**_
- [Calendar](#) **calendar**_
- [RollingConvention](#) **convention**_
- int **fixedFrequency**_
- int **floatingFrequency**_
- bool **fixedIsAdjusted**_
- [DayCounter](#) **fixedDayCount**_
- [Date](#) **settlement**_
- [Handle](#)< [Instruments::SimpleSwap](#) > **swap**_
- [RelinkableHandle](#)< [TermStructure](#) > **termStructureHandle**_

11.205.1 Detailed Description

Warning:

This class assumes that today's date does not change between calls of [setTermStructure](#)().

11.205.2 Member Function Documentation

11.205.2.1 void setTermStructure ([TermStructure](#) *) [virtual]

Warning:

Being a pointer and not a [Handle](#), the term structure is not guaranteed to remain allocated for the whole life of the rate helper. It is responsibility of the programmer to ensure that the pointer remains valid. It is advised that rate helpers be used only in term structure constructors, setting the term structure to **this**, i.e., the one being constructed.

Reimplemented from [QuantLib::TermStructures::RateHelper](#).

The documentation for this class was generated from the following files:

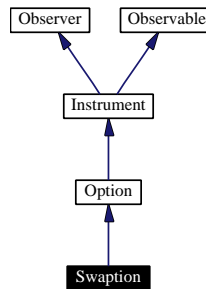
- [ratehelpers.hpp](#)
- [ratehelpers.cpp](#)

11.206 QuantLib::Instruments::Swaption Class Reference

[Swaption](#) class.

```
#include <swaption.hpp>
```

Inheritance diagram for QuantLib::Instruments::Swaption:



Public Methods

- **Swaption** (const [Handle](#)< [SimpleSwap](#) > &swap, const [Exercise](#) &exercise, const [Relinkable-Handle](#)< [TermStructure](#) > &termStructure, const [Handle](#)< [OptionPricingEngine](#) > &engine)

Protected Methods

- void [performCalculations](#) () const
- void [setupEngine](#) () const

11.206.1 Member Function Documentation

11.206.1.1 void performCalculations () const [protected, virtual]

Warning:

this method simply launches the engine and copies the returned value into NPV_. It does **not** set is-Expired_. This should be taken care of by redefining this method in derived classes and calling this implementation after checking for validity and only if the check succeeded.

Reimplemented from [QuantLib::Option](#).

The documentation for this class was generated from the following files:

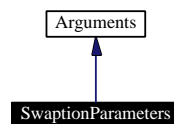
- [swaption.hpp](#)
- [swaption.cpp](#)

11.207 QuantLib::Instruments::SwaptionParameters Class Reference

parameters for swaption calculation.

```
#include <swaption.hpp>
```

Inheritance diagram for QuantLib::Instruments::SwaptionParameters:



Public Methods

- **SwaptionParameters** ()
- void **validate** () const

Public Attributes

- bool **payFixed**
- [Rate](#) **fairRate**
- [Rate](#) **fixedRate**
- double **fixedBPS**
- std::vector< [Time](#) > **fixedPayTimes**
- std::vector< double > **fixedCoupons**
- std::vector< [Time](#) > **floatingResetTimes**
- std::vector< [Time](#) > **floatingPayTimes**
- std::vector< double > **nominals**
- Exercise::Type **exerciseType**
- std::vector< [Time](#) > **exerciseTimes**

The documentation for this class was generated from the following file:

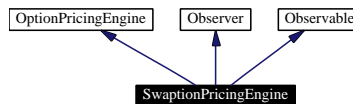
- [swaption.hpp](#)

11.208 QuantLib::Pricers::SwaptionPricingEngine Class Template Reference

base class for swaption pricing engines.

```
#include <swaption.hpp>
```

Inheritance diagram for QuantLib::Pricers::SwaptionPricingEngine:



Public Methods

- **SwaptionPricingEngine** ()
- **SwaptionPricingEngine** (const [Handle](#)< ModelType > &model)
- [Arguments](#) * **parameters** ()
- const [Results](#) * **results** () const
- void **validateParameters** () const
- void **setModel** (const [Handle](#)< ModelType > &model)
- void [update](#) ()

Protected Attributes

- [Instruments::SwaptionParameters](#) **parameters_**
- [Instruments::SwaptionResults](#) **results_**
- [Handle](#)< ModelType > **model_**

11.208.1 Detailed Description

```
template<class ModelType> class QuantLib::Pricers::SwaptionPricingEngine< ModelType >
```

Derived engines only need to implement the `calculate()` method

11.208.2 Member Function Documentation

11.208.2.1 void update() [inline, virtual]

This method must be implemented in derived classes. An instance of `Observer` does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

The documentation for this class was generated from the following file:

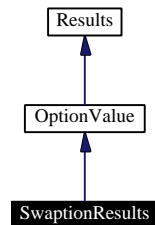
- [swaption.hpp](#)

11.209 QuantLib::Instruments::SwaptionResults Class Reference

results from swaption calculation.

```
#include <swaption.hpp>
```

Inheritance diagram for QuantLib::Instruments::SwaptionResults:



The documentation for this class was generated from the following file:

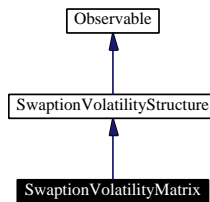
- [swaption.hpp](#)

11.210 QuantLib::Volatilities::SwaptionVolatilityMatrix Class Reference

Swaption at-the-money volatility matrix.

```
#include <swaptionvolmatrix.hpp>
```

Inheritance diagram for QuantLib::Volatilities::SwaptionVolatilityMatrix:



Public Methods

- **SwaptionVolatilityMatrix** (const [Date](#) &today'sDate, const std::vector< [Date](#) > &exerciseDates, const std::vector< [Period](#) > &lengths, const [Math::Matrix](#) &volatilities, const [DayCounter](#) &dayCounter=[DayCounters::Thirty360](#)())
- [Date](#) today'sDate () const
returns today's date.
- [DayCounter](#) dayCounter () const
returns the day counter used for internal date/time conversions.
- const std::vector< [Date](#) > & exerciseDates () const
- const std::vector< [Period](#) > & lengths () const

11.210.1 Detailed Description

This class provides the at-the-money volatility for a given swaption by interpolating a volatility matrix whose elements are the market volatilities of a set of swaption with given exercise date and length.

Todo:

Either add correct copy behavior or inhibit copy. Right now, a copied instance would end up with its own copy of the exercise date and length vector but an interpolation pointing to the original ones.

The documentation for this class was generated from the following file:

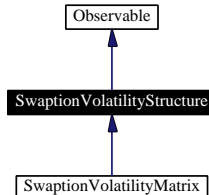
- [swaptionvolmatrix.hpp](#)

11.211 QuantLib::SwaptionVolatilityStructure Class Reference

Swaption volatility structure.

```
#include <swaptionvolstructure.hpp>
```

Inheritance diagram for QuantLib::SwaptionVolatilityStructure:



Public Methods

- virtual `~SwaptionVolatilityStructure ()`
- virtual `Date todaysDate () const=0`
returns today's date.
- virtual `DayCounter dayCounter () const=0`
returns the day counter used for internal date/time conversions.
- double `volatility (const Date &start, const Period &length, Rate strike) const`
returns the volatility for a given starting date and length.
- double `volatility (Time start, Time length, Rate strike) const`
returns the volatility for a given starting time and length.

Protected Methods

- virtual double `volatilityImpl (Time start, Time length, Rate strike) const=0`
implements the actual volatility calculation in derived classes.

11.211.1 Detailed Description

This class is purely abstract and defines the interface of concrete swaption volatility structures which will be derived from this one.

The documentation for this class was generated from the following file:

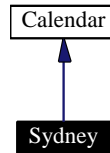
- [swaptionvolstructure.hpp](#)

11.212 QuantLib::Calendars::Sydney Class Reference

Sydney, calendar (New South Wales, Australia).

```
#include <sydney.hpp>
```

Inheritance diagram for QuantLib::Calendars::Sydney:



Public Methods

- **Sydney ()**

11.212.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st (possibly moved to Monday or Tuesday)
- Day after New Year's Day, January 2st (possibly moved to Monday or Tuesday)
- Anniversary Day, Monday nearest January 22nd
- Waitangi Day, February 6th
- Good Friday
- Easter Monday
- ANZAC Day, April 25th
- Queen's Birthday, first Monday in June
- Labour Day, fourth Monday in October
- Christmas, December 25th (possibly moved to Monday or Tuesday)
- Boxing Day, December 26th (possibly moved to Monday or Tuesday)

Note:

The holiday rules for [Sydney](#) were documented by David Gilbert for IDB (<http://www.jrefinery.com/ibd/>)

The documentation for this class was generated from the following file:

- [sydney.hpp](#)

11.213 QuantLib::Math::SymmetricSchurDecomposition Class Reference

symmetric threshold Jacobi algorithm.

```
#include <symmetricschurdecomposition.hpp>
```

Public Methods

- **SymmetricSchurDecomposition** ([Matrix](#) &s)
- [Array](#) **eigenvalues** () const
- [Matrix](#) **eigenvectors** () const

11.213.1 Detailed Description

Given a real symmetric matrix S , the Schur decomposition finds the eigenvalues and eigenvectors of S . If D is the diagonal matrix formed by the eigenvalues and U the unitarian matrix of the eigenvector we can write the Schur decomposition as

$$S = U \cdot D \cdot U^T,$$

where \cdot is the standard matrix product and T is the transpose operator. This class implements the Schur decomposition using the symmetric threshold Jacobi algorithm. For details on the different Jacobi transformations you can start from the great book on matrix computations by Golub and Van Loan: [Matrix](#) computation, second edition The Johns Hopkins University Press

The documentation for this class was generated from the following files:

- [symmetricschurdecomposition.hpp](#)
- [symmetricschurdecomposition.cpp](#)

11.214 QuantLib::Calendars::TARGET Class Reference

TARGET calendar.

```
#include <target.hpp>
```

Inheritance diagram for QuantLib::Calendars::TARGET:



Public Methods

- TARGET ()

11.214.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st
- Good Friday
- Easter Monday
- Labour Day, May 1st
- Christmas, December 25th
- Day of Goodwill, December 26th

The documentation for this class was generated from the following file:

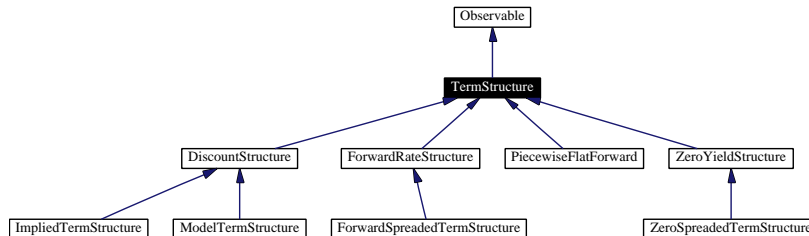
- [target.hpp](#)

11.215 QuantLib::TermStructure Class Reference

Term structure.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::TermStructure:



Public Methods

- virtual `~TermStructure()`

Rates and discount

- `Rate zeroYield` (const `Date` &, bool extrapolate=false) const
zero yield rate at a given date.
- `Rate zeroYield` (Time, bool extrapolate=false) const
zero yield rate at a given time from settlement.
- `DiscountFactor discount` (const `Date` &, bool extrapolate=false) const
discount factor at a given date.
- `DiscountFactor discount` (Time, bool extrapolate=false) const
discount factor at a given time from settlement.
- `Rate forward` (const `Date` &, bool extrapolate=false) const
instantaneous forward rate at a given date.
- `Rate forward` (Time, bool extrapolate=false) const
instantaneous forward rate at a given time from settlement.

Dates

- virtual `Date todaysDate` () const=0
returns today's date.
- virtual int `settlementDays` () const=0
returns the number of settlement days.
- virtual `Calendar calendar` () const=0
returns the calendar for settlement calculation.

- virtual [DayCounter dayCounter](#) () const=0
returns the day counter.
- virtual [Date settlementDate](#) () const=0
returns the settlement date.
- virtual [Date minDate](#) () const=0
returns the earliest date for which the curve can return rates.
- virtual [Date maxDate](#) () const=0
returns the latest date for which the curve can return rates.
- virtual [Time minTime](#) () const=0
returns the earliest time for which the curve can return rates.
- virtual [Time maxTime](#) () const=0
returns the latest date for which the curve can return rates.

Other inspectors

- virtual [Currency currency](#) () const=0
returns the currency upon which the term structure is defined.

Protected Methods

- virtual [Rate zeroYieldImpl](#) ([Time](#), bool extrapolate=false) const=0
implements the actual zero yield calculation in derived classes.
- virtual [DiscountFactor discountImpl](#) ([Time](#), bool extrapolate=false) const=0
implements the actual discount calculation in derived classes.
- virtual [Rate forwardImpl](#) ([Time](#), bool extrapolate=false) const=0
implements the actual forward rate calculation in derived classes.

11.215.1 Detailed Description

This abstract class defines the interface of concrete rate structures which will be derived from this one.

The documentation for this class was generated from the following file:

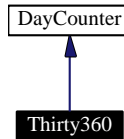
- [termstructure.hpp](#)

11.216 QuantLib::DayCounters::Thirty360 Class Reference

30/360 day count convention.

```
#include <thirty360.hpp>
```

Inheritance diagram for QuantLib::DayCounters::Thirty360:



Public Types

- enum `Convention` { `USA`, `European`, `Italian` }

Public Methods

- `Thirty360` (Convention c=`Thirty360::USA`)

11.216.1 Detailed Description

The day count can be calculated according to US, European, or Italian conventions.

US (NASD) convention: if the starting date is the 31st of a month, it becomes equal to the 30th of the same month. If the ending date is the 31st of a month and the starting date is earlier than the 30th of a month, the ending date becomes equal to the 1st of the next month, otherwise the ending date becomes equal to the 30th of the same month.

European convention: starting dates or ending dates that occur on the 31st of a month become equal to the 30th of the same month.

Italian convention: starting dates or ending dates that occur on February and are greater than 27 become equal to 30 for computational sake.

The documentation for this class was generated from the following files:

- [thirty360.hpp](#)
- [thirty360.cpp](#)

11.217 QuantLib::TimeGrid Class Reference

time grid class.

```
#include <grid.hpp>
```

Public Methods

- **TimeGrid** ()
- **TimeGrid** (const std::list< [Time](#) > ×, [Size](#) steps)
- [Size](#) **findIndex** ([Time](#) t) const
- [Time](#) **dt** ([Size](#) i) const

The documentation for this class was generated from the following file:

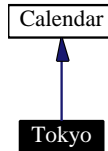
- [grid.hpp](#)

11.218 QuantLib::Calendars::Tokyo Class Reference

Tokyo calendar.

```
#include <tokyo.hpp>
```

Inheritance diagram for QuantLib::Calendars::Tokyo:



Public Methods

- **Tokyo** ()

11.218.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st
- Bank Holiday, January 2nd
- Bank Holiday, January 3rd
- Coming of Age Day, 2nd Monday in January
- National Foundation Day, February 11th
- Vernal Equinox
- Greenery Day, April 29th
- Constitution Memorial Day, May 3rd
- Holiday for a Nation, May 4th
- Children's Day, May 5th
- Marine Day, July 20th
- Respect for the Aged Day, September 15th
- Autumnal Equinox
- Health and Sports Day, 2nd Monday in October
- National Culture Day, November 3rd
- Labor Thanksgiving Day, November 23rd
- Emperor's Birthday, December 23rd
- Bank Holiday, December 31st

Holidays falling on a Sunday are observed on the Monday following except for the Bank Holidays associated with the New Year

Todo:

insert actual rules for Equinoxes

The documentation for this class was generated from the following file:

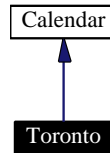
- [tokyo.hpp](#)

11.219 QuantLib::Calendars::Toronto Class Reference

Toronto calendar.

```
#include <toronto.hpp>
```

Inheritance diagram for QuantLib::Calendars::Toronto:



Public Methods

- **Toronto** ()

11.219.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st (possibly moved to Monday)
- Good Friday
- Easter Monday
- Victoria Day, The Monday on or preceding 24 May
- Canada Day, July 1st
- Provincial Holiday, first Monday of August
- Labour Day, first Monday of September
- Thanksgiving Day, second Monday of October
- Remembrance Day, November 11th
- Christmas, December 25th
- Boxing Day, December 26th

The documentation for this class was generated from the following file:

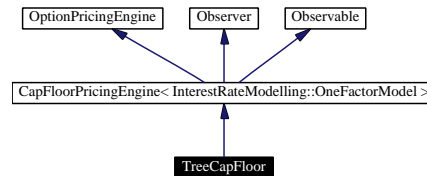
- [toronto.hpp](#)

11.220 QuantLib::Pricers::TreeCapFloor Class Reference

Cap/Floor priced in a tree.

```
#include <treecapfloor.hpp>
```

Inheritance diagram for QuantLib::Pricers::TreeCapFloor:



Public Methods

- **TreeCapFloor** (const [Handle](#)< [InterestRateModelling::OneFactorModel](#) > &model, [Size](#) time-Steps)
- **TreeCapFloor** (const [Handle](#)< [Lattices::Tree](#) > &tree)
- void **calculate** () const

The documentation for this class was generated from the following files:

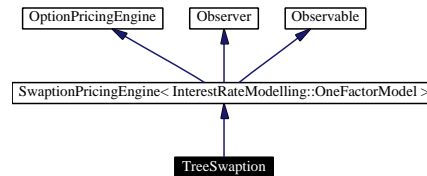
- [treecapfloor.hpp](#)
- [treecapfloor.cpp](#)

11.221 QuantLib::Pricers::TreeSwaption Class Reference

Swaption priced in a tree.

```
#include <treeswaption.hpp>
```

Inheritance diagram for QuantLib::Pricers::TreeSwaption:



Public Methods

- **TreeSwaption** (const [Handle](#)< [InterestRateModelling::OneFactorModel](#) > &model, [Size](#) time-Steps)
- **TreeSwaption** (const [Handle](#)< [Lattices::Tree](#) > &tree)
- void **calculate** () const

The documentation for this class was generated from the following files:

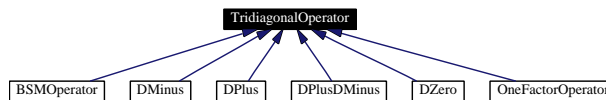
- [treeswaption.hpp](#)
- [treeswaption.cpp](#)

11.222 QuantLib::FiniteDifferences::TridiagonalOperator Class Reference

Base implementation for tridiagonal operator.

```
#include <tridiagonaloperator.hpp>
```

Inheritance diagram for QuantLib::FiniteDifferences::TridiagonalOperator:



Operator interface

- `Array applyTo` (const `Array` &`v`) const
apply operator to a given array.
- `Array solveFor` (const `Array` &`rhs`) const
solve linear system for a given right-hand side.
- `Array SOR` (const `Array` &`rhs`, double `tol`) const
solve linear system with SOR approach.
- `TridiagonalOperator identity` (`Size` `size`)
identity instance.

Public Types

- typedef `Array` `arrayType`

Public Methods

- `TridiagonalOperator` (`Size` `size`=0)
- `TridiagonalOperator` (const `Array` &`low`, const `Array` &`mid`, const `Array` &`high`)

Inspectors

- `Size size` () const
- bool `isTimeDependent` ()

Modifiers

- void `setLowerBC` (const `BoundaryCondition` &`bc`)
- void `setUpperBC` (const `BoundaryCondition` &`bc`)
- void `setFirstRow` (double, double)
- void `setMidRow` (`Size`, double, double, double)
- void `setMidRows` (double, double, double)
- void `setLastRow` (double, double)
- void `setTime` (`Time` `t`)

Protected Attributes

- [Array](#) `diagonal_`
- [Array](#) `belowDiagonal_`
- [Array](#) `aboveDiagonal_`
- [BoundaryCondition](#) `lowerBC_`
- [BoundaryCondition](#) `upperBC_`
- [Handle](#)< [TimeSetter](#) > `timeSetter_`

Friends

- `TridiagonalOperator operator+ (const TridiagonalOperator &)`
- `TridiagonalOperator operator- (const TridiagonalOperator &)`
- `TridiagonalOperator operator+ (const TridiagonalOperator &, const TridiagonalOperator &)`
- `TridiagonalOperator operator- (const TridiagonalOperator &, const TridiagonalOperator &)`
- `TridiagonalOperator operator * (double, const TridiagonalOperator &)`
- `TridiagonalOperator operator * (const TridiagonalOperator &, double)`
- `TridiagonalOperator operator/ (const TridiagonalOperator &, double)`

11.222.1 Detailed Description

Warning:

to use real time-dependant algebra, you must overload the corresponding operators in the inheriting time-dependent class

The documentation for this class was generated from the following files:

- [tridiagonaloperator.hpp](#)
- [tridiagonaloperator.cpp](#)

11.223 QuantLib::FiniteDifferences::TridiagonalOperator::TimeSetter Class Reference

encapsulation of time-setting logic.

```
#include <tridiagonaloperator.hpp>
```

Public Methods

- virtual `~TimeSetter` ()
- virtual void `setTime` ([Time](#) t, [TridiagonalOperator](#) &L) const=0

The documentation for this class was generated from the following file:

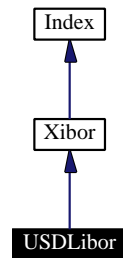
- [tridiagonaloperator.hpp](#)

11.224 QuantLib::Indexes::USDLibor Class Reference

USD Libor index.

```
#include <usdlibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::USDLibor:



Public Methods

- **USDLibor** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

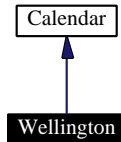
- [usdlibor.hpp](#)

11.225 QuantLib::Calendars::Wellington Class Reference

Wellington calendar.

```
#include <wellington.hpp>
```

Inheritance diagram for QuantLib::Calendars::Wellington:



Public Methods

- **Wellington ()**

11.225.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st (possibly moved to Monday or Tuesday)
- Day after New Year's Day, January 2st (possibly moved to Monday or Tuesday)
- Anniversary Day, Monday nearest January 22nd
- Waitangi Day, February 6th
- Good Friday
- Easter Monday
- ANZAC Day, April 25th
- Queen's Birthday, first Monday in June
- Labour Day, fourth Monday in October
- Christmas, December 25th (possibly moved to Monday or Tuesday)
- Boxing Day, December 26th (possibly moved to Monday or Tuesday)

Note:

The holiday rules for [Wellington](http://www.jrefinery.com/ibd/) were documented by David Gilbert for IDB (<http://www.jrefinery.com/ibd/>)

The documentation for this class was generated from the following file:

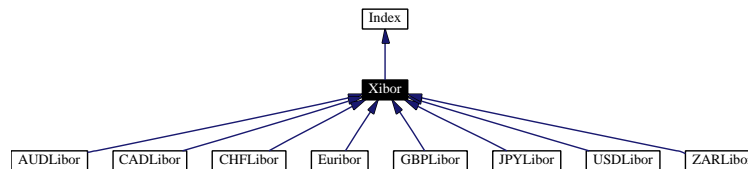
- [wellington.hpp](#)

11.226 QuantLib::Indexes::Xibor Class Reference

base class for libor indexes.

```
#include <xibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::Xibor:



Public Methods

- **Xibor** (const std::string &familyName, int n, [TimeUnit](#) units, int settlementDays, [Currency](#) currency, const [Calendar](#) &calendar, bool isAdjusted, [RollingConvention](#) rollingConvention, const [DayCounter](#) &dayCounter, const [RelinkableHandle](#)< [TermStructure](#) > &h)

Index interface

- **Rate fixing** (const [Date](#) &fixingDate) const
returns the fixing at the given date.

Inspectors

- std::string **name** () const
Returns the name of the index.
- [Period](#) **tenor** () const
- int **settlementDays** () const
- [Currency](#) **currency** () const
- [Calendar](#) **calendar** () const
- bool **isAdjusted** () const
- [RollingConvention](#) **rollingConvention** () const
- [DayCounter](#) **dayCounter** () const

11.226.1 Member Function Documentation

11.226.1.1 **Rate fixing** (const [Date](#) &fixingDate) const [virtual]

Note:

any date passed as arguments must be a value date, i.e., the real calendar date advanced by a number of settlement days.

Implements [QuantLib::Index](#).

11.226.1.2 `std::string name () const` [virtual]

Warning:

This method is used for output and comparison between indexes. It is **not** meant to be used for writing switch-on-type code.

Implements [QuantLib::Index](#).

The documentation for this class was generated from the following files:

- [xibor.hpp](#)
- [xibor.cpp](#)

11.227 QuantLib::Indexes::XiborManager Class Reference

global repository for libor histories.

```
#include <xibormanager.hpp>
```

Static Public Methods

- void **setHistory** (const std::string &name, const [History](#) &)
- const [History](#) & **getHistory** (const std::string &name)
- bool **hasHistory** (const std::string &name)
- std::vector< std::string > **histories** ()

The documentation for this class was generated from the following files:

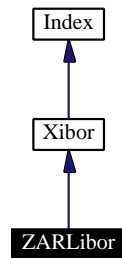
- [xibormanager.hpp](#)
- [xibormanager.cpp](#)

11.228 QuantLib::Indexes::ZARLibor Class Reference

ZAR Libor index (also known as JIBAR, check settlement days).

```
#include <zarlibor.hpp>
```

Inheritance diagram for QuantLib::Indexes::ZARLibor:



Public Methods

- **ZARLibor** (int n, [TimeUnit](#) units, const [RelinkableHandle](#)< [TermStructure](#) > &h)

The documentation for this class was generated from the following file:

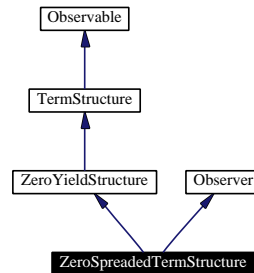
- [zarlibor.hpp](#)

11.229 QuantLib::ZeroSpreadedTermStructure Class Reference

Term structure with an added spread on the zero yield rate.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::ZeroSpreadedTermStructure:



Public Methods

- **ZeroSpreadedTermStructure** (const [RelinkableHandle](#)< [TermStructure](#) > &, const [RelinkableHandle](#)< [MarketElement](#) > &spread)

TermStructure interface

- [Currency](#) **currency** () const
returns the currency upon which the term structure is defined.
- [Date](#) **todayDate** () const
returns today's date.
- [int](#) **settlementDays** () const
returns the number of settlement days.
- [Calendar](#) **calendar** () const
returns the calendar for settlement calculation.
- [DayCounter](#) **dayCounter** () const
returns the day counter.
- [Date](#) **settlementDate** () const
returns the settlement date.
- [Date](#) **maxDate** () const
returns the latest date for which the curve can return rates.
- [Date](#) **minDate** () const
returns the earliest date for which the curve can return rates.
- [Time](#) **maxTime** () const
returns the latest date for which the curve can return rates.
- [Time](#) **minTime** () const

returns the earliest time for which the curve can return rates.

Observer interface

- void [update](#) ()

Protected Methods

- [Rate zeroYieldImpl](#) ([Time](#), bool *extrapolate*=false) const
returns the spreaded zero yield rate.
- [Rate forwardImpl](#) ([Time](#), bool *extrapolate*=false) const
returns the spreaded forward rate.

11.229.1 Detailed Description

Note:

This term structure will remain linked to the original structure, i.e., any changes in the latter will be reflected in this structure as well.

11.229.2 Member Function Documentation

11.229.2.1 void [update](#) () [inline, virtual]

This method must be implemented in derived classes. An instance of Observer does not call this method directly: instead, it will be called by the observables the instance registered with when they need to notify any changes.

Implements [QuantLib::Patterns::Observer](#).

11.229.2.2 [Rate forwardImpl](#) ([Time](#), bool *extrapolate* = false) const [inline, protected, virtual]

Warning:

This method must disappear should the spread become a curve

Reimplemented from [QuantLib::ZeroYieldStructure](#).

The documentation for this class was generated from the following file:

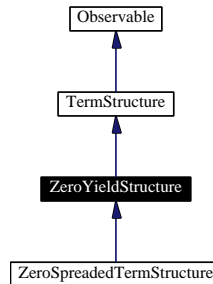
- [termstructure.hpp](#)

11.230 QuantLib::ZeroYieldStructure Class Reference

Zero yield term structure.

```
#include <termstructure.hpp>
```

Inheritance diagram for QuantLib::ZeroYieldStructure:



Public Methods

- virtual `~ZeroYieldStructure()`

Protected Methods

- `DiscountFactor discountImpl(Time, bool extrapolate=false) const`
- `Rate forwardImpl(Time, bool extrapolate=false) const`

11.230.1 Detailed Description

This abstract class acts as an adapter to [TermStructure](#) allowing the programmer to implement only the `zeroYield(Time)` method in derived classes.

11.230.2 Member Function Documentation

11.230.2.1 `DiscountFactor discountImpl(Time, bool extrapolate = false) const` [inline, protected, virtual]

Returns the discount factor for the given date calculating it from the zero yield.

Implements [QuantLib::TermStructure](#).

11.230.2.2 `Rate forwardImpl(Time, bool extrapolate = false) const` [inline, protected, virtual]

Returns the instantaneous forward rate for the given date calculating it from the zero yield.

Implements [QuantLib::TermStructure](#).

Reimplemented in [QuantLib::ZeroSpreadedTermStructure](#).

The documentation for this class was generated from the following file:

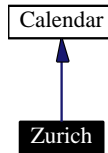
- [termstructure.hpp](#)

11.231 QuantLib::Calendars::Zurich Class Reference

Zurich calendar.

```
#include <zurich.hpp>
```

Inheritance diagram for QuantLib::Calendars::Zurich:



Public Methods

- `Zurich ()`

11.231.1 Detailed Description

Holidays:

- Saturdays
- Sundays
- New Year's Day, January 1st
- Berchtoldstag, January 2nd
- Good Friday
- Easter Monday
- Ascension Day
- Whit Monday
- Labour Day, May 1st
- National Day, August 1st
- Christmas, December 25th
- St. Stephen's Day, December 26th

The documentation for this class was generated from the following file:

- [zurich.hpp](#)

Chapter 12

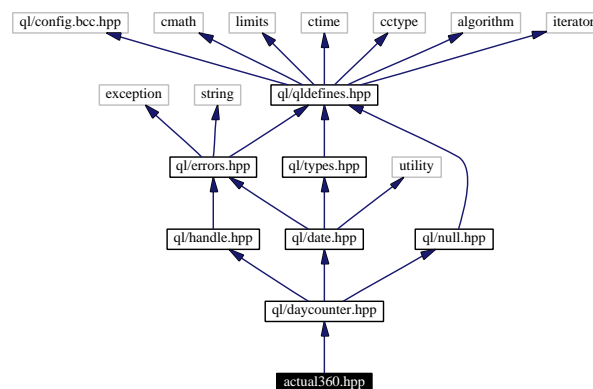
QuantLib File Documentation

12.1 actual360.hpp File Reference

act/360 day counter.

```
#include <ql/daycounter.hpp>
```

Include dependency graph for actual360.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::DayCounters](#)

12.1.1 Detailed Description

Full path:

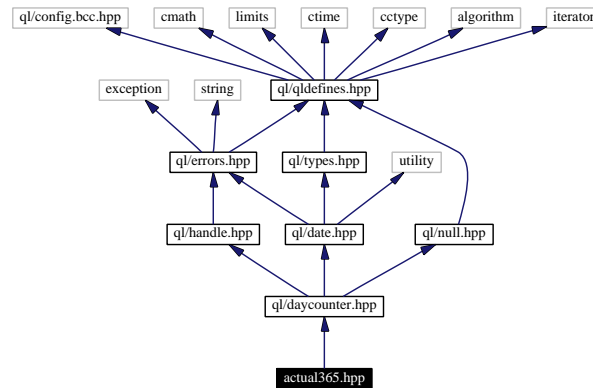
`ql/DayCounters/actual360.hpp`

12.2 actual365.hpp File Reference

act/365 day counter.

```
#include <ql/daycounter.hpp>
```

Include dependency graph for actual365.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::DayCounters](#)

12.2.1 Detailed Description

Full path:

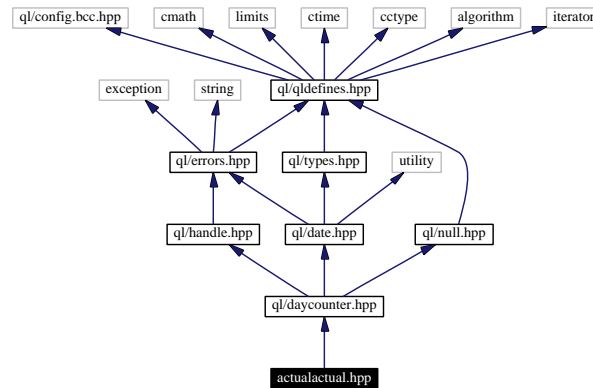
`ql/DayCounters/actual365.hpp`

12.4 actualactual.hpp File Reference

act/act day counters.

```
#include <ql/daycounter.hpp>
```

Include dependency graph for actualactual.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::DayCounters](#)

12.4.1 Detailed Description

Full path:

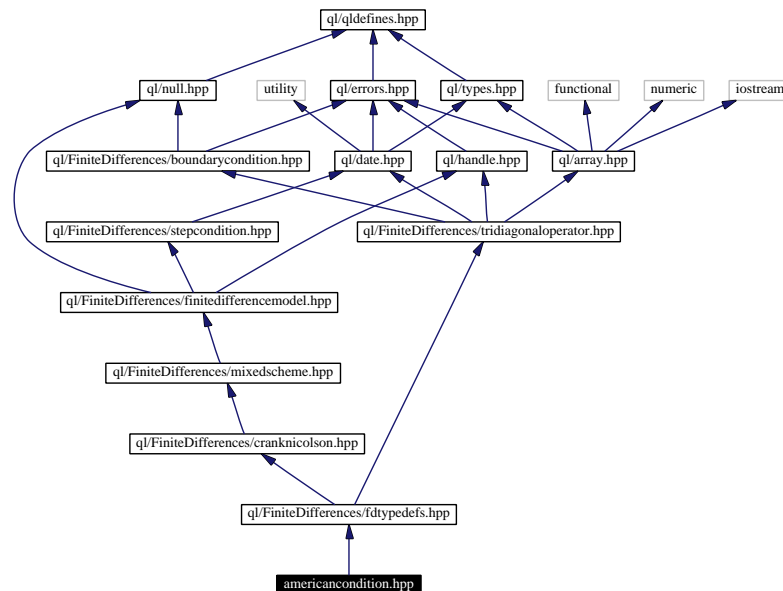
`ql/DayCounters/actualactual.hpp`

12.5 americancondition.hpp File Reference

american option exercise condition.

```
#include <ql/FiniteDifferences/fdtypedefs.hpp>
```

Include dependency graph for americancondition.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.5.1 Detailed Description

Full path:

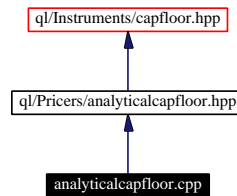
`ql/FiniteDifferences/americancondition.hpp`

12.6 analyticalcapfloor.cpp File Reference

Analytical pricer for caps/floors.

```
#include "ql/Pricers/analyticalcapfloor.hpp"
```

Include dependency graph for analyticalcapfloor.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.6.1 Detailed Description

Full path:

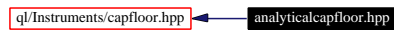
ql/Pricers/analyticalcapfloor.cpp

12.7 analyticalcapfloor.hpp File Reference

Analytical pricer for caps/floors.

```
#include <ql/Instruments/capfloor.hpp>
```

Include dependency graph for analyticalcapfloor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.7.1 Detailed Description

Full path:

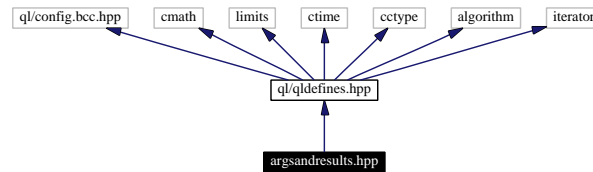
ql/Pricers/analyticalcapfloor.hpp

12.8 argsandresults.hpp File Reference

Base classes for generic arguments and results.

```
#include <ql/qldefines.hpp>
```

Include dependency graph for argsandresults.hpp:



Namespaces

- namespace [QuantLib](#)

12.8.1 Detailed Description

Full path:

`ql/argsandresults.hpp`

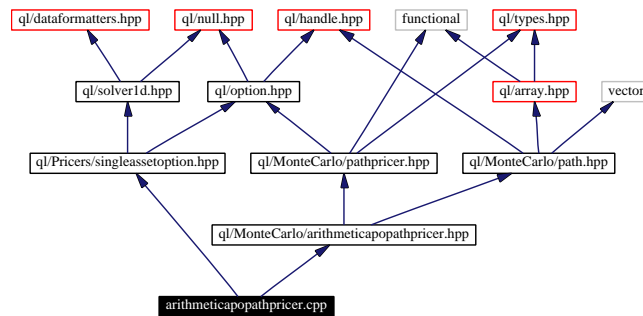
12.9 arithmeticapopathpricer.cpp File Reference

arithmetic average price option path pricer.

```
#include <ql/MonteCarlo/arithmeticapopathpricer.hpp>
```

```
#include <ql/Pricers/singleassetoption.hpp>
```

Include dependency graph for arithmeticapopathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.9.1 Detailed Description

Full path:

ql/MonteCarlo/arithmeticapopathpricer.cpp

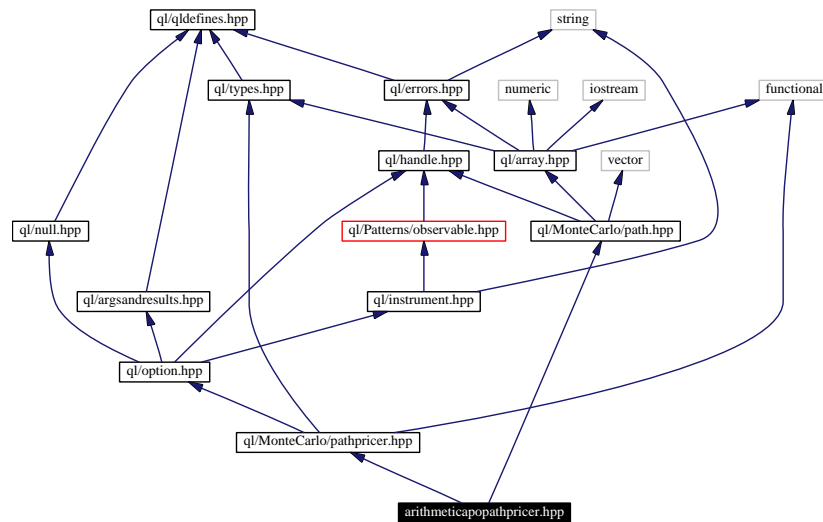
12.10 arithmeticapopathpricer.hpp File Reference

arithmetic average price option path pricer.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/path.hpp>
```

Include dependency graph for arithmeticapopathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.10.1 Detailed Description

Full path:

ql/MonteCarlo/arithmeticapopathpricer.hpp

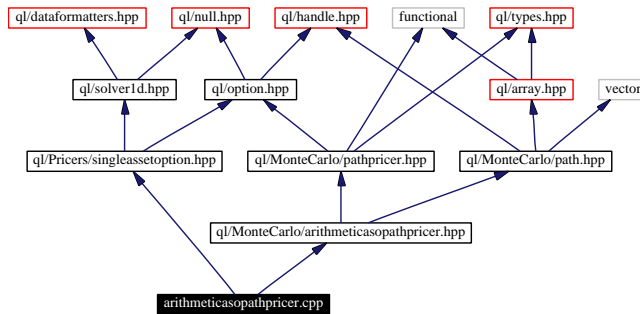
12.11 arithmeticasopathpricer.cpp File Reference

arithmetic average strike option path pricer.

```
#include <ql/MonteCarlo/arithmeticasopathpricer.hpp>
```

```
#include <ql/Pricers/singleassetoption.hpp>
```

Include dependency graph for arithmeticasopathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.11.1 Detailed Description

Full path:

ql/MonteCarlo/arithmeticasopathpricer.cpp

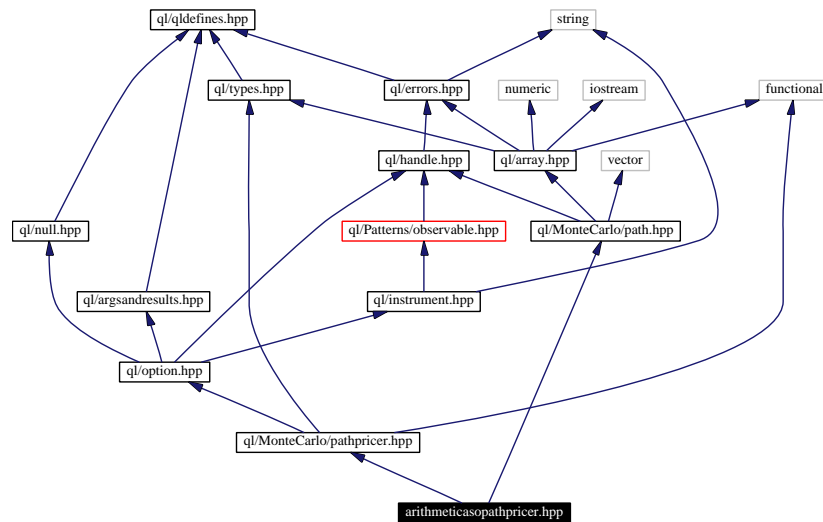
12.12 arithmeticasopathpricer.hpp File Reference

arithmetic average strike option path pricer.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/path.hpp>
```

Include dependency graph for arithmeticasopathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.12.1 Detailed Description

Full path:

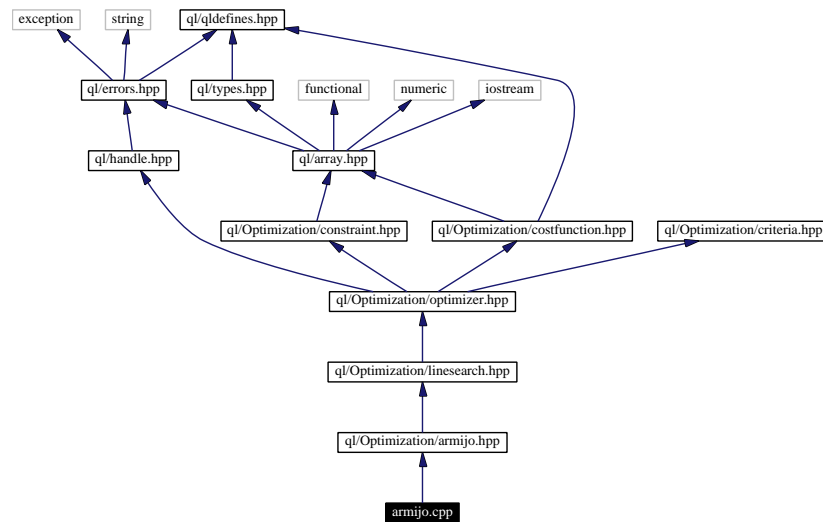
ql/MonteCarlo/arithmeticasopathpricer.hpp

12.13 armijo.cpp File Reference

Armijo line-search class.

```
#include "ql/Optimization/armijo.hpp"
```

Include dependency graph for armijo.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.13.1 Detailed Description

Full path:

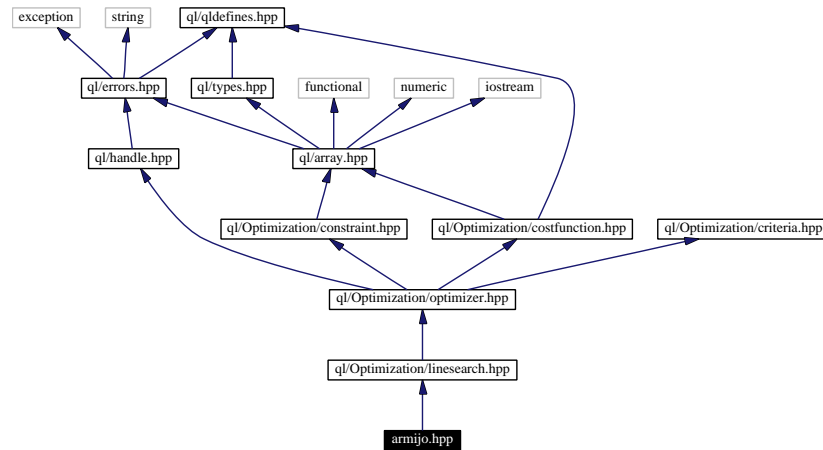
`ql/Optimization/armijo.cpp`

12.14 armijo.hpp File Reference

Armijo line-search class.

```
#include "ql/Optimization/linesearch.hpp"
```

Include dependency graph for armijo.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Optimization**

12.14.1 Detailed Description

Full path:

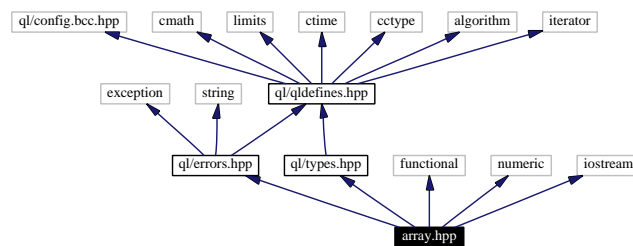
`ql/Optimization/armijo.hpp`

12.15 array.hpp File Reference

1-D array used in linear algebra.

```
#include <ql/types.hpp>
#include <ql/errors.hpp>
#include <functional>
#include <numeric>
#include <iostream>
```

Include dependency graph for array.hpp:



Namespaces

- namespace [QuantLib](#)

12.15.1 Detailed Description

Full path:

`ql/array.hpp`

12.16 audlibor.hpp File Reference

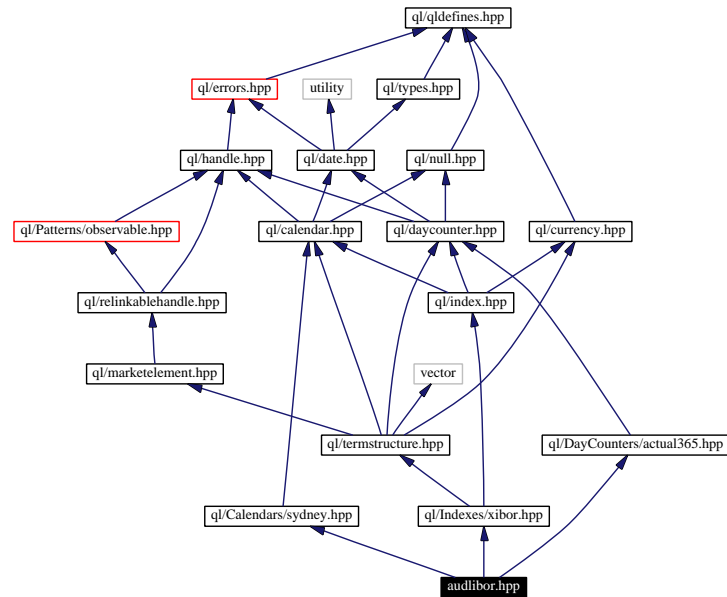
AUD Libor index (check settlement days).

```
#include <ql/Indexes/xibor.hpp>
```

```
#include <ql/Calendars/sydney.hpp>
```

```
#include <ql/DayCounters/actual365.hpp>
```

Include dependency graph for audlibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.16.1 Detailed Description

Full path:

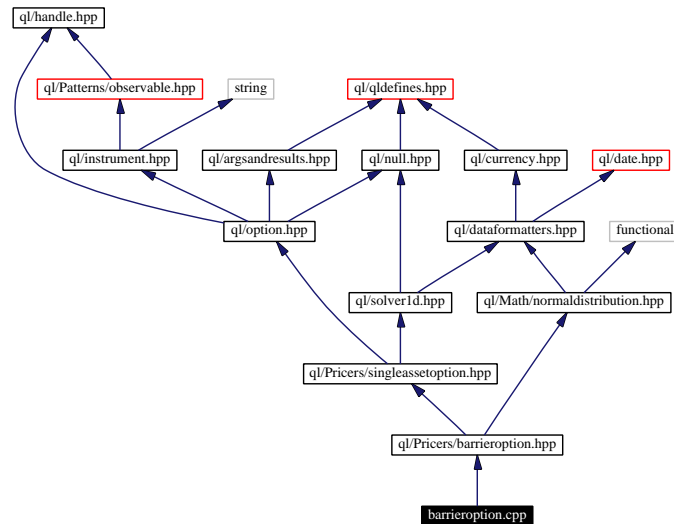
ql/Indexes/audlibor.hpp

12.17 barrieroption.cpp File Reference

barrier option.

```
#include <ql/Pricers/barrieroption.hpp>
```

Include dependency graph for barrieroption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.17.1 Detailed Description

Full path:

`ql/Pricers/barrieroption.cpp`

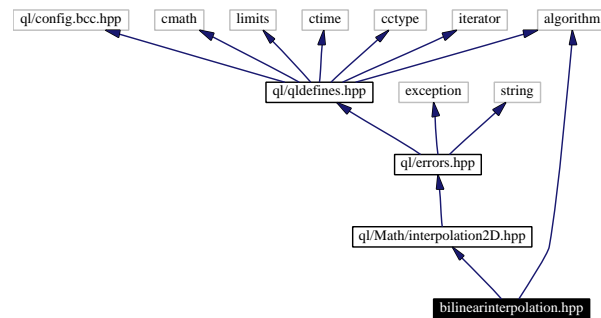
12.21 bilinearinterpolation.hpp File Reference

bilinear interpolation between discrete points.

```
#include <ql/Math/interpolation2D.hpp>
```

```
#include <algorithm>
```

Include dependency graph for bilinearinterpolation.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.21.1 Detailed Description

Full path:

`ql/Math/bilinearinterpolation.hpp`

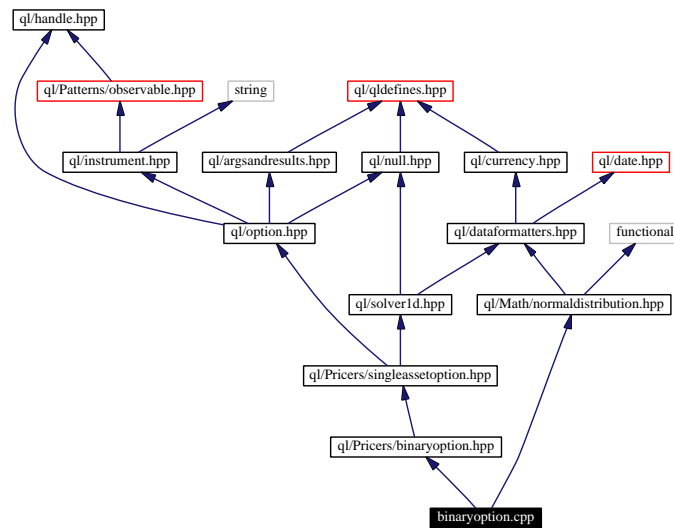
12.22 binaryoption.cpp File Reference

European style cash-or-nothing option.

```
#include <ql/Pricers/binaryoption.hpp>
```

```
#include <ql/Math/normaldistribution.hpp>
```

Include dependency graph for binaryoption.cpp:



Namespaces

- namespace `QuantLib`
- namespace `QuantLib::Pricers`

12.22.1 Detailed Description

Full path:

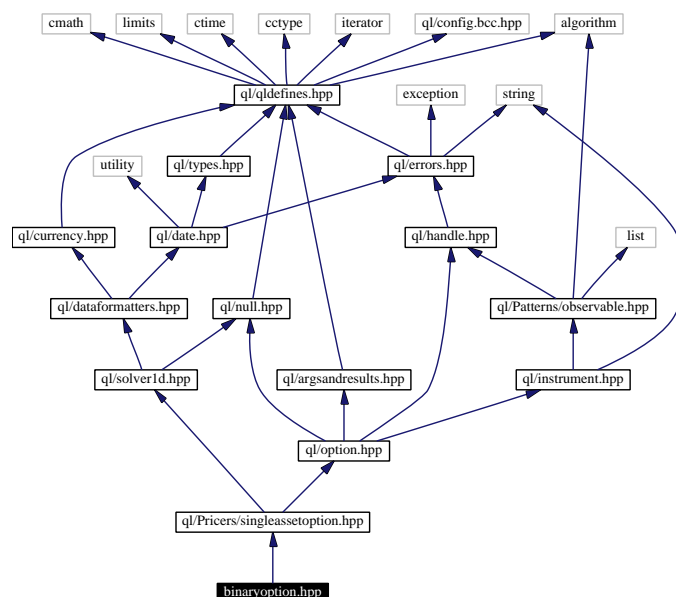
`ql/Pricers/binaryoption.cpp`

12.23 binaryoption.hpp File Reference

European style cash-or-nothing option.

```
#include <ql/Pricers/singleassetoption.hpp>
```

Include dependency graph for binaryoption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib:Pricers](#)

12.23.1 Detailed Description

Full path:

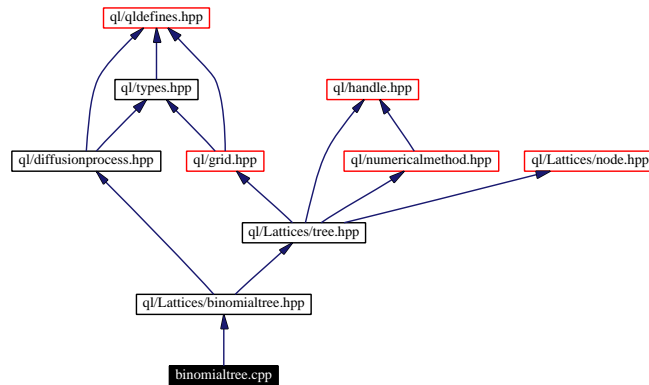
`ql/Pricers/binaryoption.hpp`

12.24 binomialtree.cpp File Reference

Binomial tree class.

```
#include "ql/Lattices/binomialtree.hpp"
```

Include dependency graph for binomialtree.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Lattices**

12.24.1 Detailed Description

Full path:

`ql/Lattices/binomialtree.cpp`

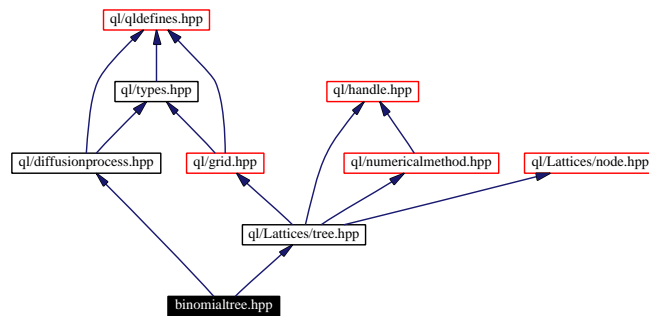
12.25 binomialtree.hpp File Reference

Binomial tree class.

```
#include <ql/diffusionprocess.hpp>
```

```
#include <ql/Lattices/tree.hpp>
```

Include dependency graph for binomialtree.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Lattices`

12.25.1 Detailed Description

Full path:

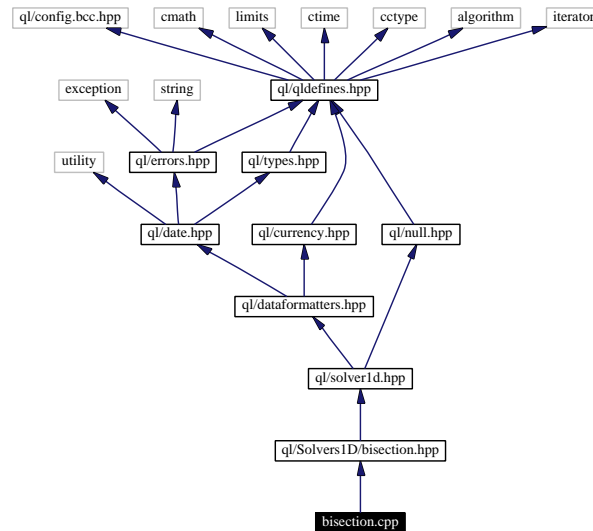
`ql/Lattices/binomialtree.hpp`

12.26 bisection.cpp File Reference

bisection 1-D solver.

```
#include <ql/Solvers1D/bisection.hpp>
```

Include dependency graph for bisection.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.26.1 Detailed Description

Full path:

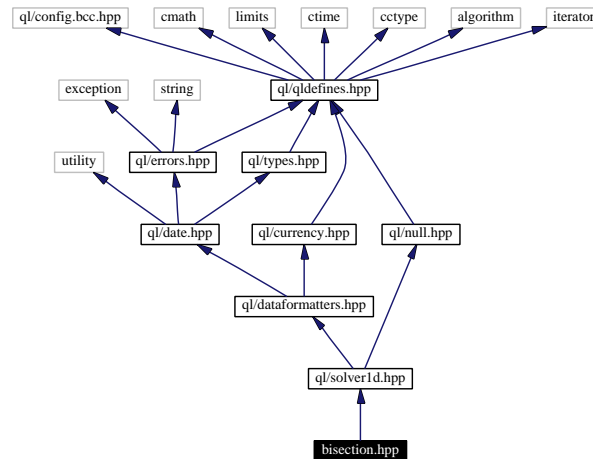
`ql/Solvers1D/bisection.cpp`

12.27 bisection.hpp File Reference

bisection 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for bisection.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.27.1 Detailed Description

Full path:

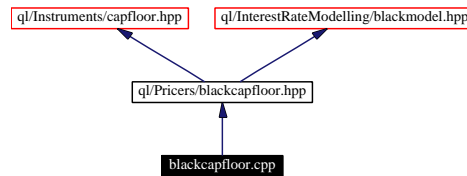
`ql/Solvers1D/bisection.hpp`

12.28 blackcapfloor.cpp File Reference

European capfloor calculated using Black formula.

```
#include "ql/Pricers/blackcapfloor.hpp"
```

Include dependency graph for blackcapfloor.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.28.1 Detailed Description

Full path:

ql/Pricers/blackcapfloor.cpp

12.29 blackcapfloor.hpp File Reference

CapFloor calculated using the Black formula.

```
#include <ql/Instruments/capfloor.hpp>
```

```
#include <ql/InterestRateModelling/blackmodel.hpp>
```

Include dependency graph for blackcapfloor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.29.1 Detailed Description

Full path:

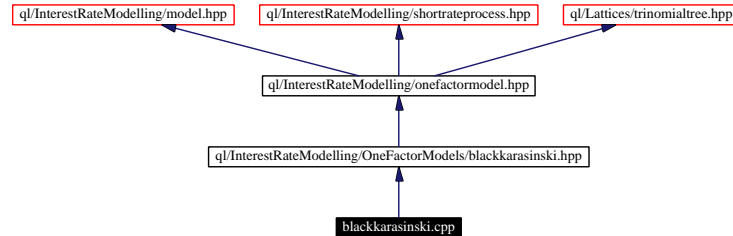
ql/Pricers/blackcapfloor.hpp

12.30 blackkarasinski.cpp File Reference

Black-Karasinski model.

```
#include "ql/InterestRateModelling/OneFactorModels/blackkarasinski.hpp"
```

Include dependency graph for blackkarasinski.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::InterestRateModelling`

12.30.1 Detailed Description

Full path:

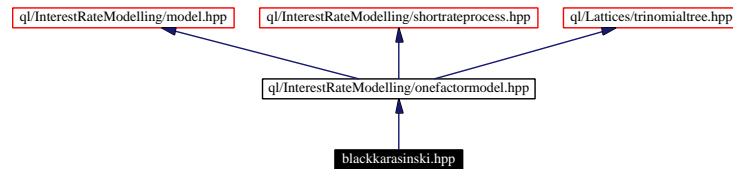
`ql/InterestRateModelling/OneFactorModels/blackkarasinski.cpp`

12.31 blackkarasinski.hpp File Reference

Black-Karasinski model.

```
#include <ql/InterestRateModelling/onefactormodel.hpp>
```

Include dependency graph for blackkarasinski.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.31.1 Detailed Description

Full path:

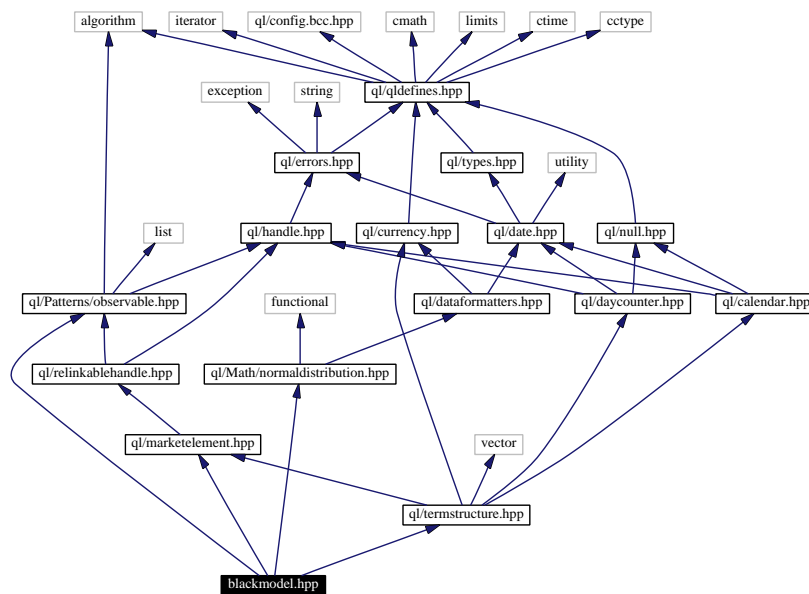
ql/InterestRateModelling/OneFactorModels/blackkarasinski.hpp

12.32 blackmodel.hpp File Reference

Abstract class for Black-type models (market models).

```
#include <ql/marketelement.hpp>
#include <ql/termstructure.hpp>
#include <ql/Math/normaldistribution.hpp>
#include <ql/Patterns/observable.hpp>
```

Include dependency graph for blackmodel.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::InterestRateModelling](#)

12.32.1 Detailed Description

Full path:

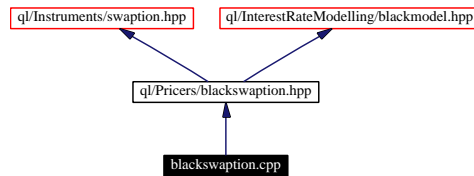
ql/InterestRateModelling/blackmodel.hpp

12.33 blackswaption.cpp File Reference

European swaption calculated using Black formula.

```
#include "ql/Pricers/blackswaption.hpp"
```

Include dependency graph for blackswaption.cpp:



Namespaces

- namespace `QuantLib`
- namespace `QuantLib::Pricers`

12.33.1 Detailed Description

Full path:

`ql/Pricers/blackswaption.cpp`

12.34 blackswaption.hpp File Reference

Swaption calculated using the Black formula.

```
#include <ql/Instruments/swaption.hpp>
```

```
#include <ql/InterestRateModelling/blackmodel.hpp>
```

Include dependency graph for blackswaption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.34.1 Detailed Description

Full path:

`ql/Pricers/blackswaption.hpp`

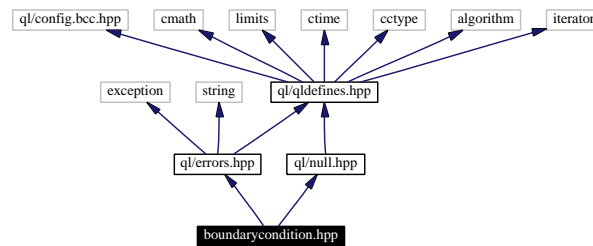
12.35 boundarycondition.hpp File Reference

boundary conditions for differential operators.

```
#include <ql/null.hpp>
```

```
#include <ql/errors.hpp>
```

Include dependency graph for boundarycondition.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.35.1 Detailed Description

Full path:

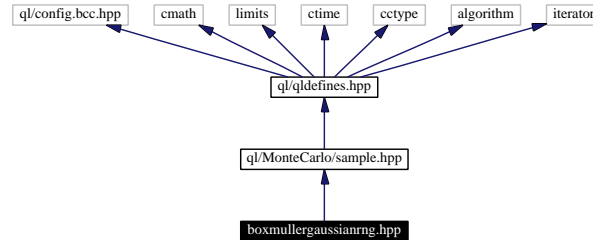
`ql/FiniteDifferences/boundarycondition.hpp`

12.36 boxmullergaussianrng.hpp File Reference

Box-Muller Gaussian random-number generator.

```
#include <ql/MonteCarlo/sample.hpp>
```

Include dependency graph for boxmullergaussianrng.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.36.1 Detailed Description

Full path:

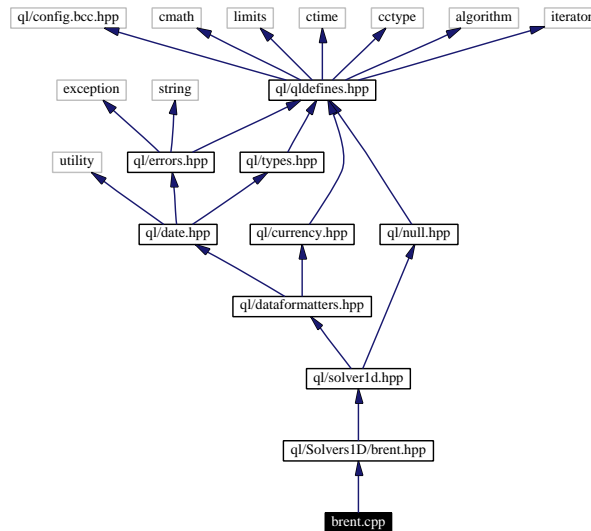
`ql/RandomNumbers/boxmullergaussianrng.hpp`

12.37 brent.cpp File Reference

Brent 1-D solver.

```
#include <ql/Solvers1D/brent.hpp>
```

Include dependency graph for brent.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

Defines

- `#define SIGN(a, b) ((b) >= 0.0 ? QL_FABS(a) : -QL_FABS(a))`

12.37.1 Detailed Description

Full path:

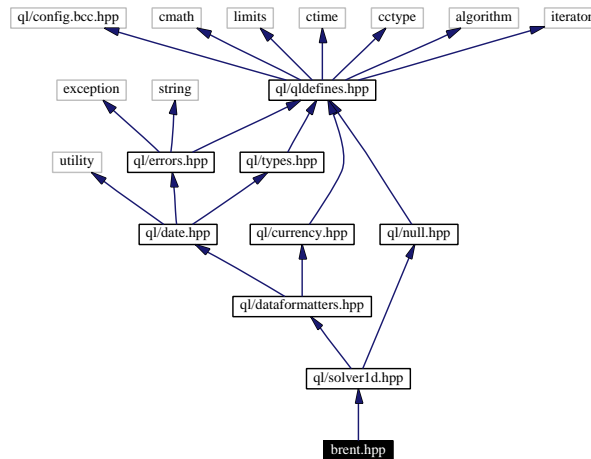
`ql/Solvers1D/brent.cpp`

12.38 brent.hpp File Reference

Brent 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for brent.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.38.1 Detailed Description

Full path:

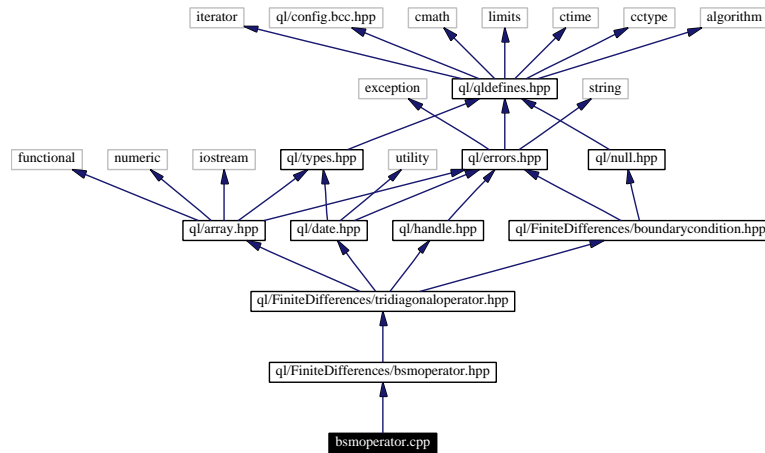
`ql/Solvers1D/brent.hpp`

12.39 bsmoperator.cpp File Reference

differential operator for Black-Scholes-Merton equation.

```
#include <ql/FiniteDifferences/bsmoperator.hpp>
```

Include dependency graph for bsmoperator.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.39.1 Detailed Description

Full path:

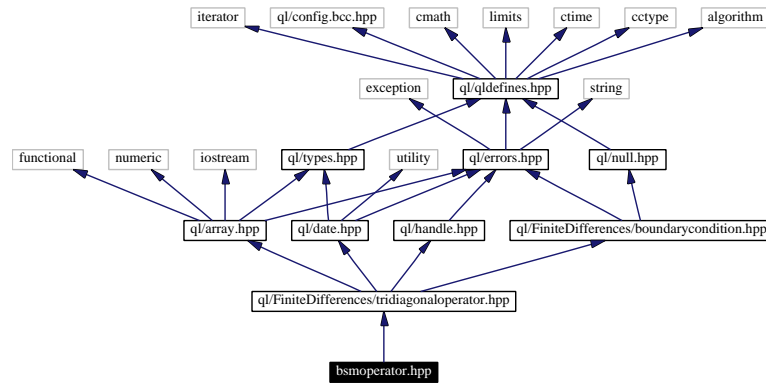
FiniteDifferences/bsmoperator.cpp

12.40 bsmoperator.hpp File Reference

differential operator for Black-Scholes-Merton equation.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

Include dependency graph for bsmoperator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.40.1 Detailed Description

Full path:

ql/FiniteDifferences/bsmoperator.hpp

12.41 cadlibor.hpp File Reference

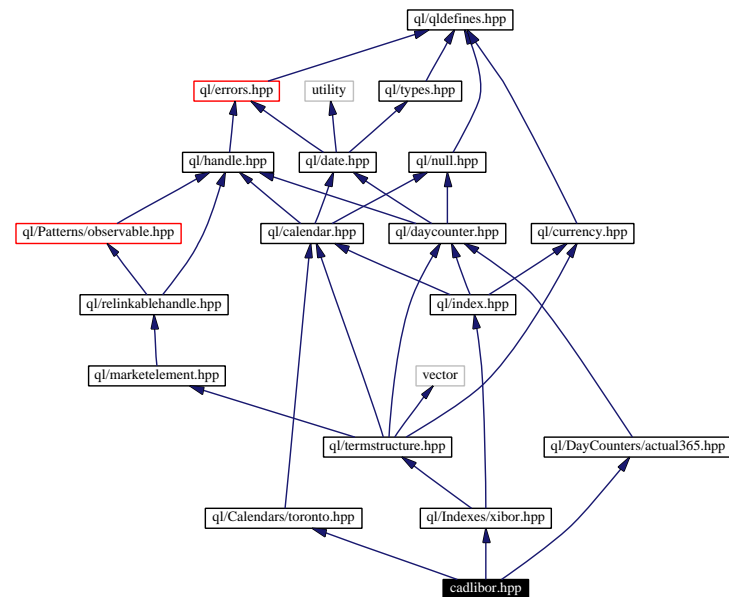
CAD Libor index (Also known as CDOR, check settlement days).

```
#include <ql/Indexes/xibor.hpp>
```

```
#include <ql/Calendars/toronto.hpp>
```

```
#include <ql/DayCounters/actual365.hpp>
```

Include dependency graph for cadlibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.41.1 Detailed Description

Full path:

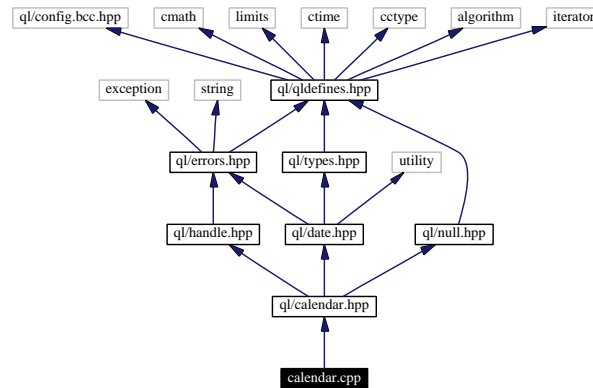
ql/Indexes/cadlibor.hpp

12.42 calendar.cpp File Reference

Abstract calendar class.

```
#include <ql/calendar.hpp>
```

Include dependency graph for calendar.cpp:



Namespaces

- namespace [QuantLib](#)

12.42.1 Detailed Description

Full path:

`ql/calendar.cpp`

12.43 calendar.hpp File Reference

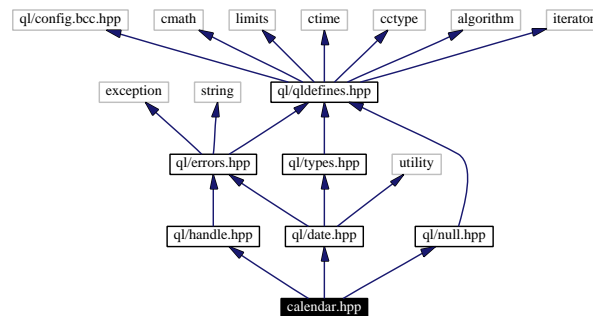
calendar class.

```
#include <ql/date.hpp>
```

```
#include <ql/handle.hpp>
```

```
#include <ql/null.hpp>
```

Include dependency graph for calendar.hpp:



Namespaces

- namespace [QuantLib::Calendars](#)
- namespace [QuantLib](#)

12.43.1 Detailed Description

Full path:

`ql/calendar.hpp`

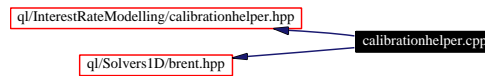
12.44 calibrationhelper.cpp File Reference

Calibration helper class.

```
#include "ql/InterestRateModelling/calibrationhelper.hpp"
```

```
#include "ql/Solvers1D/brent.hpp"
```

Include dependency graph for calibrationhelper.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.44.1 Detailed Description

Full path:

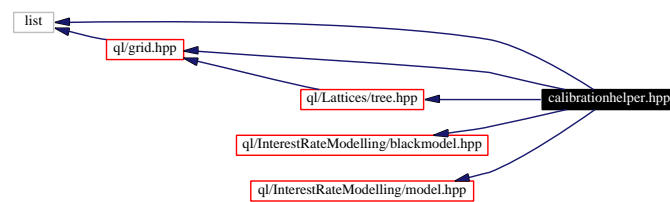
`ql/InterestRateModelling/CalibrationHelpers/calibrationhelper.hpp`

12.45 calibrationhelper.hpp File Reference

Calibration helper class.

```
#include <ql/grid.hpp>
#include <ql/InterestRateModelling/blackmodel.hpp>
#include <ql/InterestRateModelling/model.hpp>
#include <ql/Lattices/tree.hpp>
#include <list>
```

Include dependency graph for calibrationhelper.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.45.1 Detailed Description

Full path:

`ql/InterestRateModelling/calibrationhelper.hpp`

12.47 capfloor.cpp File Reference

European cap and floor class.

```
#include "ql/Instruments/capfloor.hpp"
```

```
#include "ql/CashFlows/floatingratecoupon.hpp"
```

Include dependency graph for capfloor.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.47.1 Detailed Description

Full path:

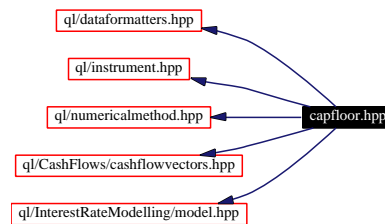
`ql/Instruments/capfloor.cpp`

12.48 capfloor.hpp File Reference

Cap and Floor class.

```
#include "ql/dataformatters.hpp"
#include <ql/instrument.hpp>
#include <ql/numericalmethod.hpp>
#include <ql/CashFlows/cashflowvectors.hpp>
#include <ql/InterestRateModelling/model.hpp>
```

Include dependency graph for capfloor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)
- namespace [QuantLib::Pricers](#)

12.48.1 Detailed Description

Full path:

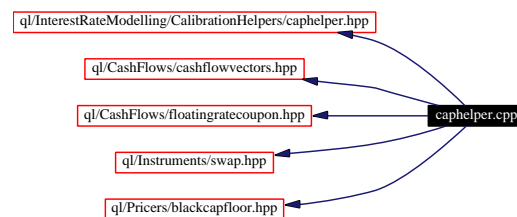
`ql/Instruments/capfloor.hpp`

12.49 caphelper.cpp File Reference

Cap calibration helper.

```
#include "ql/InterestRateModelling/CalibrationHelpers/caphelper.hpp"
#include "ql/CashFlows/cashflowvectors.hpp"
#include "ql/CashFlows/floatingratecoupon.hpp"
#include "ql/Instruments/swap.hpp"
#include "ql/Pricers/blackcapfloor.hpp"
```

Include dependency graph for caphelper.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**
- namespace **QuantLib::InterestRateModelling::CalibrationHelpers**

12.49.1 Detailed Description

Full path:

`ql/InterestRateModelling/CalibrationHelpers/caphelper.hpp`

12.50 caphelper.hpp File Reference

CapHelper calibration helper.

```
#include "ql/InterestRateModelling/calibrationhelper.hpp"
```

```
#include "ql/Instruments/capfloor.hpp"
```

Include dependency graph for caphelper.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**
- namespace **QuantLib::InterestRateModelling::CalibrationHelpers**

12.50.1 Detailed Description

Full path:

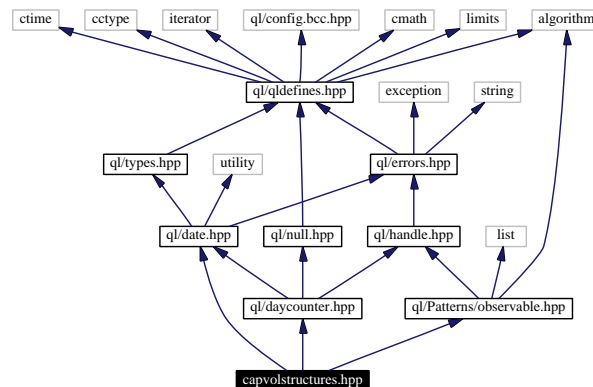
`ql/InterestRateModelling/CalibrationHelpers/caphelper.hpp`

12.51 capvolstructures.hpp File Reference

Cap/Floor volatility structures.

```
#include <ql/date.hpp>
#include <ql/daycounter.hpp>
#include <ql/Patterns/observable.hpp>
```

Include dependency graph for capvolstructures.hpp:



Namespaces

- namespace [QuantLib](#)

12.51.1 Detailed Description

Full path:

ql/capvolstructures.hpp

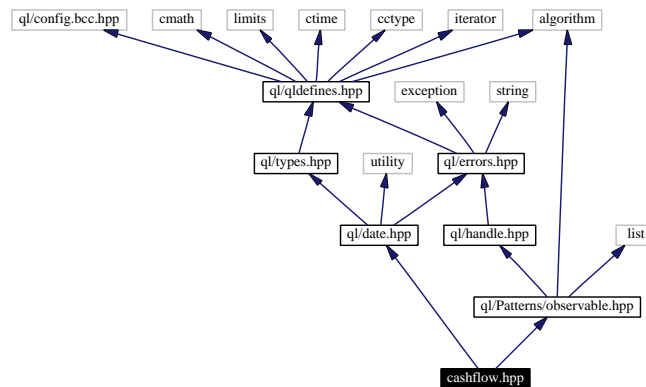
12.52 cashflow.hpp File Reference

Base class for cash flows.

```
#include <ql/date.hpp>
```

```
#include <ql/Patterns/observable.hpp>
```

Include dependency graph for cashflow.hpp:



Namespaces

- namespace [QuantLib::CashFlows](#)
- namespace [QuantLib](#)

12.52.1 Detailed Description

Full path:

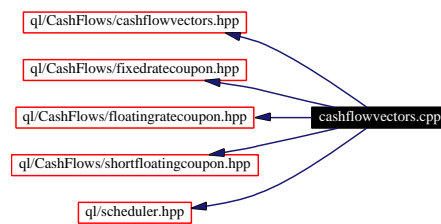
`ql/cashflow.hpp`

12.53 cashflowvectors.cpp File Reference

Cash flow vector builders.

```
#include <ql/CashFlows/cashflowvectors.hpp>
#include <ql/CashFlows/fixedratecoupon.hpp>
#include <ql/CashFlows/floatingratecoupon.hpp>
#include <ql/CashFlows/shortfloatingcoupon.hpp>
#include <ql/scheduler.hpp>
```

Include dependency graph for cashflowvectors.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.53.1 Detailed Description

Full path:

`ql/CashFlows/cashflowvectors.cpp`

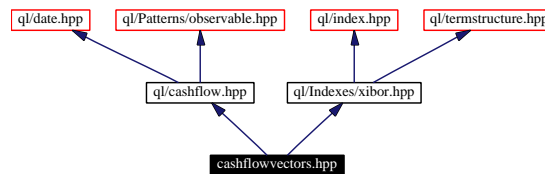
12.54 cashflowvectors.hpp File Reference

Cash flow vector builders.

```
#include <ql/cashflow.hpp>
```

```
#include <ql/Indexes/xibor.hpp>
```

Include dependency graph for cashflowvectors.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.54.1 Detailed Description

Full path:

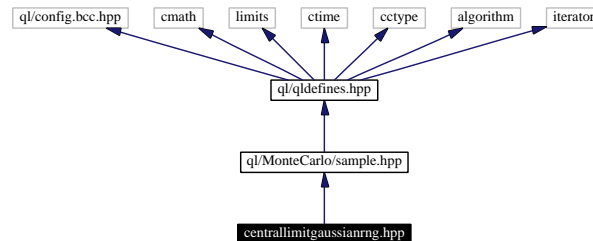
ql/CashFlows/cashflowvectors.hpp

12.55 centrallimitgaussianrng.hpp File Reference

Central limit Gaussian random-number generator.

```
#include <ql/MonteCarlo/sample.hpp>
```

Include dependency graph for centrallimitgaussianrng.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.55.1 Detailed Description

Full path:

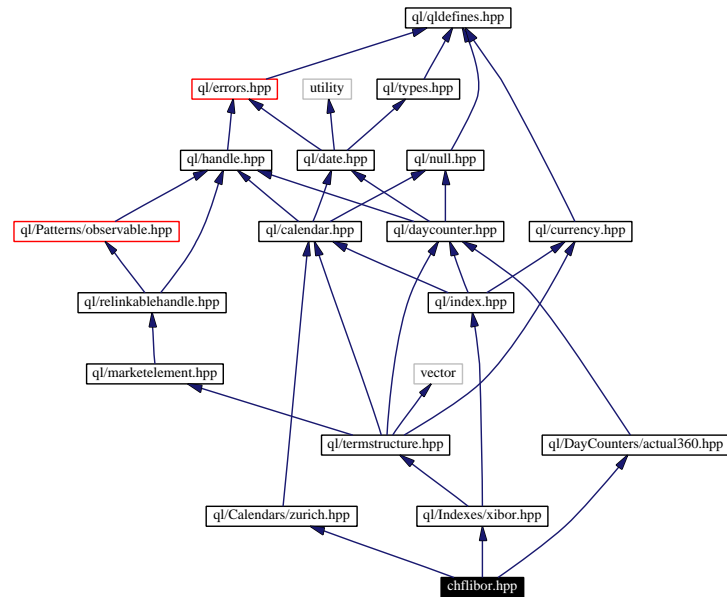
`ql/RandomNumbers/centrallimitgaussianrng.hpp`

12.56 chflibor.hpp File Reference

CHF Libor index (Also known as ZIBOR, check settlement days and day-count).

```
#include <ql/Indexes/xibor.hpp>
#include <ql/Calendars/zurich.hpp>
#include <ql/DayCounters/actual360.hpp>
```

Include dependency graph for chflibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.56.1 Detailed Description

Full path:

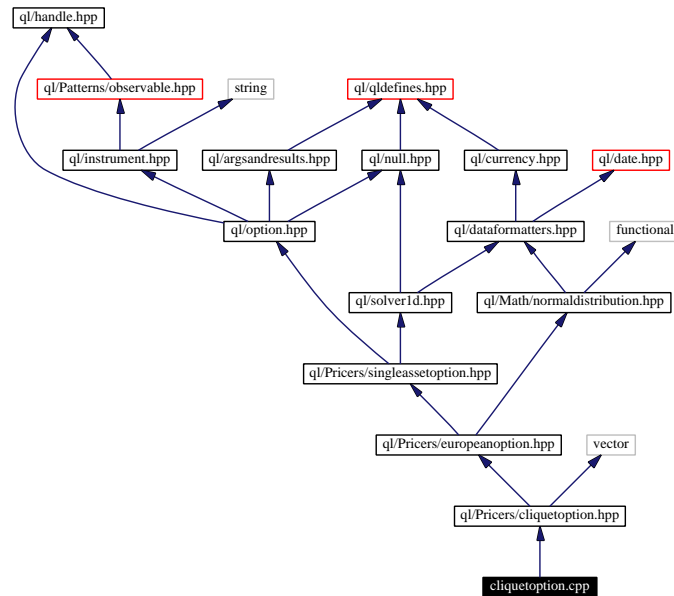
ql/Indexes/chflibor.hpp

12.57 cliquetoption.cpp File Reference

Textbook example of european-style multi-period option.

```
#include <ql/Pricers/cliquetoption.hpp>
```

Include dependency graph for cliquetoption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.57.1 Detailed Description

Full path:

ql/Pricers/cliquetoption.cpp

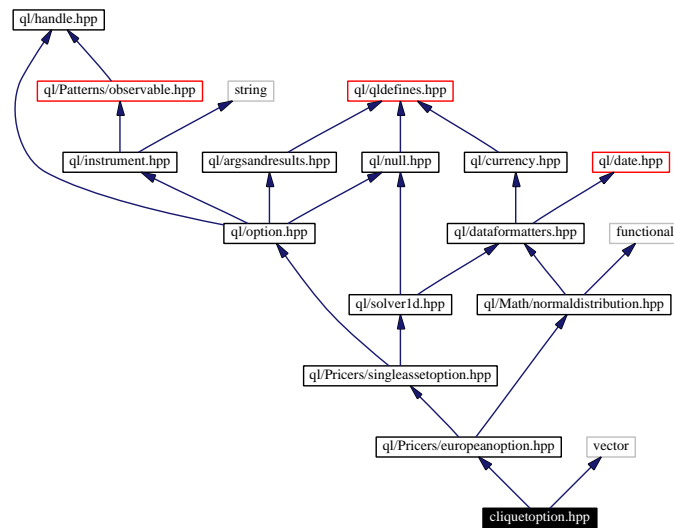
12.58 cliquetooption.hpp File Reference

Textbook example of european-style multi-period option.

```
#include <ql/Pricers/europeanoption.hpp>
```

```
#include <vector>
```

Include dependency graph for cliquetooption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.58.1 Detailed Description

Full path:

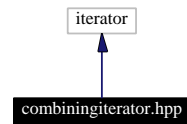
ql/Pricers/cliquetooption.hpp

12.59 combiningiterator.hpp File Reference

Iterator mapping a function to a set of underlying sequences.

```
#include <iterator>
```

Include dependency graph for combiningiterator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Utilities](#)

12.59.1 Detailed Description

Full path:

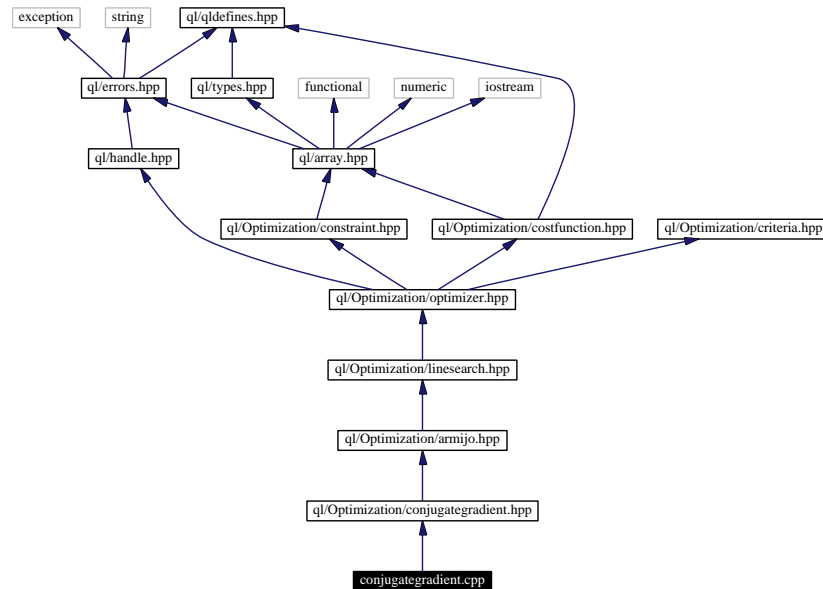
ql/Utilities/combiningiterator.hpp

12.60 conjugategradient.cpp File Reference

Conjugate gradient optimization method.

```
#include "ql/Optimization/conjugategradient.hpp"
```

Include dependency graph for conjugategradient.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Optimization**

12.60.1 Detailed Description

Full path:

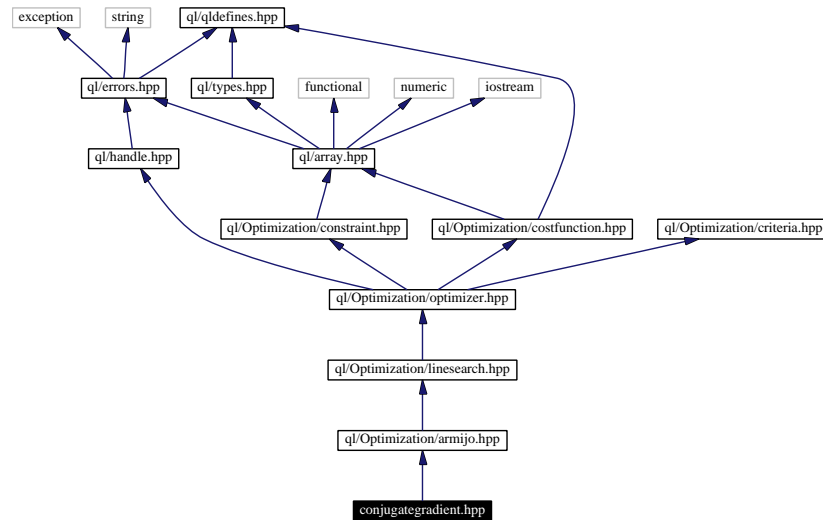
`ql/Optimization/conjugategradient.cpp`

12.61 conjugategradient.hpp File Reference

Conjugate gradient optimization method.

```
#include "ql/Optimization/armijo.hpp"
```

Include dependency graph for conjugategradient.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.61.1 Detailed Description

Full path:

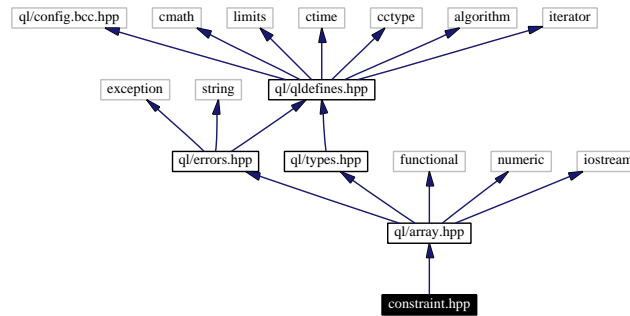
`ql/Optimization/conjugategradient.hpp`

12.62 constraint.hpp File Reference

Abstract constraint class.

```
#include "ql/array.hpp"
```

Include dependency graph for constraint.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.62.1 Detailed Description

Full path:

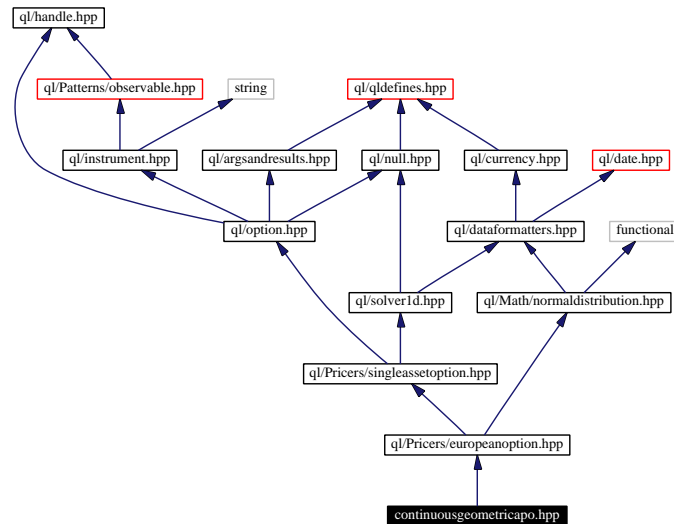
`ql/Optimization/constraint.hpp`

12.63 continuousgeometricapo.hpp File Reference

Continuous Geometric Average Price Option (European exercise).

```
#include <ql/Pricers/europeanoption.hpp>
```

Include dependency graph for continuousgeometricapo.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.63.1 Detailed Description

Full path:

ql/Pricers/continuousgeometricapo.hpp

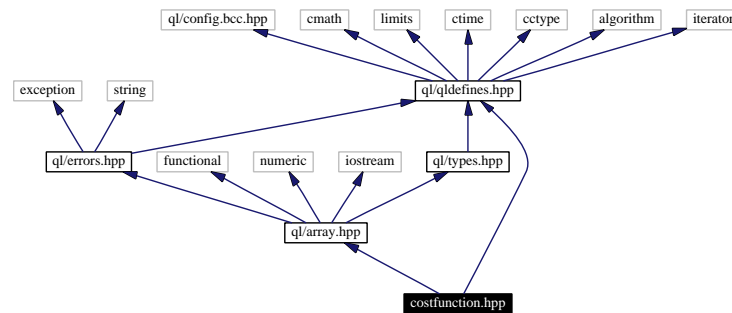
12.64 costfunction.hpp File Reference

Optimization cost function class.

```
#include <ql/qldefines.hpp>
```

```
#include <ql/array.hpp>
```

Include dependency graph for costfunction.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.64.1 Detailed Description

Full path:

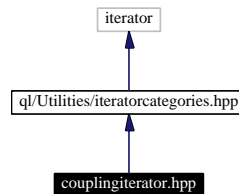
`ql/Optimization/costfunction.hpp`

12.65 couplingiterator.hpp File Reference

Iterator mapping a function to a pair of underlying sequences.

```
#include <ql/Utilities/iteratorcategories.hpp>
```

Include dependency graph for couplingiterator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Utilities](#)

12.65.1 Detailed Description

Full path:

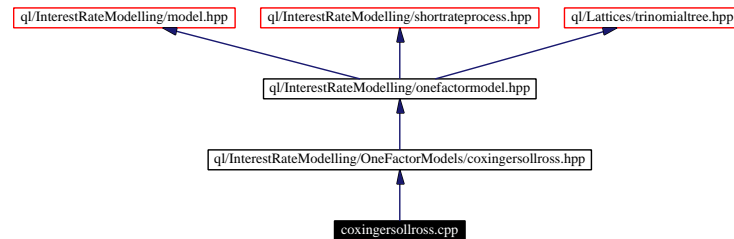
ql/Utilities/couplingiterator.hpp

12.67 coxingersollross.cpp File Reference

Cox-Ingersoll-Ross model.

```
#include "ql/InterestRateModelling/OneFactorModels/coxingersollross.hpp"
```

Include dependency graph for coxingersollross.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::InterestRateModelling`

12.67.1 Detailed Description

Full path:

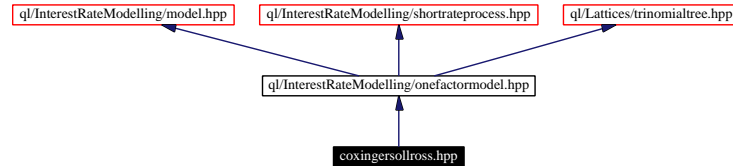
`ql/InterestRateModelling/OneFactorModels/coxingersollross.cpp`

12.68 coxingersollross.hpp File Reference

Cox-Ingersoll-Ross model.

```
#include "ql/InterestRateModelling/onefactormodel.hpp"
```

Include dependency graph for coxingersollross.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::InterestRateModelling](#)

12.68.1 Detailed Description

Full path:

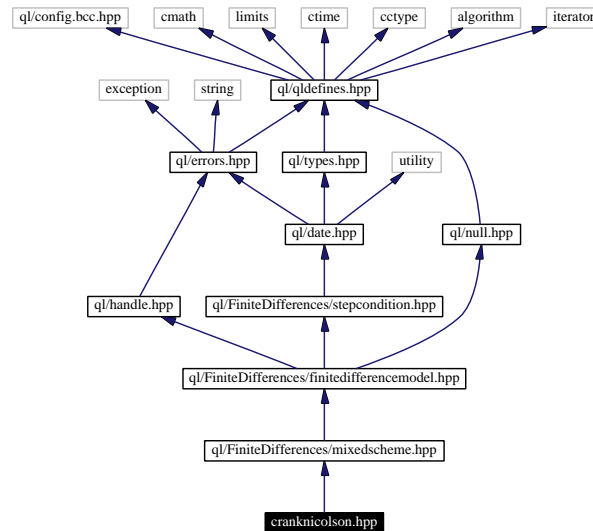
ql/InterestRateModelling/OneFactorModels/coxingersollross.hpp

12.69 cranknicolson.hpp File Reference

Crank-Nicolson scheme for finite difference methods.

```
#include <ql/FiniteDifferences/mixedscheme.hpp>
```

Include dependency graph for cranknicolson.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.69.1 Detailed Description

Full path:

`ql/FiniteDifferences/cranknicolson.hpp`

12.70 criteria.hpp File Reference

Optimization criteria class.

Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Optimization**

12.70.1 Detailed Description

Full path:

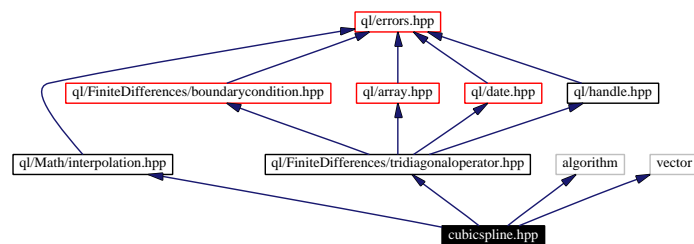
ql/Optimization/criteria.hpp

12.71 cubicspline.hpp File Reference

cubic spline interpolation between discrete points.

```
#include <ql/Math/interpolation.hpp>
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
#include <algorithm>
#include <vector>
```

Include dependency graph for cubicspline.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.71.1 Detailed Description

Full path:

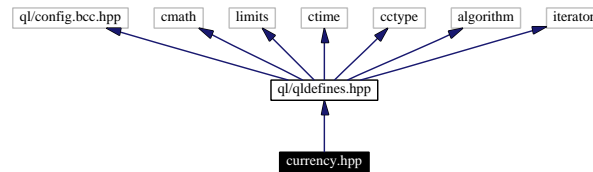
`ql/Math/cubicspline.hpp`

12.72 currency.hpp File Reference

Known currencies.

```
#include <ql/qldefines.hpp>
```

Include dependency graph for currency.hpp:



Namespaces

- namespace [QuantLib](#)

12.72.1 Detailed Description

Full path:

`ql/currency.hpp`

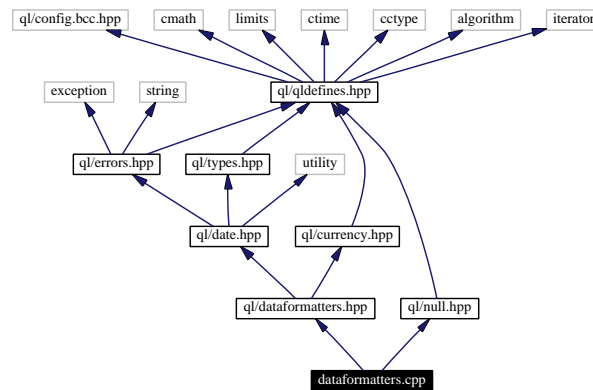
12.73 dataformatters.cpp File Reference

classes used to format data for output.

```
#include <ql/dataformatters.hpp>
```

```
#include <ql/null.hpp>
```

Include dependency graph for dataformatters.cpp:



Namespaces

- namespace [QuantLib](#)

12.73.1 Detailed Description

Full path:

ql/dataformatters.cpp

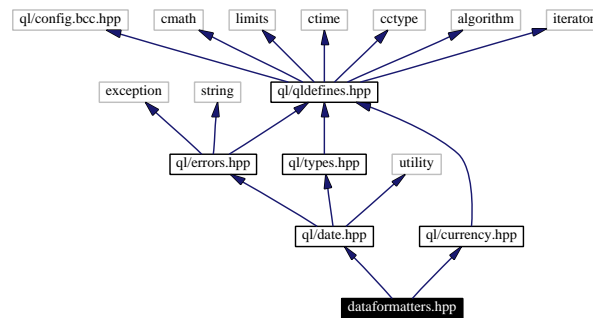
12.74 dataformatters.hpp File Reference

Classes used to format data for output.

```
#include <ql/date.hpp>
```

```
#include <ql/currency.hpp>
```

Include dependency graph for dataformatters.hpp:



Namespaces

- namespace [QuantLib](#)

12.74.1 Detailed Description

Full path:

`ql/dataformatters.hpp`

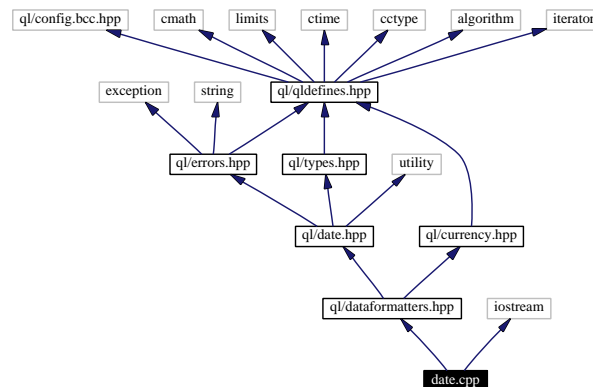
12.75 date.cpp File Reference

date- and time-related classes, typedefs and enumerations.

```
#include <ql/dataformatters.hpp>
```

```
#include <iostream>
```

Include dependency graph for date.cpp:



Namespaces

- namespace [QuantLib](#)

12.75.1 Detailed Description

Full path:

`ql/date.cpp`

12.76 date.hpp File Reference

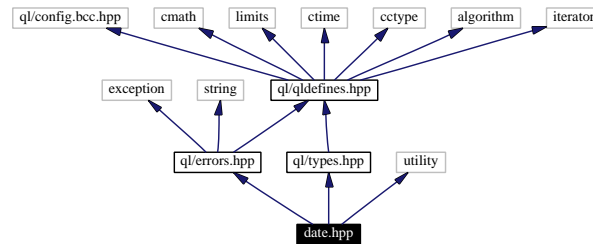
date- and time-related classes, typedefs and enumerations.

```
#include <ql/errors.hpp>
```

```
#include <ql/types.hpp>
```

```
#include <utility>
```

Include dependency graph for date.hpp:



Namespaces

- namespace [QuantLib](#)

12.76.1 Detailed Description

Full path:

`ql/date.hpp`

12.77 daycounter.hpp File Reference

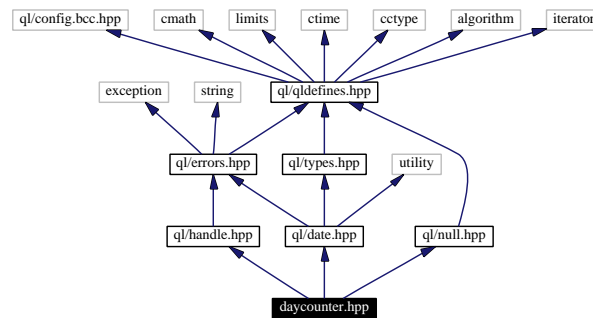
day counter class.

```
#include <ql/date.hpp>
```

```
#include <ql/handle.hpp>
```

```
#include <ql/null.hpp>
```

Include dependency graph for daycounter.hpp:



Namespaces

- namespace [QuantLib::DayCounters](#)
- namespace [QuantLib](#)

12.77.1 Detailed Description

Full path:

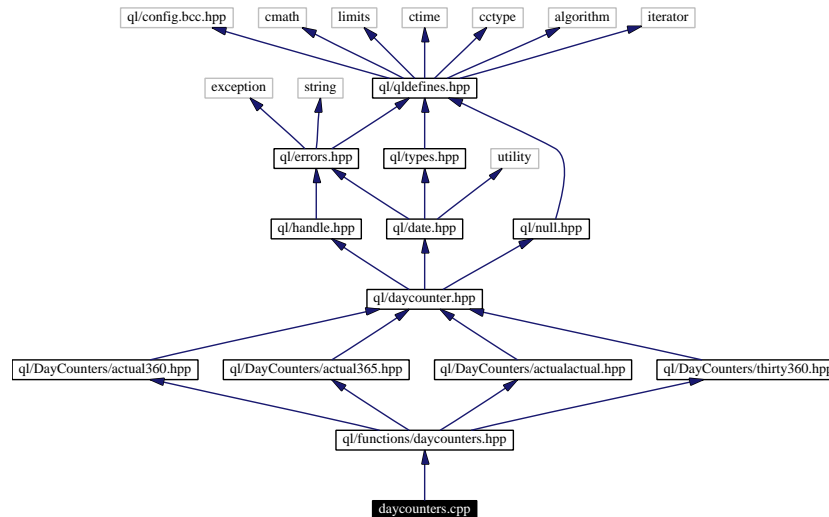
`ql/daycounter.hpp`

12.78 daycounters.cpp File Reference

day counters functions.

```
#include <ql/functions/daycounters.hpp>
```

Include dependency graph for daycounters.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Functions](#)

12.78.1 Detailed Description

Full path:

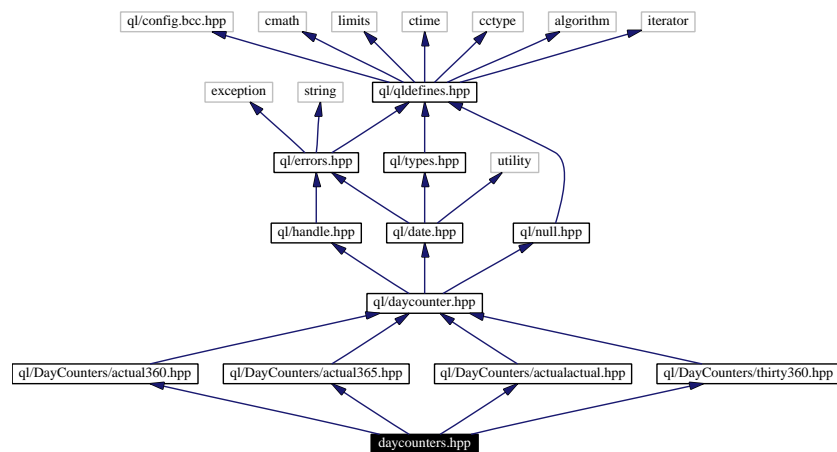
`ql/functions/daycounters.cpp`

12.79 daycounters.hpp File Reference

day counters functions.

```
#include <ql/DayCounters/actual360.hpp>
#include <ql/DayCounters/actual365.hpp>
#include <ql/DayCounters/actualactual.hpp>
#include <ql/DayCounters/thirty360.hpp>
```

Include dependency graph for daycounters.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Functions**

12.79.1 Detailed Description

Full path:

ql/functions/daycounters.hpp

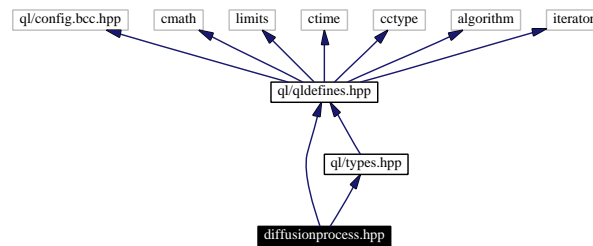
12.80 diffusionprocess.hpp File Reference

Diffusion process.

```
#include <ql/qldefines.hpp>
```

```
#include <ql/types.hpp>
```

Include dependency graph for diffusionprocess.hpp:



Namespaces

- namespace [QuantLib](#)

12.80.1 Detailed Description

Full path:

`ql/diffusionprocess.hpp`

12.81 discretegeometricapo.cpp File Reference

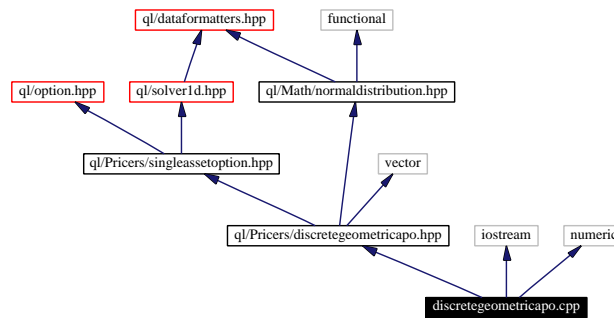
Discrete Geometric Average Price Option.

```
#include <ql/Pricers/discretegeometricapo.hpp>
```

```
#include <iostream>
```

```
#include <numeric>
```

Include dependency graph for discretegeometricapo.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.81.1 Detailed Description

Full path:

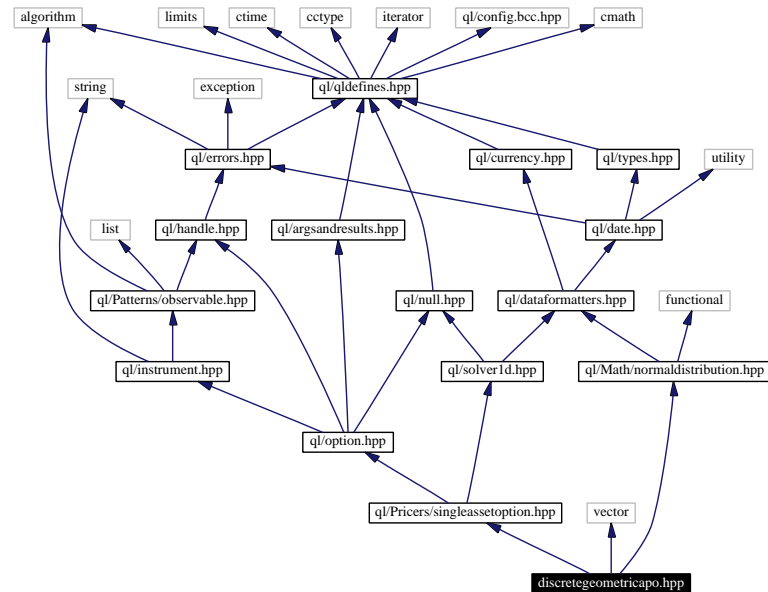
`ql/Pricers/discretegeometricapo.cpp`

12.82 discretegeometricapo.hpp File Reference

Discrete Geometric Average Price Option.

```
#include <ql/Pricers/singleassetoption.hpp>
#include <ql/Math/normaldistribution.hpp>
#include <vector>
```

Include dependency graph for discretegeometricapo.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.82.1 Detailed Description

Full path:

ql/Pricers/discretegeometricapo.hpp

12.83 discretegeometricaso.cpp File Reference

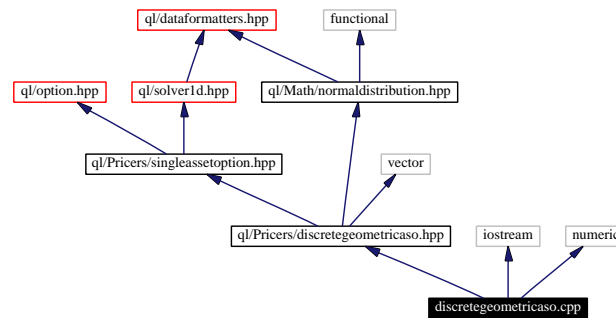
Discrete Geometric Average Strike Option.

```
#include <ql/Pricers/discretegeometricaso.hpp>
```

```
#include <iostream>
```

```
#include <numeric>
```

Include dependency graph for discretegeometricaso.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.83.1 Detailed Description

Full path:

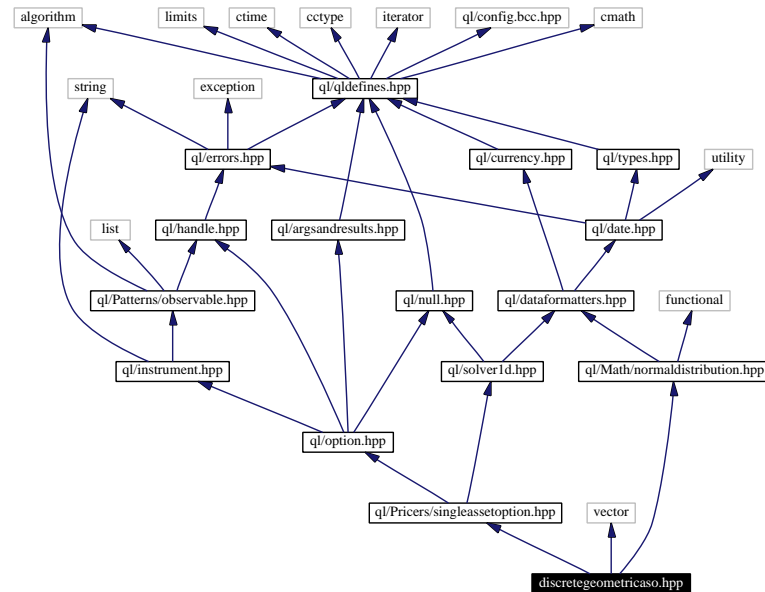
`ql/Pricers/discretegeometricaso.cpp`

12.84 discretegeometricaso.hpp File Reference

Discrete Geometric Average Strike Option.

```
#include <ql/Pricers/singleassetoption.hpp>
#include <ql/Math/normaldistribution.hpp>
#include <vector>
```

Include dependency graph for discretegeometricaso.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.84.1 Detailed Description

Full path:

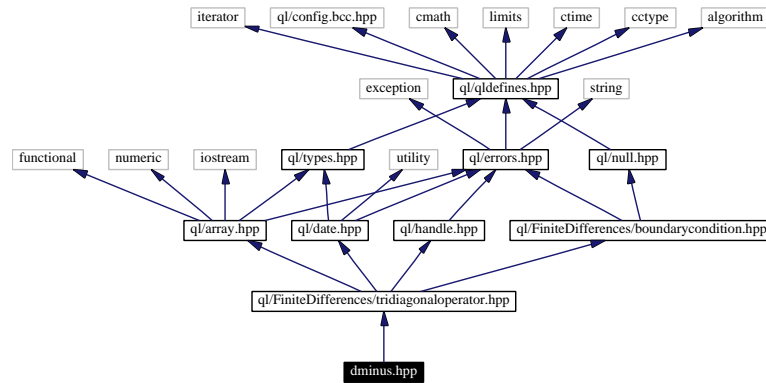
ql/Pricers/discretegeometricaso.hpp

12.85 dminus.hpp File Reference

D_- matricial representation.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

Include dependency graph for dminus.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.85.1 Detailed Description

Full path:

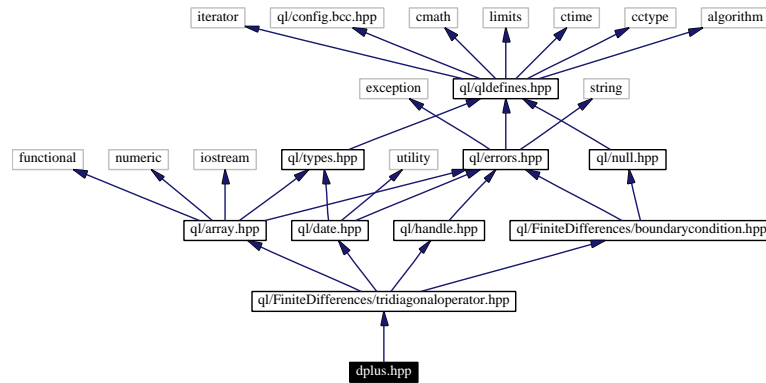
`ql/FiniteDifferences/dminus.hpp`

12.86 dplus.hpp File Reference

D_+ matricial representation.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

Include dependency graph for dplus.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.86.1 Detailed Description

Full path:

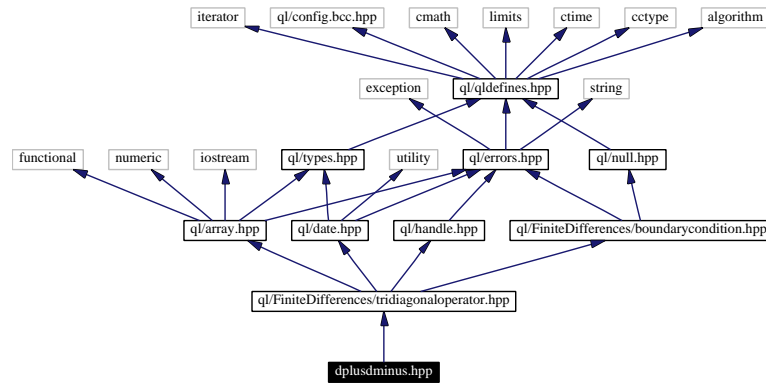
ql/FiniteDifferences/dplus.hpp

12.87 dplusdminus.hpp File Reference

D_+D_- matricial representation.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

Include dependency graph for dplusdminus.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.87.1 Detailed Description

Full path:

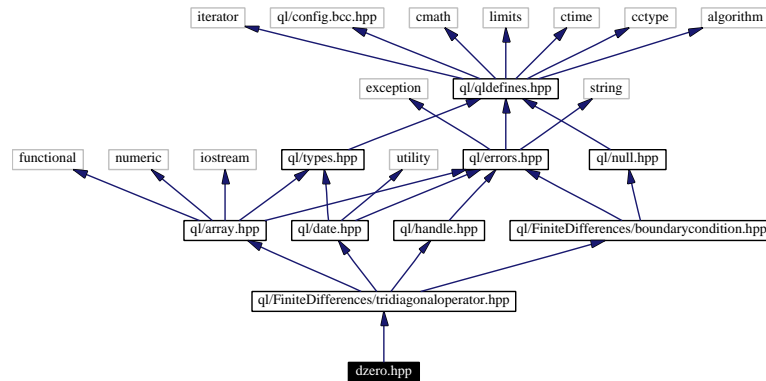
ql/FiniteDifferences/dplusdminus.hpp

12.88 dzero.hpp File Reference

D_0 matricial representation.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

Include dependency graph for dzero.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.88.1 Detailed Description

Full path:

ql/FiniteDifferences/dzero.hpp

12.89 errors.hpp File Reference

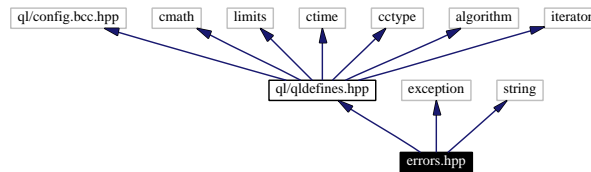
Classes and functions for error handling.

```
#include <ql/qldefines.hpp>
```

```
#include <exception>
```

```
#include <string>
```

Include dependency graph for errors.hpp:



Namespaces

- namespace [QuantLib](#)

Defines

- #define [QL_ASSERT](#)(condition, description)
throw an error if the given condition is not verified \relates Error.
- #define [QL_REQUIRE](#)(condition, description)
throw an error if the given pre-condition is not verified \relates Error.
- #define [QL_ENSURE](#)(condition, description)
throw an error if the given post-condition is not verified \relates Error.

12.89.1 Detailed Description

Full path:

ql/errors.hpp

12.89.2 Define Documentation

12.89.2.1 #define QL_ASSERT(condition, description)

Value:

```
do { \
    if (!(condition)) \
        throw QuantLib::AssertionFailedError(description); \
} while (false)
```

12.89.2.2 #define QL_REQUIRE(condition, description)**Value:**

```
do { \
    if (!(condition)) \
        throw QuantLib::PreconditionNotSatisfiedError(description); \
} while (false)
```

Examples:

[DiscreteHedging.cpp](#), and [swapvaluation.cpp](#).

12.89.2.3 #define QL_ENSURE(condition, description)**Value:**

```
do { \
    if (!(condition)) \
        throw QuantLib::PostconditionNotSatisfiedError(description); \
} while (false)
```

12.90 euribor.hpp File Reference

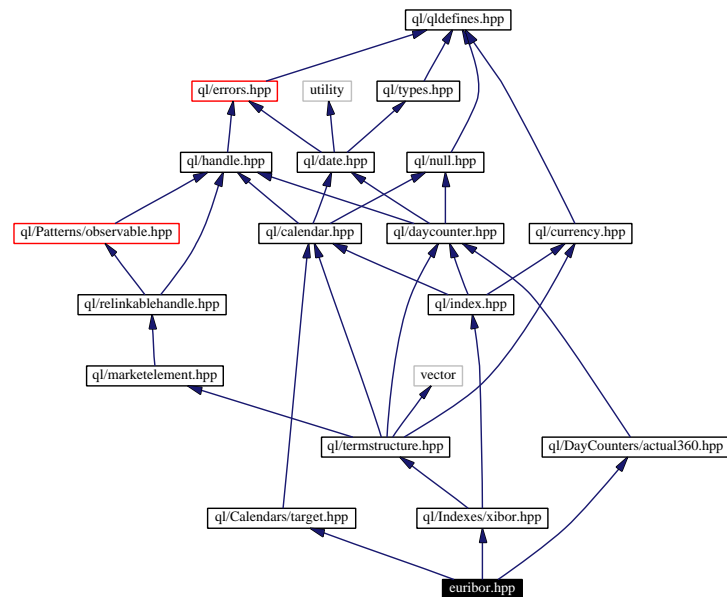
Euribor index.

```
#include <ql/Indexes/xibor.hpp>
```

```
#include <ql/Calendars/target.hpp>
```

```
#include <ql/DayCounters/actual360.hpp>
```

Include dependency graph for euribor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.90.1 Detailed Description

Full path:

ql/Indexes/euribor.hpp

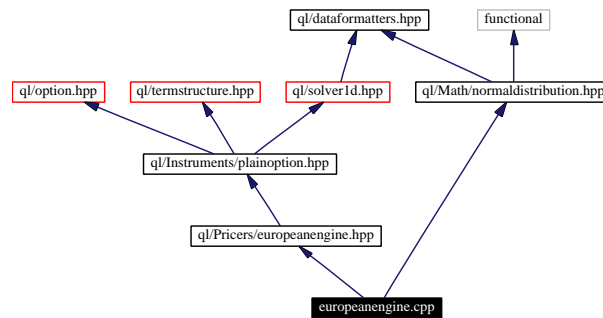
12.91 europeanengine.cpp File Reference

analytic pricing engine for European options.

```
#include <ql/Pricers/europeanengine.hpp>
```

```
#include <ql/Math/normaldistribution.hpp>
```

Include dependency graph for europeanengine.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.91.1 Detailed Description

Full path:

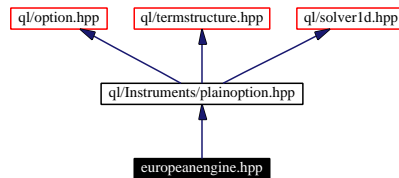
`ql/Pricers/europeanengine.cpp`

12.92 europeanengine.hpp File Reference

analytic pricing engine for European options.

```
#include <ql/Instruments/plainoption.hpp>
```

Include dependency graph for europeanengine.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.92.1 Detailed Description

Full path:

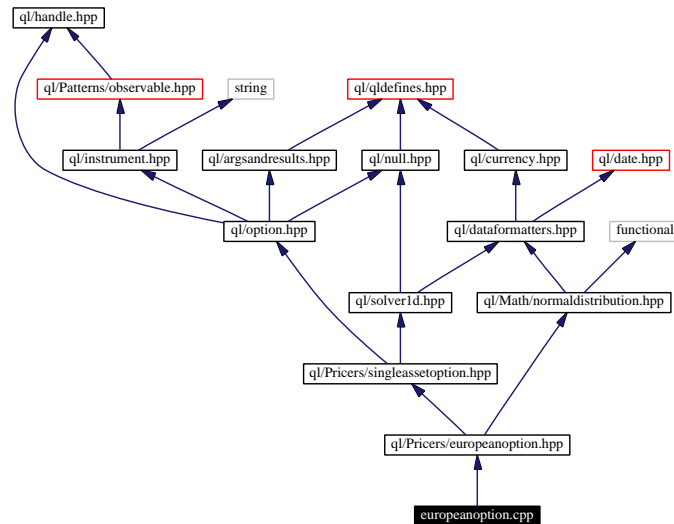
`ql/Pricers/europeanengine.hpp`

12.93 europeanoption.cpp File Reference

european option.

```
#include <ql/Pricers/europeanoption.hpp>
```

Include dependency graph for europeanoption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.93.1 Detailed Description

Full path:

`ql/Pricers/europeanoption.cpp`

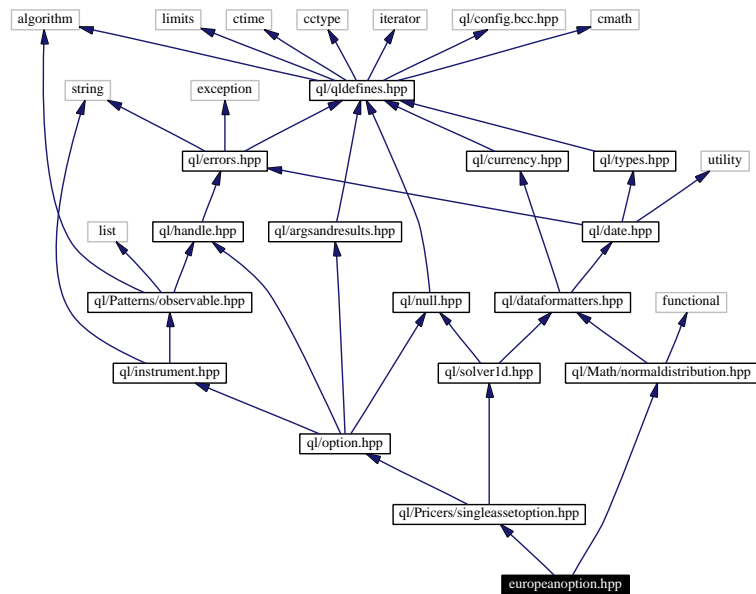
12.94 europeanoption.hpp File Reference

european option.

```
#include <ql/Pricers/singleassetoption.hpp>
```

```
#include <ql/Math/normaldistribution.hpp>
```

Include dependency graph for europeanoption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.94.1 Detailed Description

Full path:

ql/Pricers/europeanoption.hpp

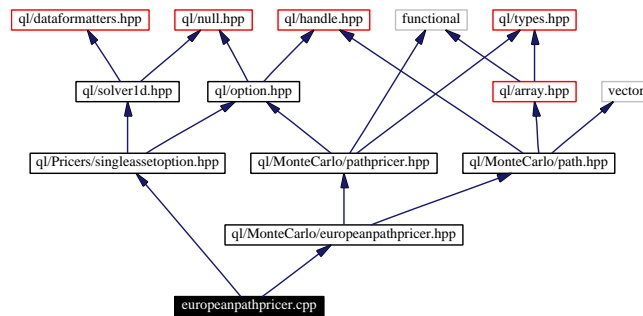
12.95 europeanpathpricer.cpp File Reference

path pricer for European options.

```
#include <ql/MonteCarlo/europeanpathpricer.hpp>
```

```
#include <ql/Pricers/singleassetoption.hpp>
```

Include dependency graph for europeanpathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.95.1 Detailed Description

Full path:

`ql/MonteCarlo/europeanpathpricer.cpp`

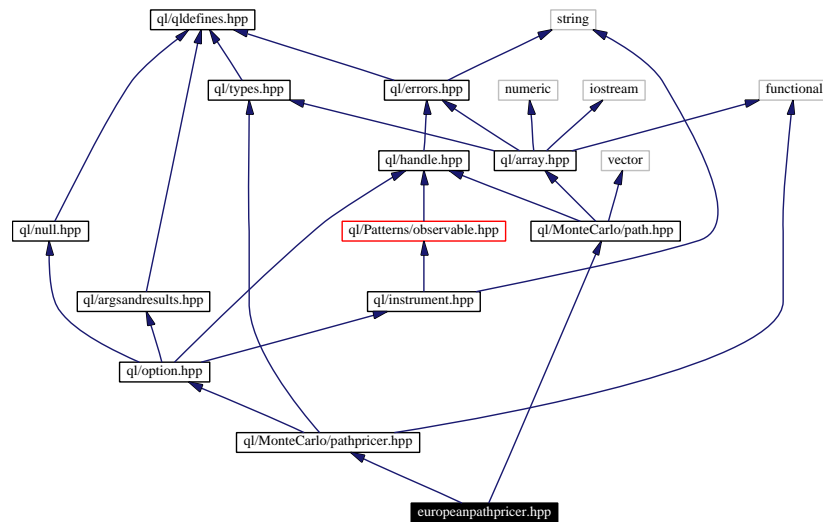
12.96 europeanpathpricer.hpp File Reference

path pricer for European options.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/path.hpp>
```

Include dependency graph for europeanpathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.96.1 Detailed Description

Full path:

ql/MonteCarlo/europeanpathpricer.hpp

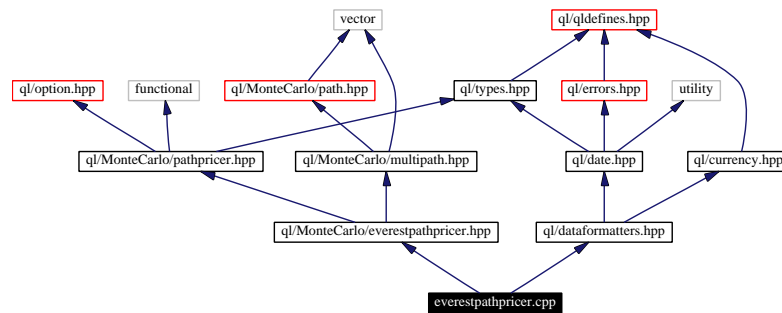
12.97 everestpathpricer.cpp File Reference

path pricer for European-type Everest option.

```
#include <ql/MonteCarlo/everestpathpricer.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for everestpathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.97.1 Detailed Description

Full path:

MonteCarlo/everestpathpricer.cpp

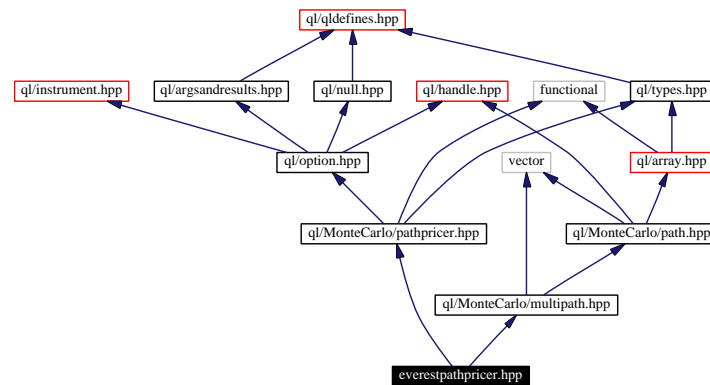
12.98 everestpathpricer.hpp File Reference

path pricer for European-type Everest option.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/multipath.hpp>
```

Include dependency graph for everestpathpricer.hpp:



Namespaces

- namespace `QuantLib`
- namespace `QuantLib::MonteCarlo`

12.98.1 Detailed Description

Full path:

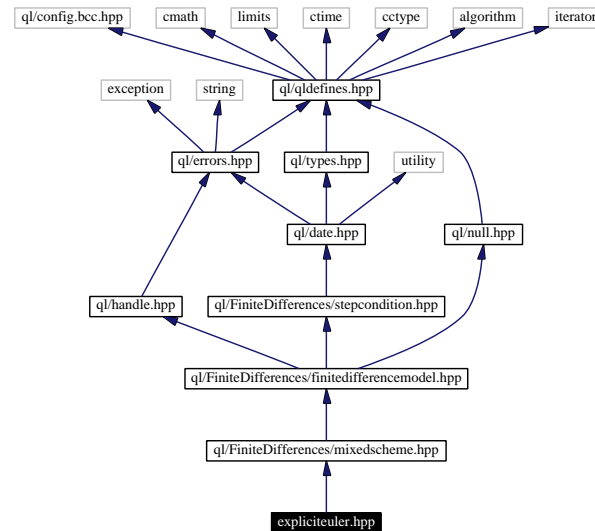
ql/MonteCarlo/everestpathpricer.hpp

12.99 expliciteuler.hpp File Reference

explicit Euler scheme for finite difference methods.

```
#include <ql/FiniteDifferences/mixedscheme.hpp>
```

Include dependency graph for expliciteuler.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.99.1 Detailed Description

Full path:

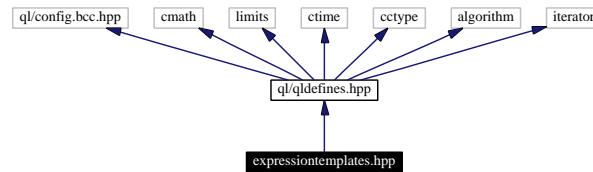
`ql/FiniteDifferences/expliciteuler.hpp`

12.100 expressiontemplates.hpp File Reference

expression template implementation.

```
#include <ql/qldefines.hpp>
```

Include dependency graph for expressiontemplates.hpp:



Namespaces

- namespace [QuantLib](#)

12.100.1 Detailed Description

Full path:

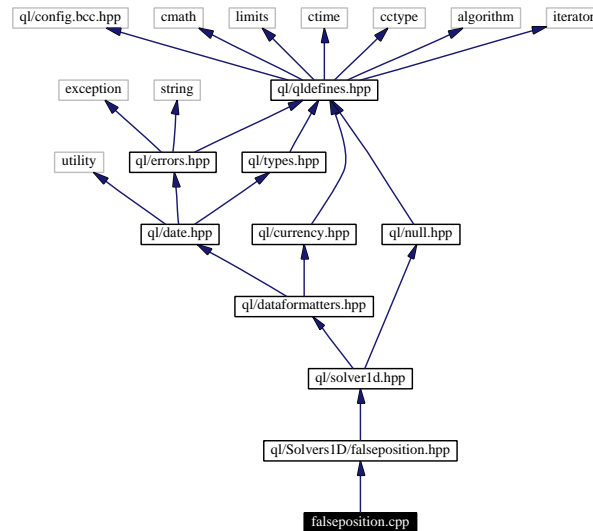
`ql/expressiontemplates.hpp`

12.101 falseposition.cpp File Reference

false-position 1-D solver.

```
#include <ql/Solvers1D/falseposition.hpp>
```

Include dependency graph for falseposition.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.101.1 Detailed Description

Full path:

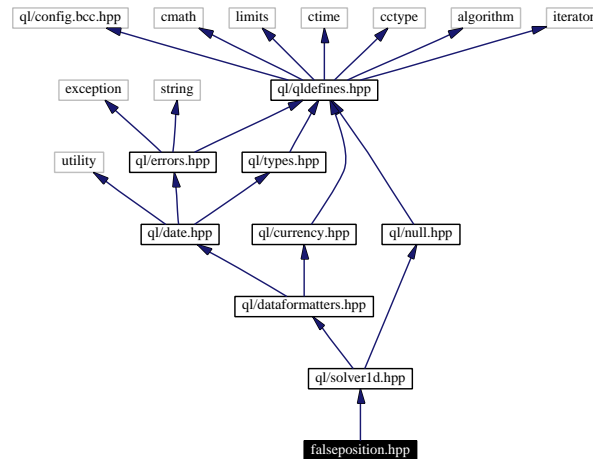
`ql/Solvers1D/falseposition.cpp`

12.102 falseposition.hpp File Reference

false-position 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for falseposition.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.102.1 Detailed Description

Full path:

`ql/Solvers1D/falseposition.hpp`

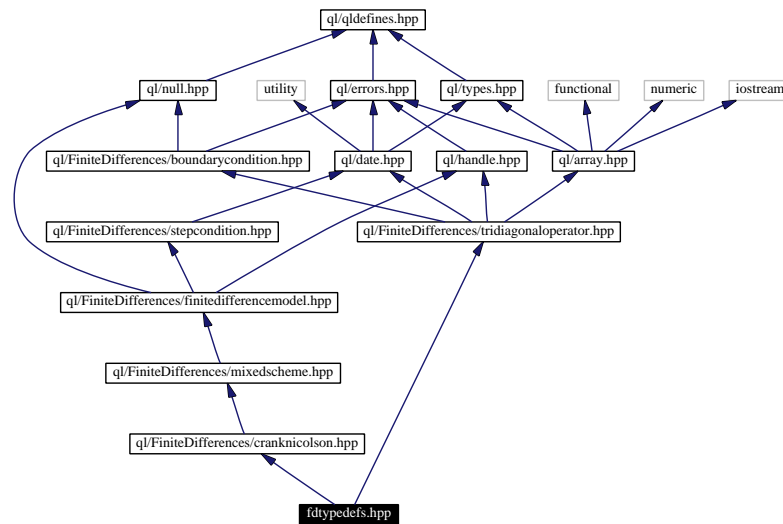
12.103 fdtypedefs.hpp File Reference

default choices for template instantiations.

```
#include <ql/FiniteDifferences/cranknicolson.hpp>
```

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

Include dependency graph for fdtypedefs.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.103.1 Detailed Description

Full path:

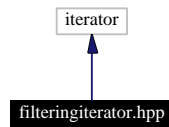
ql/FiniteDifferences/fdtypedefs.hpp

12.104 filteringiterator.hpp File Reference

Iterator filtering undesired data.

```
#include <iterator>
```

Include dependency graph for filteringiterator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Utilities](#)

12.104.1 Detailed Description

Full path:

ql/Utilities/filteringiterator.hpp

12.105 finitedifferencemodel.hpp File Reference

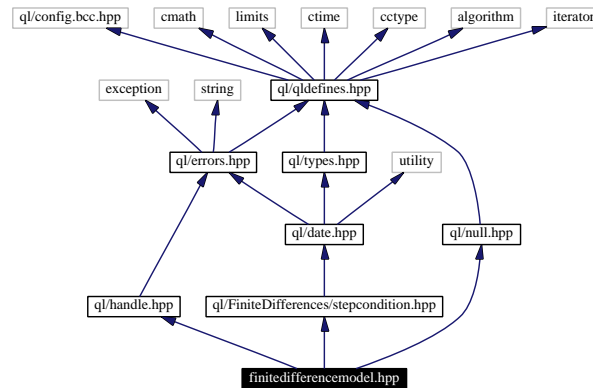
generic finite difference model.

```
#include <ql/FiniteDifferences/stepcondition.hpp>
```

```
#include <ql/handle.hpp>
```

```
#include <ql/null.hpp>
```

Include dependency graph for finitedifferencemodel.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.105.1 Detailed Description

Full path:

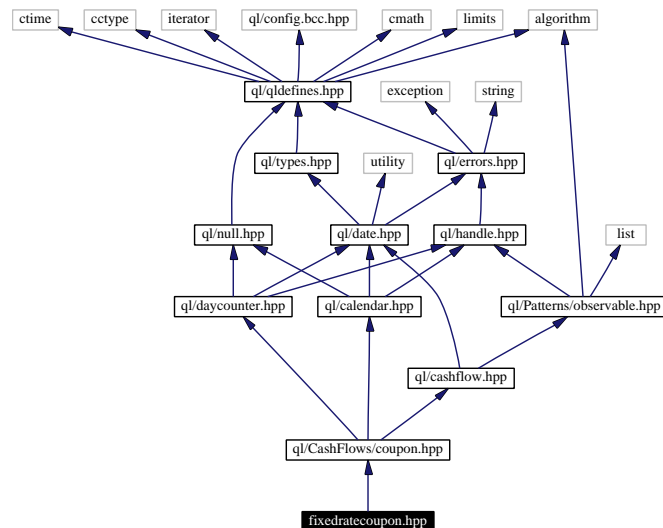
`ql/FiniteDifferences/finitedifferencemodel.hpp`

12.106 fixedratecoupon.hpp File Reference

Coupon paying a fixed annual rate.

```
#include <ql/CashFlows/coupon.hpp>
```

Include dependency graph for fixedratecoupon.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.106.1 Detailed Description

Full path:

`ql/CashFlows/fixedratecoupon.hpp`

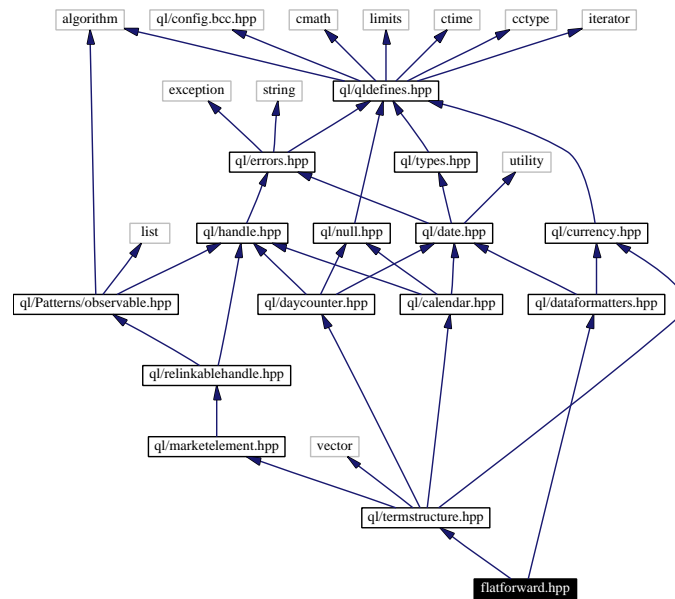
12.107 flatforward.hpp File Reference

flat forward rate term structure.

```
#include <ql/termstructure.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for flatforward.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::TermStructures](#)

12.107.1 Detailed Description

Full path:

ql/TermStructures/flatforward.hpp

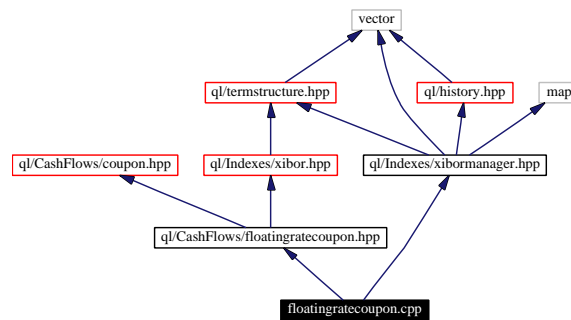
12.108 floatingratecoupon.cpp File Reference

Coupon at par on a term structure.

```
#include <ql/CashFlows/floatingratecoupon.hpp>
```

```
#include <ql/Indexes/xibormanager.hpp>
```

Include dependency graph for floatingratecoupon.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.108.1 Detailed Description

Full path:

`ql/CashFlows/floatingratecoupon.cpp`

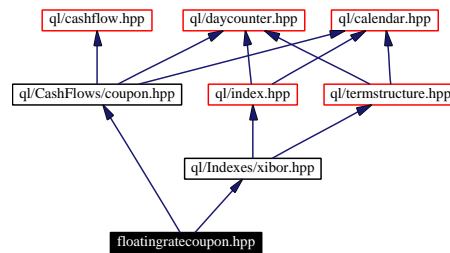
12.109 floatingratecoupon.hpp File Reference

Coupon at par on a term structure.

```
#include <ql/CashFlows/coupon.hpp>
```

```
#include <ql/Indexes/xibor.hpp>
```

Include dependency graph for floatingratecoupon.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.109.1 Detailed Description

Full path:

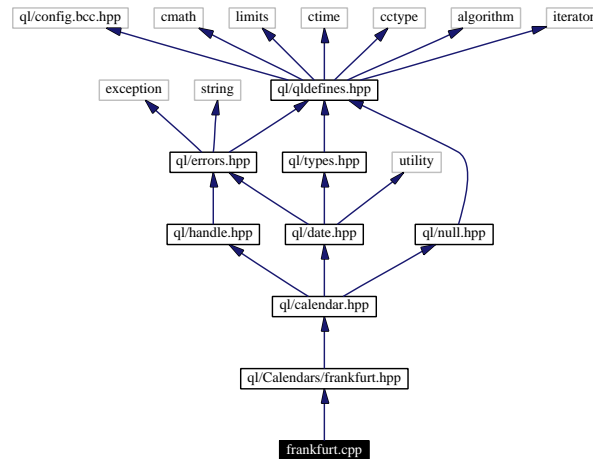
ql/CashFlows/floatingratecoupon.hpp

12.110 frankfurt.cpp File Reference

Frankfurt calendar.

```
#include <ql/Calendars/frankfurt.hpp>
```

Include dependency graph for frankfurt.cpp:



Namespaces

- namespace QuantLib
- namespace QuantLib::Calendars

12.110.1 Detailed Description

Full path:

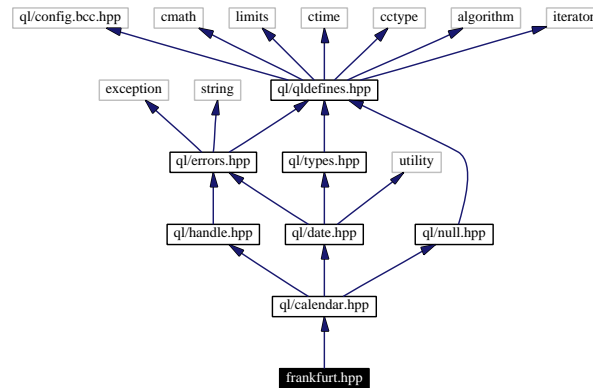
Calendars/frankfurt.cpp

12.111 frankfurt.hpp File Reference

Frankfurt calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for frankfurt.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.111.1 Detailed Description

Full path:

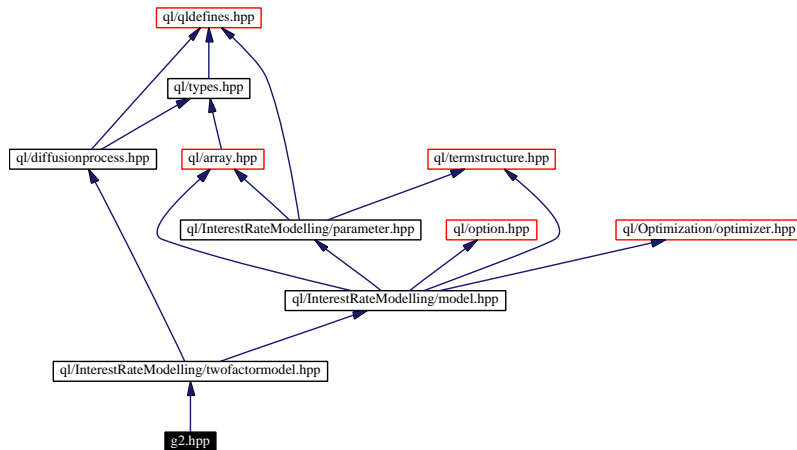
`ql/Calendars/frankfurt.hpp`

12.112 g2.hpp File Reference

Two-additive-factor Gaussian Model G2++.

```
#include "ql/InterestRateModelling/twofactormodel.hpp"
```

Include dependency graph for g2.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.112.1 Detailed Description

Full path:

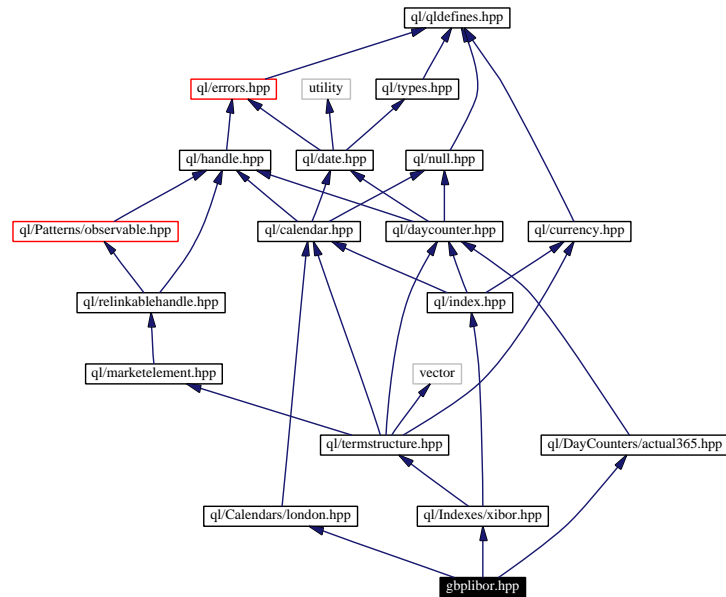
ql/InterestRateModelling/TwoFactorModels/g2.hpp

12.113 gbplibor.hpp File Reference

GBP Libor index.

```
#include <ql/Indexes/xibor.hpp>
#include <ql/Calendars/london.hpp>
#include <ql/DayCounters/actual365.hpp>
```

Include dependency graph for gbplibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.113.1 Detailed Description

Full path:

ql/Indexes/gbplibor.hpp

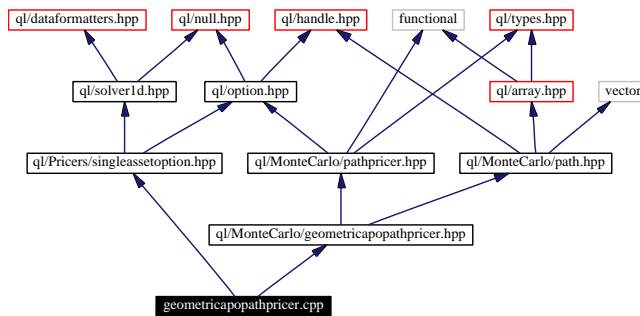
12.114 `geometricapopathpricer.cpp` File Reference

path pricer for geometric average price option.

```
#include <ql/MonteCarlo/geometricapopathpricer.hpp>
```

```
#include <ql/Pricers/singleassetoption.hpp>
```

Include dependency graph for `geometricapopathpricer.cpp`:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.114.1 Detailed Description

Full path:

`ql/MonteCarlo/geometricapopathpricer.cpp`

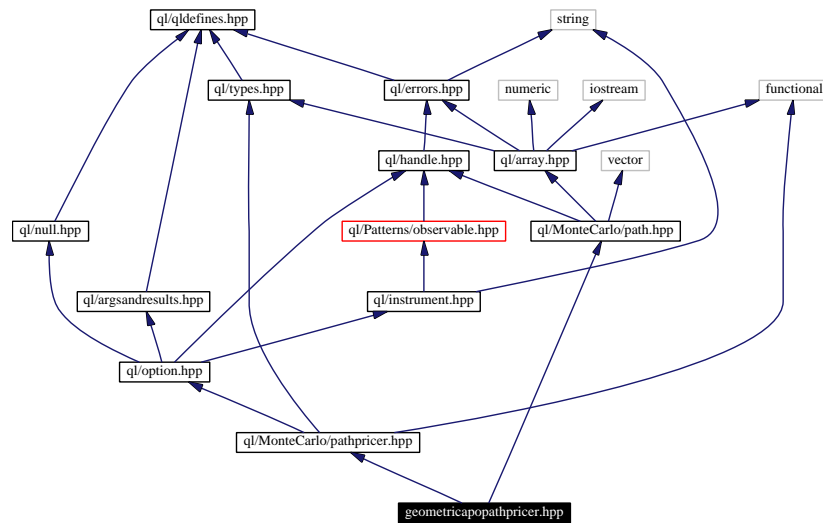
12.115 geometricapopathpricer.hpp File Reference

path pricer for geometric average price option.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/path.hpp>
```

Include dependency graph for geometricapopathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.115.1 Detailed Description

Full path:

ql/MonteCarlo/geometricapopathpricer.hpp

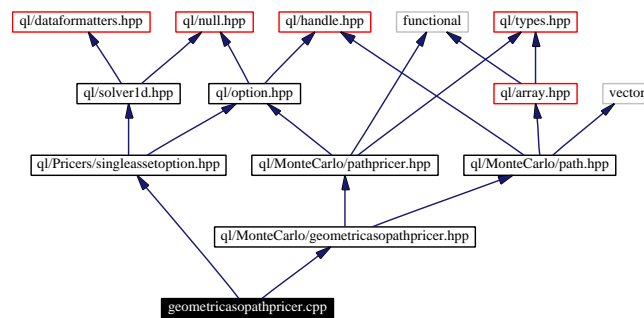
12.116 geometriasopathpricer.cpp File Reference

path pricer for geometric average strike option.

```
#include <ql/MonteCarlo/geometriasopathpricer.hpp>
```

```
#include <ql/Pricers/singleassetoption.hpp>
```

Include dependency graph for geometriasopathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.116.1 Detailed Description

Full path:

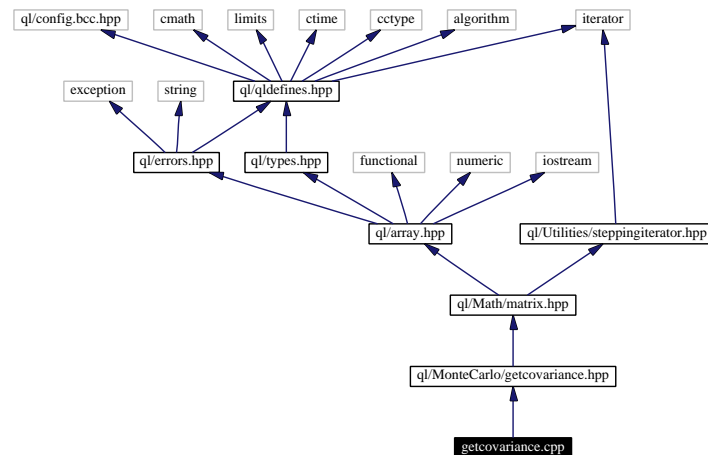
`ql/MonteCarlo/geometriasopathpricer.cpp`

12.118 getcovariance.cpp File Reference

Covariance matrix calculation.

```
#include <ql/MonteCarlo/getcovariance.hpp>
```

Include dependency graph for getcovariance.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.118.1 Detailed Description

Full path:

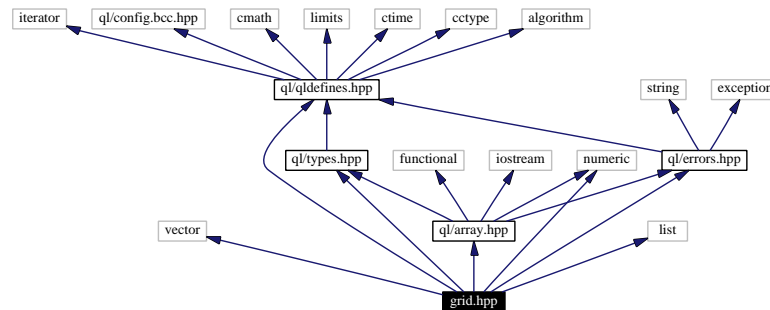
`ql/MonteCarlo/getcovariance.cpp`

12.119 grid.hpp File Reference

Grid classes with useful constructors for trees and finite diffs.

```
#include <ql/array.hpp>
#include <ql/errors.hpp>
#include <ql/qldefines.hpp>
#include <ql/types.hpp>
#include <list>
#include <numeric>
#include <vector>
```

Include dependency graph for grid.hpp:



Namespaces

- namespace [QuantLib](#)

12.119.1 Detailed Description

Full path:

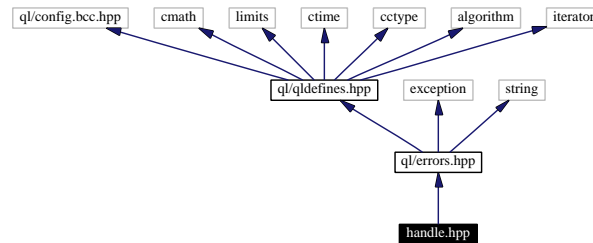
`ql/grid.hpp`

12.120 handle.hpp File Reference

Reference-counted pointer.

```
#include <ql/errors.hpp>
```

Include dependency graph for handle.hpp:



Namespaces

- namespace [QuantLib](#)

12.120.1 Detailed Description

Full path:

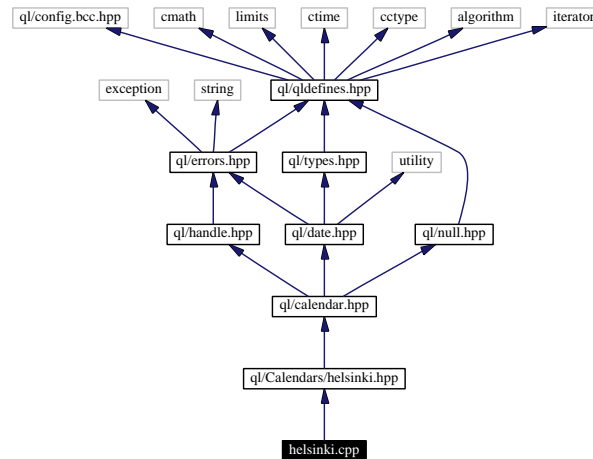
`ql/handle.hpp`

12.121 helsinki.cpp File Reference

Helsinki calendar.

```
#include <ql/Calendars/helsinki.hpp>
```

Include dependency graph for helsinki.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.121.1 Detailed Description

Full path:

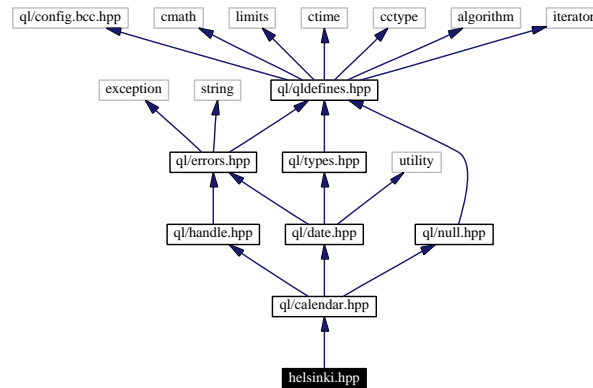
Calendars/helsinki.cpp

12.122 helsinki.hpp File Reference

Helsinki calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for helsinki.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.122.1 Detailed Description

Full path:

`ql/Calendars/helsinki.hpp`

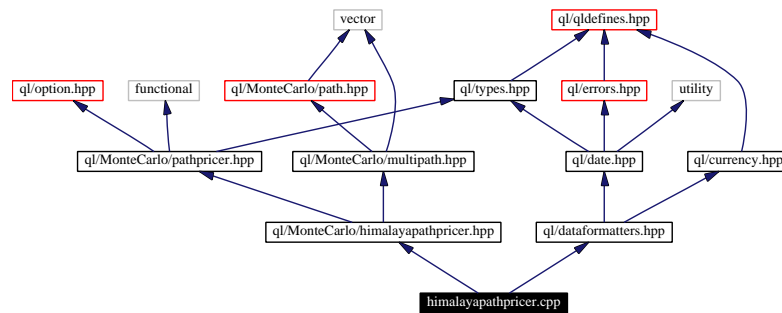
12.123 himalayapathpricer.cpp File Reference

multipath pricer for European-type Himalaya option.

```
#include <ql/MonteCarlo/himalayapathpricer.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for himalayapathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.123.1 Detailed Description

Full path:

ql/MonteCarlo/himalayapathpricer.cpp

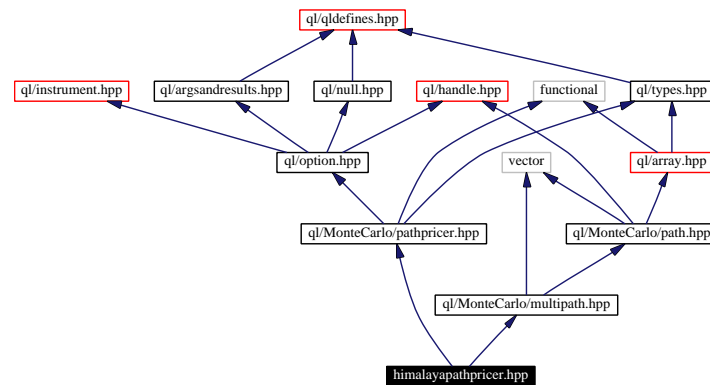
12.124 himalayapathpricer.hpp File Reference

multipath pricer for European-type Himalaya option.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/multipath.hpp>
```

Include dependency graph for himalayapathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.124.1 Detailed Description

Full path:

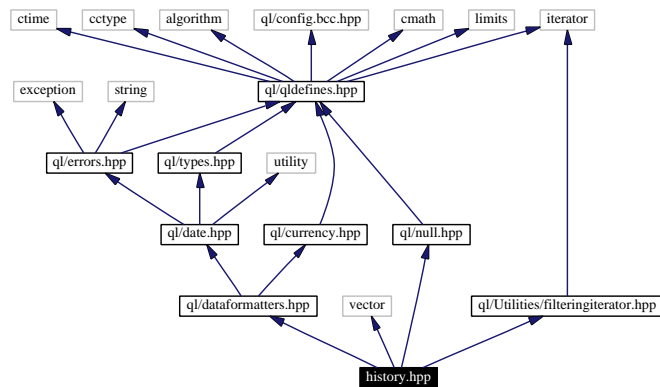
`ql/MonteCarlo/himalayapathpricer.hpp`

12.125 history.hpp File Reference

history class.

```
#include <ql/null.hpp>
#include <ql/Utilities/filteringiterator.hpp>
#include <ql/dataformatters.hpp>
#include <vector>
```

Include dependency graph for history.hpp:



Namespaces

- namespace [QuantLib](#)

12.125.1 Detailed Description

Full path:

ql/history.hpp

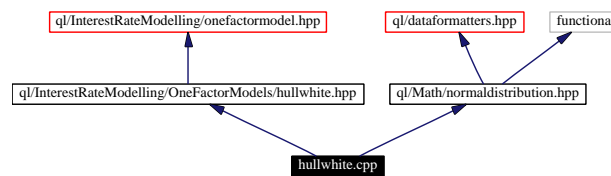
12.126 hullwhite.cpp File Reference

Hull & White model.

```
#include "ql/InterestRateModelling/OneFactorModels/hullwhite.hpp"
```

```
#include "ql/Math/normaldistribution.hpp"
```

Include dependency graph for hullwhite.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.126.1 Detailed Description

Full path:

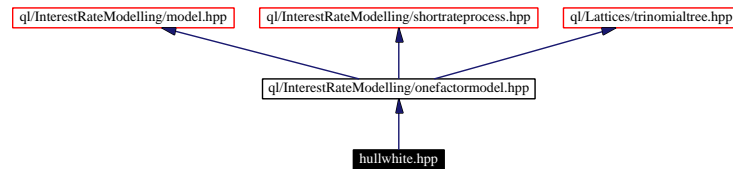
`ql/InterestRateModelling/OneFactorModels/hullwhite.cpp`

12.127 hullwhite.hpp File Reference

Hull & White (HW) model.

```
#include "ql/InterestRateModelling/onefactormodel.hpp"
```

Include dependency graph for hullwhite.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::InterestRateModelling`

12.127.1 Detailed Description

Full path:

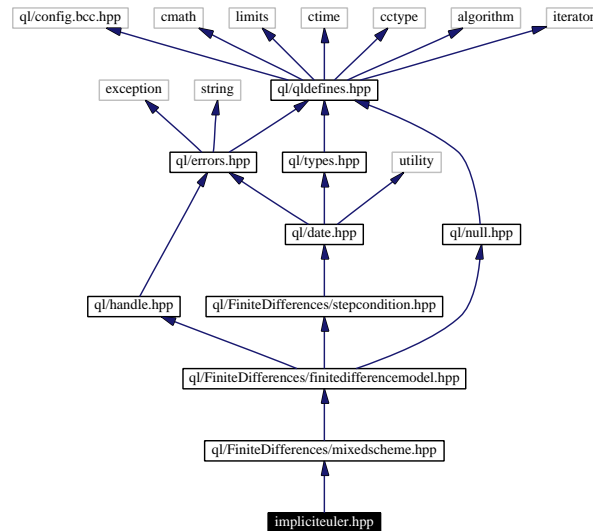
`ql/InterestRateModelling/OneFactorModels/hullwhite.hpp`

12.128 `impliciteuler.hpp` File Reference

implicit Euler scheme for finite difference methods.

```
#include <ql/FiniteDifferences/mixedscheme.hpp>
```

Include dependency graph for `impliciteuler.hpp`:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.128.1 Detailed Description

Full path:

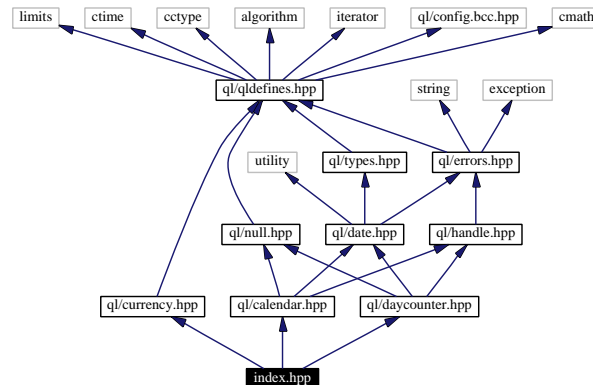
`ql/FiniteDifferences/impliciteuler.hpp`

12.129 index.hpp File Reference

purely virtual base class for indexes.

```
#include <ql/calendar.hpp>
#include <ql/currency.hpp>
#include <ql/daycounter.hpp>
```

Include dependency graph for index.hpp:



Namespaces

- namespace [QuantLib::Indexes](#)
- namespace [QuantLib](#)

12.129.1 Detailed Description

Full path:

ql/index.hpp

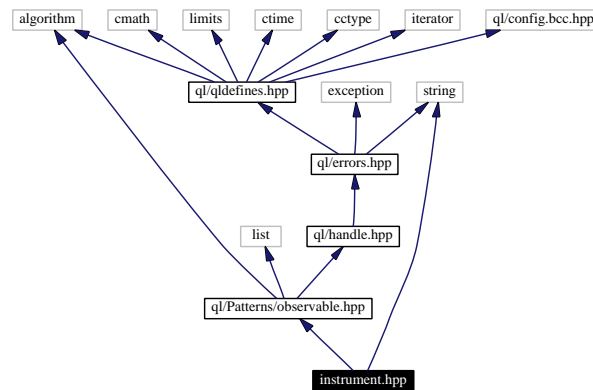
12.130 instrument.hpp File Reference

Abstract instrument class.

```
#include <ql/Patterns/observable.hpp>
```

```
#include <string>
```

Include dependency graph for instrument.hpp:



Namespaces

- namespace [QuantLib::Instruments](#)
- namespace [QuantLib](#)

12.130.1 Detailed Description

Full path:

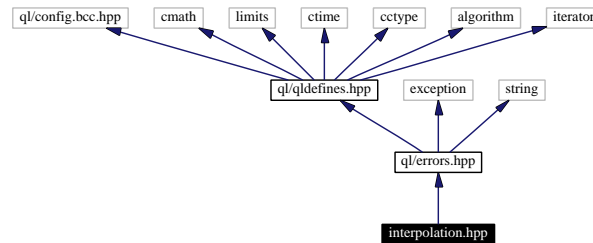
`ql/instrument.hpp`

12.131 interpolation.hpp File Reference

abstract base classes for interpolations.

```
#include <ql/errors.hpp>
```

Include dependency graph for interpolation.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.131.1 Detailed Description

Full path:

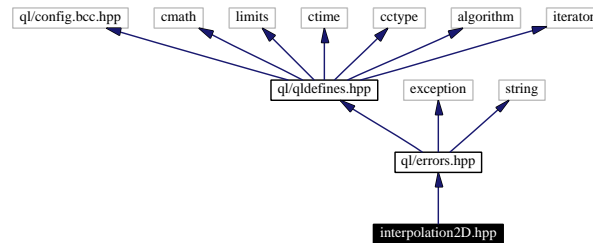
`ql/Math/interpolation.hpp`

12.132 interpolation2D.hpp File Reference

abstract base classes for 2-D interpolations.

```
#include <ql/errors.hpp>
```

Include dependency graph for interpolation2D.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.132.1 Detailed Description

Full path:

`ql/Math/interpolation2D.hpp`

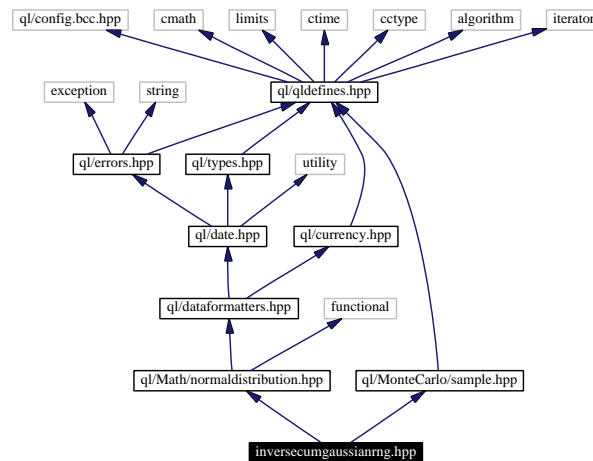
12.133 invsecumgaussianrng.hpp File Reference

Inverse cumulative Gaussian random-number generator.

```
#include <ql/Math/normaldistribution.hpp>
```

```
#include <ql/MonteCarlo/sample.hpp>
```

Include dependency graph for invsecumgaussianrng.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.133.1 Detailed Description

Full path:

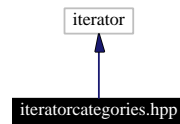
`ql/RandomNumbers/invsecumgaussianrng.hpp`

12.134 iteratorcategories.hpp File Reference

Lowest common denominator between two iterator categories.

```
#include <iterator>
```

Include dependency graph for iteratorcategories.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Utilities](#)

12.134.1 Detailed Description

Full path:

ql/Utilities/iteratorcategories.hpp

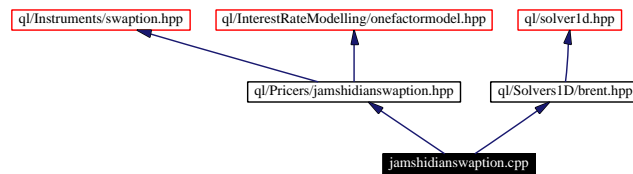
12.135 jamshidianswaption.cpp File Reference

Swaption pricer using Jamshidian's decomposition.

```
#include "ql/Pricers/jamshidianswaption.hpp"
```

```
#include "ql/Solvers1D/brent.hpp"
```

Include dependency graph for jamshidianswaption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.135.1 Detailed Description

Full path:

ql/Pricers/jamshidianswaption.cpp

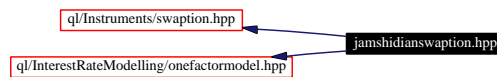
12.136 jamshidianswaption.hpp File Reference

Swaption pricer using Jamshidian's decomposition.

```
#include "ql/Instruments/swaption.hpp"
```

```
#include "ql/InterestRateModelling/onefactormodel.hpp"
```

Include dependency graph for jamshidianswaption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.136.1 Detailed Description

Full path:

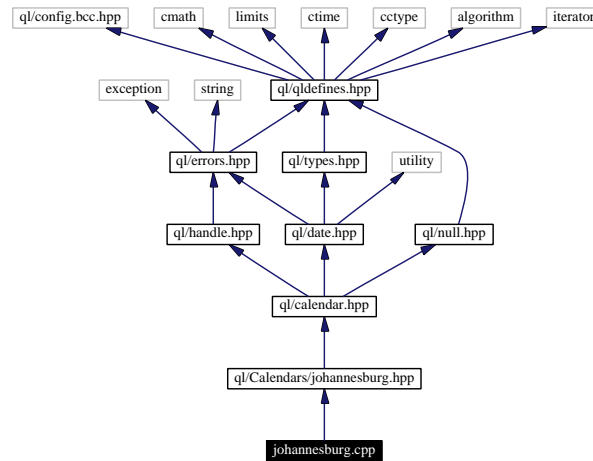
`ql/Pricers/jamshidianswaption.hpp`

12.137 johannesburg.cpp File Reference

Johannesburg calendar.

```
#include <ql/Calendars/johannesburg.hpp>
```

Include dependency graph for johannesburg.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.137.1 Detailed Description

Full path:

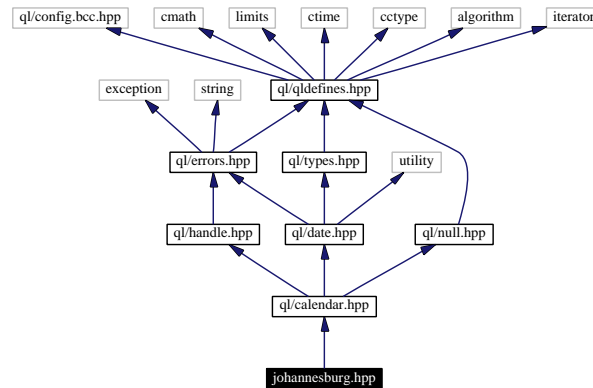
Calendars/johannesburg.cpp

12.138 johannesburg.hpp File Reference

Johannesburg calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for johannesburg.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.138.1 Detailed Description

Full path:

`ql/Calendars/johannesburg.hpp`

12.139 jpylibor.hpp File Reference

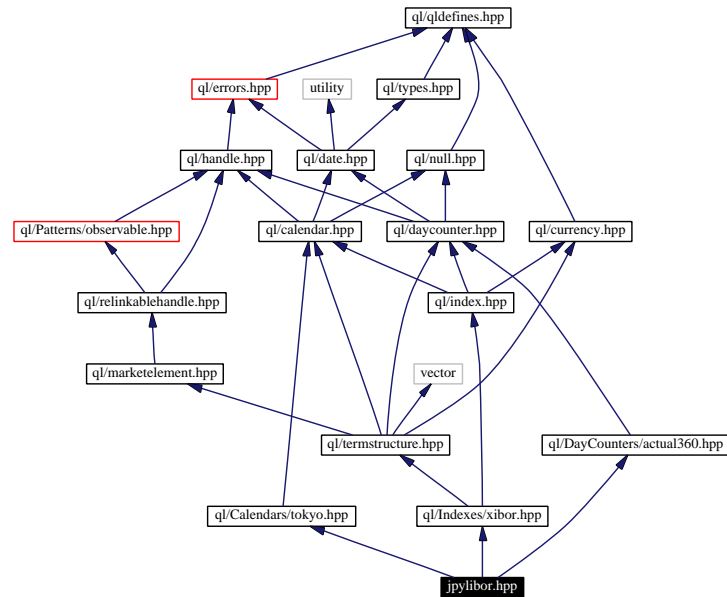
JPY Libor index (Also known as TIBOR, check settlement days).

```
#include <ql/Indexes/xibor.hpp>
```

```
#include <ql/Calendars/tokyo.hpp>
```

```
#include <ql/DayCounters/actual360.hpp>
```

Include dependency graph for jpylibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.139.1 Detailed Description

Full path:

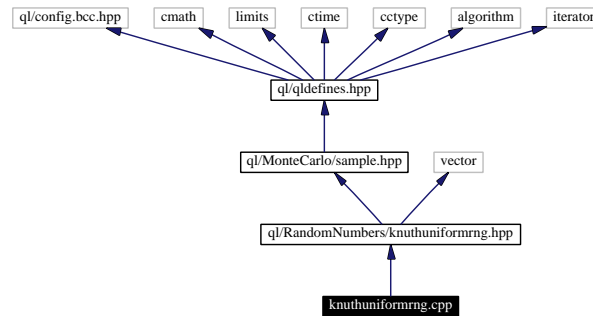
ql/Indexes/jpylibor.hpp

12.140 knuthuniformrng.cpp File Reference

Knuth uniform random number generator.

```
#include <ql/RandomNumbers/knuthuniformrng.hpp>
```

Include dependency graph for knuthuniformrng.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.140.1 Detailed Description

Full path:

`ql/RandomNumbers/knuthuniformrng.cpp`

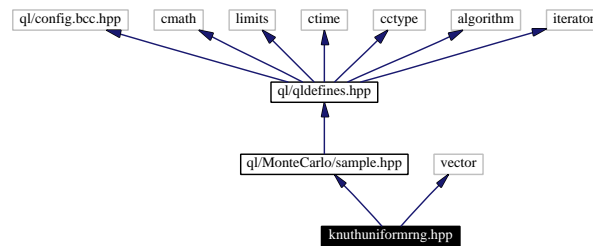
12.141 knuthuniformrng.hpp File Reference

Knuth uniform random number generator.

```
#include <ql/MonteCarlo/sample.hpp>
```

```
#include <vector>
```

Include dependency graph for knuthuniformrng.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.141.1 Detailed Description

Full path:

`ql/RandomNumbers/knuthuniformrng.hpp`

12.142 leastsquare.hpp File Reference

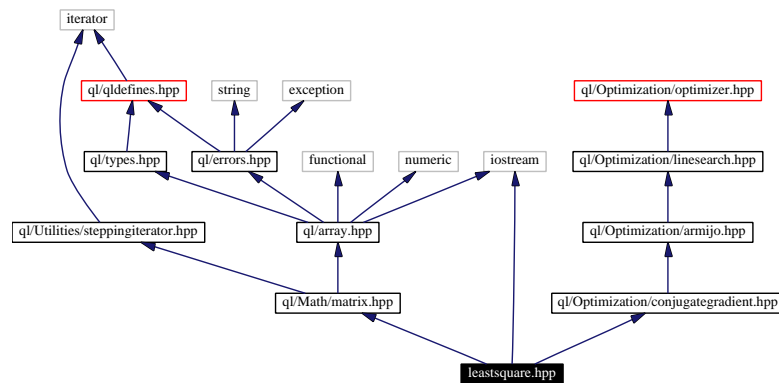
Least square cost function.

```
#include "ql/Math/matrix.hpp"
```

```
#include "ql/Optimization/conjugategradient.hpp"
```

```
#include <iostream>
```

Include dependency graph for leastsquare.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.142.1 Detailed Description

Full path:

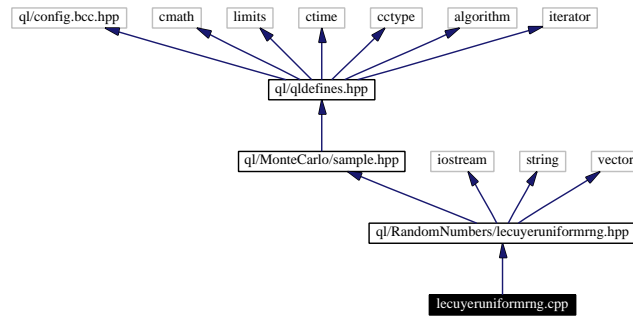
`ql/Optimization/leastsquare.hpp`

12.143 lecuyeruniformrng.cpp File Reference

L'Ecuyer uniform random number generator.

```
#include <ql/RandomNumbers/lecuyeruniformrng.hpp>
```

Include dependency graph for lecuyeruniformrng.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.143.1 Detailed Description

Full path:

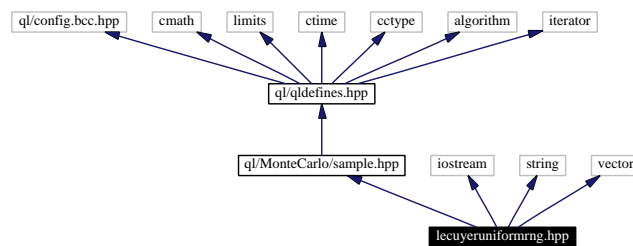
`ql/RandomNumbers/lecuyeruniformrng.cpp`

12.144 lecuyeruniformrng.hpp File Reference

L'Ecuyer uniform random number generator.

```
#include <ql/MonteCarlo/sample.hpp>
#include <iostream>
#include <string>
#include <vector>
```

Include dependency graph for lecuyeruniformrng.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.144.1 Detailed Description

Full path:

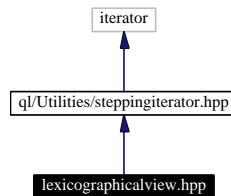
`ql/RandomNumbers/lecuyeruniformrng.hpp`

12.145 lexicographicalview.hpp File Reference

Lexicographical 2-D view of a contiguous set of data.

```
#include <ql/Utilities/steppingiterator.hpp>
```

Include dependency graph for lexicographicalview.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.145.1 Detailed Description

Full path:

`ql/Math/lexicographicalview.hpp`

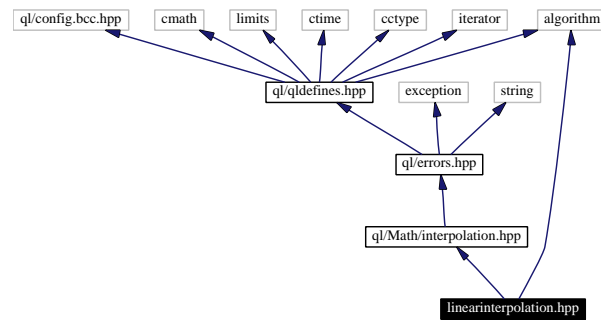
12.146 linearinterpolation.hpp File Reference

linear interpolation between discrete points.

```
#include <ql/Math/interpolation.hpp>
```

```
#include <algorithm>
```

Include dependency graph for linearinterpolation.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.146.1 Detailed Description

Full path:

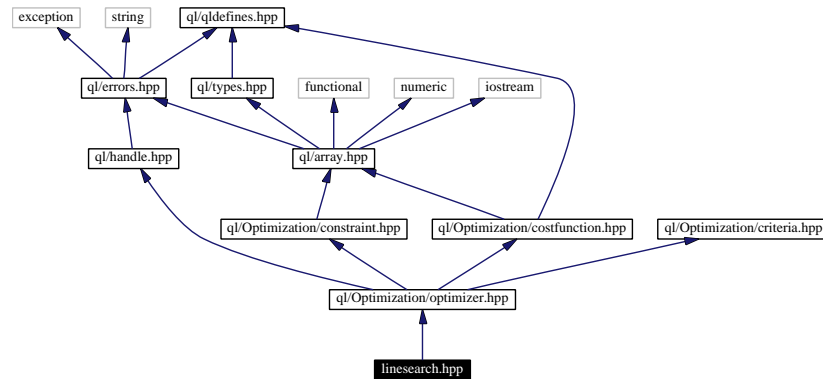
`ql/Math/linearinterpolation.hpp`

12.147 linesearch.hpp File Reference

Line search abstract class.

```
#include "ql/Optimization/optimizer.hpp"
```

Include dependency graph for linesearch.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.147.1 Detailed Description

Full path:

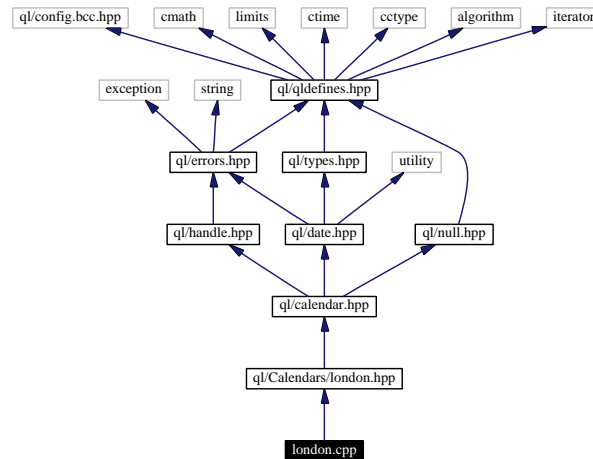
`ql/Optimization/linesearch.hpp`

12.148 london.cpp File Reference

London calendar.

```
#include <ql/Calendars/london.hpp>
```

Include dependency graph for london.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.148.1 Detailed Description

Full path:

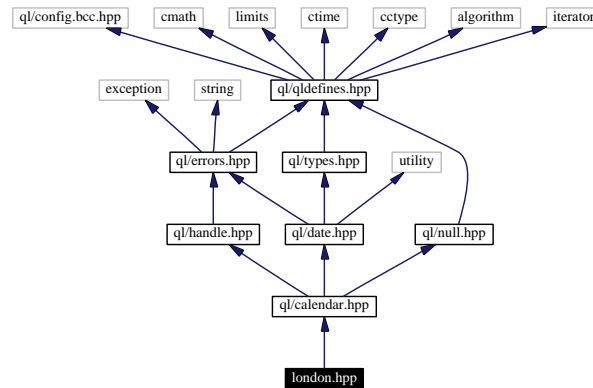
Calendars/london.cpp

12.149 london.hpp File Reference

London calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for london.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.149.1 Detailed Description

Full path:

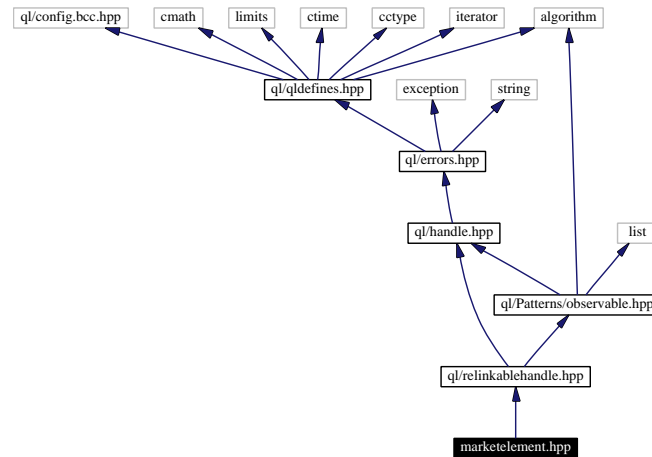
`ql/Calendars/london.hpp`

12.150 marketelement.hpp File Reference

purely virtual base class for market observables.

```
#include <ql/relinkablehandle.hpp>
```

Include dependency graph for marketelement.hpp:



Namespaces

- namespace [QuantLib](#)

12.150.1 Detailed Description

Full path:

`ql/marketelement.hpp`

12.151 mathf.cpp File Reference

math functions.

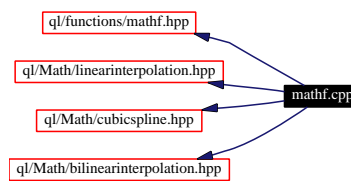
```
#include <ql/functions/mathf.hpp>
```

```
#include <ql/Math/linearinterpolation.hpp>
```

```
#include <ql/Math/cubicspline.hpp>
```

```
#include <ql/Math/bilinearinterpolation.hpp>
```

Include dependency graph for mathf.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Functions**

12.151.1 Detailed Description

Full path:

`ql/functions/mathf.cpp`

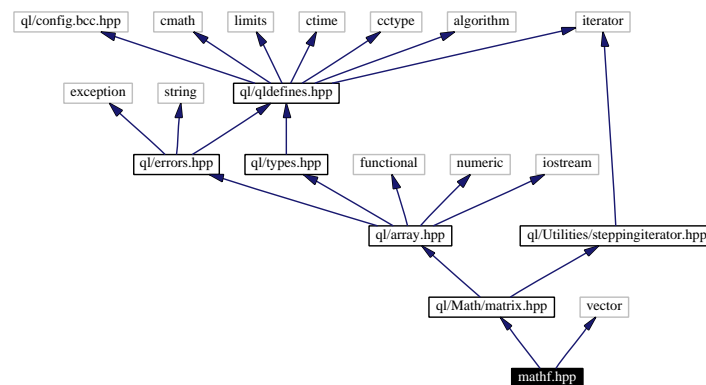
12.152 mathf.hpp File Reference

math functions.

```
#include <ql/Math/matrix.hpp>
```

```
#include <vector>
```

Include dependency graph for mathf.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Functions`

12.152.1 Detailed Description

Full path:

`ql/functions/mathf.hpp`

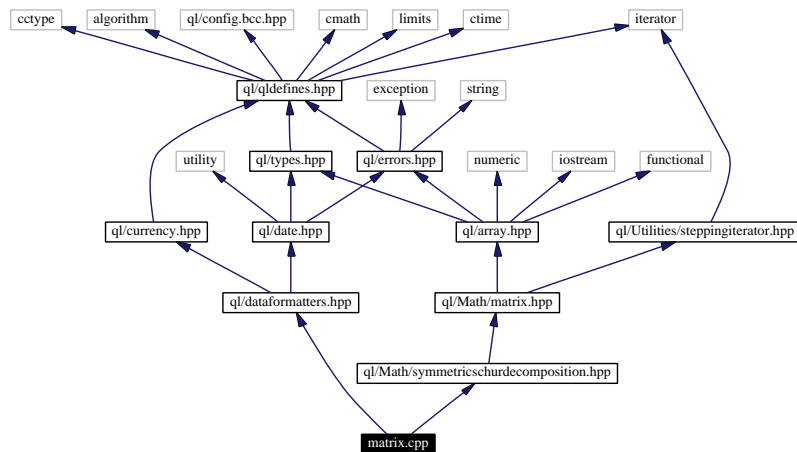
12.153 matrix.cpp File Reference

matrix used in linear algebra.

```
#include <ql/Math/symmetricschurdecomposition.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for matrix.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.153.1 Detailed Description

Full path:

ql/Math/matrix.cpp

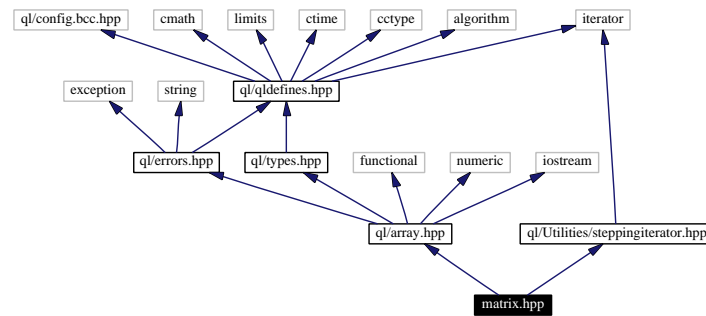
12.154 matrix.hpp File Reference

matrix used in linear algebra.

```
#include <ql/array.hpp>
```

```
#include <ql/Utilities/steppingiterator.hpp>
```

Include dependency graph for matrix.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.154.1 Detailed Description

Full path:

ql/Math/matrix.hpp

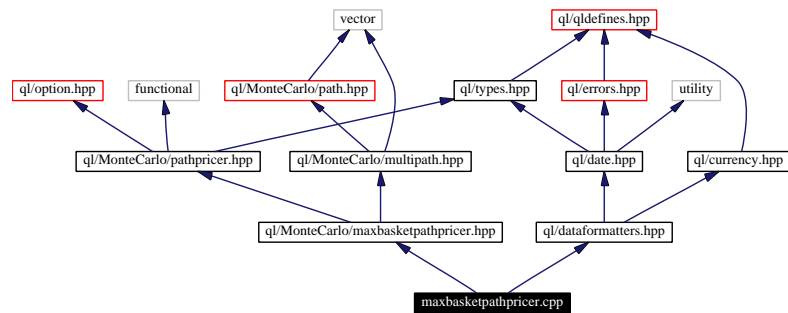
12.155 maxbasketpathpricer.cpp File Reference

multipath pricer for max basket option.

```
#include <ql/MonteCarlo/maxbasketpathpricer.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for maxbasketpathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.155.1 Detailed Description

Full path:

MonteCarlo/basketpathpricer.cpp

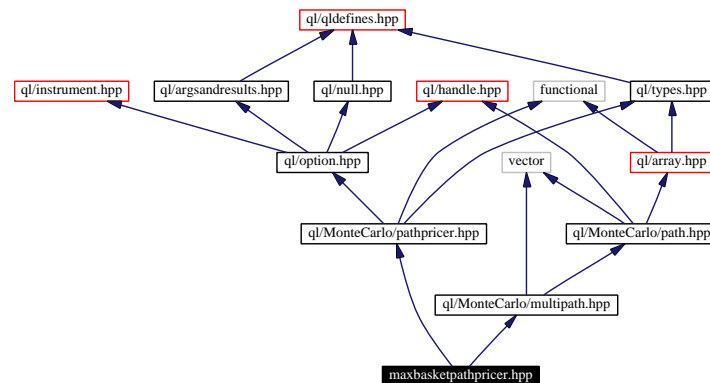
12.156 maxbasketpathpricer.hpp File Reference

multipath pricer for max basket option.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/multipath.hpp>
```

Include dependency graph for maxbasketpathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.156.1 Detailed Description

Full path:

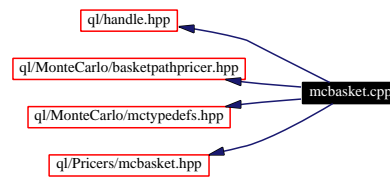
`ql/MonteCarlo/maxbasketpathpricer.hpp`

12.157 mcbasket.cpp File Reference

simple example of multi-factor Monte Carlo pricer.

```
#include <ql/handle.hpp>
#include <ql/MonteCarlo/basketpathpricer.hpp>
#include <ql/MonteCarlo/mctypedefs.hpp>
#include <ql/Pricers/mcbasket.hpp>
```

Include dependency graph for mcbasket.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.157.1 Detailed Description

Full path:

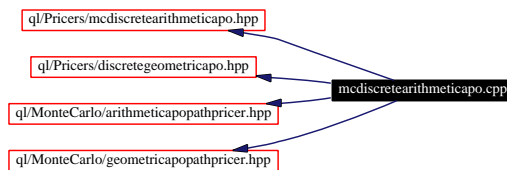
ql/Pricers/mcbasket.cpp

12.158 mcdiscretearithmeticapo.cpp File Reference

Discrete Arithmetic Average Price Option.

```
#include <ql/Pricers/mcdiscretearithmeticapo.hpp>
#include <ql/Pricers/discretegeometricapo.hpp>
#include <ql/MonteCarlo/arithmeticapopathpricer.hpp>
#include <ql/MonteCarlo/geometricapopathpricer.hpp>
```

Include dependency graph for mcdiscretearithmeticapo.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.158.1 Detailed Description

Full path:

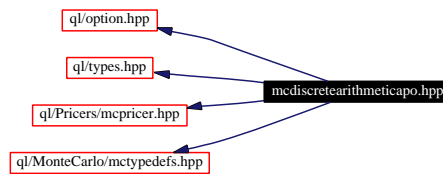
ql/Pricers/mcdiscretearithmeticapo.cpp

12.159 mcdiscretearithmeticapo.hpp File Reference

Discrete Arithmetic Average Price Option.

```
#include <ql/option.hpp>
#include <ql/types.hpp>
#include <ql/Pricers/mcpricer.hpp>
#include <ql/MonteCarlo/mctypedefs.hpp>
```

Include dependency graph for mcdiscretearithmeticapo.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.159.1 Detailed Description

Full path:

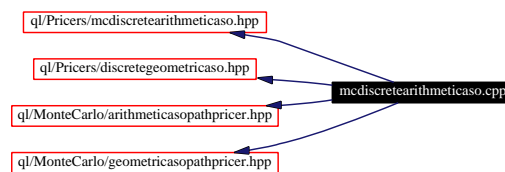
ql/Pricers/mcdiscretearithmeticapo.hpp

12.160 mcdiscretearithmeticaso.cpp File Reference

Discrete Arithmetic Average Strike Option.

```
#include <ql/Pricers/mcdiscretearithmeticaso.hpp>
#include <ql/Pricers/discretegeometricaso.hpp>
#include <ql/MonteCarlo/arithmeticasopathpricer.hpp>
#include <ql/MonteCarlo/geometricasopathpricer.hpp>
```

Include dependency graph for mcdiscretearithmeticaso.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.160.1 Detailed Description

Full path:

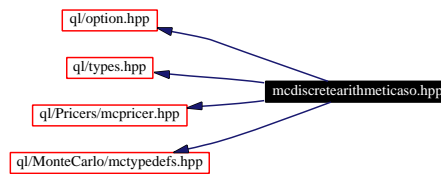
ql/Pricers/mcdiscretearithmeticaso.cpp

12.161 mcdiscretearithmeticaso.hpp File Reference

Discrete Arithmetic Average Strike Option.

```
#include <ql/option.hpp>
#include <ql/types.hpp>
#include <ql/Pricers/mcpricer.hpp>
#include <ql/MonteCarlo/mctypedefs.hpp>
```

Include dependency graph for mcdiscretearithmeticaso.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.161.1 Detailed Description

Full path:

ql/Pricers/mcdiscretearithmeticaso.hpp

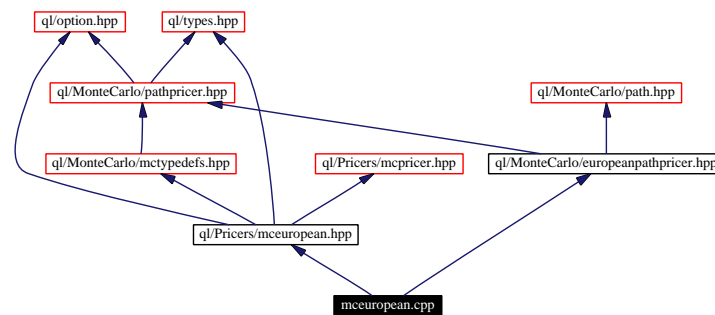
12.162 mceuropean.cpp File Reference

simple example of Monte Carlo pricer.

```
#include <ql/Pricers/mceuropean.hpp>
```

```
#include <ql/MonteCarlo/europeanpathpricer.hpp>
```

Include dependency graph for mceuropean.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.162.1 Detailed Description

Full path:

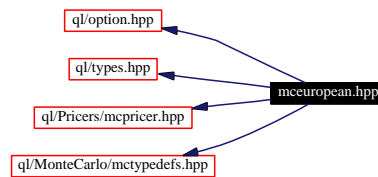
ql/Pricers/mceuropean.cpp

12.163 mceuropean.hpp File Reference

simple example of Monte Carlo pricer.

```
#include <ql/option.hpp>
#include <ql/types.hpp>
#include <ql/Pricers/mcpricer.hpp>
#include <ql/MonteCarlo/mctypedefs.hpp>
```

Include dependency graph for mceuropean.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.163.1 Detailed Description

Full path:

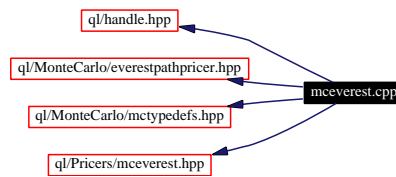
ql/Pricers/mceuropean.hpp

12.164 mceverest.cpp File Reference

Everest-type option pricer.

```
#include <ql/handle.hpp>
#include <ql/MonteCarlo/everestpathpricer.hpp>
#include <ql/MonteCarlo/mctypedefs.hpp>
#include <ql/Pricers/mceverest.hpp>
```

Include dependency graph for mceverest.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.164.1 Detailed Description

Full path:

ql/Pricers/mceverest.cpp

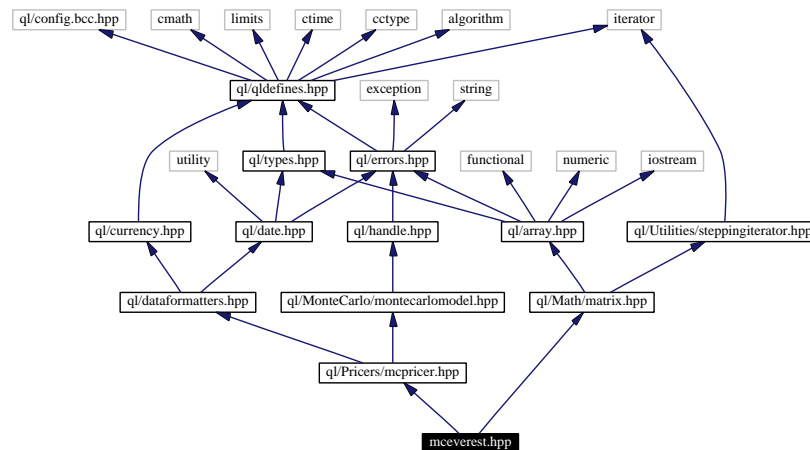
12.165 mceverest.hpp File Reference

Everest-type option pricer.

```
#include <ql/Pricers/mcpricer.hpp>
```

```
#include <ql/Math/matrix.hpp>
```

Include dependency graph for mceverest.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.165.1 Detailed Description

Full path:

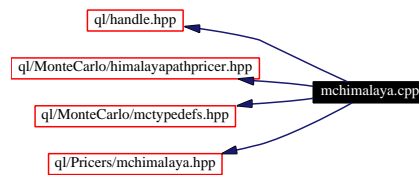
ql/Pricers/mceverest.hpp

12.166 mchimalaya.cpp File Reference

Himalayan-type option pricer.

```
#include <ql/handle.hpp>
#include <ql/MonteCarlo/himalayapathpricer.hpp>
#include <ql/MonteCarlo/mctypedefs.hpp>
#include <ql/Pricers/mchimalaya.hpp>
```

Include dependency graph for mchimalaya.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.166.1 Detailed Description

Full path:

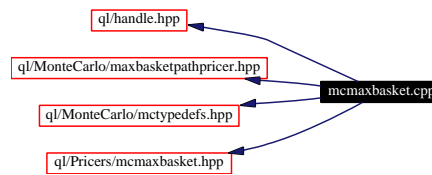
ql/Pricers/mchimalaya.cpp

12.167 mcmAXBasket.cpp File Reference

Max Basket Monte Carlo pricer.

```
#include <ql/handle.hpp>
#include <ql/MonteCarlo/maxBasketPathPricer.hpp>
#include <ql/MonteCarlo/mcTypeDefs.hpp>
#include <ql/Pricers/mcmaxBasket.hpp>
```

Include dependency graph for mcmAXBasket.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.167.1 Detailed Description

Full path:

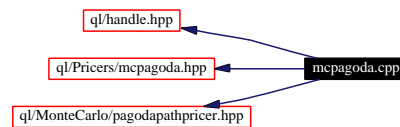
ql/Pricers/mcmaxBasket.cpp

12.168 mcpagoda.cpp File Reference

Roofed multi asset Asian option.

```
#include <ql/handle.hpp>
#include <ql/Pricers/mcpagoda.hpp>
#include <ql/MonteCarlo/pagodapathpricer.hpp>
```

Include dependency graph for mcpagoda.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.168.1 Detailed Description

Full path:

ql/Pricers/mcpagoda.cpp

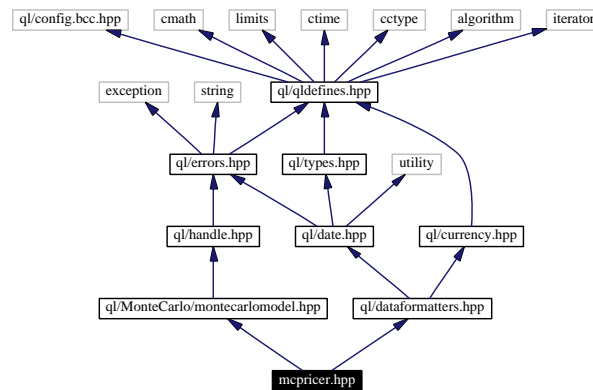
12.169 mcpricer.hpp File Reference

base class for Monte Carlo pricers.

```
#include <ql/dataformatters.hpp>
```

```
#include <ql/MonteCarlo/montecarlomodel.hpp>
```

Include dependency graph for mcpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.169.1 Detailed Description

Full path:

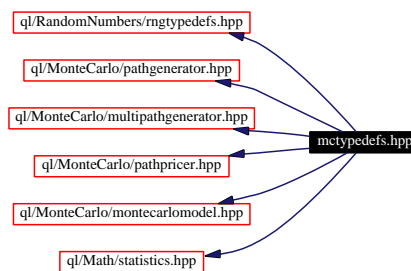
ql/Pricers/mcpricer.hpp

12.170 mctypedefs.hpp File Reference

Default choices for template instantiations.

```
#include <ql/RandomNumbers/rngtypedefs.hpp>
#include <ql/MonteCarlo/pathgenerator.hpp>
#include <ql/MonteCarlo/multipathgenerator.hpp>
#include <ql/MonteCarlo/pathpricer.hpp>
#include <ql/MonteCarlo/montecarlomodel.hpp>
#include <ql/Math/statistics.hpp>
```

Include dependency graph for mctypedefs.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.170.1 Detailed Description

Full path:

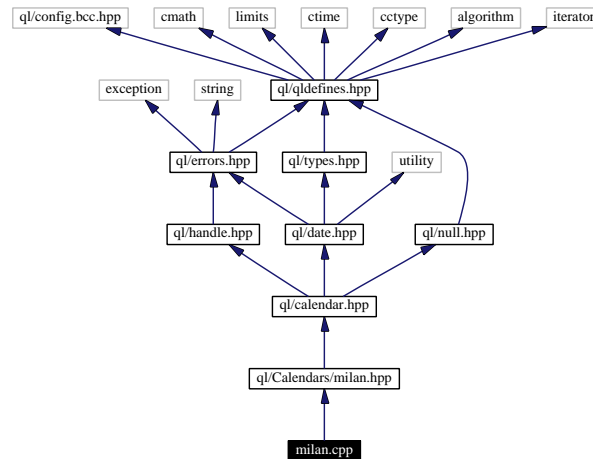
ql/MonteCarlo/mctypedefs.hpp

12.171 milan.cpp File Reference

Milan calendar.

```
#include <ql/Calendars/milan.hpp>
```

Include dependency graph for milan.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.171.1 Detailed Description

Full path:

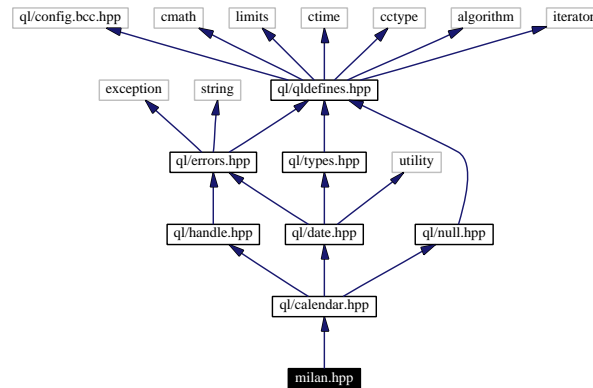
`ql/Calendars/milan.cpp`

12.172 milan.hpp File Reference

Milan calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for milan.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.172.1 Detailed Description

Full path:

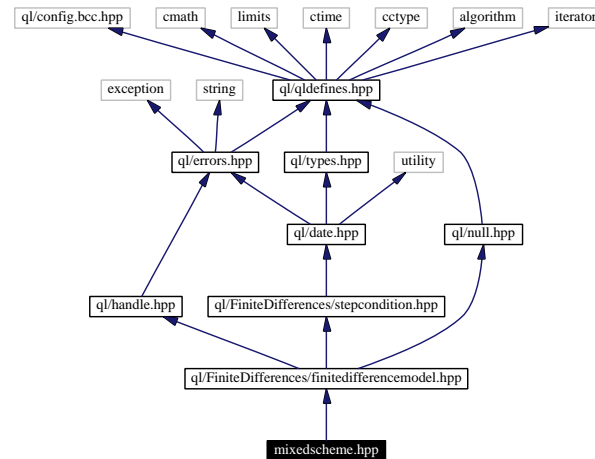
`ql/Calendars/milan.hpp`

12.173 mixedscheme.hpp File Reference

Mixed (explicit/implicit) scheme for finite difference methods.

```
#include <ql/FiniteDifferences/finitedifferencemodel.hpp>
```

Include dependency graph for mixedscheme.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.173.1 Detailed Description

Full path:

`ql/FiniteDifferences/mixedscheme.hpp`

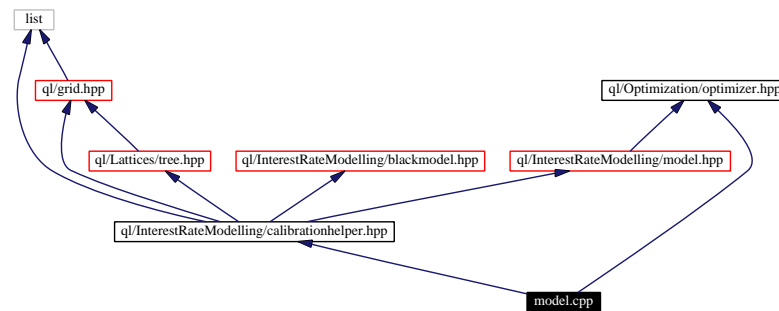
12.174 model.cpp File Reference

Abstract interest rate model class.

```
#include "ql/InterestRateModelling/calibrationhelper.hpp"
```

```
#include "ql/Optimization/optimizer.hpp"
```

Include dependency graph for model.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.174.1 Detailed Description

Full path:

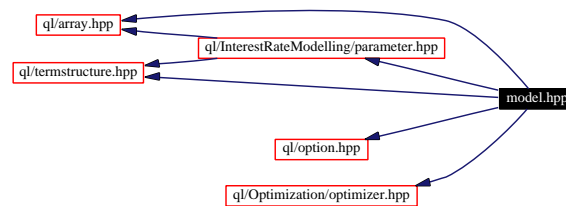
ql/InterestRateModelling/model.cpp

12.175 model.hpp File Reference

Abstract interest rate model class.

```
#include <ql/array.hpp>
#include <ql/option.hpp>
#include <ql/termstructure.hpp>
#include <ql/InterestRateModelling/parameter.hpp>
#include <ql/Optimization/optimizer.hpp>
```

Include dependency graph for model.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.175.1 Detailed Description

Full path:

ql/InterestRateModelling/model.hpp

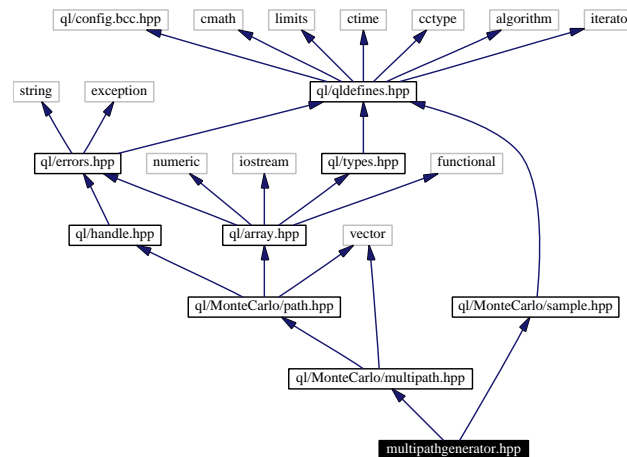
12.176 multipathgenerator.hpp File Reference

Generates a multi path from a random-array generator.

```
#include <ql/MonteCarlo/multipath.hpp>
```

```
#include <ql/MonteCarlo/sample.hpp>
```

Include dependency graph for multipathgenerator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.176.1 Detailed Description

Full path:

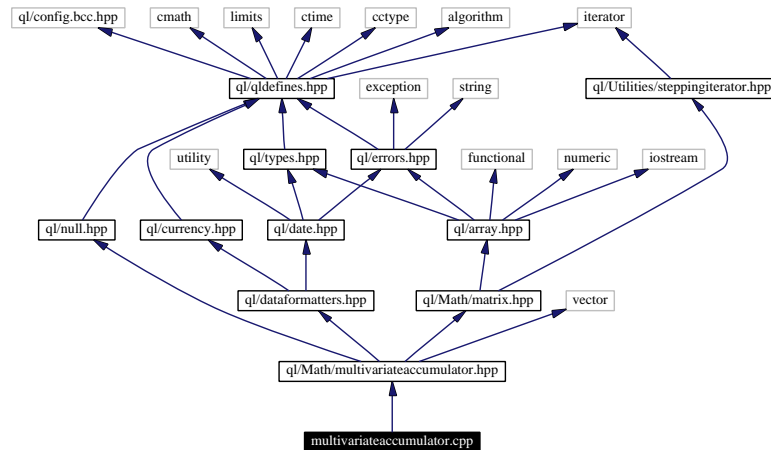
ql/MonteCarlo/multipathgenerator.hpp

12.177 multivariateaccumulator.cpp File Reference

A simple accumulator for vector-type samples.

```
#include <ql/Math/multivariateaccumulator.hpp>
```

Include dependency graph for multivariateaccumulator.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.177.1 Detailed Description

Full path:

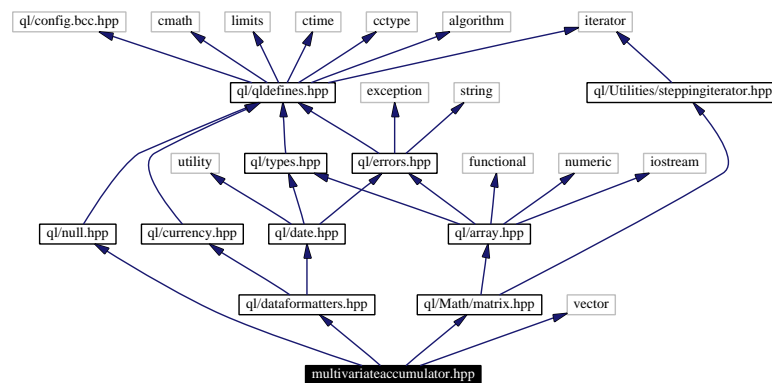
ql/Math/multivariateaccumulator.cpp

12.178 multivariateaccumulator.hpp File Reference

A simple accumulator for vector-type samples.

```
#include <ql/null.hpp>
#include <ql/dataformatters.hpp>
#include <ql/Math/matrix.hpp>
#include <vector>
```

Include dependency graph for multivariateaccumulator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.178.1 Detailed Description

Full path:

ql/Math/multivariateaccumulator.hpp

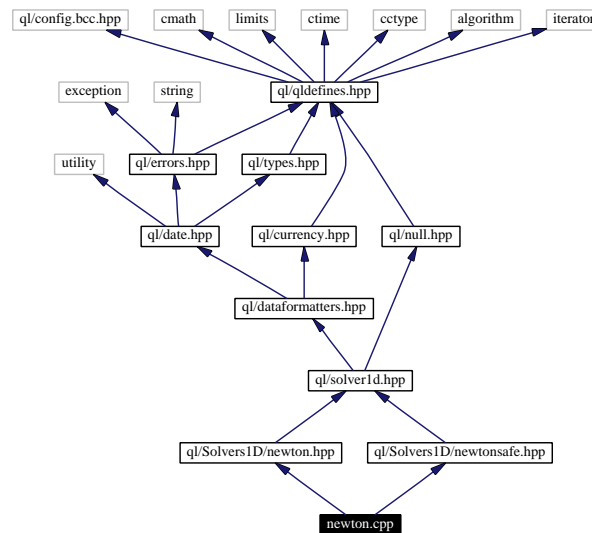
12.179 newton.cpp File Reference

Newton 1-D solver.

```
#include <ql/Solvers1D/newton.hpp>
```

```
#include <ql/Solvers1D/newtonsafe.hpp>
```

Include dependency graph for newton.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.179.1 Detailed Description

Full path:

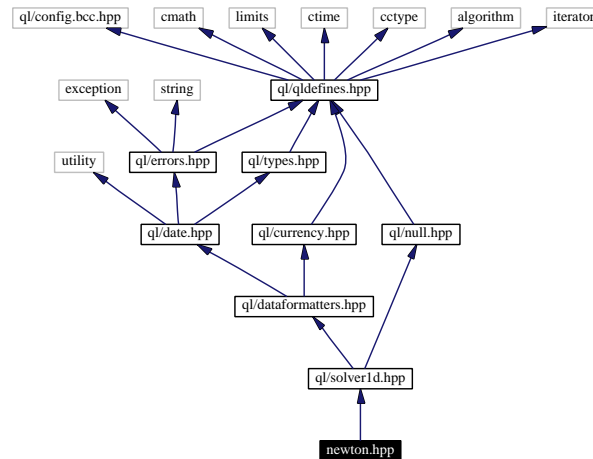
`ql/Solvers1D/newton.cpp`

12.180 newton.hpp File Reference

Newton 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for newton.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.180.1 Detailed Description

Full path:

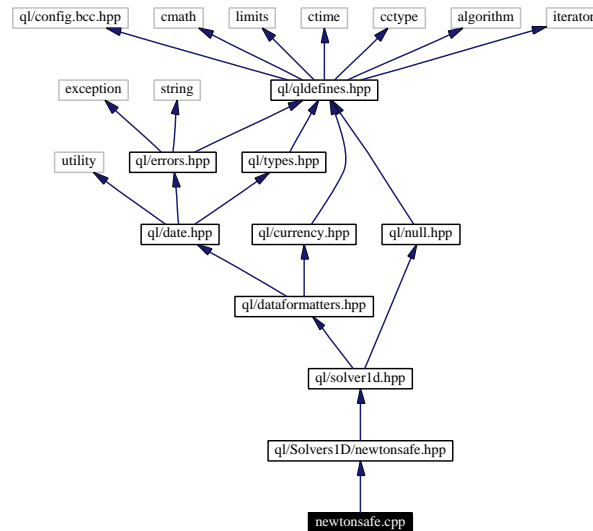
ql/Solvers1D/newton.hpp

12.181 newtonsafe.cpp File Reference

Safe (braketed) Newton 1-D solver.

```
#include <ql/Solvers1D/newtonsafe.hpp>
```

Include dependency graph for newtonsafe.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.181.1 Detailed Description

Full path:

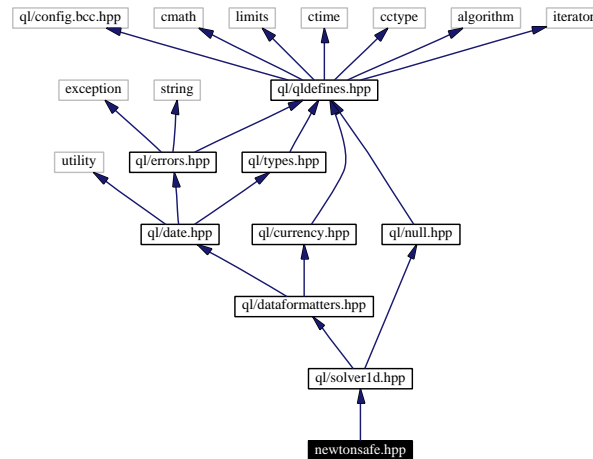
`ql/Solvers1D/newtonsafe.cpp`

12.182 newtonsafe.hpp File Reference

Safe (braketed) Newton 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for newtonsafe.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.182.1 Detailed Description

Full path:

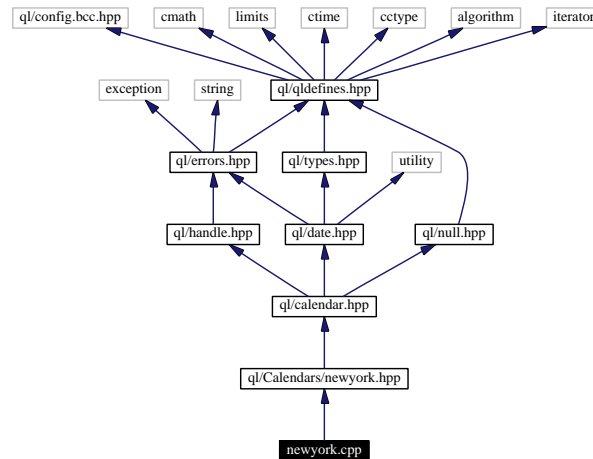
`ql/Solvers1D/newtonsafe.hpp`

12.183 newyork.cpp File Reference

New York calendar.

```
#include <ql/Calendars/newyork.hpp>
```

Include dependency graph for newyork.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.183.1 Detailed Description

Full path:

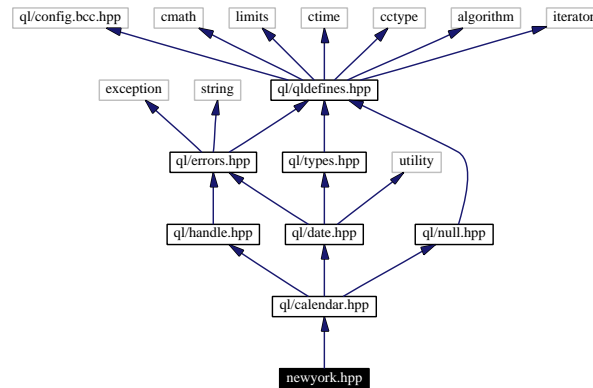
ql/Calendars/newyork.cpp

12.184 newyork.hpp File Reference

New York calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for newyork.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.184.1 Detailed Description

Full path:

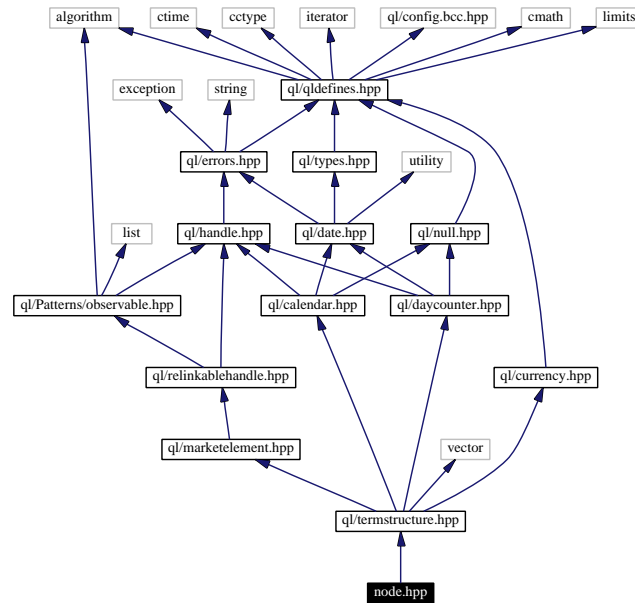
ql/Calendars/newyork.hpp

12.185 node.hpp File Reference

Node class.

```
#include <ql/termstructure.hpp>
```

Include dependency graph for node.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Lattices](#)

12.185.1 Detailed Description

Full path:

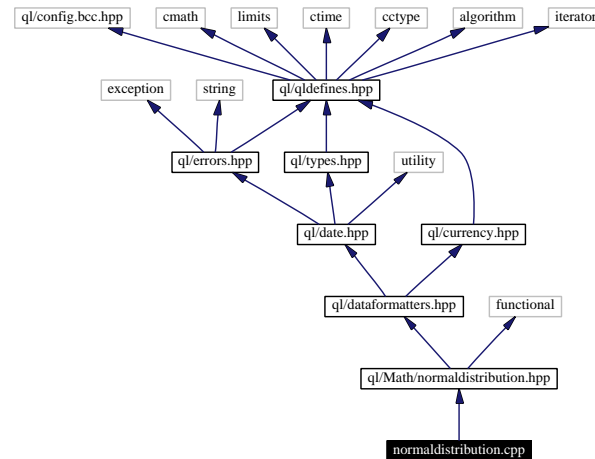
ql/Lattices/node.hpp

12.186 normaldistribution.cpp File Reference

normal, cumulative and inverse cumulative distributions.

```
#include <ql/Math/normaldistribution.hpp>
```

Include dependency graph for normaldistribution.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.186.1 Detailed Description

Full path:

`ql/Math/normaldistribution.cpp`

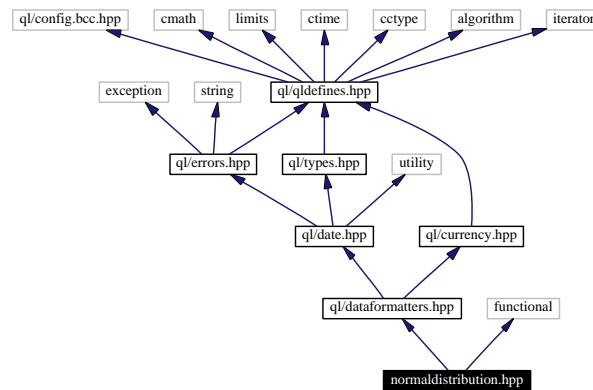
12.187 normaldistribution.hpp File Reference

normal, cumulative and inverse cumulative distributions.

```
#include <ql/dataformatters.hpp>
```

```
#include <functional>
```

Include dependency graph for normaldistribution.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.187.1 Detailed Description

Full path:

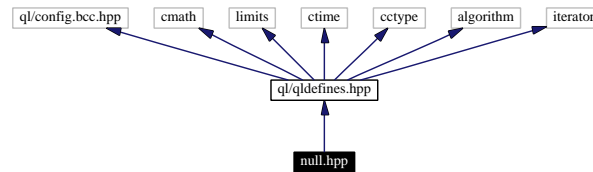
`ql/Math/normaldistribution.hpp`

12.188 null.hpp File Reference

null values.

```
#include <ql/qldefines.hpp>
```

Include dependency graph for null.hpp:



Namespaces

- namespace [QuantLib](#)

12.188.1 Detailed Description

Full path:

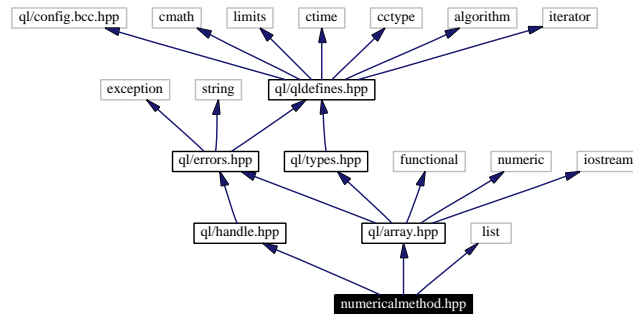
`ql/null.hpp`

12.189 numericalmethod.hpp File Reference

Numerical method class.

```
#include <ql/array.hpp>
#include <ql/handle.hpp>
#include <list>
```

Include dependency graph for numericalmethod.hpp:



Namespaces

- namespace [QuantLib](#)

12.189.1 Detailed Description

Full path:

ql/numericalmethod.hpp

12.190 observable.hpp File Reference

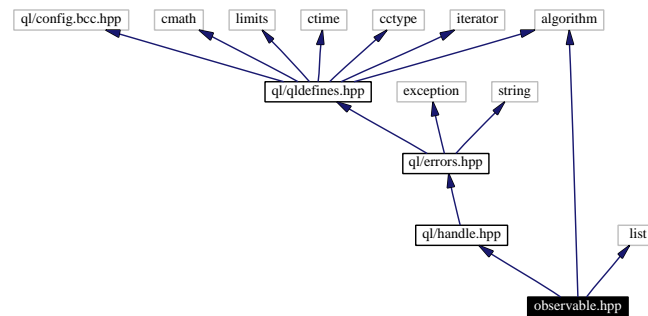
observer/observable pattern.

```
#include <ql/handle.hpp>
```

```
#include <list>
```

```
#include <algorithm>
```

Include dependency graph for observable.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Patterns](#)

12.190.1 Detailed Description

Full path:

`ql/Patterns/observable.hpp`

12.191 onefactormodel.cpp File Reference

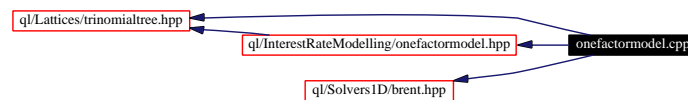
Abstract one-factor interest rate model class.

```
#include "ql/InterestRateModelling/onefactormodel.hpp"
```

```
#include "ql/Lattices/trinomialtree.hpp"
```

```
#include "ql/Solvers1D/brent.hpp"
```

Include dependency graph for onefactormodel.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.191.1 Detailed Description

Full path:

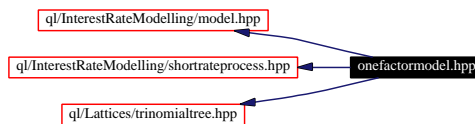
`ql/InterestRateModelling/onefactormodel.cpp`

12.192 onefactormodel.hpp File Reference

Abstract one-factor interest rate model class.

```
#include <ql/InterestRateModelling/model.hpp>
#include <ql/InterestRateModelling/shortrateprocess.hpp>
#include <ql/Lattices/trinomialtree.hpp>
```

Include dependency graph for onefactormodel.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.192.1 Detailed Description

Full path:

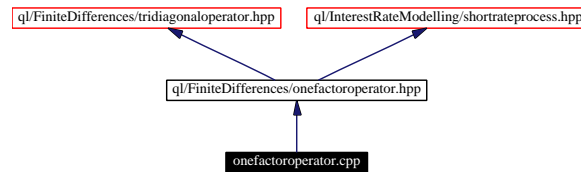
`ql/InterestRateModelling/onefactormodel.hpp`

12.193 onefactoroperator.cpp File Reference

differential operator for one-factor interest rate models.

```
#include "ql/FiniteDifferences/onefactoroperator.hpp"
```

Include dependency graph for onefactoroperator.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.193.1 Detailed Description

Full path:

`ql/FiniteDifferences/onefactoroperator.cpp`

12.194 onefactoroperator.hpp File Reference

general differential operator for one-factor interest rate models.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
```

```
#include <ql/InterestRateModelling/shortrateprocess.hpp>
```

Include dependency graph for onefactoroperator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.194.1 Detailed Description

Full path:

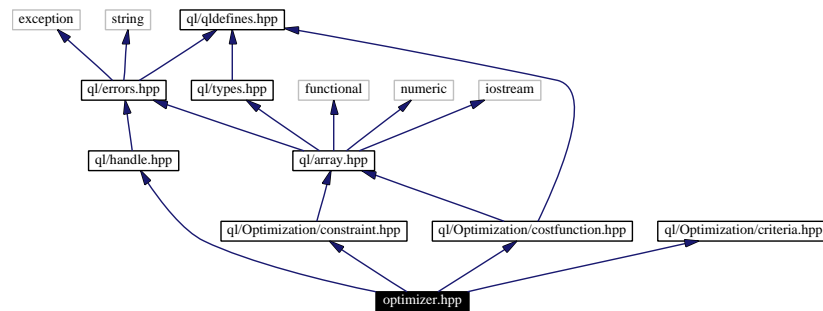
`ql/FiniteDifferences/onefactoroperator.hpp`

12.195 optimizer.hpp File Reference

Abstract optimization class.

```
#include <ql/handle.hpp>
#include <ql/Optimization/constraint.hpp>
#include <ql/Optimization/costfunction.hpp>
#include <ql/Optimization/criteria.hpp>
```

Include dependency graph for optimizer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Optimization**

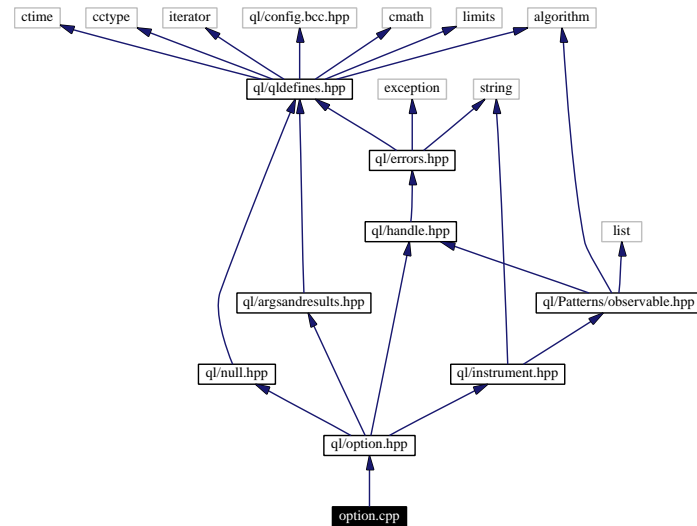
12.195.1 Detailed Description

Full path:

ql/Optimization/optimizer.hpp

Base option class.

Include dependency graph for option.cpp:



- namespace **QuantLib**

Full path:

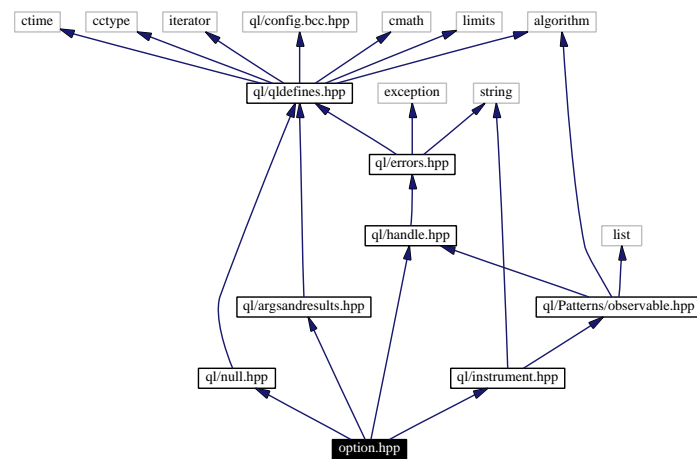
ql/option.cpp

12.197 option.hpp File Reference

Base option class.

```
#include <ql/instrument.hpp>
#include <ql/argsandresults.hpp>
#include <ql/handle.hpp>
#include <ql/null.hpp>
```

Include dependency graph for option.hpp:



Namespaces

- namespace [QuantLib](#)

12.197.1 Detailed Description

Full path:

ql/option.hpp

12.198 pagodapathpricer.cpp File Reference

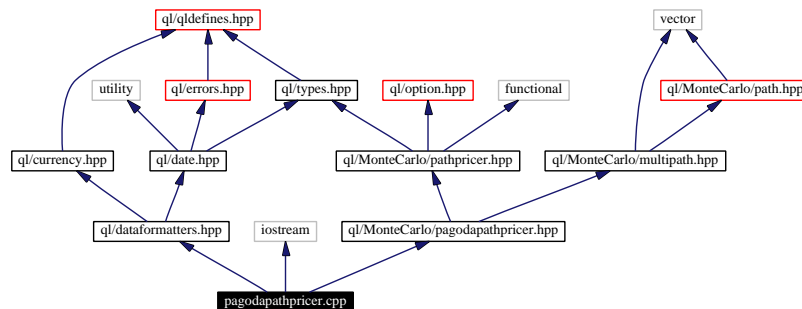
path pricer for pagoda options.

```
#include <ql/MonteCarlo/pagodapathpricer.hpp>
```

```
#include <ql/dataformatters.hpp>
```

```
#include <iostream>
```

Include dependency graph for pagodapathpricer.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.198.1 Detailed Description

Full path:

MonteCarlo/pagodapathpricer.cpp

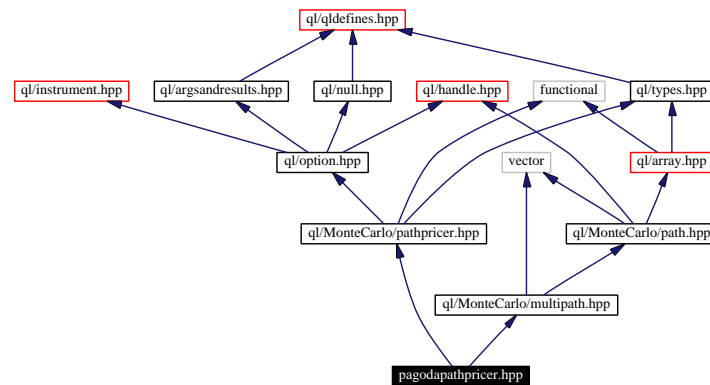
12.199 pagodapathpricer.hpp File Reference

path pricer for pagoda options.

```
#include <ql/MonteCarlo/pathpricer.hpp>
```

```
#include <ql/MonteCarlo/multipath.hpp>
```

Include dependency graph for pagodapathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.199.1 Detailed Description

Full path:

`ql/MonteCarlo/pagodapathpricer.hpp`

12.200 parameter.hpp File Reference

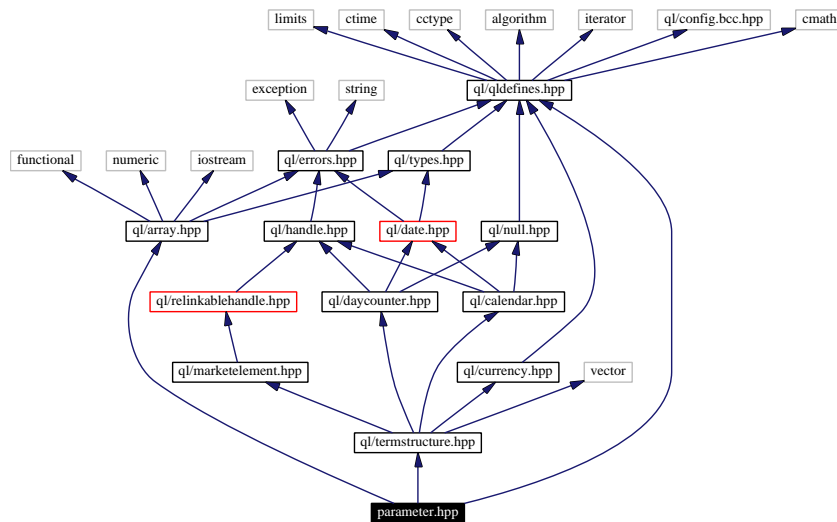
Model parameter classes.

```
#include <ql/qldefines.hpp>
```

```
#include <ql/array.hpp>
```

```
#include <ql/termstructure.hpp>
```

Include dependency graph for parameter.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::InterestRateModelling](#)

12.200.1 Detailed Description

Full path:

ql/InterestRateModelling/parameter.hpp

12.201 pathpricer.hpp File Reference

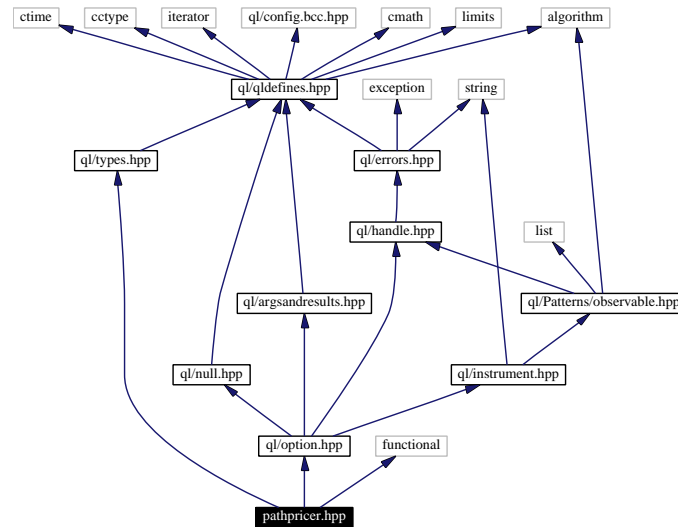
base class for single-path pricers.

```
#include <ql/option.hpp>
```

```
#include <ql/types.hpp>
```

```
#include <functional>
```

Include dependency graph for pathpricer.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.201.1 Detailed Description

Full path:

ql/MonteCarlo/pathpricer.hpp

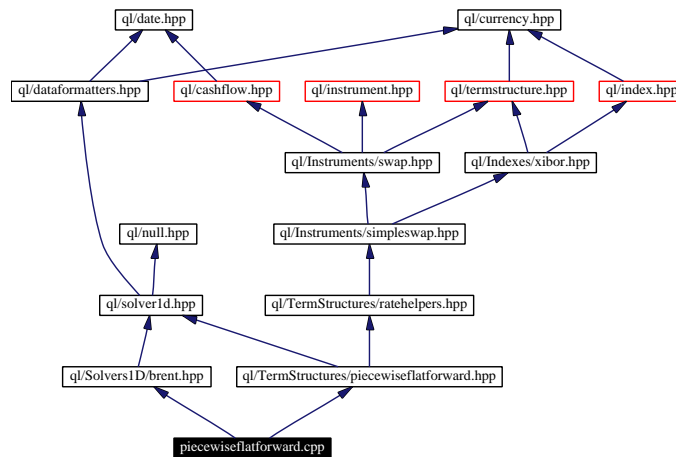
12.202 piecewiseflatforward.cpp File Reference

piecewise flat forward term structure.

```
#include <ql/TermStructures/piecewiseflatforward.hpp>
```

```
#include <ql/Solvers1D/brent.hpp>
```

Include dependency graph for piecewiseflatforward.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::TermStructures](#)

12.202.1 Detailed Description

Full path:

`ql/TermStructures/piecewiseflatforward.cpp`

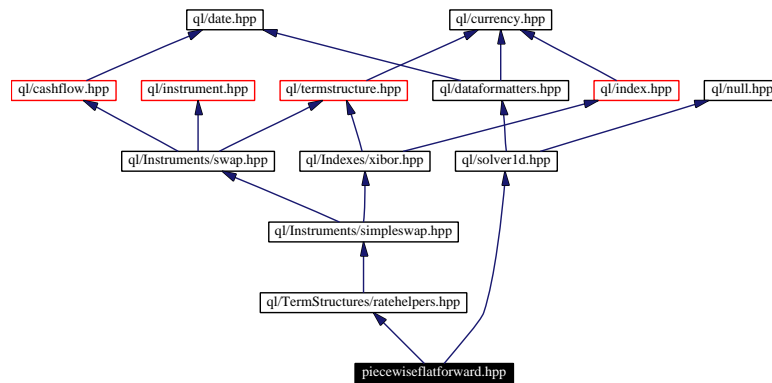
12.203 piecewiseflatforward.hpp File Reference

piecewise flat forward term structure.

```
#include <ql/TermStructures/ratehelpers.hpp>
```

```
#include <ql/solver1d.hpp>
```

Include dependency graph for piecewiseflatforward.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::TermStructures](#)

12.203.1 Detailed Description

Full path:

ql/TermStructures/piecewiseflatforward.hpp

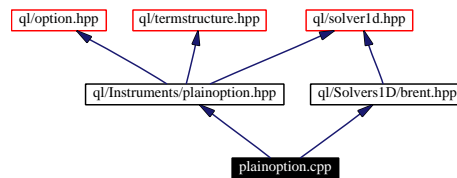
12.204 plainoption.cpp File Reference

Plain (no dividends, no barriers) option on a single asset.

```
#include <ql/Instruments/plainoption.hpp>
```

```
#include <ql/Solvers1D/brent.hpp>
```

Include dependency graph for plainoption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)
- namespace [QuantLib::Pricers](#)

12.204.1 Detailed Description

Full path:

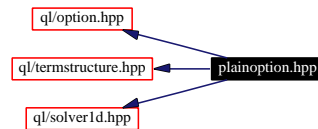
ql/Instruments/plainoption.cpp

12.205 plainoption.hpp File Reference

Plain (no dividends, no barriers) option on a single asset.

```
#include <ql/option.hpp>
#include <ql/termstructure.hpp>
#include <ql/solver1d.hpp>
```

Include dependency graph for plainoption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)
- namespace [QuantLib::Pricers](#)

12.205.1 Detailed Description

Full path:

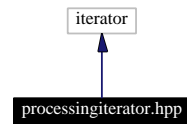
`ql/Instruments/plainoption.hpp`

12.206 processingiterator.hpp File Reference

Iterator mapping a unary function to an underlying sequence.

```
#include <iterator>
```

Include dependency graph for processingiterator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Utilities](#)

12.206.1 Detailed Description

Full path:

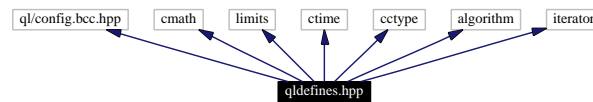
ql/Utilities/processingiterator.hpp

12.207 qldefines.hpp File Reference

Global definitions and compiler switches.

```
#include <ql/config.bcc.hpp>
#include <cmath>
#include <limits>
#include <ctime>
#include <cctype>
#include <algorithm>
#include <iterator>
```

Include dependency graph for qldefines.hpp:



Namespaces

- namespace [QuantLib](#)

Defines

- #define [QL_HEX_VERSION](#) 0x000300b1
version hexadecimal number.
- #define [QL_VERSION](#) "0.3.0b1-20020322"
version string.
- #define [QL_TRACE_LEVEL](#) 0
global trace level (may be superseded locally by a greater value).
- #define [QL_DUMMY_RETURN](#)(x)
Is a dummy return statement required?
- #define [QL_MIN_INT](#) ((std::numeric_limits<int>::min)())
- #define [QL_MAX_INT](#) ((std::numeric_limits<int>::max)())
- #define [QL_MIN_DOUBLE](#) -((std::numeric_limits<double>::max)())
- #define [QL_MAX_DOUBLE](#) ((std::numeric_limits<double>::max)())
- #define [QL_EPSILON](#) ((std::numeric_limits<double>::epsilon)())
- #define [QL_MIN_POSITIVE_DOUBLE](#) ((std::numeric_limits<double>::min)())
- #define [QL_DECLARE_TEMPLATE_SPECIALIZATIONS](#)
Blame Microsoft for this one...
- #define [QL_ALLOW_TEMPLATE_METHOD_CALLS](#) 1
Blame Microsoft for this one...

- `#define QL_EXPRESSION_TEMPLATES_WORK 1`
- `#define QL_TEMPLATE_METAPROGRAMMING_WORKS 1`
- `#define QL_SPECIALIZE_ITERATOR_TRAITS(T)`
- `#define QL_REVERSE_ITERATOR(iterator, type) std::reverse_iterator< iterator >`
Blame Microsoft for this one...
- `#define QL_FULL_ITERATOR_SUPPORT`

12.207.1 Detailed Description

Full path:

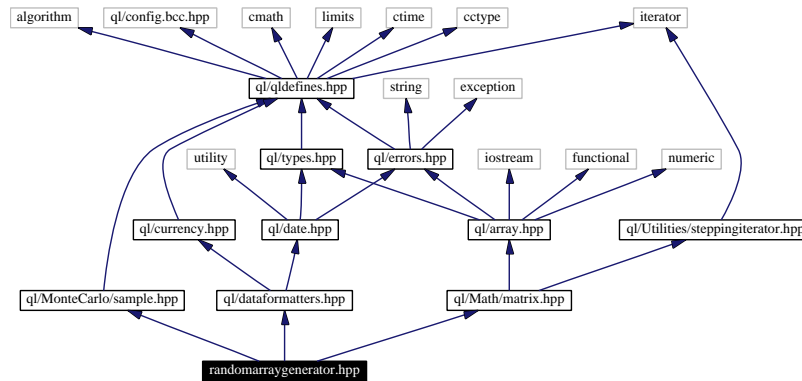
ql/qldefines.hpp

12.208 randomarraygenerator.hpp File Reference

Generates random arrays from a random number generator.

```
#include <ql/Math/matrix.hpp>
#include <ql/MonteCarlo/sample.hpp>
#include <ql/dataformatters.hpp>
```

Include dependency graph for randomarraygenerator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.208.1 Detailed Description

Full path:

ql/RandomNumbers/randomarraygenerator.hpp

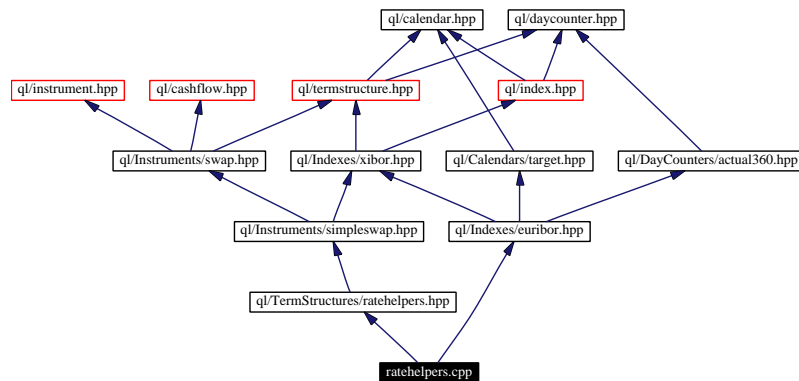
12.209 ratehelpers.cpp File Reference

rate helpers base class.

```
#include <ql/TermStructures/ratehelpers.hpp>
```

```
#include <ql/Indexes/euribor.hpp>
```

Include dependency graph for ratehelpers.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::TermStructures](#)

12.209.1 Detailed Description

Full path:

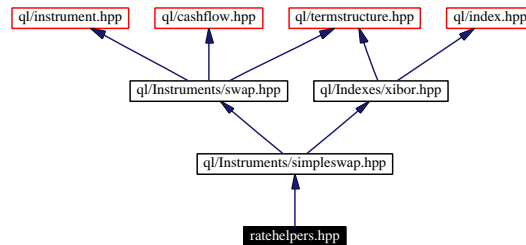
`ql/TermStructures/ratehelpers.cpp`

12.210 ratehelpers.hpp File Reference

rate helpers base class.

```
#include <ql/Instruments/simpleswap.hpp>
```

Include dependency graph for ratehelpers.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::TermStructures](#)

12.210.1 Detailed Description

Full path:

`ql/TermStructures/ratehelpers.hpp`

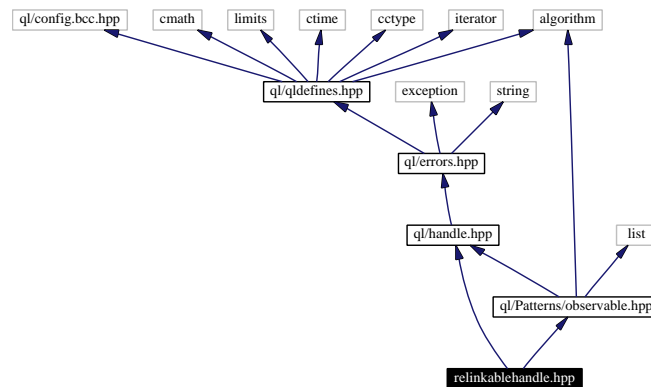
12.211 relinkablehandle.hpp File Reference

Globally accessible relinkable pointer.

```
#include <ql/handle.hpp>
```

```
#include <ql/Patterns/observable.hpp>
```

Include dependency graph for relinkablehandle.hpp:



Namespaces

- namespace [QuantLib](#)

12.211.1 Detailed Description

Full path:

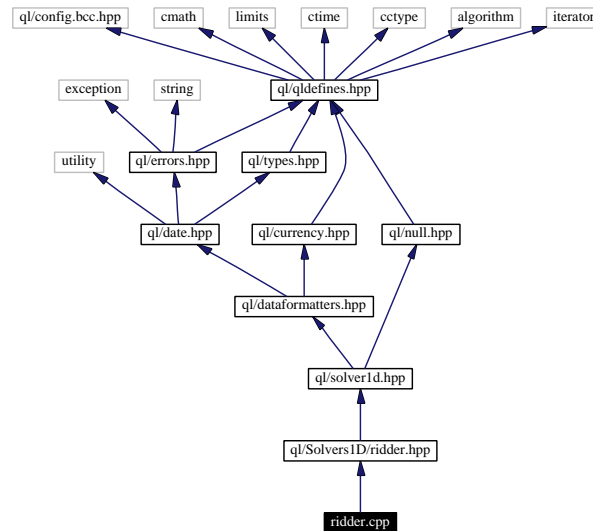
`ql/relinkablehandle.hpp`

12.212 ridder.cpp File Reference

Ridder 1-D solver.

```
#include <ql/Solvers1D/ridder.hpp>
```

Include dependency graph for ridder.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

Defines

- #define **SIGN**(a, b) ((b) >= 0.0 ? QL_FABS(a) : -QL_FABS(a))

12.212.1 Detailed Description

Full path:

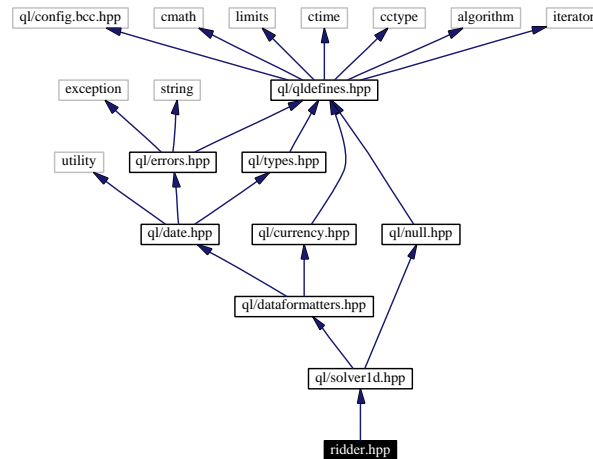
ql/Solvers1D/ridder.cpp

12.213 ridder.hpp File Reference

Ridder 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for ridder.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.213.1 Detailed Description

Full path:

`ql/Solvers1D/ridder.hpp`

12.214 riskmeasures.hpp File Reference

Risk functions.

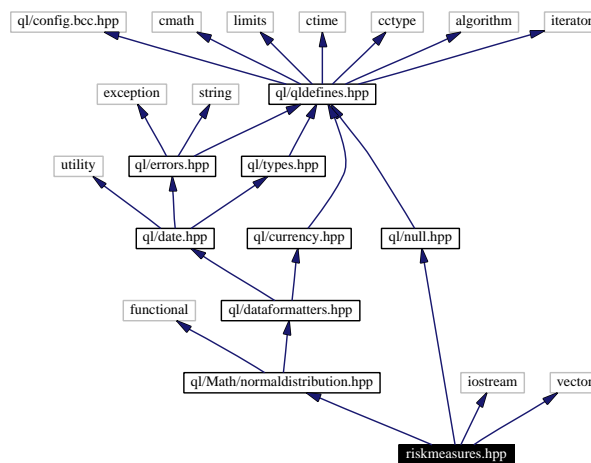
```
#include <ql/null.hpp>
```

```
#include <ql/Math/normaldistribution.hpp>
```

```
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for riskmeasures.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.214.1 Detailed Description

Full path:

ql/Math/riskmeasures.hpp

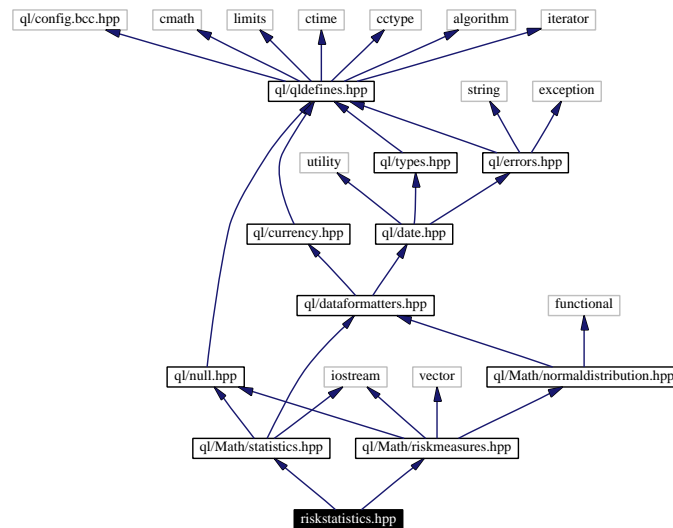
12.215 riskstatistics.hpp File Reference

Normal distribution risk analysis tool: VAR, (average) shortfall.

```
#include <ql/Math/statistics.hpp>
```

```
#include <ql/Math/riskmeasures.hpp>
```

Include dependency graph for riskstatistics.hpp:



Namespaces

- namespace [QuantLib](#)

12.215.1 Detailed Description

Full path:

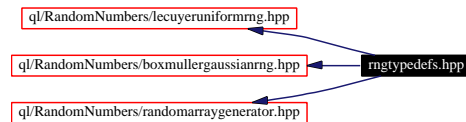
ql/riskstatistics.hpp

12.216 rngtypedefs.hpp File Reference

Default choices for template instantiations.

```
#include <ql/RandomNumbers/lecuyeruniformrng.hpp>
#include <ql/RandomNumbers/boxmullergaussianrng.hpp>
#include <ql/RandomNumbers/randomarraygenerator.hpp>
```

Include dependency graph for rngtypedefs.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::RandomNumbers](#)

12.216.1 Detailed Description

Full path:

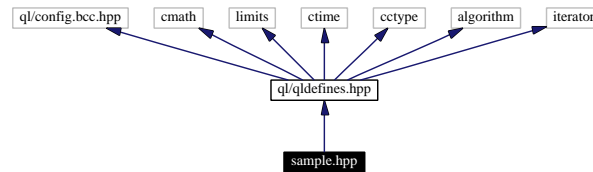
`ql/RandomNumbers/rngtypedefs.hpp`

12.217 sample.hpp File Reference

weighted sample.

```
#include <ql/qldefines.hpp>
```

Include dependency graph for sample.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::MonteCarlo](#)

12.217.1 Detailed Description

Full path:

`ql/MonteCarlo/sample.hpp`

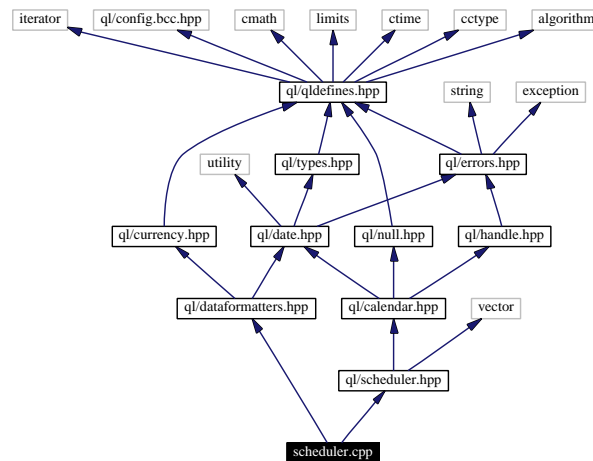
12.218 scheduler.cpp File Reference

date scheduler.

```
#include <ql/scheduler.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for scheduler.cpp:



Namespaces

- namespace [QuantLib](#)

12.218.1 Detailed Description

Full path:

ql/scheduler.cpp

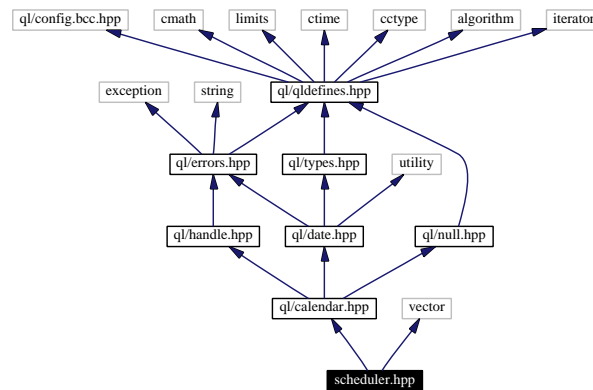
12.219 scheduler.hpp File Reference

date scheduler.

```
#include <ql/calendar.hpp>
```

```
#include <vector>
```

Include dependency graph for scheduler.hpp:



Namespaces

- namespace [QuantLib](#)

12.219.1 Detailed Description

Full path:

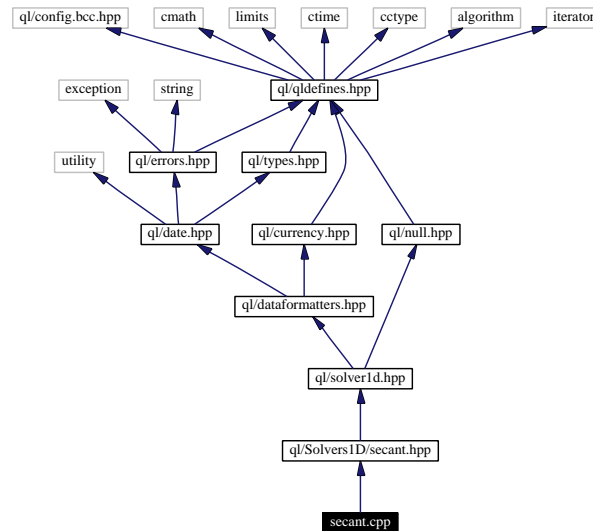
ql/scheduler.hpp

12.220 secant.cpp File Reference

secant 1-D solver.

```
#include <ql/Solvers1D/secant.hpp>
```

Include dependency graph for secant.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.220.1 Detailed Description

Full path:

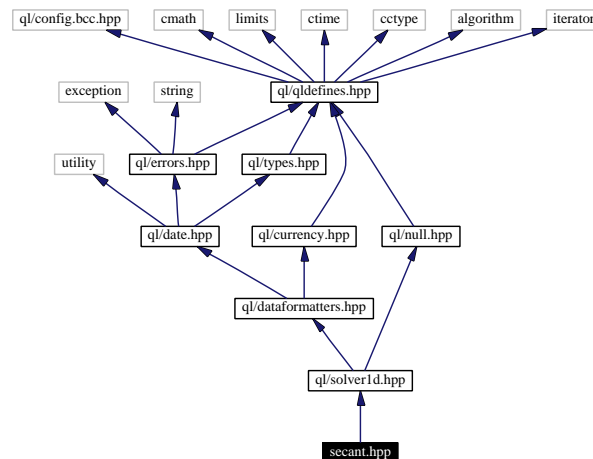
`ql/Solvers1D/secant.cpp`

12.221 secant.hpp File Reference

secant 1-D solver.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for secant.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

12.221.1 Detailed Description

Full path:

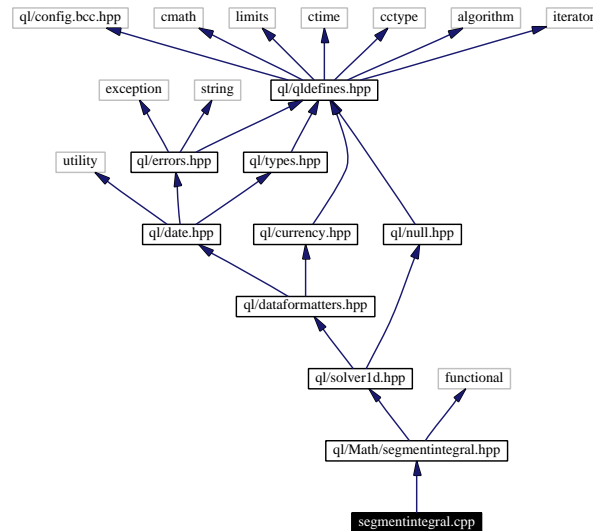
`ql/Solvers1D/secant.hpp`

12.222 segmentintegral.cpp File Reference

Integral of a function over a segment.

```
#include <ql/Math/segmentintegral.hpp>
```

Include dependency graph for segmentintegral.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.222.1 Detailed Description

Full path:

`ql/Math/segmentintegral.cpp`

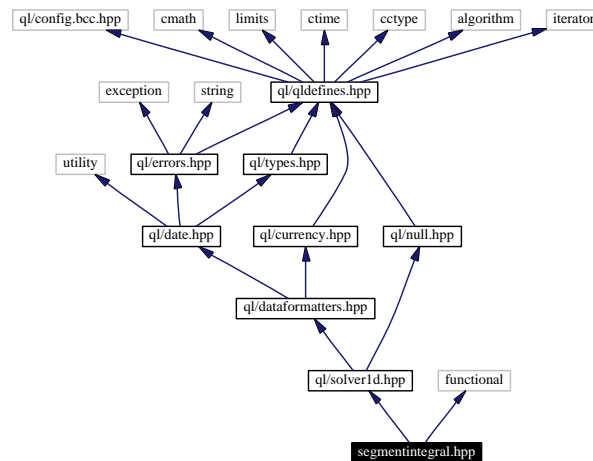
12.223 segmentintegral.hpp File Reference

Integral of a one-dimensional function.

```
#include <ql/solver1d.hpp>
```

```
#include <functional>
```

Include dependency graph for segmentintegral.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.223.1 Detailed Description

Full path:

`ql/Math/integral.hpp`

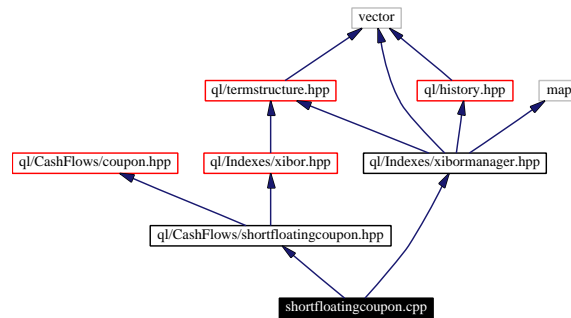
12.224 shortfloatingcoupon.cpp File Reference

Short coupon at par on a term structure.

```
#include <ql/CashFlows/shortfloatingcoupon.hpp>
```

```
#include <ql/Indexes/xibormanager.hpp>
```

Include dependency graph for shortfloatingcoupon.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.224.1 Detailed Description

Full path:

`ql/CashFlows/shortfloatingcoupon.cpp`

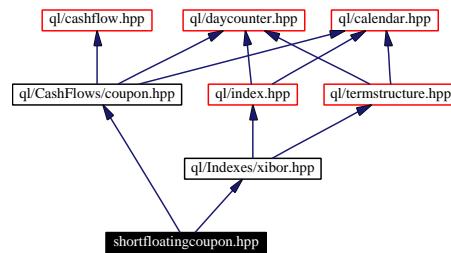
12.225 shortfloatingcoupon.hpp File Reference

Short (or long) coupon at par on a term structure.

```
#include <ql/CashFlows/coupon.hpp>
```

```
#include <ql/Indexes/xibor.hpp>
```

Include dependency graph for shortfloatingcoupon.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::CashFlows](#)

12.225.1 Detailed Description

Full path:

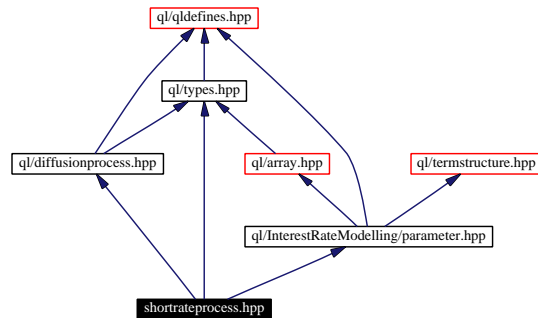
`ql/CashFlows/shortfloatingcoupon.hpp`

12.226 shortrateprocess.hpp File Reference

Short rate process.

```
#include <ql/diffusionprocess.hpp>
#include <ql/types.hpp>
#include <ql/InterestRateModelling/parameter.hpp>
```

Include dependency graph for shortrateprocess.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**

12.226.1 Detailed Description

Full path:

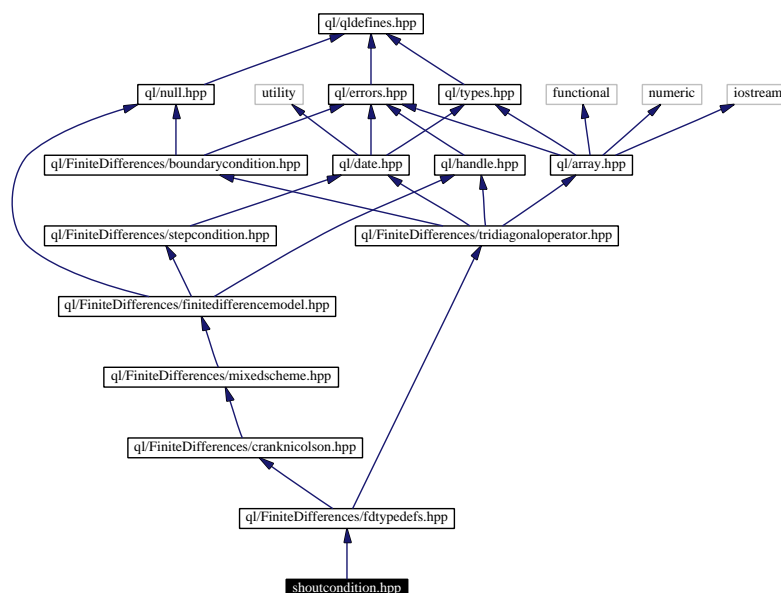
`ql/InterestRateModelling/shortrateprocess.hpp`

12.227 shoutcondition.hpp File Reference

shout option exercise condition.

```
#include <ql/FiniteDifferences/fdtypedefs.hpp>
```

Include dependency graph for shoutcondition.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.227.1 Detailed Description

Full path:

ql/FiniteDifferences/shoutcondition.hpp

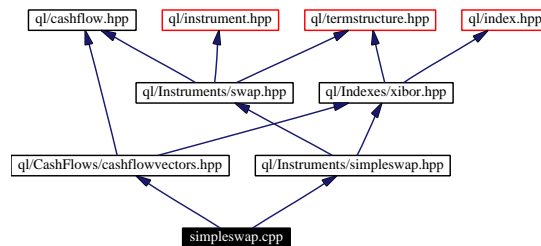
12.229 simpleswap.cpp File Reference

Simple fixed-rate vs Libor swap.

```
#include <ql/Instruments/simpleswap.hpp>
```

```
#include <ql/CashFlows/cashflowvectors.hpp>
```

Include dependency graph for simpleswap.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.229.1 Detailed Description

Full path:

`ql/Instruments/simpleswap.cpp`

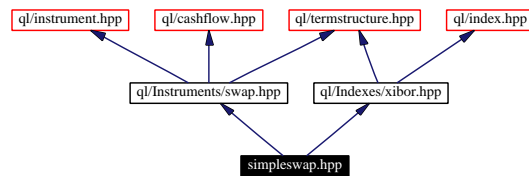
12.230 simpleswap.hpp File Reference

Simple fixed-rate vs Libor swap.

```
#include <ql/Instruments/swap.hpp>
```

```
#include <ql/Indexes/xibor.hpp>
```

Include dependency graph for simpleswap.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.230.1 Detailed Description

Full path:

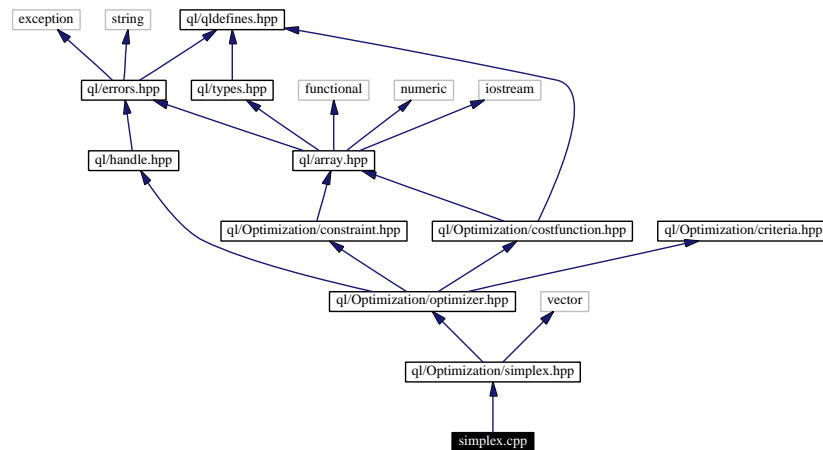
`ql/Instruments/simpleswap.hpp`

12.231 simplex.cpp File Reference

Simplex optimization method.

```
#include "ql/Optimization/simplex.hpp"
```

Include dependency graph for simplex.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Optimization**

12.231.1 Detailed Description

Full path:

`ql/Optimization/simplex.cpp`

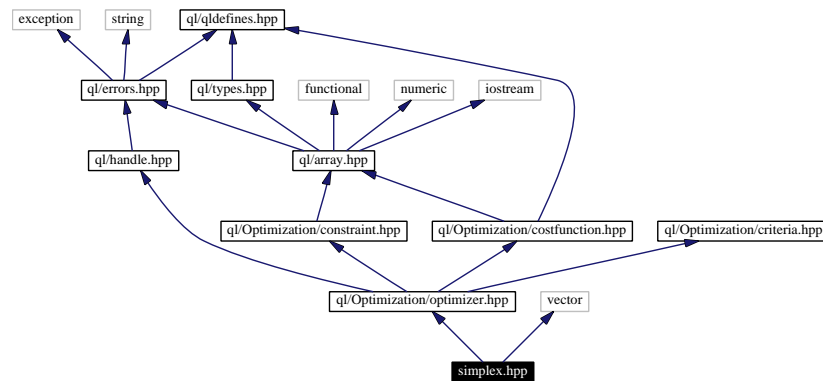
12.232 simplex.hpp File Reference

Simplex optimization method.

```
#include <ql/Optimization/optimizer.hpp>
```

```
#include <vector>
```

Include dependency graph for simplex.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.232.1 Detailed Description

Full path:

`ql/Optimization/simplex.hpp`

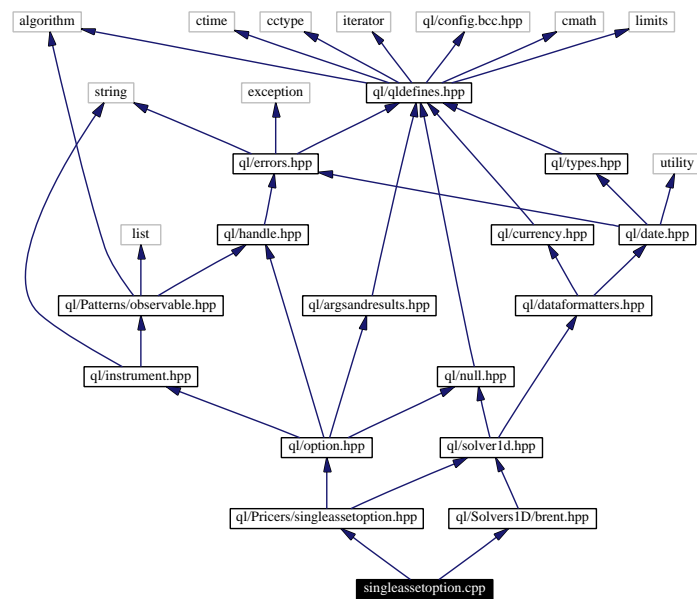
12.233 singleassetoption.cpp File Reference

common code for option evaluation.

```
#include <ql/Pricers/singleassetoption.hpp>
```

```
#include <ql/Solvers1D/brent.hpp>
```

Include dependency graph for singleassetoption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.233.1 Detailed Description

Full path:

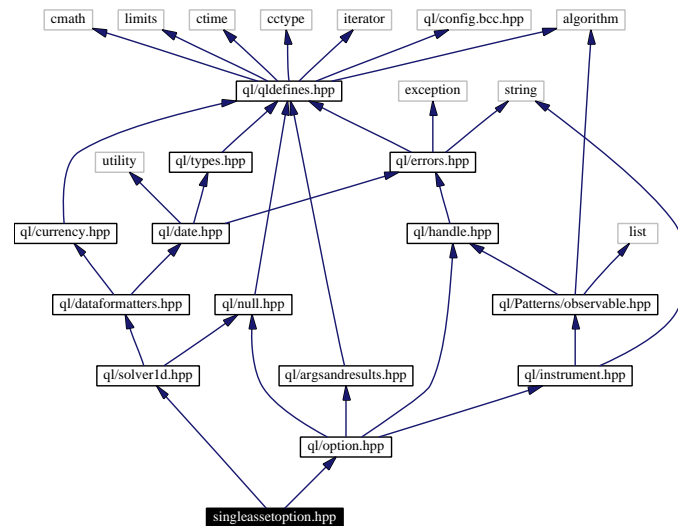
ql/Pricers/singleassetoption.cpp

12.234 singleassetoption.hpp File Reference

common code for option evaluation.

```
#include <ql/option.hpp>
#include <ql/solver1d.hpp>
```

Include dependency graph for singleassetoption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

Defines

- `#define QL_MIN_VOLATILITY 0.0001`
- `#define QL_MAX_VOLATILITY 4.0`

12.234.1 Detailed Description

Full path:

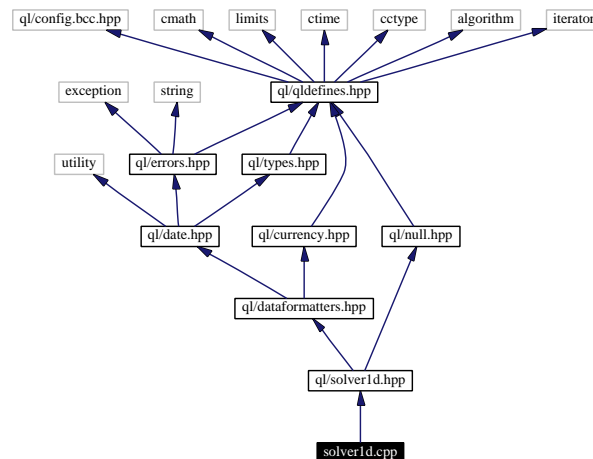
ql/Pricers/singleassetoption.hpp

12.235 solver1d.cpp File Reference

Abstract 1-D solver class.

```
#include <ql/solver1d.hpp>
```

Include dependency graph for solver1d.cpp:



Namespaces

- namespace [QuantLib](#)

12.235.1 Detailed Description

Full path:

`ql/solver1d.cpp`

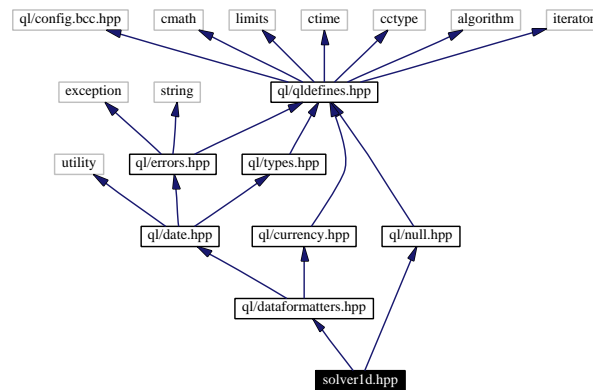
12.236 solver1d.hpp File Reference

Abstract 1-D solver class.

```
#include <ql/null.hpp>
```

```
#include <ql/dataformatters.hpp>
```

Include dependency graph for solver1d.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Solvers1D](#)

Defines

- #define `MAX_FUNCTION_EVALUATIONS` 100

12.236.1 Detailed Description

Full path:

ql/solver1d.hpp

12.238 statistics.hpp File Reference

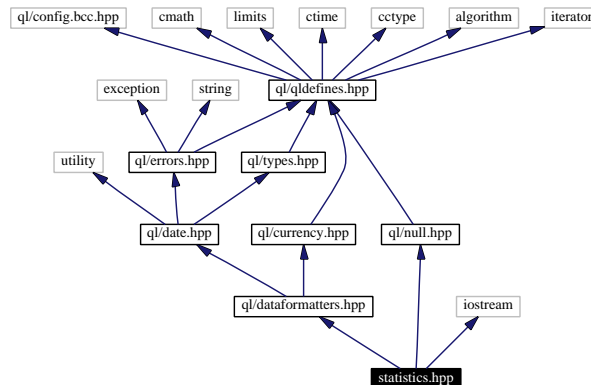
statistic tools.

```
#include <ql/null.hpp>
```

```
#include <ql/dataformatters.hpp>
```

```
#include <iostream>
```

Include dependency graph for statistics.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.238.1 Detailed Description

Full path:

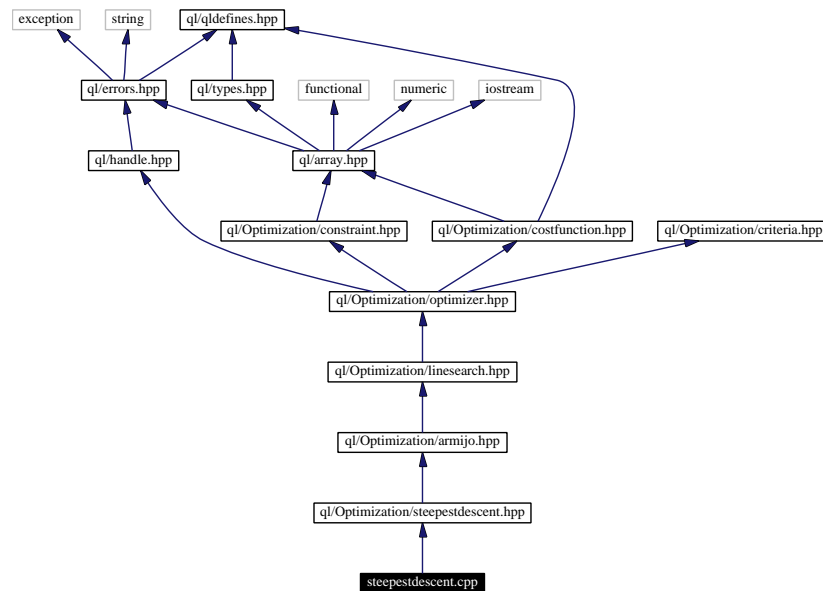
ql/Math/statistics.hpp

12.239 steepestdescent.cpp File Reference

Steepest descent optimization method.

```
#include "ql/Optimization/steepestdescent.hpp"
```

Include dependency graph for steepestdescent.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Optimization**

12.239.1 Detailed Description

Full path:

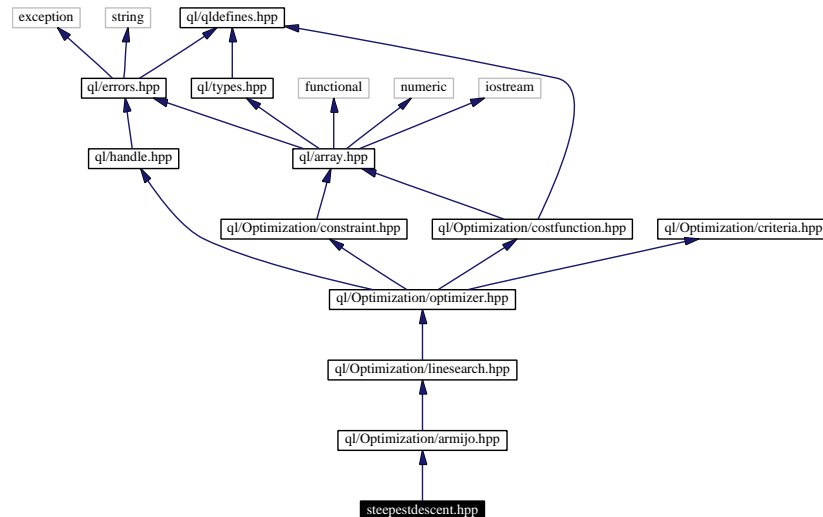
`ql/Optimization/steepestdescent.cpp`

12.240 steepestdescent.hpp File Reference

Steepest descent optimization method.

```
#include <ql/Optimization/armijo.hpp>
```

Include dependency graph for steepestdescent.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Optimization`

12.240.1 Detailed Description

Full path:

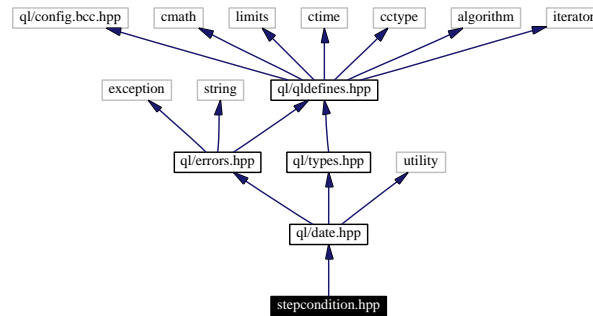
`ql/Optimization/steepestdescent.hpp`

12.241 stepcondition.hpp File Reference

conditions to be applied at every time step.

```
#include <ql/date.hpp>
```

Include dependency graph for stepcondition.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.241.1 Detailed Description

Full path:

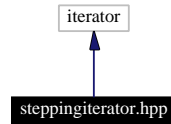
`ql/FiniteDifferences/stepcondition.hpp`

12.242 steppingiterator.hpp File Reference

Iterator advancing in constant steps.

```
#include <iterator>
```

Include dependency graph for steppingiterator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Utilities](#)

12.242.1 Detailed Description

Full path:

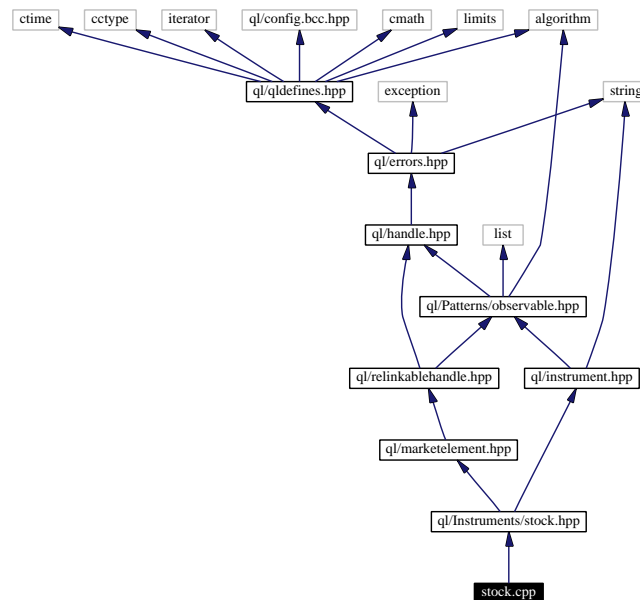
ql/Utilities/steppingiterator.hpp

12.243 stock.cpp File Reference

concrete stock class.

```
#include <ql/Instruments/stock.hpp>
```

Include dependency graph for stock.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.243.1 Detailed Description

Full path:

ql/Instruments/stock.cpp

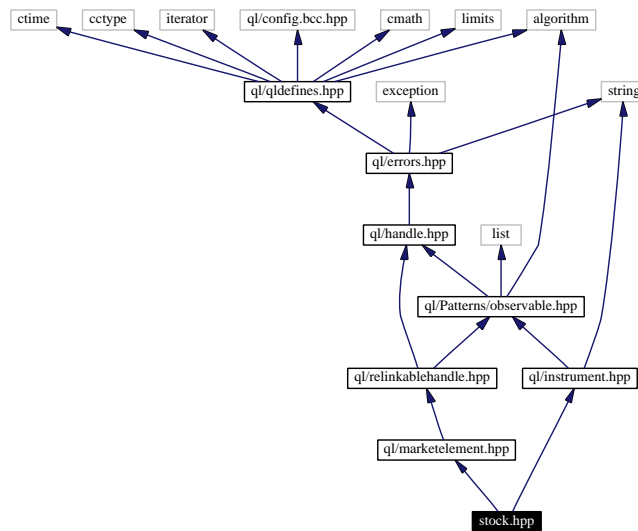
12.244 stock.hpp File Reference

concrete stock class.

```
#include <ql/instrument.hpp>
```

```
#include <ql/marketelement.hpp>
```

Include dependency graph for stock.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.244.1 Detailed Description

Full path:

`ql/Instruments/stock.hpp`

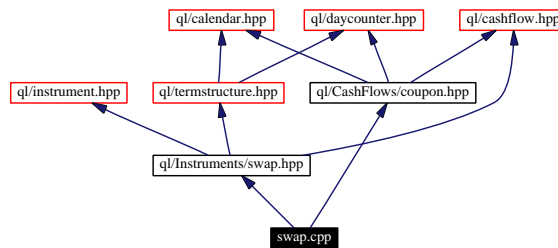
12.245 swap.cpp File Reference

Interest rate swap.

```
#include <ql/Instruments/swap.hpp>
```

```
#include <ql/CashFlows/coupon.hpp>
```

Include dependency graph for swap.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.245.1 Detailed Description

Full path:

ql/Instruments/swap.cpp

12.246 swap.hpp File Reference

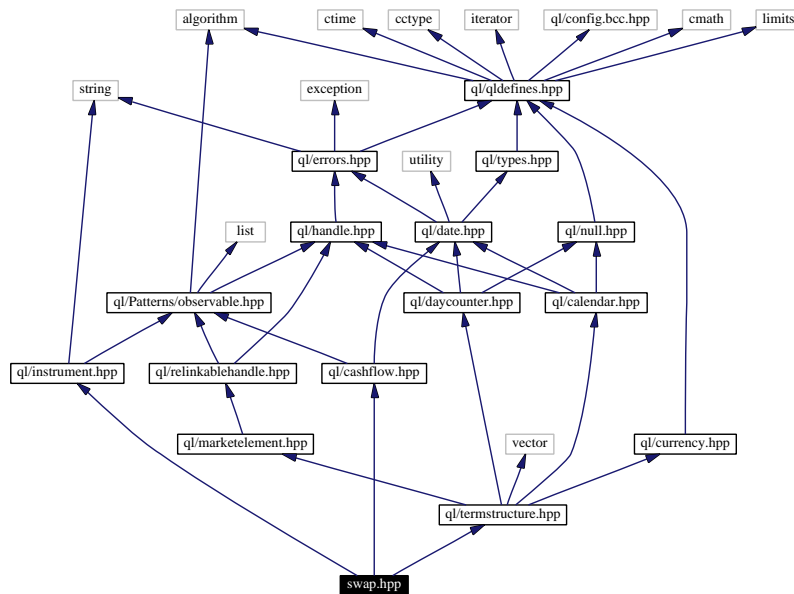
Interest rate swap.

```
#include <ql/instrument.hpp>
```

```
#include <ql/cashflow.hpp>
```

```
#include <ql/termstructure.hpp>
```

Include dependency graph for swap.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.246.1 Detailed Description

Full path:

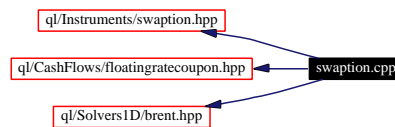
ql/Instruments/swap.hpp

12.247 swaption.cpp File Reference

Swaption.

```
#include <ql/Instruments/swaption.hpp>
#include <ql/CashFlows/floatingratecoupon.hpp>
#include <ql/Solvers1D/brent.hpp>
```

Include dependency graph for swaption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)

12.247.1 Detailed Description

Full path:

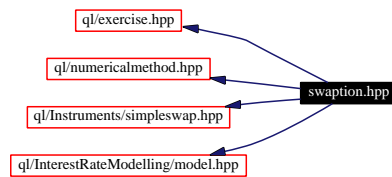
ql/Instruments/swaption.cpp

12.248 swaption.hpp File Reference

Swaption class.

```
#include <ql/exercise.hpp>
#include <ql/numericalmethod.hpp>
#include <ql/Instruments/simpleswap.hpp>
#include <ql/InterestRateModelling/model.hpp>
```

Include dependency graph for swaption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Instruments](#)
- namespace [QuantLib::Pricers](#)

12.248.1 Detailed Description

Full path:

ql/Instruments/swaption.hpp

12.249 swaptionhelper.cpp File Reference

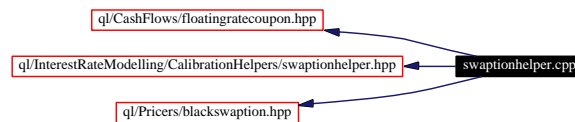
Swaption calibration helper.

```
#include "ql/CashFlows/floatingratecoupon.hpp"
```

```
#include "ql/InterestRateModelling/CalibrationHelpers/swaptionhelper.hpp"
```

```
#include "ql/Pricers/blackswaption.hpp"
```

Include dependency graph for swaptionhelper.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**
- namespace **QuantLib::InterestRateModelling::CalibrationHelpers**

12.249.1 Detailed Description

Full path:

ql/InterestRateModelling/CalibrationHelpers/swaptionhelper.cpp

12.250 swaptionhelper.hpp File Reference

Swaption calibration helper.

```
#include <ql/InterestRateModelling/calibrationhelper.hpp>
```

```
#include <ql/Instruments/swaption.hpp>
```

Include dependency graph for swaptionhelper.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::InterestRateModelling**
- namespace **QuantLib::InterestRateModelling::CalibrationHelpers**

12.250.1 Detailed Description

Full path:

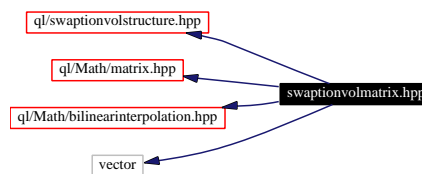
`ql/InterestRateModelling/CalibrationHelpers/swaptionhelper.hpp`

12.251 swaptionvolmatrix.hpp File Reference

Swaption at-the-money volatility matrix.

```
#include "ql/swaptionvolstructure.hpp"
#include "ql/Math/matrix.hpp"
#include "ql/Math/bilinearinterpolation.hpp"
#include <vector>
```

Include dependency graph for swaptionvolmatrix.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Volatilities**

12.251.1 Detailed Description

Full path:

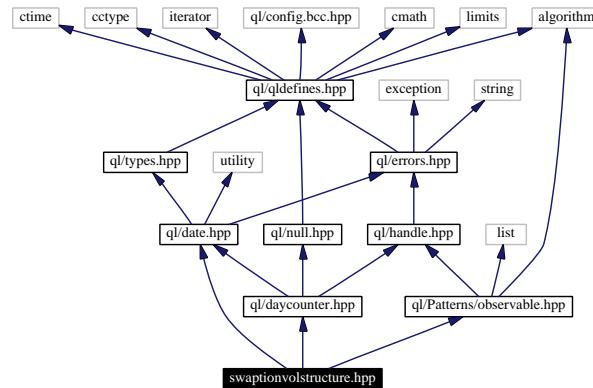
`ql/Volatilities/swaptionvolmatrix.hpp`

12.252 swaptionvolstructure.hpp File Reference

Swaption volatility structure.

```
#include <ql/date.hpp>
#include <ql/daycounter.hpp>
#include <ql/Patterns/observable.hpp>
```

Include dependency graph for swaptionvolstructure.hpp:



Namespaces

- namespace [QuantLib](#)

12.252.1 Detailed Description

Full path:

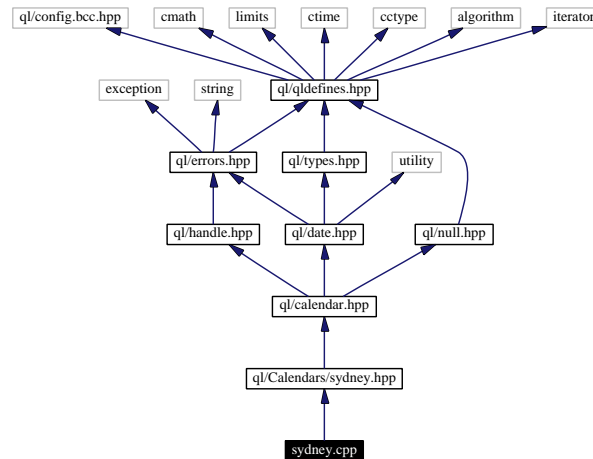
ql/swaptionvolstructure.hpp

12.253 sydney.cpp File Reference

Sydney calendar.

```
#include <ql/Calendars/sydney.hpp>
```

Include dependency graph for sydney.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.253.1 Detailed Description

Full path:

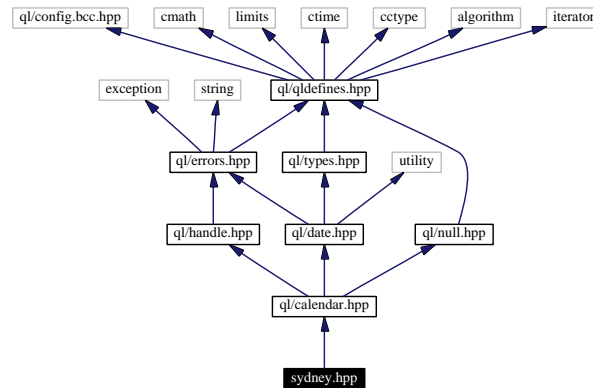
ql/Calendars/sydney.cpp

12.254 sydney.hpp File Reference

Sydney calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for sydney.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.254.1 Detailed Description

Full path:

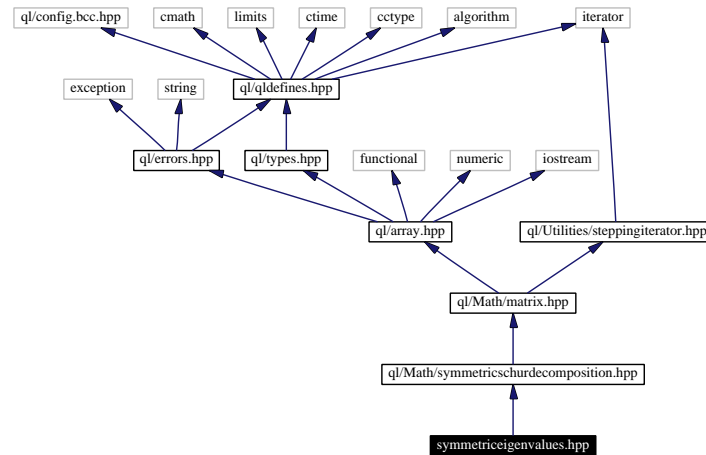
`ql/Calendars/sydney.hpp`

12.255 symmetriceigenvalues.hpp File Reference

Eigenvalues / eigenvectors of a real symmetric matrix.

```
#include <ql/Math/symmetricschurdecomposition.hpp>
```

Include dependency graph for symmetriceigenvalues.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.255.1 Detailed Description

Full path:

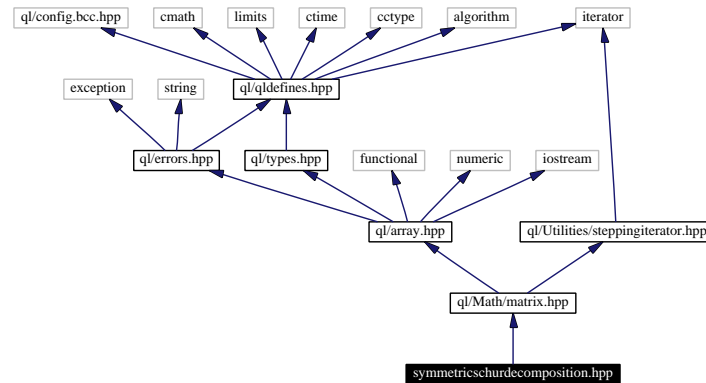
`ql/Math/symmetriceigenvalues.hpp`

12.257 symmetricsschurdecomposition.hpp File Reference

Eigenvalues / eigenvectors of a real symmetric matrix.

```
#include <ql/Math/matrix.hpp>
```

Include dependency graph for symmetricsschurdecomposition.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Math](#)

12.257.1 Detailed Description

Full path:

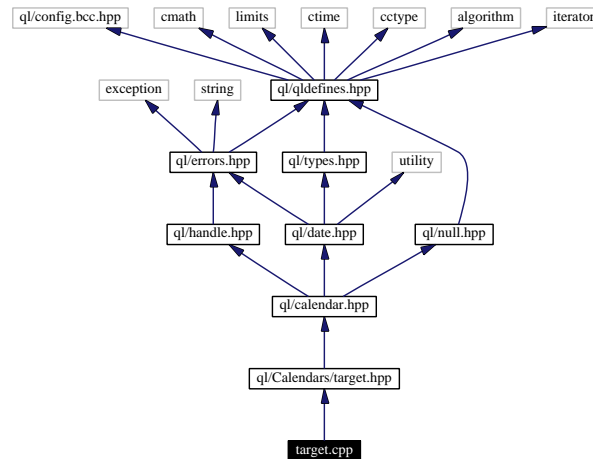
`ql/Math/symmetricsschurdecomposition.hpp`

12.258 target.cpp File Reference

TARGET calendar.

```
#include <ql/Calendars/target.hpp>
```

Include dependency graph for target.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.258.1 Detailed Description

Full path:

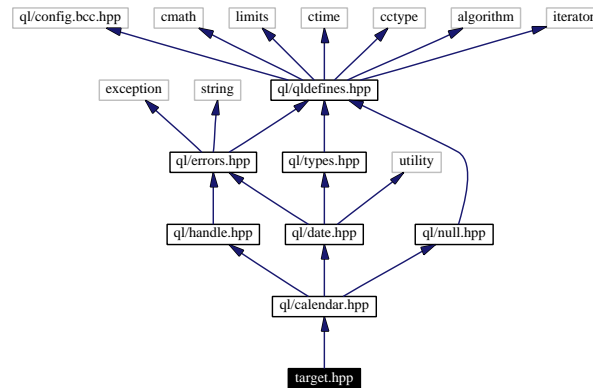
ql/Calendars/target.cpp

12.259 target.hpp File Reference

TARGET calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for target.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.259.1 Detailed Description

Full path:

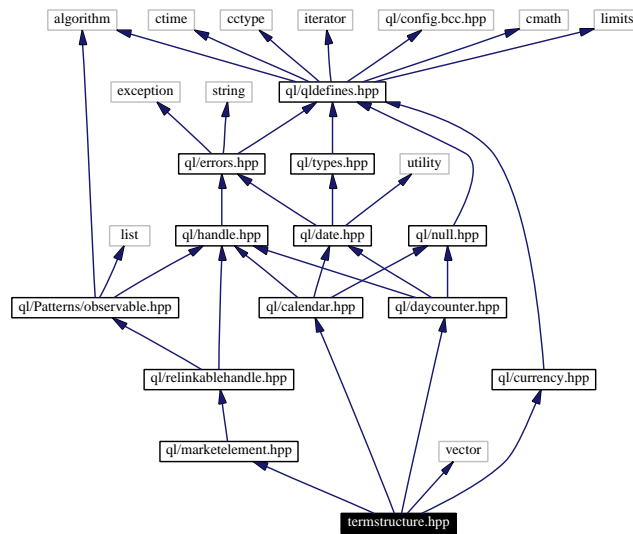
ql/Calendars/target.hpp

12.260 termstructure.hpp File Reference

Term structure.

```
#include <ql/calendar.hpp>
#include <ql/currency.hpp>
#include <ql/daycounter.hpp>
#include <ql/marketelement.hpp>
#include <vector>
```

Include dependency graph for termstructure.hpp:



Namespaces

- namespace [QuantLib::TermStructures](#)
- namespace [QuantLib](#)

12.260.1 Detailed Description

Full path:

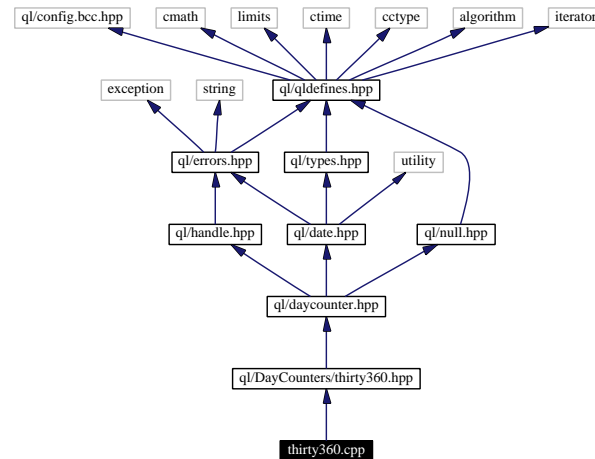
ql/termstructure.hpp

12.261 thirty360.cpp File Reference

30/360 day counters.

```
#include <ql/DayCounters/thirty360.hpp>
```

Include dependency graph for thirty360.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::DayCounters](#)

12.261.1 Detailed Description

Full path:

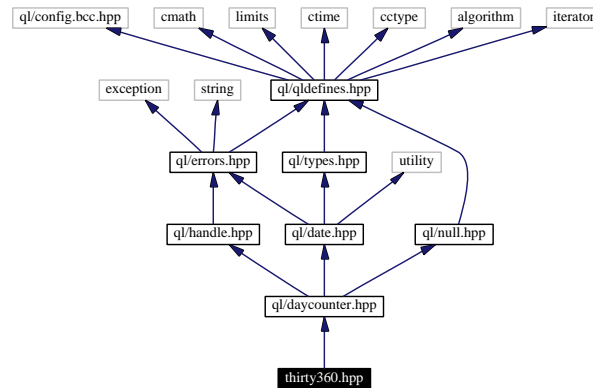
`ql/DayCounters/thirty360.cpp`

12.262 thirty360.hpp File Reference

30/360 day counters.

```
#include <ql/daycounter.hpp>
```

Include dependency graph for thirty360.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::DayCounters](#)

12.262.1 Detailed Description

Full path:

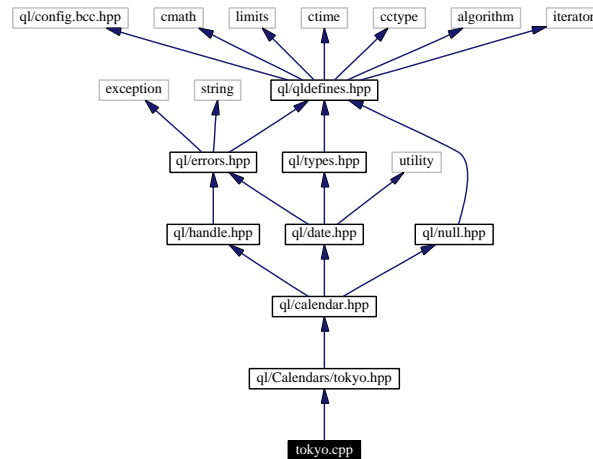
`ql/DayCounters/thirty360.hpp`

12.263 tokyo.cpp File Reference

Tokyo calendar.

```
#include <ql/Calendars/tokyo.hpp>
```

Include dependency graph for tokyo.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.263.1 Detailed Description

Full path:

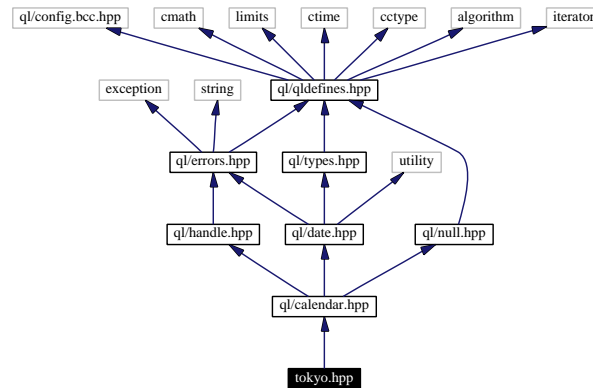
ql/Calendars/tokyo.cpp

12.264 tokyo.hpp File Reference

Tokyo calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for tokyo.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.264.1 Detailed Description

Full path:

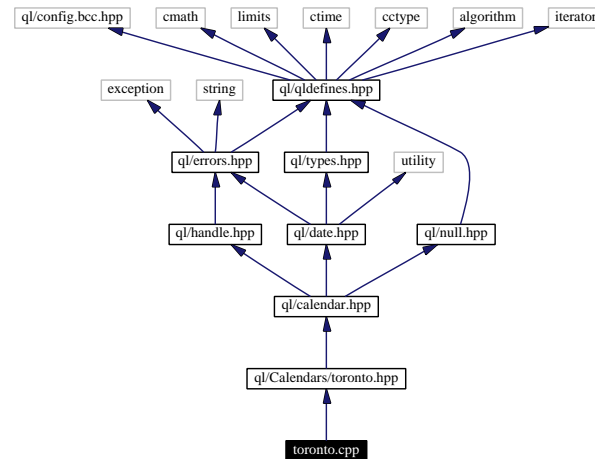
`ql/Calendars/tokyo.hpp`

12.265 toronto.cpp File Reference

Toronto calendar.

```
#include <ql/Calendars/toronto.hpp>
```

Include dependency graph for toronto.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.265.1 Detailed Description

Full path:

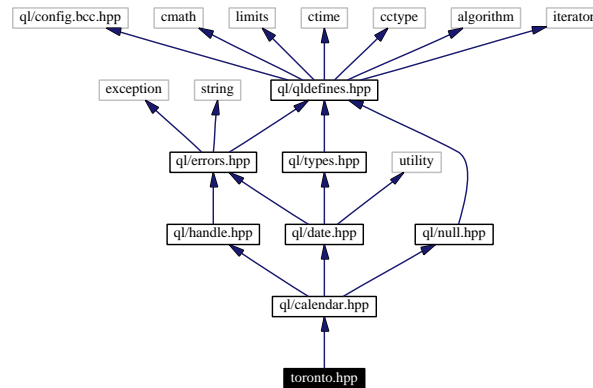
Calendars/toronto.cpp

12.266 toronto.hpp File Reference

Toronto calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for toronto.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.266.1 Detailed Description

Full path:

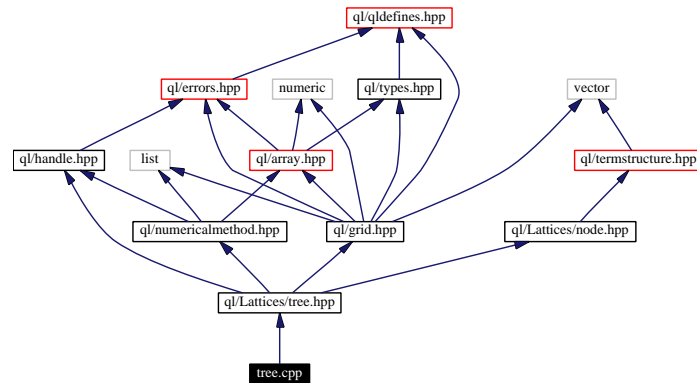
`ql/Calendars/toronto.hpp`

12.267 tree.cpp File Reference

Tree class.

```
#include "ql/Lattices/tree.hpp"
```

Include dependency graph for tree.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Lattices**

12.267.1 Detailed Description

Full path:

ql/Lattices/tree.cpp

12.268 tree.hpp File Reference

Tree class.

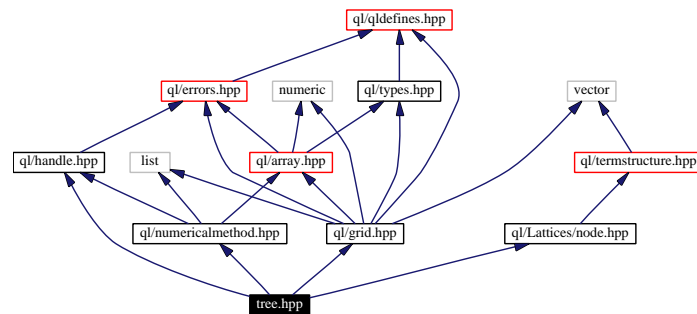
```
#include <ql/numericalmethod.hpp>
```

```
#include <ql/handle.hpp>
```

```
#include <ql/grid.hpp>
```

```
#include <ql/Lattices/node.hpp>
```

Include dependency graph for tree.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Lattices](#)

12.268.1 Detailed Description

Full path:

ql/Lattices/tree.hpp

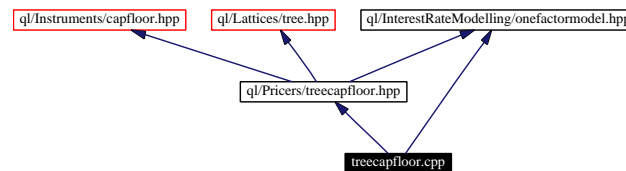
12.269 treecapfloor.cpp File Reference

Cap/Floor calculated using a tree.

```
#include "ql/Pricers/treecapfloor.hpp"
```

```
#include "ql/InterestRateModelling/onefactormodel.hpp"
```

Include dependency graph for treecapfloor.cpp:



Namespaces

- namespace `QuantLib`
- namespace `QuantLib::Pricers`

12.269.1 Detailed Description

Full path:

`ql/Pricers/treecapfloor.cpp`

12.270 treecapfloor.hpp File Reference

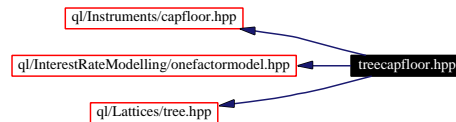
Cap/Floor calculated using a tree.

```
#include <ql/Instruments/capfloor.hpp>
```

```
#include <ql/InterestRateModelling/onefactormodel.hpp>
```

```
#include <ql/Lattices/tree.hpp>
```

Include dependency graph for treecapfloor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.270.1 Detailed Description

Full path:

`ql/Pricers/treecapfloor.hpp`

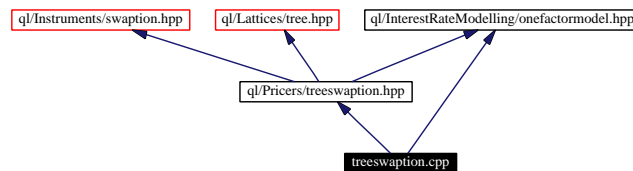
12.271 treeswaption.cpp File Reference

European swaption calculated using finite differences.

```
#include "ql/Pricers/treeswaption.hpp"
```

```
#include "ql/InterestRateModelling/onefactormodel.hpp"
```

Include dependency graph for treeswaption.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.271.1 Detailed Description

Full path:

ql/Pricers/treeswaption.cpp

12.272 treeswaption.hpp File Reference

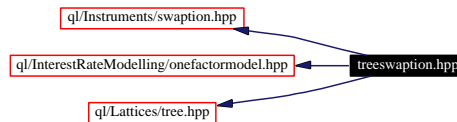
Swaption calculated using a tree.

```
#include <ql/Instruments/swaption.hpp>
```

```
#include <ql/InterestRateModelling/onefactormodel.hpp>
```

```
#include <ql/Lattices/tree.hpp>
```

Include dependency graph for treeswaption.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Pricers](#)

12.272.1 Detailed Description

Full path:

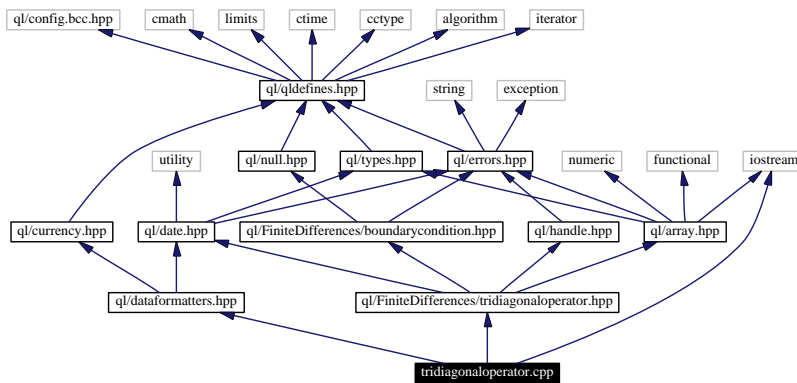
`ql/Pricers/treeswaption.hpp`

12.273 tridiagonaloperator.cpp File Reference

tridiagonal operator.

```
#include <ql/FiniteDifferences/tridiagonaloperator.hpp>
#include <ql/dataformatters.hpp>
#include <iostream>
```

Include dependency graph for tridiagonaloperator.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.273.1 Detailed Description

Full path:

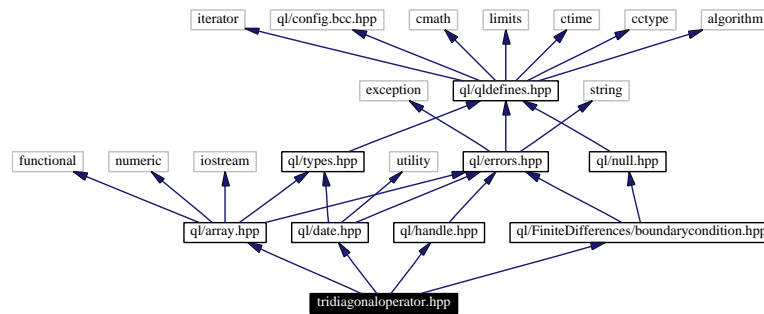
ql/FiniteDifferences/tridiagonaloperator.cpp

12.274 tridiagonaloperator.hpp File Reference

tridiagonal operator.

```
#include <ql/FiniteDifferences/boundarycondition.hpp>
#include <ql/array.hpp>
#include <ql/date.hpp>
#include <ql/handle.hpp>
```

Include dependency graph for tridiagonaloperator.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.274.1 Detailed Description

Full path:

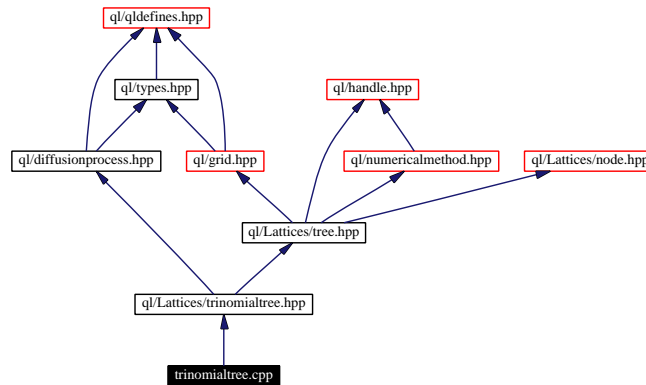
ql/FiniteDifferences/tridiagonaloperator.hpp

12.275 trinomialtree.cpp File Reference

Trinomial tree class.

```
#include "ql/Lattices/trinomialtree.hpp"
```

Include dependency graph for trinomialtree.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace **QuantLib::Lattices**

12.275.1 Detailed Description

Full path:

`ql/Lattices/trinomialtree.cpp`

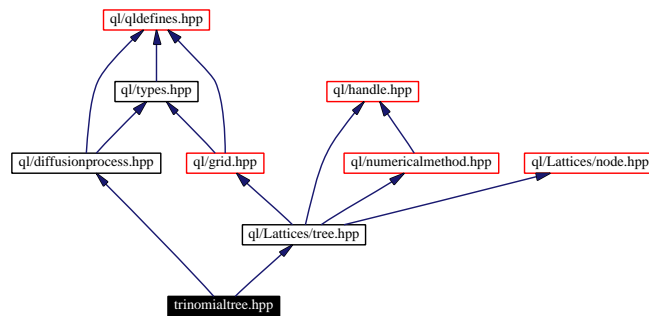
12.276 trinomialtree.hpp File Reference

Trinomial tree class.

```
#include <ql/diffusionprocess.hpp>
```

```
#include <ql/Lattices/tree.hpp>
```

Include dependency graph for trinomialtree.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::Lattices`

12.276.1 Detailed Description

Full path:

`ql/Lattices/trinomialtree.hpp`

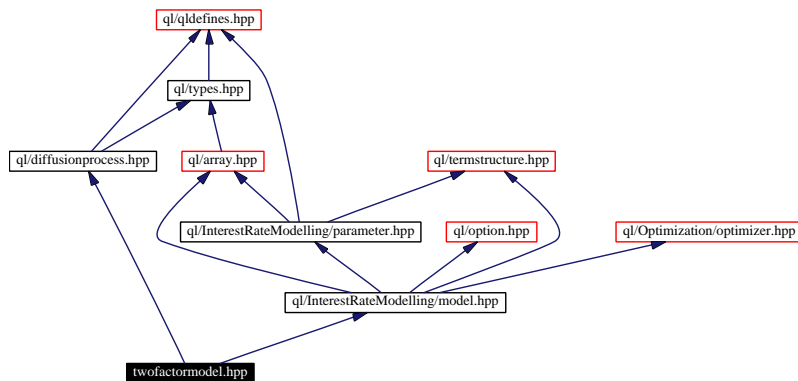
12.277 twofactormodel.hpp File Reference

Abstract two-factor interest rate model class.

```
#include <ql/diffusionprocess.hpp>
```

```
#include <ql/InterestRateModelling/model.hpp>
```

Include dependency graph for twofactormodel.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace `QuantLib::InterestRateModelling`

12.277.1 Detailed Description

Full path:

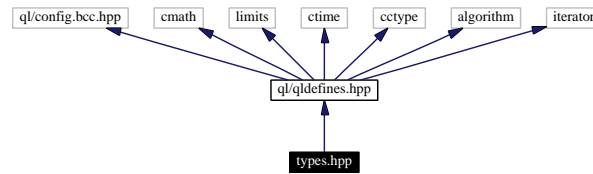
`ql/InterestRateModelling/twofactormodel.hpp`

12.278 types.hpp File Reference

Custom types.

```
#include <ql/qldefines.hpp>
```

Include dependency graph for types.hpp:



Namespaces

- namespace [QuantLib](#)

12.278.1 Detailed Description

Full path:

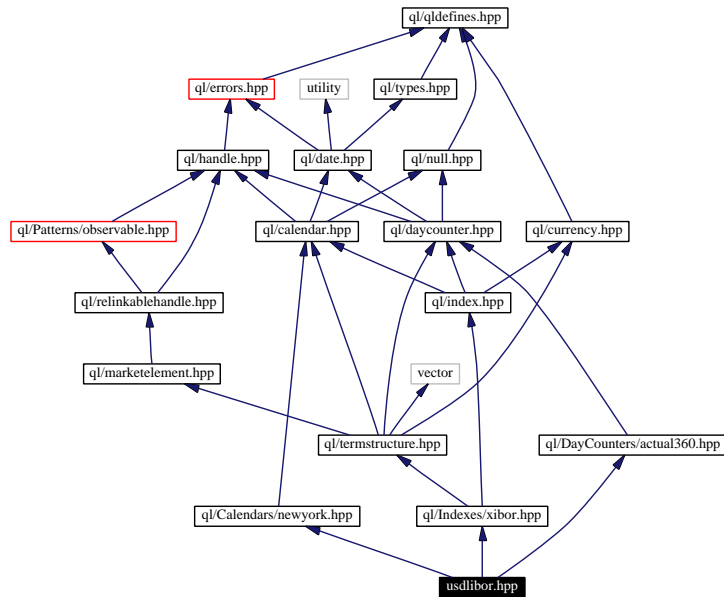
`ql/types.hpp`

12.279 usdlibor.hpp File Reference

USD Libor index.

```
#include <ql/Indexes/xibor.hpp>
#include <ql/Calendars/newyork.hpp>
#include <ql/DayCounters/actual360.hpp>
```

Include dependency graph for usdlibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.279.1 Detailed Description

Full path:

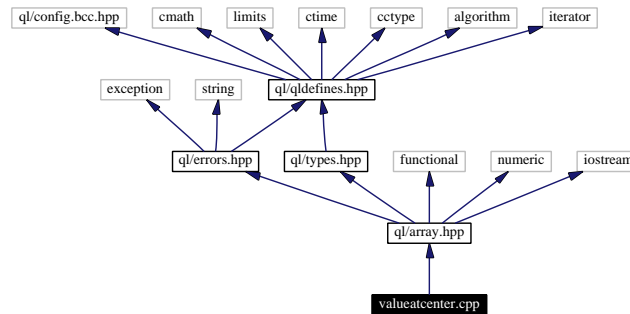
ql/Indexes/usdlibor.hpp

12.280 valueatcenter.cpp File Reference

compute value, first, and second derivatives at grid center.

```
#include <ql/array.hpp>
```

Include dependency graph for valueatcenter.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.280.1 Detailed Description

Full path:

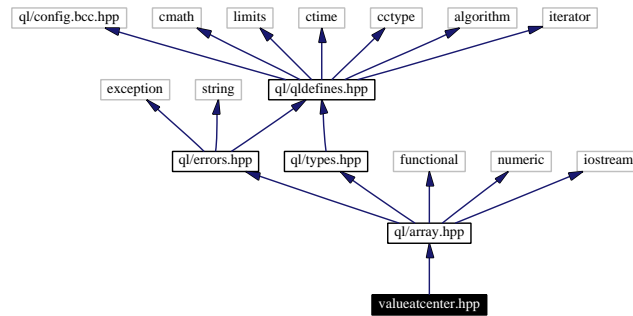
ql/FiniteDifferences/valueatcenter.cpp

12.281 valueatcenter.hpp File Reference

compute value, first, and second derivatives at grid center.

```
#include <ql/array.hpp>
```

Include dependency graph for valueatcenter.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::FiniteDifferences](#)

12.281.1 Detailed Description

Full path:

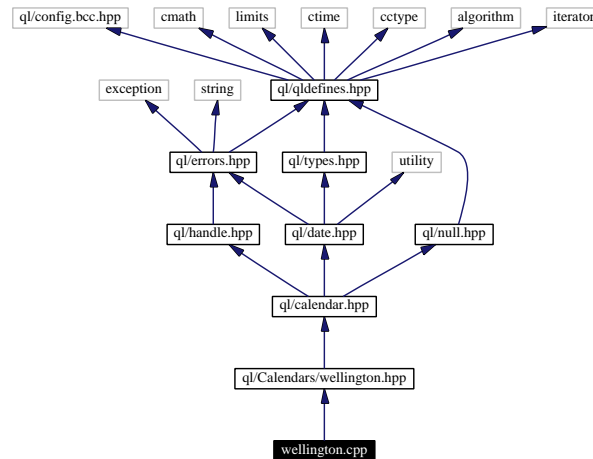
`ql/FiniteDifferences/valueatcenter.hpp`

12.282 wellington.cpp File Reference

Wellington calendar.

```
#include <ql/Calendars/wellington.hpp>
```

Include dependency graph for wellington.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.282.1 Detailed Description

Full path:

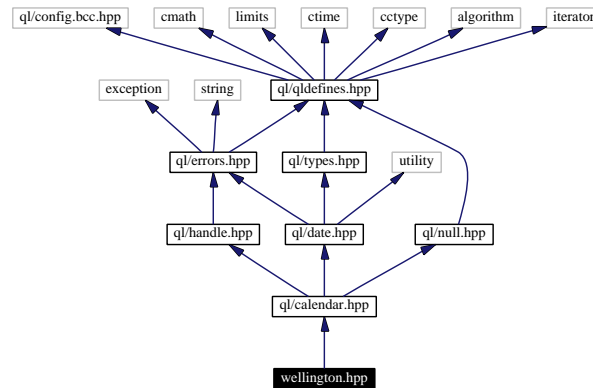
`ql/Calendars/wellington.cpp`

12.283 wellington.hpp File Reference

Wellington calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for wellington.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.283.1 Detailed Description

Full path:

`ql/Calendars/wellington.hpp`

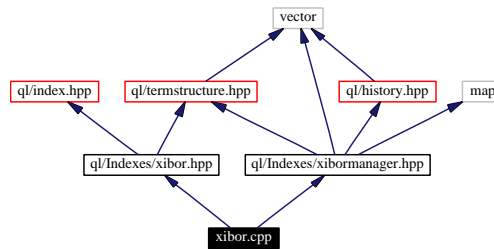
12.284 xibor.cpp File Reference

purely virtual base class for libor indexes.

```
#include <ql/Indexes/xibor.hpp>
```

```
#include <ql/Indexes/xibormanager.hpp>
```

Include dependency graph for xibor.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.284.1 Detailed Description

Full path:

ql/Indexes/xibor.cpp

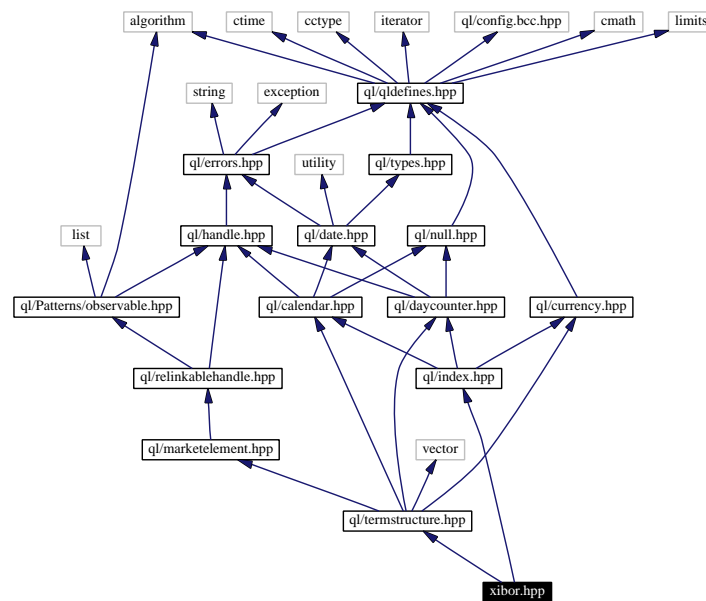
12.285 xibor.hpp File Reference

purely virtual base class for libor indexes.

```
#include <ql/index.hpp>
```

```
#include <ql/termstructure.hpp>
```

Include dependency graph for xibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.285.1 Detailed Description

Full path:

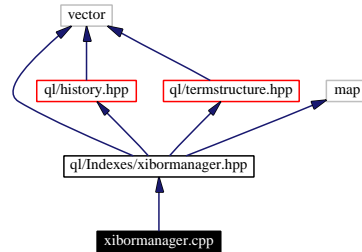
ql/Indexes/xibor.hpp

12.286 xibormanager.cpp File Reference

global repository for Xibor histories.

```
#include <ql/Indexes/xibormanager.hpp>
```

Include dependency graph for xibormanager.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.286.1 Detailed Description

Full path:

`ql/Indexes/xibormanager.cpp`

12.287 xibormanager.hpp File Reference

global repository for Xibor histories.

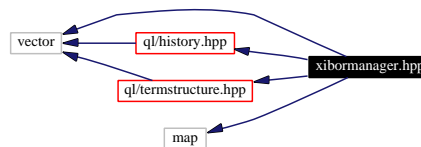
```
#include <ql/history.hpp>
```

```
#include <ql/termstructure.hpp>
```

```
#include <map>
```

```
#include <vector>
```

Include dependency graph for xibormanager.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.287.1 Detailed Description

Full path:

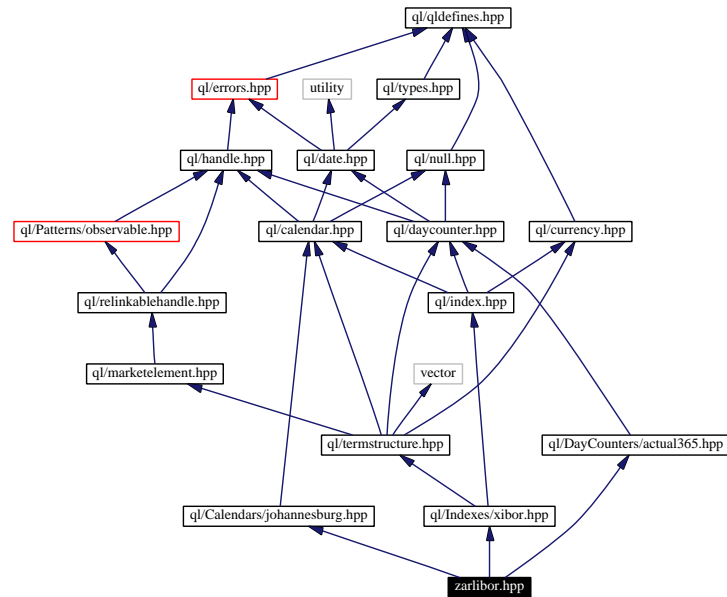
`ql/Indexes/xibormanager.hpp`

12.288 zarlibor.hpp File Reference

ZAR Libor index (also known as JIBAR, check settlement days).

```
#include <ql/Indexes/xibor.hpp>
#include <ql/Calendars/johannesburg.hpp>
#include <ql/DayCounters/actual365.hpp>
```

Include dependency graph for zarlibor.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Indexes](#)

12.288.1 Detailed Description

Full path:

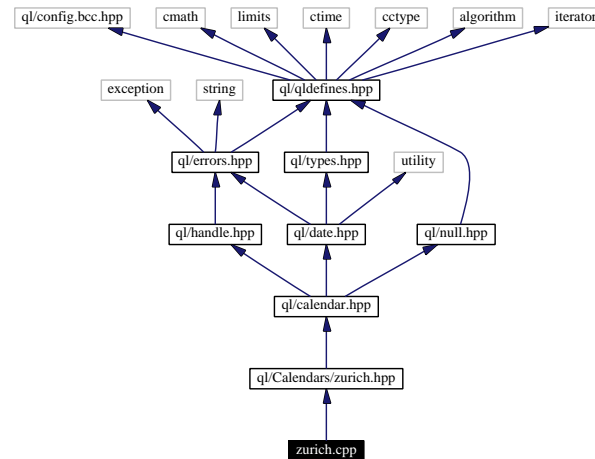
ql/Indexes/zarlibor.hpp

12.289 zurich.cpp File Reference

Zurich calendar.

```
#include <ql/Calendars/zurich.hpp>
```

Include dependency graph for zurich.cpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.289.1 Detailed Description

Full path:

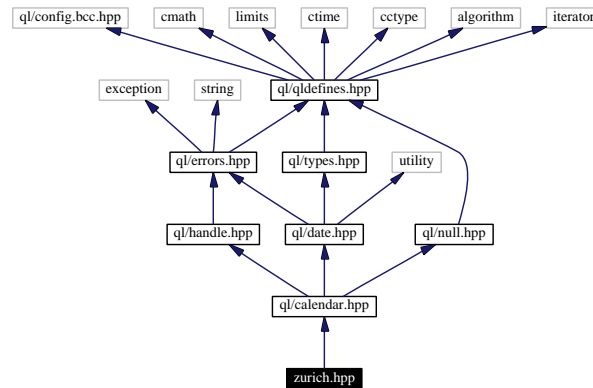
`ql/Calendars/zurich.cpp`

12.290 zurich.hpp File Reference

Zurich calendar.

```
#include <ql/calendar.hpp>
```

Include dependency graph for zurich.hpp:



Namespaces

- namespace [QuantLib](#)
- namespace [QuantLib::Calendars](#)

12.290.1 Detailed Description

Full path:

`ql/Calendars/zurich.hpp`

Chapter 13

QuantLib Example Documentation

13.1 DiscreteHedging.cpp

This is an example of using the QuantLib Monte Carlo framework.

```
/* This example computes profit and loss of a discrete interval hedging
   strategy and compares with the results of Derman & Kamal's (Goldman Sachs
   Equity Derivatives Research) Research Note: "When You Cannot Hedge
   Continuously: The Corrections to Black-Scholes"
   (http://www.gs.com/qs/doc/when\_you\_cannot\_hedge.pdf)

   Suppose an option hedger sells an European option and receives the
   Black-Scholes value as the options premium.
   Then he follows a Black-Scholes hedging strategy, rehedging at discrete,
   evenly spaced time intervals as the underlying stock changes. At
   expiration, the hedger delivers the option payoff to the option holder,
   and unwinds the hedge. We are interested in understanding the final
   profit or loss of this strategy.

   If the hedger had followed the exact Black-Scholes replication strategy,
   re-hedging continuously as the underlying stock evolved towards its final
   value at expiration, then, no matter what path the stock took, the final
   P&L would be exactly zero. When the replication strategy deviates from
   the exact Black-Scholes method, the final P&L may deviate from zero. This
   deviation is called the replication error. When the hedger rebalances at
   discrete rather than continuous intervals, the hedge is imperfect and the
   replication is inexact. The more often hedging occurs, the smaller the
   replication error.

   We examine the range of possibilities, computing the replication error.
*/

// the only header you need to use QuantLib
#include <ql/quantlib.hpp>

// introducing the players ....

// Rate and Time are just double, but having their own types allows for
// a stronger check at compile time
using QuantLib::Rate;
using QuantLib::Time;
```

```

// Option is a helper class that holds the enumeration {Call, Put, Straddle}
using QuantLib::Option;

// Handle is the QuantLib way to have reference-counted objects
using QuantLib::Handle;

// helper function for option payoff: MAX((strike-underlying),0), etc.
using QuantLib::Pricers::ExercisePayoff;

// the classic Black Scholes analytic solution for European Option
using QuantLib::Pricers::EuropeanOption;

// class for statistical analysis
using QuantLib::Math::Statistics;

// single Path of a random variable
// It contains the list of variations
using QuantLib::MonteCarlo::Path;

// the pricer computes final portfolio's value for each random variable path
using QuantLib::MonteCarlo::PathPricer;

// the path generator
using QuantLib::MonteCarlo::GaussianPathGenerator;

// the Montecarlo pricing model for option on a single asset
using QuantLib::MonteCarlo::OneFactorMonteCarloOption;

// to format the output of doubles
using QuantLib::DoubleFormatter;

/* The ReplicationError class carries out Monte Carlo simulations to evaluate
   the outcome (the replication error) of the discrete hedging strategy over
   different, randomly generated scenarios of future stock price evolution.
*/
class ReplicationError
{
public:
    ReplicationError(Option::Type type,
                    Time maturity,
                    double strike,
                    double s0,
                    double sigma,
                    Rate r)
    : type_(type), maturity_(maturity), strike_(strike), s0_(s0),
      sigma_(sigma), r_(r) {

        // value of the option
        EuropeanOption option = EuropeanOption(type_, s0_, strike_, 0.0, r_,
                                                maturity_, sigma_);
        std::cout << "Option value: " << option.value() << std::endl;

        // store option's vega, since Derman and Kamal's formula needs it
        vega_ = option.vega();

        std::cout << std::endl;
        std::cout <<
            "          |          | P&L \t| P&L          | Derman&Kamal | P&L"
            "          \t| P&L" << std::endl;

        std::cout <<
            "samples | trades | Mean \t| Std Dev | Formula          |"
            " skewness \t| kurt." << std::endl;

        std::cout << "-----"

```

```

        "-----" << std::endl;
    }

    // the actual replication error computation
    void compute(int nTimeSteps, int nSamples);
private:
    Option::Type type_;
    Time maturity_;
    double strike_, s0_;
    double sigma_;
    Rate r_;
    double vega_;
};

// The key for the MonteCarlo simulation is to have a PathPricer that
// implements a value(const Path& path) method.
// This method prices the portfolio for each Path of the random variable
class ReplicationPathPricer : public PathPricer<Path>
{
public:
    // real constructor
    ReplicationPathPricer(Option::Type type,
                          double underlying,
                          double strike,
                          Rate r,
                          Time maturity,
                          double sigma)
    : PathPricer<Path>(1.0, false), type_(type), underlying_(underlying),
      strike_(strike), r_(r), maturity_(maturity), sigma_(sigma) {
        QL_REQUIRE(strike_ > 0.0,
                    "ReplicationPathPricer: strike must be positive");
        QL_REQUIRE(underlying_ > 0.0,
                    "ReplicationPathPricer: underlying must be positive");
        QL_REQUIRE(r_ >= 0.0,
                    "ReplicationPathPricer: risk free rate (r) must"
                    " be positive or zero");
        QL_REQUIRE(maturity_ > 0.0,
                    "ReplicationPathPricer: maturity must be positive");
        QL_REQUIRE(sigma_ >= 0.0,
                    "ReplicationPathPricer: volatility (sigma)"
                    " must be positive or zero");
    }

    // The value() method encapsulates the pricing code
    double operator()(const Path& path) const;

private:
    Option::Type type_;
    double underlying_, strike_;
    Rate r_;
    Time maturity_;
    double sigma_;
};

// Compute Replication Error as in the Derman and Kamal's research note
int main(int argc, char* argv[])
{
    try {
        Time maturity = 1./12.; // 1 month
        double strike = 100;
        double underlying = 100;
        double volatility = 0.20; // 20%
        Rate riskFreeRate = 0.05; // 5%
        ReplicationError rp(Option::Call, maturity, strike, underlying,

```

```

        volatility, riskFreeRate);

    int scenarios = 50000;
    int hedgesNum;

    hedgesNum = 21;
    rp.compute(hedgesNum, scenarios);

    hedgesNum = 84;
    rp.compute(hedgesNum, scenarios);

    return 0;
} catch (std::exception& e) {
    std::cout << e.what() << std::endl;
    return 1;
} catch (...) {
    std::cout << "unknown error" << std::endl;
    return 1;
}
}

/* The actual computation of the Profit&Loss for each single path.

In each scenario N reheding trades spaced evenly in time over
the life of the option are carried out, using the Black-Scholes
hedge ratio.
*/
double ReplicationPathPricer::operator()(const Path& path) const
{
    // path is an instance of QuantLib::MonteCarlo::Path
    // It contains the list of variations.
    // It can be used as an array: it has a size() method
    int n = path.size();
    QL_REQUIRE(n>0,
        "ReplicationPathPricer: the path cannot be empty");

    // discrete hedging interval
    Time dt = maturity_/n;

    // For simplicity, we assume the stock pays no dividends.
    double stockDividendYield = 0.0;

    // let's start
    Time t = 0;

    // stock value at t=0
    double stock = underlying_;
    double stockLogGrowth = 0.0;

    // money account at t=0
    double money_account = 0.0;

    /**** the initial deal ****/
    // option fair price (Black-Scholes) at t=0
    EuropeanOption option = EuropeanOption(type_, stock, strike_,
        stockDividendYield, r_, maturity_, sigma_);
    // sell the option, cash in its premium
    money_account += option.value();
    // compute delta
    double delta = option.delta();
    // delta-hedge the option buying stock

```



```

double stockAmount = delta;
money_account -= stockAmount*stock;

/*****/
/** hedging during option life **/
/*****/
for(int step = 0; step < n-1; step++){

    // time flows
    t += dt;

    // accruing on the money account
    money_account *= QL_EXP( r_*dt );

    // stock growth:
    // path contains the list of Gaussian variations
    // and path[n] is the n-th variation
    stockLogGrowth += path[step];
    stock = underlying_*QL_EXP(stockLogGrowth);

    // recalculate option value at the current stock value,
    // and the current time to maturity
    EuropeanOption option = EuropeanOption(type_, stock, strike_,
        stockDividendYield, r_, maturity_-t, sigma_);

    // recalculate delta
    delta = option.delta();

    // re-hedging
    money_account -= (delta - stockAmount)*stock;
    stockAmount = delta;
}

/*****/
/** option expiration **/
/*****/
// last accrual on my money account
money_account *= QL_EXP( r_*dt );
// last stock growth
stockLogGrowth += path[n-1];
stock = underlying_*QL_EXP(stockLogGrowth);

// the hedger delivers the option payoff to the option holder
double optionPayoff = ExercisePayoff(type_, stock, strike_);
money_account -= optionPayoff;

// and unwinds the hedge selling his stock position
money_account += stockAmount*stock;

// final Profit&Loss
return money_account;
}

// The computation over nSamples paths of the P&L distribution
void ReplicationError::compute(int nTimeSteps, int nSamples)
{
    QL_REQUIRE(nTimeSteps>0,
        "ReplicationError::compute : the number of steps must be > 0");

    // hedging interval
    // double tau = maturity_ / nTimeSteps;

    /* Black-Scholes framework: the underlying stock price evolves
       lognormally with a fixed known volatility that stays constant

```

```

        throughout time.
    */
    double drift = r_ - 0.5*sigma_*sigma_;

    // Black Scholes equation rules the path generator:
    // at each step the log of the stock
    // will have drift and sigma^2 variance
    Handle<GaussianPathGenerator> myPathGenerator(
        new GaussianPathGenerator(drift, sigma_*sigma_,
            maturity_, nTimeSteps));

    // The replication strategy's Profit&Loss is computed for each path
    // of the stock. The path pricer knows how to price a path using its
    // value() method
    Handle<PathPricer<Path> > myPathPricer =
        Handle<PathPricer<Path> >(new ReplicationPathPricer(type_, s0_, strike_, r_,
            maturity_, sigma_));

    // a statistic accumulator for the path-dependant Profit&Loss values
    Statistics statisticAccumulator;

    // The OneFactorMontecarloModel generates paths using myPathGenerator
    // each path is priced using myPathPricer
    // prices will be accumulated into statisticAccumulator
    OneFactorMonteCarloOption MCSimulation(myPathGenerator,
        myPathPricer, statisticAccumulator);

    // the model simulates nSamples paths
    MCSimulation.addSamples(nSamples);

    // the sampleAccumulator method of OneFactorMonteCarloOption
    // gives access to all the methods of statisticAccumulator
    double PLMean = MCSimulation.sampleAccumulator().mean();
    double PLStDev = MCSimulation.sampleAccumulator().standardDeviation();
    double PLSkew = MCSimulation.sampleAccumulator().skewness();
    double PLKurt = MCSimulation.sampleAccumulator().kurtosis();

    // Derman and Kamal's formula
    double theorStD = QL_SQRT(3.1415926535/4/nTimeSteps)*vega_*sigma_;

    std::cout << nSamples << "\t| "
        << nTimeSteps << "\t | "
        << DoubleFormatter::toString(PLMean, 3) << " \t| "
        << DoubleFormatter::toString(PLStDev, 2) << " \t | "
        << DoubleFormatter::toString(theorStD, 2) << " \t | "
        << DoubleFormatter::toString(PLSkew, 2) << " \t| "
        << DoubleFormatter::toString(PLKurt, 2) << std::endl;
}

```

13.2 EuropeanOption.cpp

This example calculates European call options using different methods while testing call-put parity.

```
#include <ql/quantlib.hpp>

using namespace QuantLib;

using QuantLib::Pricers::EuropeanOption;
using QuantLib::Pricers::McEuropean;
using QuantLib::Pricers::FdEuropean;

// helper function for option payoff: MAX((stike-underlying),0), etc.
using QuantLib::Pricers::ExercisePayoff;

// This will be included in the library after a bit of redesign
class Payoff : public QL::ObjectiveFunction{
public:
    Payoff(Time maturity,
           double strike,
           double s0,
           double sigma,
           Rate r)
        : maturity_(maturity),
          strike_(strike),
          s0_(s0),
          sigma_(sigma), r_(r){}

    double operator()(double x) const {
        double nuT = (r_-0.5*sigma_*sigma_)*maturity_;
        return QL_EXP(-r_*maturity_)
            *ExercisePayoff(Option::Call, s0_*QL_EXP(x), strike_)
            *QL_EXP(-(x - nuT)*(x - nuT)/(2*sigma_*sigma_*maturity_))
            /QL_SQRT(2.0*3.141592*sigma_*sigma_*maturity_);
    }
private:
    Time maturity_;
    double strike_;
    double s0_;
    double sigma_;
    Rate r_;
};

int main(int argc, char* argv[])
{
    try {
        // our option
        double underlying = 102;
        double strike = 100; // at the money
        Spread dividendYield = 0.0; // no dividends
        Rate riskFreeRate = 0.05; // 5%
        Time maturity = 0.25; // 3 months
        double volatility = 0.20; // 20%
        std::cout << "Time to maturity = " << maturity
                  << std::endl;
        std::cout << "Underlying price = " << underlying
                  << std::endl;
        std::cout << "Strike = " << strike
                  << std::endl;
        std::cout << "Risk-free interest rate = " << riskFreeRate
                  << std::endl;
    }
}
```

```

std::cout << "Volatility = " << volatility
          << std::endl;
std::cout << std::endl;

// write column headings
std::cout << "Method\t\tValue\tEstimatedError\tDiscrepancy"
          << "\tRel. Discr." << std::endl;

// first method: Black Scholes analytic solution
std::string method = "Black Scholes";
double value = EuropeanOption(Option::Call, underlying, strike,
                              dividendYield, riskFreeRate, maturity, volatility).value();
double estimatedError = 0.0;
double discrepancy = 0.0;
double relativeDiscrepancy = 0.0;
std::cout << method << "\t"
          << DoubleFormatter::toString(value, 4) << "\t"
          << DoubleFormatter::toString(estimatedError, 4) << "\t\t"
          << DoubleFormatter::toString(discrepancy, 6) << "\t"
          << DoubleFormatter::toString(relativeDiscrepancy, 6)
          << std::endl;

// store the Black Scholes value as the correct one
double rightValue = value;

// second method: Call-Put parity
method = "Call-Put parity";
value = EuropeanOption(Option::Put, underlying, strike,
                      dividendYield, riskFreeRate, maturity, volatility).value()
      + underlying - strike*QL_EXP(- riskFreeRate*maturity);
discrepancy = QL_FABS(value-rightValue);
relativeDiscrepancy = discrepancy/rightValue;
std::cout << method << "\t"
          << DoubleFormatter::toString(value, 4) << "\t"
          << "N/A\t\t"
          << discrepancy << "\t"
          << DoubleFormatter::toString(relativeDiscrepancy, 6)
          << std::endl;

// third method: Integral
method = "Integral";
using QuantLib::Math::SegmentIntegral;
Payoff po(maturity, strike, underlying, volatility, riskFreeRate);
SegmentIntegral integrator(5000);

double nuT = (riskFreeRate - 0.5*volatility*volatility)*maturity;
double infinity = 10.0*volatility*QL_SQRT(maturity);

value = integrator(po, nuT-infinity, nuT+infinity);
discrepancy = QL_FABS(value-rightValue);
relativeDiscrepancy = discrepancy/rightValue;
std::cout << method << "\t"
          << DoubleFormatter::toString(value, 4) << "\t"
          << "N/A\t\t"
          << DoubleFormatter::toString(discrepancy, 6) << "\t"
          << DoubleFormatter::toString(relativeDiscrepancy, 6)
          << std::endl;

```

```

// fourth method: Finite Differences
method = "Finite Diff.";
Size grid = 100;
value = FdEuropean(Option::Call, underlying, strike,
    dividendYield, riskFreeRate, maturity, volatility, grid).value();
discrepancy = QL_FABS(value-rightValue);
relativeDiscrepancy = discrepancy/rightValue;
std::cout << method << "\t"
    << DoubleFormatter::toString(value, 4) << "\t"
    << "N/A\t\t"
    << DoubleFormatter::toString(discrepancy, 6) << "\t"
    << DoubleFormatter::toString(relativeDiscrepancy, 6)
    << std::endl;

// fifth method: Monte Carlo (crude)
method = "MC (crude)";
bool antitheticVariance = false;
McEuropean mcEur(Option::Call, underlying, strike, dividendYield,
    riskFreeRate, maturity, volatility, antitheticVariance);
// let's require a tolerance of 0.002%
value = mcEur.value(0.002);
estimatedError = mcEur.errorEstimate();
discrepancy = QL_FABS(value-rightValue);
relativeDiscrepancy = discrepancy/rightValue;
std::cout << method << "\t"
    << DoubleFormatter::toString(value, 4) << "\t"
    << DoubleFormatter::toString(estimatedError, 4) << "\t\t"
    << DoubleFormatter::toString(discrepancy, 6) << "\t"
    << DoubleFormatter::toString(relativeDiscrepancy, 6)
    << std::endl;

// sixth method: Monte Carlo with antithetic variance reduction
method = "MC (antithetic)";
// let's use the same number of samples as in the crude Monte Carlo
Size nSamples = mcEur.sampleAccumulator().samples();
antitheticVariance = true;
McEuropean mcEur2(Option::Call, underlying, strike, dividendYield,
    riskFreeRate, maturity, volatility, antitheticVariance);
value = mcEur2.valueWithSamples(nSamples);
estimatedError = mcEur2.errorEstimate();
discrepancy = QL_FABS(value-rightValue);
relativeDiscrepancy = discrepancy/rightValue;
std::cout << method << "\t"
    << DoubleFormatter::toString(value, 4) << "\t"
    << DoubleFormatter::toString(estimatedError, 4) << "\t\t"
    << DoubleFormatter::toString(discrepancy, 6) << "\t"
    << DoubleFormatter::toString(relativeDiscrepancy, 6)
    << std::endl;

return 0;
} catch (std::exception& e) {
    std::cout << e.what() << std::endl;
    return 1;
} catch (...) {
    std::cout << "unknown error" << std::endl;
    return 1;
}
}

```

}

13.3 history_iterators.cpp

This code exemplifies how to use History iterators to perform statistic analyses on historical data.

```
// initialize a History
History h(...);

// print out the mean value and its standard deviation.

Statistics s;
s.addSequence(h.vdbegin(),h.vdend());
cout << "Historical mean: " << s.mean() << endl;
cout << "Std. deviation:  " << s.standardDeviation() << endl;

// Another possibility: print out the maximum value.

History::const_valid_iterator max = h.vbegin(), i=max, end = h.vend();
for (i++; i!=end; i++)
    if (i->value() > max->value())
        max = i;
cout << "Maximum value: " << max->value()
    << " assumed " << DateFormatter::toString(max->date()) << endl;

// or the minimum, this time the STL way:

bool lessthan(const History::Entry& i, const History::Entry& j) {
    return i.value() < j.value();
}

History::const_valid_iterator min =
    std::min_element(h.vbegin(),h.vend(),lessthan);
cout << "Minimum value: " << min->value()
    << " assumed " << DateFormatter::toString(min->date()) << endl;
```

13.4 swapvaluation.cpp

This is an example of using the QuantLib Term Structure for pricing a simple swap.

```

/* This example shows how to set up a Term Structure and then price a simple
   swap.
*/

// the only header you need to use QuantLib
#include <ql/quantlib.hpp>

using namespace QuantLib;
using Calendars::TARGET;
using DayCounters::ActualActual;
using DayCounters::Actual360;
using DayCounters::Thirty360;
using Indexes::Xibor;
using Indexes::Euribor;
using Instruments::SimpleSwap;
using TermStructures::PiecewiseFlatForward;
using TermStructures::FlatForward;
using TermStructures::RateHelper;
using TermStructures::DepositRateHelper;
using TermStructures::FraRateHelper;
using TermStructures::FuturesRateHelper;
using TermStructures::SwapRateHelper;

int main(int argc, char* argv[])
{
    try {
        Calendar calendar = TARGET();
        Currency currency = EUR;
        int settlementDays = 2;
        int fixingDays = 2;

        /*****
        *** MARKET DATA ***
        *****/

        Date todaysDate(6, November, 2001);

        // deposits
        double d1wQuote=0.0382;
        double d1mQuote=0.0372;
        double d3mQuote=0.0363;
        double d6mQuote=0.0353;
        double d9mQuote=0.0348;
        double d1yQuote=0.0345;
        // FRAs
        double fra3x6Quote=0.037125;
        double fra6x9Quote=0.037125;
        double fra6x12Quote=0.037125;
        // futures
        double fut1Quote=96.2875;
        double fut2Quote=96.7875;
        double fut3Quote=96.9875;
        double fut4Quote=96.6875;
        double fut5Quote=96.4875;
        double fut6Quote=96.3875;
        double fut7Quote=96.2875;
    }
}

```



```

double fut8Quote=96.0875;
// swaps
double s2yQuote=0.037125;
double s3yQuote=0.0398;
double s5yQuote=0.0443;
double s10yQuote=0.05165;
double s15yQuote=0.055175;

/*****
***  RATE HELPERS ***
*****/

// RateHelpers are built from the above quotes together with other
// instrument dependant infos.

// setup deposits
DayCounter depositDayCounter = Actual360();

Handle<RateHelper> dlw(new DepositRateHelper(
    dlwQuote, settlementDays,
    1, Weeks, calendar, ModifiedFollowing, depositDayCounter));
Handle<RateHelper> d1m(new DepositRateHelper(
    d1mQuote, settlementDays,
    1, Months, calendar, ModifiedFollowing, depositDayCounter));
Handle<RateHelper> d3m(new DepositRateHelper(
    d3mQuote, settlementDays,
    3, Months, calendar, ModifiedFollowing, depositDayCounter));
Handle<RateHelper> d6m(new DepositRateHelper(
    d6mQuote, settlementDays,
    6, Months, calendar, ModifiedFollowing, depositDayCounter));
Handle<RateHelper> d9m(new DepositRateHelper(
    d9mQuote, settlementDays,
    9, Months, calendar, ModifiedFollowing, depositDayCounter));
Handle<RateHelper> d1y(new DepositRateHelper(
    d1yQuote, settlementDays,
    1, Years, calendar, ModifiedFollowing, depositDayCounter));

// setup swaps
int swFixedLegFrequency = 1;
bool swFixedLegIsAdjusted = false;
DayCounter swFixedLegDayCounter = Thirty360(Thirty360::European);
int swFloatingLegFrequency = 2;

Handle<RateHelper> s2y(new SwapRateHelper(
    s2yQuote, settlementDays,
    2, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
Handle<RateHelper> s3y(new SwapRateHelper(
    s3yQuote, settlementDays,
    3, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
Handle<RateHelper> s5y(new SwapRateHelper(
    s5yQuote, settlementDays,
    5, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
Handle<RateHelper> s10y(new SwapRateHelper(
    s10yQuote, settlementDays,
    10, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));

```

```

        swFloatingLegFrequency));
Handle<RateHelper> s15y(new SwapRateHelper(
    s15yQuote, settlementDays,
    15, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));

/*****
**   CURVE BUILDING **
*****/

// Any DayCounter would be fine.
// ActualActual::ISDA ensures that 30 years is 30.0
DayCounter termStructureDayCounter =
    ActualActual(ActualActual::ISDA);

// A depo-swap curve
std::vector<Handle<RateHelper> > depoSwapInstruments;
depoSwapInstruments.push_back(d1w);
depoSwapInstruments.push_back(d1m);
depoSwapInstruments.push_back(d3m);
depoSwapInstruments.push_back(d6m);
depoSwapInstruments.push_back(d9m);
depoSwapInstruments.push_back(d1y);
depoSwapInstruments.push_back(s2y);
depoSwapInstruments.push_back(s3y);
depoSwapInstruments.push_back(s5y);
depoSwapInstruments.push_back(s10y);
depoSwapInstruments.push_back(s15y);
Handle<TermStructure> depoSwapTermStructure(new
    PiecewiseFlatForward(currency, termStructureDayCounter,
        todaysDate, calendar, settlementDays, depoSwapInstruments));

/*****
*   SWAPS TO BE PRICED *
*****/

// Term structures that will be used for pricing:
// the one used for discounting cash flows
RelinkableHandle<TermStructure> discountingTermStructure;
// the one used for forward rate forecasting
RelinkableHandle<TermStructure> forecastingTermStructure;

// spot start
Date spotDate = calendar.advance(todaysDate, settlementDays, Days,
    Following);
// constant nominal 1,000,000 Euro
double nominal = 1000000.0;
// fixed leg
int fixedLegFrequency = 1; // annual
bool fixedLegIsAdjusted = false;
RollingConvention roll = ModifiedFollowing;
DayCounter fixedLegDayCounter = Thirty360(Thirty360::European);
Rate fixedRate = 0.04;

// floating leg
int floatingLegFrequency = 2;
Handle<Xibor> euriborIndex(new Euribor(6, Months,
    forecastingTermStructure)); // using the forecasting curve
Spread spread = 0.0;

int lenghtInYears = 5;

```

```

bool payFixedRate = true;
SimpleSwap spot5YearSwap(payFixedRate, spotDate, lenghtInYears,
    Years, calendar, roll, nominal, fixedLegFrequency, fixedRate,
    fixedLegIsAdjusted, fixedLegDayCounter, floatingLegFrequency,
    euriborIndex, fixingDays, spread,
    discountingTermStructure); // using the discounting curve
SimpleSwap oneYearForward5YearSwap(payFixedRate,
    calendar.advance(spotDate, 1, Years, ModifiedFollowing),
    lenghtInYears, Years,
    calendar, roll, nominal, fixedLegFrequency, fixedRate,
    fixedLegIsAdjusted, fixedLegDayCounter, floatingLegFrequency,
    euriborIndex, fixingDays, spread,
    discountingTermStructure); // using the discounting curve

/*****
 * SWAP PRICING *
 *****/

// let's price in term of NPV, fixed rate, and spread
double NPV;
Rate fairFixedRate;
Spread fairFloatingSpread;

// Of course, you're not forced to really use different curves
forecastingTermStructure.linkTo(depoSwapTermStructure);
discountingTermStructure.linkTo(depoSwapTermStructure);
std::cout << "*** using Depo-Fut-Swap term structure:" << std::endl;

NPV = spot5YearSwap.NPV();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = - NPV / spot5YearSwap.floatingLegBPS();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " spread: "
    << RateFormatter::toString(fairFloatingSpread,8)
    << std::endl;
fairFixedRate = fixedRate - NPV / spot5YearSwap.fixedLegBPS();
std::cout << "5Y fixed rate: "
    << RateFormatter::toString(fairFixedRate,8)
    << std::endl;
// let's check that the 5 years swap has been correctly re-priced
QL_REQUIRE(QL_FABS(fairFixedRate-s5yQuote)<1e-8,
    "5 years swap mispriced!");

// now let's price the 1Y forward 5Y swap
NPV = oneYearForward5YearSwap.NPV();
std::cout << "1Yx5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = -NPV / spot5YearSwap.floatingLegBPS();
std::cout << "1Yx5Y "
    << RateFormatter::toString(fixedRate,2)
    << " spread: "
    << RateFormatter::toString(fairFloatingSpread,8)
    << std::endl;
fairFixedRate = fixedRate - NPV/oneYearForward5YearSwap.fixedLegBPS();
std::cout << "1Yx5Y fixed rate: "

```

```

    << RateFormatter::toString(fairFixedRate,8)
    << std::endl;

// now, let's get serious

/*****
***  MARKET DATA  ***
*****/

// market elements are containers for quotes.
// SimpleMarketElement stores a value which can be manually changed;
// other MarketElement subclasses could read the value from a
// database or some kind of data feed.

// deposits
Handle<MarketElement> dlwRate(new SimpleMarketElement(dlwQuote));
Handle<MarketElement> dlmRate(new SimpleMarketElement(dlmQuote));
Handle<MarketElement> d3mRate(new SimpleMarketElement(d3mQuote));
Handle<MarketElement> d6mRate(new SimpleMarketElement(d6mQuote));
Handle<MarketElement> d9mRate(new SimpleMarketElement(d9mQuote));
Handle<MarketElement> d1yRate(new SimpleMarketElement(d1yQuote));
// FRAs
Handle<MarketElement> fra3x6Rate(new SimpleMarketElement(fra3x6Quote));
Handle<MarketElement> fra6x9Rate(new SimpleMarketElement(fra6x9Quote));
Handle<MarketElement> fra6x12Rate(new SimpleMarketElement(fra6x12Quote));
// futures
Handle<MarketElement> fut1Price(new SimpleMarketElement(fut1Quote));
Handle<MarketElement> fut2Price(new SimpleMarketElement(fut2Quote));
Handle<MarketElement> fut3Price(new SimpleMarketElement(fut3Quote));
Handle<MarketElement> fut4Price(new SimpleMarketElement(fut4Quote));
Handle<MarketElement> fut5Price(new SimpleMarketElement(fut5Quote));
Handle<MarketElement> fut6Price(new SimpleMarketElement(fut6Quote));
Handle<MarketElement> fut7Price(new SimpleMarketElement(fut7Quote));
Handle<MarketElement> fut8Price(new SimpleMarketElement(fut8Quote));
// swaps
Handle<MarketElement> s2yRate(new SimpleMarketElement(s2yQuote));
Handle<MarketElement> s3yRate(new SimpleMarketElement(s3yQuote));
Handle<MarketElement> s5yRate(new SimpleMarketElement(s5yQuote));
Handle<MarketElement> s10yRate(new SimpleMarketElement(s10yQuote));
Handle<MarketElement> s15yRate(new SimpleMarketElement(s15yQuote));

/*****
***  RATE HELPERS  ***
*****/

// RateHelpers are built from the above quotes together with other
// instrument dependant infos.
// This time quotes are passed in relinkable
// handles which could be relinked to some other data source later.

// setup deposits
dlw = Handle<RateHelper>(new DepositRateHelper(
    RelinkableHandle<MarketElement>(dlwRate), settlementDays,
    1, Weeks, calendar, ModifiedFollowing, depositDayCounter));
dlm = Handle<RateHelper>(new DepositRateHelper(

```

```

        RelinkableHandle<MarketElement>(d1mRate), settlementDays,
        1, Months, calendar, ModifiedFollowing, depositDayCounter));
d3m=Handle<RateHelper>(new DepositRateHelper(
    RelinkableHandle<MarketElement>(d3mRate), settlementDays,
    3, Months, calendar, ModifiedFollowing, depositDayCounter));
d6m=Handle<RateHelper>(new DepositRateHelper(
    RelinkableHandle<MarketElement>(d6mRate), settlementDays,
    6, Months, calendar, ModifiedFollowing, depositDayCounter));
d9m=Handle<RateHelper>(new DepositRateHelper(
    RelinkableHandle<MarketElement>(d9mRate), settlementDays,
    9, Months, calendar, ModifiedFollowing, depositDayCounter));
d1y=Handle<RateHelper>(new DepositRateHelper(
    RelinkableHandle<MarketElement>(d1yRate), settlementDays,
    1, Years, calendar, ModifiedFollowing, depositDayCounter));

// setup swaps
s2y=Handle<RateHelper>(new SwapRateHelper(
    RelinkableHandle<MarketElement>(s2yRate), settlementDays,
    2, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
s3y=Handle<RateHelper>(new SwapRateHelper(
    RelinkableHandle<MarketElement>(s3yRate), settlementDays,
    3, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
s5y=Handle<RateHelper>(new SwapRateHelper(
    RelinkableHandle<MarketElement>(s5yRate), settlementDays,
    5, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
s10y=Handle<RateHelper>(new SwapRateHelper(
    RelinkableHandle<MarketElement>(s10yRate), settlementDays,
    10, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));
s15y=Handle<RateHelper>(new SwapRateHelper(
    RelinkableHandle<MarketElement>(s15yRate), settlementDays,
    15, calendar, ModifiedFollowing, swFixedLegFrequency,
    swFixedLegIsAdjusted, swFixedLegDayCounter,
    swFloatingLegFrequency));

// let's add FRA and futures

// setup FRAs
Handle<RateHelper> fra3x6(new FraRateHelper(
    RelinkableHandle<MarketElement>(fra3x6Rate),
    settlementDays, 3, 6, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fra6x9(new FraRateHelper(
    RelinkableHandle<MarketElement>(fra6x9Rate),
    settlementDays, 6, 9, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fra6x12(new FraRateHelper(
    RelinkableHandle<MarketElement>(fra6x12Rate),
    settlementDays, 6, 12, calendar, ModifiedFollowing,
    depositDayCounter));

// setup futures
int futMonths = 3;
Handle<RateHelper> fut1(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(19, December, 2001),
    settlementDays, futMonths, calendar, ModifiedFollowing,

```

```

        depositDayCounter));
Handle<RateHelper> fut2(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(20, March, 2002),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fut3(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(19, June, 2002),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fut4(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(18, September, 2002),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fut5(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(18, December, 2002),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fut6(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(19, March, 2003),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fut7(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(18, June, 2003),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));
Handle<RateHelper> fut8(new FuturesRateHelper(
    RelinkableHandle<MarketElement>(fut1Price),
    Date(17, September, 2003),
    settlementDays, futMonths, calendar, ModifiedFollowing,
    depositDayCounter));

/*****
** CURVE BUILDING **
*****/

// A depo-futures-swap curve
std::vector<Handle<RateHelper> > depoFutSwapInstruments;
depoFutSwapInstruments.push_back(dlw);
depoFutSwapInstruments.push_back(dlm);
depoFutSwapInstruments.push_back(fut1);
depoFutSwapInstruments.push_back(fut2);
depoFutSwapInstruments.push_back(fut3);
depoFutSwapInstruments.push_back(fut4);
depoFutSwapInstruments.push_back(fut5);
depoFutSwapInstruments.push_back(fut6);
depoFutSwapInstruments.push_back(fut7);
depoFutSwapInstruments.push_back(fut8);
depoFutSwapInstruments.push_back(s3y);
depoFutSwapInstruments.push_back(s5y);
depoFutSwapInstruments.push_back(s10y);
depoFutSwapInstruments.push_back(s15y);
Handle<TermStructure> depoFutSwapTermStructure(new
    PiecewiseFlatForward(currency, termStructureDayCounter,
        todaysDate, calendar, settlementDays, depoFutSwapInstruments));

```

```

// A depo-FRA-swap curve
std::vector<Handle<RateHelper> > depoFRASwapInstruments;
depoFRASwapInstruments.push_back(d1w);
depoFRASwapInstruments.push_back(d1m);
depoFRASwapInstruments.push_back(d3m);
depoFRASwapInstruments.push_back(fra3x6);
depoFRASwapInstruments.push_back(fra6x9);
depoFRASwapInstruments.push_back(fra6x12);
depoFRASwapInstruments.push_back(s2y);
depoFRASwapInstruments.push_back(s3y);
depoFRASwapInstruments.push_back(s5y);
depoFRASwapInstruments.push_back(s10y);
depoFRASwapInstruments.push_back(s15y);
Handle<TermStructure> depoFRASwapTermStructure(new
    PiecewiseFlatForward(currency, termStructureDayCounter,
        todaysDate, calendar, settlementDays, depoFRASwapInstruments));

/*****
 * SWAP PRICING *
 *****/

// switch the curve used by the swaps to be priced
forecastingTermStructure.linkTo(depoFutSwapTermStructure);
discountingTermStructure.linkTo(depoFutSwapTermStructure);
std::cout << "**** using Depo-Fut-Swap term structure:" << std::endl;

NPV = spot5YearSwap.NPV();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = - NPV / spot5YearSwap.floatingLegBPS();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " spread: "
    << RateFormatter::toString(fairFloatingSpread,8)
    << std::endl;
fairFixedRate = fixedRate - NPV / spot5YearSwap.fixedLegBPS();
std::cout << "5Y fixed rate: "
    << RateFormatter::toString(fairFixedRate,8)
    << std::endl;
// let's check that the 5 years swap has been correctly re-priced
QL_REQUIRE(QL_FABS(fairFixedRate-s5yQuote)<1e-8,
    "5 years swap mispriced!");

// now let's price the 1Y forward 5Y swap
NPV = oneYearForward5YearSwap.NPV();
std::cout << "1Yx5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = -NPV / spot5YearSwap.floatingLegBPS();
std::cout << "1Yx5Y "
    << RateFormatter::toString(fixedRate,2)

```

```

        << " spread:                "
        << RateFormatter::toString(fairFloatingSpread,8)
        << std::endl;
fairFixedRate = fixedRate - NPV/oneYearForward5YearSwap.fixedLegBPS();
std::cout << "1Yx5Y fixed rate:    "
        << RateFormatter::toString(fairFixedRate,8)
        << std::endl;

// switch the curve used by the swaps to be priced
forecastingTermStructure.linkTo(depoFRASwapTermStructure);
discountingTermStructure.linkTo(depoFRASwapTermStructure);
std::cout << "**** using Depo-FRA-Swap term structure:" << std::endl;

NPV = spot5YearSwap.NPV();
std::cout << "5Y "
        << RateFormatter::toString(fixedRate,2)
        << " NPV:                "
        << DoubleFormatter::toString(NPV,2)
        << std::endl;
fairFloatingSpread = - NPV / spot5YearSwap.floatingLegBPS();
std::cout << "5Y "
        << RateFormatter::toString(fixedRate,2)
        << " spread:                "
        << RateFormatter::toString(fairFloatingSpread,8)
        << std::endl;
fairFixedRate = fixedRate - NPV / spot5YearSwap.fixedLegBPS();
std::cout << "5Y fixed rate:      "
        << RateFormatter::toString(fairFixedRate,8)
        << std::endl;
// let's check that the 5 years swap has been correctly re-priced
QL_REQUIRE(QL_FABS(fairFixedRate-s5yRate->value())<1e-8,
           "5 years swap mispriced!");

// now let's price the 1Y forward 5Y swap
NPV = oneYearForward5YearSwap.NPV();
std::cout << "1Yx5Y "
        << RateFormatter::toString(fixedRate,2)
        << " NPV:                "
        << DoubleFormatter::toString(NPV,2)
        << std::endl;
fairFloatingSpread = -NPV / spot5YearSwap.floatingLegBPS();
std::cout << "1Yx5Y "
        << RateFormatter::toString(fixedRate,2)
        << " spread:                "
        << RateFormatter::toString(fairFloatingSpread,8)
        << std::endl;
fairFixedRate = fixedRate - NPV/oneYearForward5YearSwap.fixedLegBPS();
std::cout << "1Yx5Y fixed rate:    "
        << RateFormatter::toString(fairFixedRate,8)
        << std::endl;

// now let's say that the 5-years swap rate goes up to 4.60%.
// A smarter market element--say, connected to a data source-- would
// notice the change itself. Since we're using SimpleMarketElements,
// we'll have to change the value manually--which forces us to
// downcast the handle and use the SimpleMarketElement
// interface. In any case, the point here is that a change in the
// value contained in the MarketElement triggers a new bootstrapping
// of the curve and a repricing of the swap.

Handle<SimpleMarketElement> fiveYearsRate = s5yRate;
fiveYearsRate->setValue(0.0460);
std::cout << "**** 5Y swap goes up to 4.60%" << std::endl;

```



```

// now get the updated results
forecastingTermStructure.linkTo(depoFutSwapTermStructure);
discountingTermStructure.linkTo(depoFutSwapTermStructure);
std::cout << "*** using Depo-Fut-Swap term structure:" << std::endl;

NPV = spot5YearSwap.NPV();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = - NPV / spot5YearSwap.floatingLegBPS();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " spread: "
    << RateFormatter::toString(fairFloatingSpread,8)
    << std::endl;
fairFixedRate = fixedRate - NPV / spot5YearSwap.fixedLegBPS();
std::cout << "5Y fixed rate: "
    << RateFormatter::toString(fairFixedRate,8)
    << std::endl;
// let's check that the 5 years swap has been correctly re-priced
QL_REQUIRE(QL_FABS(fairFixedRate-s5yRate->value())<1e-8,
    "5 years swap mispriced!");

NPV = oneYearForward5YearSwap.NPV();
std::cout << "1Yx5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = -NPV / spot5YearSwap.floatingLegBPS();
std::cout << "1Yx5Y "
    << RateFormatter::toString(fixedRate,2)
    << " spread: "
    << RateFormatter::toString(fairFloatingSpread,8)
    << std::endl;
fairFixedRate = fixedRate - NPV/oneYearForward5YearSwap.fixedLegBPS();
std::cout << "1Yx5Y fixed rate: "
    << RateFormatter::toString(fairFixedRate,8)
    << std::endl;

forecastingTermStructure.linkTo(depoFRASwapTermStructure);
discountingTermStructure.linkTo(depoFRASwapTermStructure);
std::cout << "*** using Depo-FRA-Swap term structure:" << std::endl;

NPV = spot5YearSwap.NPV();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " NPV: "
    << DoubleFormatter::toString(NPV,2)
    << std::endl;
fairFloatingSpread = - NPV / spot5YearSwap.floatingLegBPS();
std::cout << "5Y "
    << RateFormatter::toString(fixedRate,2)
    << " spread: "
    << RateFormatter::toString(fairFloatingSpread,8)
    << std::endl;
fairFixedRate = fixedRate - NPV / spot5YearSwap.fixedLegBPS();
std::cout << "5Y fixed rate: "
    << RateFormatter::toString(fairFixedRate,8)
    << std::endl;

```

```

// let's check that the 5 years swap has been correctly re-priced
QL_REQUIRE(QL_FABS(fairFixedRate-s5yRate->value())<1e-8,
            "5 years swap mispriced!");

NPV = oneYearForward5YearSwap.NPV();
std::cout << "1Yx5Y "
            << RateFormatter::toString(fixedRate,2)
            << " NPV: "
            << DoubleFormatter::toString(NPV,2)
            << std::endl;
fairFloatingSpread = -NPV / spot5YearSwap.floatingLegBPS();
std::cout << "1Yx5Y "
            << RateFormatter::toString(fixedRate,2)
            << " spread: "
            << RateFormatter::toString(fairFloatingSpread,8)
            << std::endl;
fairFixedRate = fixedRate - NPV/oneYearForward5YearSwap.fixedLegBPS();
std::cout << "1Yx5Y fixed rate: "
            << RateFormatter::toString(fairFixedRate,8)
            << std::endl;

return 0;

} catch (std::exception& e) {
    std::cout << e.what() << std::endl;
    return 1;
} catch (...) {
    std::cout << "unknown error" << std::endl;
    return 1;
}
}

```

Index

- ~Arguments
 - QuantLib::Arguments, [111](#)
 - ~Array
 - QuantLib::Array, [114](#)
 - ~BlackKarasinski
 - QuantLib::InterestRateModelling::Black-Karasinski, [125](#)
 - ~CapFlatVolatilityStructure
 - QuantLib::CapFlatVolatilityStructure, [139](#)
 - ~CapletForwardVolatilityStructure
 - QuantLib::CapletForwardVolatility-Structure, [144](#)
 - ~CashFlow
 - QuantLib::CashFlow, [145](#)
 - ~ConjugateGradient
 - QuantLib::Optimization::Conjugate-Gradient, [152](#)
 - ~CubicSpline
 - QuantLib::Math::CubicSpline, [160](#)
 - ~DiffusionProcess
 - QuantLib::DiffusionProcess, [171](#)
 - ~DiscountStructure
 - QuantLib::DiscountStructure, [172](#)
 - ~Error
 - QuantLib::Error, [180](#)
 - ~ExtendedVasicek
 - QuantLib::InterestRate-Modelling::ExtendedVasicek, [189](#)
 - ~ForwardRateStructure
 - QuantLib::ForwardRateStructure, [207](#)
 - ~GeneralBlackKarasinski
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, [216](#)
 - ~GeneralCoxIngersollRoss
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, [218](#)
 - ~Handle
 - QuantLib::Handle, [223](#)
 - ~HullWhite
 - QuantLib::InterestRateModelling::Hull-White, [232](#)
 - ~Index
 - QuantLib::Index, [240](#)
 - ~Instrument
 - QuantLib::Instrument, [242](#)
 - ~Interpolation2D
 - QuantLib::Math::Interpolation2D, [248](#)
 - ~LeastSquareFunction
 - QuantLib::Optimization::LeastSquare-Function, [254](#)
 - ~MarketElement
 - QuantLib::MarketElement, [263](#)
 - ~Matrix
 - QuantLib::Math::Matrix, [264](#)
 - ~McPricer
 - QuantLib::Pricers::McPricer, [276](#)
 - ~Model
 - QuantLib::InterestRateModelling::Model, [280](#)
 - ~NonLinearLeastSquare
 - QuantLib::Optimization::NonLinearLeast-Square, [291](#)
 - ~ObjectiveFunction
 - QuantLib::ObjectiveFunction, [294](#)
 - ~Observable
 - QuantLib::Patterns::Observable, [295](#)
 - ~Observer
 - QuantLib::Patterns::Observer, [297](#)
 - ~OneFactorModel
 - QuantLib::InterestRateModelling::One-FactorModel, [299](#)
 - ~OneFactorOperator
 - QuantLib::FiniteDifferences::OneFactor-Operator, [300](#)
 - ~OptimizationMethod
 - QuantLib::Optimization::Optimization-Method, [301](#)
 - ~OptimizationProblem
 - QuantLib::Optimization::Optimization-Problem, [303](#)
 - ~Option
 - QuantLib::Option, [304](#)
 - ~OptionPricingEngine
 - QuantLib::OptionPricingEngine, [307](#)
 - ~PathPricer
 - QuantLib::MonteCarlo::PathPricer, [314](#)
 - ~RateHelper
-

- QuantLib::TermStructures::RateHelper, 330
- ~Results
 - QuantLib::Results, 334
- ~Simplex
 - QuantLib::Optimization::Simplex, 348
- ~SingleAssetOption
 - QuantLib::Pricers::SingleAssetOption, 349
- ~Solver1D
 - QuantLib::Solver1D, 351
- ~SteepestDescent
 - QuantLib::Optimization::SteepestDescent, 356
- ~StepCondition
 - QuantLib::FiniteDifferences::StepCondition, 357
- ~SwaptionVolatilityStructure
 - QuantLib::SwaptionVolatilityStructure, 371
- ~TermStructure
 - QuantLib::TermStructure, 375
- ~TimeSetter
 - QuantLib::FiniteDifferences::TridiagonalOperator::TimeSetter, 385
- ~ZeroYieldStructure
 - QuantLib::ZeroYieldStructure, 394
- a_
 - QuantLib::InterestRateModelling::ExtendedVasicek, 189
 - QuantLib::InterestRateModelling::GeneralBlackKarasinski, 216
- aboveDiagonal_
 - QuantLib::FiniteDifferences::TridiagonalOperator, 384
- Abs
 - QuantLib, 84
 - QuantLib::Array, 115
- accrualDays
 - QuantLib::CashFlows::Coupon, 157
- accrualEndDate
 - QuantLib::CashFlows::Coupon, 157
- accrualPeriod
 - QuantLib::CashFlows::Coupon, 157
- accrualStartDate
 - QuantLib::CashFlows::Coupon, 157
- accrualTimes
 - QuantLib::Instruments::CapFloorParameters, 141
- accruedAmount
 - QuantLib::CashFlows::Coupon, 157
 - QuantLib::CashFlows::FixedRateCoupon, 202
- QuantLib::CashFlows::FloatingRateCoupon, 204
- QuantLib::CashFlows::ShortFloatingRateCoupon, 343
- Actual360
 - QuantLib::DayCounters::Actual360, 107
- actual360.hpp, 397
- Actual365
 - QuantLib::DayCounters::Actual365, 108
- actual365.hpp, 398
- ActualActual
 - QuantLib::DayCounters::ActualActual, 109
- actualactual.cpp, 399
- actualactual.hpp, 400
- add
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::Math::Statistics, 355
 - QuantLib::RiskStatistics, 338
- addSamples
 - QuantLib::MonteCarlo::MonteCarloModel, 283
- addSequence
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::Math::Statistics, 353
 - QuantLib::RiskStatistics, 337
- advance
 - QuantLib::Calendar, 135
- algoMacros
 - QL_MAX, 76
 - QL_MIN, 76
- allowExtrapolation_
 - QuantLib::Math::Interpolation, 246
 - QuantLib::Math::Interpolation2D, 248
- americancondition.hpp, 401
- amount
 - QuantLib::CashFlow, 145
 - QuantLib::CashFlows::FixedRateCoupon, 202
 - QuantLib::CashFlows::FloatingRateCoupon, 205
 - QuantLib::CashFlows::ShortFloatingRateCoupon, 344
 - QuantLib::CashFlows::SimpleCashFlow, 345
- AnalyticalCapFloor
 - QuantLib::Pricers::AnalyticalCapFloor, 110
- analyticalcapfloor.cpp, 402
- analyticalcapfloor.hpp, 403
- applyTo

- QuantLib::FiniteDifferences::Step-Condition, 357
- QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- argsandresults.hpp, 404
- argument_type
 - QuantLib::Math::CubicSpline, 160
 - QuantLib::Math::Interpolation, 246
 - QuantLib::Math::LinearInterpolation, 258
- ArithmeticAOPATHPricer
 - QuantLib::MonteCarlo::Arithmetic-AOPATHPricer, 112
- arithmeticapopathpricer.cpp, 405
- arithmeticapopathpricer.hpp, 406
- ArithmeticASOPATHPricer
 - QuantLib::MonteCarlo::Arithmetic-ASOPATHPricer, 113
- arithmeticasopathpricer.cpp, 407
- arithmeticasopathpricer.hpp, 408
- armijo.cpp, 409
- armijo.hpp, 410
- Array
 - QuantLib::Array, 114
- array.hpp, 411
- arrayType
 - QuantLib::FiniteDifferences::Finite-DifferenceModel, 201
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- AssertionFailedError
 - QuantLib::AssertionFailedError, 117
- assetNumber
 - QuantLib::MonteCarlo::MultiPath, 284
- AUD
 - QuantLib, 85
- AUDLiber
 - QuantLib::Indexes::AUDLiber, 118
- audlibor.hpp, 412
- averageShortfall
 - QuantLib::Math::RiskMeasures, 336
 - QuantLib::RiskStatistics, 337
- BarrierOption
 - QuantLib::Pricers::BarrierOption, 119
- barrieroption.cpp, 413
- barrieroption.hpp, 414
- BasketPathPricer
 - QuantLib::MonteCarlo::BasketPathPricer, 120
- basketpathpricer.cpp, 415
- basketpathpricer.hpp, 416
- begin
 - QuantLib::Array, 115
 - QuantLib::History, 228
 - QuantLib::Math::Matrix, 264
 - QuantLib::Scheduler, 340
- belowDiagonal_
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 384
- BGL
 - QuantLib, 85
- BilinearInterpolation
 - QuantLib::Math::BilinearInterpolation, 121
- bilinearinterpolation.hpp, 417
- BinaryOption
 - QuantLib::Pricers::BinaryOption, 122
- binaryoption.cpp, 418
- binaryoption.hpp, 419
- binomialtree.cpp, 420
- binomialtree.hpp, 421
- bisection.cpp, 422
- bisection.hpp, 423
- BlackCapFloor
 - QuantLib::Pricers::BlackCapFloor, 124
- blackcapfloor.cpp, 424
- blackcapfloor.hpp, 425
- BlackKarasinski
 - QuantLib::InterestRateModelling::Black-Karasinski, 125
- blackkarasinski.cpp, 426
- blackkarasinski.hpp, 427
- BlackModel
 - QuantLib::InterestRateModelling::Black-Model, 126
- blackmodel.hpp, 428
- BlackScholesProcess
 - QuantLib::BlackScholesProcess, 127
- BlackSwaption
 - QuantLib::Pricers::BlackSwaption, 128
- blackswaption.cpp, 429
- blackswaption.hpp, 430
- BoundaryCondition
 - QuantLib::FiniteDifferences::Boundary-Condition, 129
- boundarycondition.hpp, 431
- BoxMullerGaussianRng
 - QuantLib::RandomNumbers::BoxMuller-GaussianRng, 130
- boxmullergaussianrng.hpp, 432
- brent.cpp, 433
- SIGN, 433
- brent.hpp, 434
- BSMOperator
 - QuantLib::Finite-Differences::BSMOperator, 132
- bsmoperator.cpp, 435

- bsmoperator.hpp, 436
- CAD
 - QuantLib, 85
- CADLiber
 - QuantLib::Indexes::CADLiber, 133
- cadlibor.hpp, 437
- calculate
 - QuantLib::Instrument, 243
 - QuantLib::OptionPricingEngine, 307
 - QuantLib::Pricers::AnalyticalCapFloor, 110
 - QuantLib::Pricers::BlackCapFloor, 124
 - QuantLib::Pricers::BlackSwaption, 128
 - QuantLib::Pricers::EuropeanEngine, 183
 - QuantLib::Pricers::FdBsmOption, 194
 - QuantLib::Pricers::FdEuropean, 198
 - QuantLib::Pricers::FdStepCondition-Option, 199
 - QuantLib::Pricers::JamshidianSwaption, 250
 - QuantLib::Pricers::TreeCapFloor, 381
 - QuantLib::Pricers::TreeSwaption, 382
- calculate_
 - QuantLib::Pricers::BarrierOption, 119
- Calendar
 - QuantLib::Calendar, 135
- calendar
 - QuantLib::Exercise, 187
 - QuantLib::ForwardSpreadedTerm-Structure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::Indexes::Xibor, 388
 - QuantLib::TermStructure, 375
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 316
 - QuantLib::ZeroSpreadedTermStructure, 392
- calendar.cpp, 438
- calendar.hpp, 439
- calendar_
 - QuantLib::CashFlows::Coupon, 158
 - QuantLib::Exercise, 187
 - QuantLib::TermStructures::SwapRate-Helper, 364
- calibrate
 - QuantLib::InterestRateModelling::Model, 280
- CalibrationFunction
 - QuantLib::InterestRateModelling::Model, 280
- calibrationhelper.cpp, 440
- calibrationhelper.hpp, 441
- CapFlatVolatilityVector
 - QuantLib::Volatilities::CapFlatVolatility-Vector, 140
- capflatvolvector.hpp, 442
- capfloor.cpp, 443
- capfloor.hpp, 444
- CapFloorParameters
 - QuantLib::Instruments::CapFloor-Parameters, 141
- CapFloorPricingEngine
 - QuantLib::Pricers::CapFloorPricing-Engine, 142
- caphelper.cpp, 445
- caphelper.hpp, 446
- capRates
 - QuantLib::Instruments::CapFloor-Parameters, 141
- capvolstructures.hpp, 447
- cashflow.hpp, 448
- cashflowvectors.cpp, 449
- cashflowvectors.hpp, 450
- center_
 - QuantLib::Pricers::FdBsmOption, 194
- centrallimitgaussianrng.hpp, 451
- Character functions, 74
- charMacros
 - QL_STRLEN, 74
 - QL_TOLOWER, 74
 - QL_TOUPPER, 74
- CHF
 - QuantLib, 85
- CHFLiber
 - QuantLib::Indexes::CHFLiber, 146
- chfliber.hpp, 452
- CLGaussianRng
 - QuantLib::RandomNumbers::CLGaussian-Rng, 147
- CliquetOption
 - QuantLib::Pricers::CliquetOption, 148
- cliquetoption.cpp, 453
- cliquetoption.hpp, 454
- clone
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::ContinuousGeometric-APO, 153
 - QuantLib::Pricers::DiscreteGeometric-APO, 173
 - QuantLib::Pricers::DiscreteGeometric-ASO, 174
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdAmericanOption, 192

- QuantLib::Pricers::FdBermudanOption, 193
- QuantLib::Pricers::FdDividendEuropeanOption, 196
- QuantLib::Pricers::FdDividendShoutOption, 197
- QuantLib::Pricers::FdEuropean, 198
- QuantLib::Pricers::SingleAssetOption, 349
- column_begin
 - QuantLib::Math::Matrix, 265
- column_end
 - QuantLib::Math::Matrix, 265
- column_iterator
 - QuantLib::Math::Matrix, 264
- column_rbegin
 - QuantLib::Math::Matrix, 265
- column_rend
 - QuantLib::Math::Matrix, 265
- columns
 - QuantLib::Math::Matrix, 265
- combining_iterator
 - QuantLib::Utilities::combining_iterator, 149
- combiningiterator.hpp, 455
- CompositeMarketElement
 - QuantLib::CompositeMarketElement, 151
- ConjugateGradient
 - QuantLib::Optimization::ConjugateGradient, 152
- conjugategradient.cpp, 456
- conjugategradient.hpp, 457
- const_column_iterator
 - QuantLib::Math::Matrix, 264
- const_data_iterator
 - QuantLib::History, 227
- const_iterator
 - QuantLib::Array, 114
 - QuantLib::History::Entry, 231
 - QuantLib::Math::Matrix, 264
 - QuantLib::Scheduler, 340
- const_row_iterator
 - QuantLib::Math::Matrix, 264
- const_valid_data_iterator
 - QuantLib::History, 227
- const_valid_iterator
 - QuantLib::History, 227
- constraint
 - QuantLib::Optimization::OptimizationProblem, 303
- constraint.hpp, 458
- constraint_
 - QuantLib::InterestRateModelling::Model, 280
- QuantLib::Optimization::OptimizationProblem, 303
- ContinuousGeometricAPO
 - QuantLib::Pricers::ContinuousGeometricAPO, 153
- continuousgeometricapo.hpp, 459
- convention_
 - QuantLib::Exercise, 187
 - QuantLib::TermStructures::SwapRateHelper, 364
- correlation
 - QuantLib::Math::MultivariateAccumulator, 286
- costFunction
 - QuantLib::Optimization::OptimizationProblem, 303
- costfunction.hpp, 460
- costFunction_
 - QuantLib::Optimization::OptimizationProblem, 303
- coupling_iterator
 - QuantLib::Utilities::coupling_iterator, 155
- couplingiterator.hpp, 461
- Coupon
 - QuantLib::CashFlows::Coupon, 157
- coupon.hpp, 462
- covariance
 - QuantLib::Math::MultivariateAccumulator, 287
- coxingersollross.cpp, 463
- coxingersollross.hpp, 464
- cranknicolson.hpp, 465
- criteria.hpp, 466
- CubicSpline
 - QuantLib::Math::CubicSpline, 160
- cubicspline.hpp, 467
- Currency
 - QuantLib, 85
- currency
 - QuantLib::ForwardSpreadedTermStructure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::Indexes::Xibor, 388
 - QuantLib::TermStructure, 376
 - QuantLib::TermStructures::PiecewiseFlatForward, 316
 - QuantLib::ZeroSpreadedTermStructure, 392
- currency.hpp, 468
- currentLink
 - QuantLib::Link, 259
- CYP
 - QuantLib, 85
- CZK

- QuantLib, 85
- data_
 - QuantLib::Math::Interpolation2D, 248
- data_iterator
 - QuantLib::History, 228
- dataformatters.cpp, 469
- dataformatters.hpp, 470
- Date
 - QuantLib::Date, 162
- date
 - QuantLib::CashFlow, 145
 - QuantLib::CashFlows::Coupon, 157
 - QuantLib::CashFlows::SimpleCashFlow, 345
 - QuantLib::Exercise, 187
 - QuantLib::History::Entry, 231
 - QuantLib::Scheduler, 340
- date.cpp, 471
- date.hpp, 472
- dates
 - QuantLib::Exercise, 187
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 316
- dates_
 - QuantLib::Exercise, 187
- Day
 - QuantLib, 83
- dayCount
 - QuantLib::DayCounter, 165
 - QuantLib::DayCounter::DayCounterImpl, 167
- DayCounter
 - QuantLib::DayCounter, 165
- dayCounter
 - QuantLib::CapFlatVolatilityStructure, 139
 - QuantLib::CapletForwardVolatility-Structure, 144
 - QuantLib::ForwardSpreadTerm-Structure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::Indexes::Xibor, 388
 - QuantLib::SwaptionVolatilityStructure, 371
 - QuantLib::TermStructure, 376
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 316
 - QuantLib::Volatilities::CapFlatVolatility-Vector, 140
 - QuantLib::Volatilities::SwaptionVolatility-Matrix, 370
 - QuantLib::ZeroSpreadTermStructure, 392
- daycounter.hpp, 473
- dayCounter_
 - QuantLib::CashFlows::Coupon, 158
- daycounters.cpp, 474
- daycounters.hpp, 475
- dayOfMonth
 - QuantLib::Date, 162
- dayOfYear
 - QuantLib::Date, 162
- dbegin
 - QuantLib::History, 228
- delta
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::OptionGreeks, 306
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::DiscreteGeometric-APO, 173
 - QuantLib::Pricers::DiscreteGeometric-ASO, 174
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdBsmOption, 194
 - QuantLib::Pricers::SingleAssetOption, 349
- delta_
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::FdBsmOption, 194
- DEM
 - QuantLib, 85
- dend
 - QuantLib::History, 228
- DepositRateHelper
 - QuantLib::TermStructures::DepositRate-Helper, 168
- Deprecated classes, 69
- derivative
 - QuantLib::ObjectiveFunction, 294
- DerivedMarketElement
 - QuantLib::DerivedMarketElement, 170
- description
 - QuantLib::Instrument, 242
- diagonal
 - QuantLib::Math::Matrix, 265
- diagonal_
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 384
- difference_type
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- diffusion

- QuantLib::BlackScholesProcess, 127
- QuantLib::DiffusionProcess, 171
- QuantLib::MonteCarlo::Path, 312
- QuantLib::OrnsteinUhlenbeckProcess, 309
- DiffusionProcess
 - QuantLib::DiffusionProcess, 171
- diffusionprocess.hpp, 476
- discount
 - QuantLib::TermStructure, 375
- discount_
 - QuantLib::MonteCarlo::PathPricer, 314
- discountBond
 - QuantLib::InterestRateModelling::Hull-White, 232
- discountBondOption
 - QuantLib::InterestRateModelling::Hull-White, 232
- DiscountFactor
 - QuantLib, 83
- discountGuess
 - QuantLib::TermStructures::DepositRateHelper, 168
 - QuantLib::TermStructures::FraRateHelper, 212
 - QuantLib::TermStructures::FuturesRateHelper, 214
 - QuantLib::TermStructures::RateHelper, 330
- discountImpl
 - QuantLib::ForwardRateStructure, 207
 - QuantLib::ImpliedTermStructure, 239
 - QuantLib::InterestRateModelling::ModelTermStructure, 282
 - QuantLib::TermStructure, 376
 - QuantLib::TermStructures::PiecewiseFlatForward, 317
 - QuantLib::ZeroYieldStructure, 394
- DiscreteGeometricAPO
 - QuantLib::Pricers::DiscreteGeometricAPO, 173
- discretegeometricapo.cpp, 477
- discretegeometricapo.hpp, 478
- DiscreteGeometricASO
 - QuantLib::Pricers::DiscreteGeometricASO, 174
- discretegeometricaso.cpp, 479
- discretegeometricaso.hpp, 480
- dividendRho
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::OptionGreeks, 306
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdDividendEuropeanOption, 196
 - QuantLib::Pricers::FdDividendShoutOption, 197
 - QuantLib::Pricers::SingleAssetOption, 349
- dividendRho_
 - QuantLib::Pricers::SingleAssetOption, 349
- dividendRhoComputed_
 - QuantLib::Pricers::SingleAssetOption, 350
- dividendYield
 - QuantLib::Instruments::PlainOptionParameters, 322
- dividendYield_
 - QuantLib::Pricers::SingleAssetOption, 349
- DKK
 - QuantLib, 85
- DMinus
 - QuantLib::FiniteDifferences::DMinus, 175
- dminus.hpp, 481
- DotProduct
 - QuantLib, 84
 - QuantLib::Array, 115
- downsideDeviation
 - QuantLib::Math::Statistics, 354
- downsideVariance
 - QuantLib::Math::Statistics, 354
- DPlus
 - QuantLib::FiniteDifferences::DPlus, 177
- dplus.hpp, 482
- DPlusDMinus
 - QuantLib::FiniteDifferences::DPlus-DMinus, 178
- dplusdminus.hpp, 483
- drift
 - QuantLib::BlackScholesProcess, 127
 - QuantLib::DiffusionProcess, 171
 - QuantLib::MonteCarlo::Path, 312
 - QuantLib::OrnsteinUhlenbeckProcess, 309
- dRMultiplier_
 - QuantLib::Pricers::SingleAssetOption, 350
- dt
 - QuantLib::TimeGrid, 378
- dt_
 - QuantLib::FiniteDifferences::MixedScheme, 278
- dVolMultiplier_
 - QuantLib::Pricers::SingleAssetOption, 350
- DZero
 - QuantLib::FiniteDifferences::DZero, 179
- dzero.hpp, 484
- easterMonday
 - QuantLib::Calendar::WesternCalendarImpl, 138
- EEK
 - QuantLib, 85

- eigenvalues
 - QuantLib::Math::SymmetricSchur-
Decomposition, 373
- eigenvectors
 - QuantLib::Math::SymmetricSchur-
Decomposition, 373
- end
 - QuantLib::Array, 115
 - QuantLib::History, 228
 - QuantLib::Math::Matrix, 264, 265
 - QuantLib::Scheduler, 340
- endCriteria
 - QuantLib::Optimization::Optimization-
Method, 301
- endCriteria_
 - QuantLib::Optimization::Optimization-
Method, 302
- endDate_
 - QuantLib::CashFlows::Coupon, 158
- endTime
 - QuantLib::Instruments::CapFloor-
Parameters, 141
- engine_
 - QuantLib::Option, 304
- Error
 - QuantLib::Error, 180
- errorEstimate
 - QuantLib::Math::Statistics, 354
 - QuantLib::Pricers::McPricer, 276
 - QuantLib::RiskStatistics, 337
- errors.hpp, 485
 - QL_ASSERT, 485
 - QL_ENSURE, 486
 - QL_REQUIRE, 485
- EUR
 - QuantLib, 85
- Euribor
 - QuantLib::Indexes::Euribor, 181
- euribor.hpp, 487
- europeanengine.cpp, 488
- europeanengine.hpp, 489
- EuropeanOption
 - QuantLib::Pricers::EuropeanOption, 184
- europeanoption.cpp, 490
- europeanoption.hpp, 491
- EuropeanPathPricer
 - QuantLib::MonteCarlo::EuropeanPath-
Pricer, 185
- europeanpathpricer.cpp, 492
- europeanpathpricer.hpp, 493
- evaluationNumber_
 - QuantLib::Solver1D, 351
- EverestPathPricer
 - QuantLib::MonteCarlo::EverestPathPricer,
186
- everestpathpricer.cpp, 494
- everestpathpricer.hpp, 495
- executeIntermediateStep
 - QuantLib::Pricers::FdBermudanOption,
193
- Exercise
 - QuantLib::Exercise, 187
- exerciseDates
 - QuantLib::Volatilities::SwaptionVolatility-
Matrix, 370
- ExercisePayoff
 - QuantLib::Pricers, 101
- exerciseTimes
 - QuantLib::Instruments::Swaption-
Parameters, 367
- exerciseType
 - QuantLib::Instruments::Swaption-
Parameters, 367
- exitFlag
 - QuantLib::Optimization::NonLinearLeast-
Square, 291
- Exp
 - QuantLib, 84
 - QuantLib::Array, 115
- expectation
 - QuantLib::BlackScholesProcess, 127
 - QuantLib::DiffusionProcess, 171
 - QuantLib::OrnsteinUhlenbeckProcess, 309
- expectedShortfall
 - QuantLib::Math::RiskMeasures, 336
 - QuantLib::RiskStatistics, 337
- expliciteuler.hpp, 496
- explicitPart_
 - QuantLib::FiniteDifferences::Mixed-
Scheme, 278
- expressiontemplates.hpp, 497
- ExtendedVasicek
 - QuantLib::InterestRate-
Modelling::ExtendedVasicek, 189
- extrapolate
 - QuantLib::Optimization::Simplex, 348
- extraTermInBermudan
 - QuantLib::Pricers::FdBermudanOption,
193
- f_
 - QuantLib::InterestRate-
Modelling::GeneralBlackKarasinski,
216
- fairRate
 - QuantLib::Instruments::SimpleSwap, 347

- QuantLib::Instruments::Swaption-Parameters, 367
- falseposition.cpp, 498
- falseposition.hpp, 499
- FdAmericanOption
 - QuantLib::Pricers::FdAmericanOption, 192
- FdBermudanOption
 - QuantLib::Pricers::FdBermudanOption, 193
- FdBsmOption
 - QuantLib::Pricers::FdBsmOption, 194
- FdDividendEuropeanOption
 - QuantLib::Pricers::FdDividendEuropeanOption, 196
- FdDividendShoutOption
 - QuantLib::Pricers::FdDividendShoutOption, 197
- FdEuropean
 - QuantLib::Pricers::FdEuropean, 198
- FdStepConditionOption
 - QuantLib::Pricers::FdStepConditionOption, 199
- fdtypedefs.hpp, 500
- FFObjFunction
 - QuantLib::TermStructures::PiecewiseFlatForward, 317
- filtering_iterator
 - QuantLib::Utilities::filtering_iterator, 200
- filteringiterator.hpp, 501
- findIndex
 - QuantLib::TimeGrid, 378
- finiteDifferenceEpsilon
 - QuantLib::Optimization::CostFunction, 154
- FiniteDifferenceModel
 - QuantLib::FiniteDifferences::FiniteDifferenceModel, 201
- finitedifferencemodel.hpp, 502
- FiniteDifferenceModel< CrankNicolson< Operator > >
 - QuantLib::FiniteDifferences::CrankNicolson, 159
- FiniteDifferenceModel< ExplicitEuler< Operator > >
 - QuantLib::FiniteDifferences::ExplicitEuler, 188
- FiniteDifferenceModel< ImplicitEuler< Operator > >
 - QuantLib::FiniteDifferences::ImplicitEuler, 237
- FiniteDifferenceModel< MixedScheme< Operator > >
 - QuantLib::FiniteDifferences::MixedScheme, 278
- finiteDifferenceOperator_
 - QuantLib::Pricers::FdBsmOption, 194
- first_argument_type
 - QuantLib::Math::BilinearInterpolation, 121
 - QuantLib::Math::Interpolation2D, 248
- firstDate
 - QuantLib::History, 227
- firstDerivativeAtCenter
 - QuantLib::FiniteDifferences, 91
- firstLeg_
 - QuantLib::Instruments::Swap, 362
- firstLegBPS
 - QuantLib::Instruments::Swap, 362
- firstLegBPS_
 - QuantLib::Instruments::Swap, 362
- fixedBPS
 - QuantLib::Instruments::SwaptionParameters, 367
- fixedCoupons
 - QuantLib::Instruments::SwaptionParameters, 367
- fixedDayCount_
 - QuantLib::TermStructures::SwapRateHelper, 364
- fixedFrequency_
 - QuantLib::TermStructures::SwapRateHelper, 364
- fixedIsAdjusted_
 - QuantLib::TermStructures::SwapRateHelper, 364
- fixedLeg
 - QuantLib::Instruments::SimpleSwap, 347
- fixedLegBPS
 - QuantLib::Instruments::SimpleSwap, 347
- fixedPayTimes
 - QuantLib::Instruments::SwaptionParameters, 367
- fixedRate
 - QuantLib::Instruments::SimpleSwap, 347
 - QuantLib::Instruments::SwaptionParameters, 367
- FixedRateCoupon
 - QuantLib::CashFlows::FixedRateCoupon, 202
- fixedratecoupon.hpp, 503
- FixedRateCouponVector
 - QuantLib::CashFlows::FixedRateCouponVector, 203
- fixing
 - QuantLib::CashFlows::FloatingRateCoupon, 204

- QuantLib::CashFlows::ShortFloatingRate-
Coupon, 343
- QuantLib::Index, 240
- QuantLib::Indexes::Xibor, 388
- fixingDays
 - QuantLib::CashFlows::FloatingRate-
Coupon, 204
- flatforward.hpp, 504
- floatingFrequency_
 - QuantLib::TermStructures::SwapRate-
Helper, 364
- floatingLeg
 - QuantLib::Instruments::SimpleSwap, 347
- floatingLegBPS
 - QuantLib::Instruments::SimpleSwap, 347
- floatingPayTimes
 - QuantLib::Instruments::Swaption-
Parameters, 367
- FloatingRateCoupon
 - QuantLib::CashFlows::FloatingRate-
Coupon, 204
- floatingratecoupon.cpp, 505
- floatingratecoupon.hpp, 506
- FloatingRateCouponVector
 - QuantLib::CashFlows::FloatingRate-
CouponVector, 206
- floatingResetTimes
 - QuantLib::Instruments::Swaption-
Parameters, 367
- floorRates
 - QuantLib::Instruments::CapFloor-
Parameters, 141
- Following
 - QuantLib, 85
- formula
 - QuantLib::InterestRateModelling::Black-
Model, 126
- forward
 - QuantLib::TermStructure, 375
- forwardImpl
 - QuantLib::DiscountStructure, 172
 - QuantLib::ForwardSpreadedTerm-
Structure, 210
 - QuantLib::TermStructure, 376
 - QuantLib::TermStructures::PiecewiseFlat-
Forward, 317
 - QuantLib::ZeroSpreadedTermStructure,
393
 - QuantLib::ZeroYieldStructure, 394
- forwards
 - QuantLib::Instruments::CapFloor-
Parameters, 141
- ForwardSpreadedTermStructure
 - QuantLib::ForwardSpreadedTerm-
Structure, 209
- Frankfurt
 - QuantLib::Calendars::Frankfurt, 211
- frankfurt.cpp, 507
- frankfurt.hpp, 508
- FraRateHelper
 - QuantLib::TermStructures::FraRateHelper,
212
- functionEvaluation
 - QuantLib::Optimization::Optimization-
Method, 301
- functionEvaluation_
 - QuantLib::Optimization::Optimization-
Method, 302
- functionValue
 - QuantLib::Optimization::Optimization-
Method, 301
- functionValue_
 - QuantLib::Optimization::Optimization-
Method, 302
- FuturesRateHelper
 - QuantLib::TermStructures::FuturesRate-
Helper, 214
- fxMax_
 - QuantLib::Solver1D, 351
- fxMin_
 - QuantLib::Solver1D, 351
- g2.hpp, 509
- gamma
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::OptionGreeks, 306
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::DiscreteGeometric-
APO, 173
 - QuantLib::Pricers::DiscreteGeometric-
ASO, 174
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdBsmOption, 194
 - QuantLib::Pricers::SingleAssetOption, 349
- gamma_
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::FdBsmOption, 194
- GaussianArrayGenerator
 - QuantLib::RandomNumbers, 102
- GaussianDistribution
 - QuantLib::Math, 94
- GaussianMultiPathGenerator
 - QuantLib::MonteCarlo, 97
- GaussianPathGenerator
 - QuantLib::MonteCarlo, 97

- GaussianRandomGenerator
 - QuantLib::RandomNumbers, [102](#)
- GBP
 - QuantLib, [85](#)
- GBPLibor
 - QuantLib::Indexes::GBPLibor, [215](#)
- gbplibor.hpp, [510](#)
- GeneralBlackKarasinski
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, [216](#)
- GeneralCoxIngersollRoss
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, [218](#)
- generateParameters
 - QuantLib::InterestRate-Modelling::ExtendedVasicek, [189](#)
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, [216](#)
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, [218](#)
 - QuantLib::InterestRateModelling::Hull-White, [232](#)
 - QuantLib::InterestRateModelling::Model, [280](#)
- GeometricAOPATHPricer
 - QuantLib::MonteCarlo::Geometric-AOPATHPricer, [220](#)
- geometricapopathpricer.cpp, [511](#)
- geometricapopathpricer.hpp, [512](#)
- GeometricASOPATHPricer
 - QuantLib::MonteCarlo::Geometric-ASOPATHPricer, [221](#)
- geometricasopathpricer.cpp, [513](#)
- geometricasopathpricer.hpp, [514](#)
- getCovariance
 - QuantLib::MonteCarlo, [97](#)
- getcovariance.cpp, [515](#)
- getGrid
 - QuantLib::Pricers::FdBsmOption, [194](#)
- getHistory
 - QuantLib::Indexes::XiborManager, [390](#)
- getPrices
 - QuantLib::Pricers::FdEuropean, [198](#)
- Global QuantLib macros, [70](#)
- gradient
 - QuantLib::Optimization::CostFunction, [154](#)
 - QuantLib::Optimization::LeastSquare-Function, [254](#)
 - QuantLib::Optimization::Optimization-Problem, [303](#)
- gradientEvaluation
 - QuantLib::Optimization::Optimization-Method, [301](#)
- gradientEvaluation_
 - QuantLib::Optimization::Optimization-Method, [302](#)
- gradientNormValue
 - QuantLib::Optimization::Optimization-Method, [301](#)
- GRD
 - QuantLib, [85](#)
- greeksCalculated_
 - QuantLib::Pricers::BarrierOption, [119](#)
- Grid
 - QuantLib::Grid, [222](#)
- grid.hpp, [516](#)
- grid_
 - QuantLib::Pricers::FdBsmOption, [194](#)
- gridPoints_
 - QuantLib::Pricers::FdBsmOption, [194](#)
- growthFactor
 - QuantLib, [84](#)
- Handle
 - QuantLib::Handle, [223](#), [224](#)
- handle.hpp, [517](#)
- HandleCopier
 - QuantLib::Handle, [223](#)
- hasBeenCalculated_
 - QuantLib::Pricers::SingleAssetOption, [349](#)
- hasHistory
 - QuantLib::Indexes::XiborManager, [390](#)
- Helsinki
 - QuantLib::Calendars::Helsinki, [225](#)
- helsinki.cpp, [518](#)
- helsinki.hpp, [519](#)
- HimalayaPathPricer
 - QuantLib::MonteCarlo::HimalayaPath-Pricer, [226](#)
- himalayapathpricer.cpp, [520](#)
- himalayapathpricer.hpp, [521](#)
- histories
 - QuantLib::Indexes::XiborManager, [390](#)
- History
 - QuantLib::History, [227](#), [228](#)
 - QuantLib::History::const_iterator, [230](#)
- history.hpp, [522](#)
- HKD
 - QuantLib, [85](#)
- HUF
 - QuantLib, [85](#)
- HullWhite

- QuantLib::InterestRateModelling::Hull-White, 232
- hullwhite.cpp, 523
- hullwhite.hpp, 524
- L
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- ICGaussianRng
 - QuantLib::RandomNumbers::ICGaussianRng, 234
- identity
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- IllegalArgumentError
 - QuantLib::IllegalArgumentError, 235
- IllegalResultError
 - QuantLib::IllegalResultError, 236
- impliciteuler.hpp, 525
- implicitPart_
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- impliedQuote
 - QuantLib::TermStructures::DepositRateHelper, 168
 - QuantLib::TermStructures::FraRateHelper, 212
 - QuantLib::TermStructures::FuturesRateHelper, 214
 - QuantLib::TermStructures::RateHelper, 330
 - QuantLib::TermStructures::SwapRateHelper, 364
- ImpliedTermStructure
 - QuantLib::ImpliedTermStructure, 238
- impliedVolatility
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::Pricers::SingleAssetOption, 350
- index
 - QuantLib::CashFlows::FloatingRate-Coupon, 204
 - QuantLib::CashFlows::ShortFloatingRate-Coupon, 343
- index.hpp, 526
- IndexError
 - QuantLib::IndexError, 241
- initializeGrid
 - QuantLib::Pricers::FdBsmOption, 194
- initializeInitialCondition
 - QuantLib::Pricers::FdBsmOption, 194
- initializeOperator
 - QuantLib::Pricers::FdBsmOption, 194
- initializeStepCondition
 - QuantLib::Pricers::FdAmericanOption, 192
 - QuantLib::Pricers::FdBermudanOption, 193
 - QuantLib::Pricers::FdDividendShout-Option, 197
 - QuantLib::Pricers::FdStepCondition-Option, 199
- initialPrices_
 - QuantLib::Pricers::FdBsmOption, 194
- initialValue_
 - QuantLib::Optimization::Optimization-Method, 302
- Input/output functions, 75
- Instrument
 - QuantLib::Instrument, 242
- instrument.hpp, 527
- Integer
 - QuantLib, 83
- Interpolation
 - QuantLib::Math::Interpolation, 246
- interpolation.hpp, 528
- Interpolation2D
 - QuantLib::Math::Interpolation2D, 248
- interpolation2D.hpp, 529
- inversecumgaussianrng.hpp, 530
- isAdjusted
 - QuantLib::Indexes::Xibor, 388
- isBusinessDay
 - QuantLib::Calendar, 135
 - QuantLib::Calendar::CalendarImpl, 137
- isExpired
 - QuantLib::Instrument, 242
- isExpired_
 - QuantLib::Instrument, 243
- isHoliday
 - QuantLib::Calendar, 135
- isinCode
 - QuantLib::Instrument, 242
- ISK
 - QuantLib, 85
- isLeap
 - QuantLib::Date, 163
- isNull
 - QuantLib::Handle, 223
 - QuantLib::Link, 259
 - QuantLib::RelinkableHandle, 332
- isRegular
 - QuantLib::Scheduler, 340
- isTimeDependent
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- iterationNumber

- QuantLib::Optimization::Optimization-Method, 301
- iterationNumber_
 - QuantLib::Optimization::Optimization-Method, 302
- iterationsNumber
 - QuantLib::Optimization::NonLinearLeast-Square, 291
- iterator
 - QuantLib::Array, 114
 - QuantLib::History, 228
 - QuantLib::Math::Matrix, 264
- Iterator support, 78
- iteratorcategories.hpp, 531
- iteratorMacros
 - QL_FULL_ITERATOR_SUPPORT, 78
- iteratorMacros
 - QL_ITERATOR, 78
 - QL_ITERATOR_TRAITS, 78
 - QL_REVERSE_ITERATOR, 78
 - QL_SPECIALIZE_ITERATOR_TRAITS, 78
- ITL
 - QuantLib, 85
- JamshidianSwaption
 - QuantLib::Pricers::JamshidianSwaption, 250
- jamshidianswaption.cpp, 532
- jamshidianswaption.hpp, 533
- Johannesburg
 - QuantLib::Calendars::Johannesburg, 251
- johannesburg.cpp, 534
- johannesburg.hpp, 535
- JPY
 - QuantLib, 85
- JPYLibor
 - QuantLib::Indexes::JPYLibor, 252
- jpylibor.hpp, 536
- k_
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, 218
- KnuthUniformRng
 - QuantLib::RandomNumbers::Knuth-UniformRng, 253
- knuthuniformrng.cpp, 537
- knuthuniformrng.hpp, 538
- KRW
 - QuantLib, 85
- kurtosis
 - QuantLib::Math::Statistics, 354
 - QuantLib::RiskStatistics, 337
- L_
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- lastDate
 - QuantLib::History, 227
- lastValue
 - QuantLib::Optimization::NonLinearLeast-Square, 291
- leastsquare.hpp, 539
- LeastSquareFunction
 - QuantLib::Optimization::LeastSquare-Function, 254
- LecuyerUniformRng
 - QuantLib::RandomNumbers::Lecuyer-UniformRng, 255
- lecuyeruniformrng.cpp, 540
- lecuyeruniformrng.hpp, 541
- length
 - QuantLib::Period, 315
- lengthInYears_
 - QuantLib::TermStructures::SwapRate-Helper, 364
- lengths
 - QuantLib::Volatilities::SwaptionVolatility-Matrix, 370
- LexicographicalView
 - QuantLib::Math::LexicographicalView, 256
- lexicographicalview.hpp, 542
- limitMacros
 - QL_MIN_POSITIVE_DOUBLE, 72
- limitMacros
 - QL_EPSILON, 72
 - QL_MAX_DOUBLE, 72
 - QL_MAX_INT, 72
 - QL_MIN_DOUBLE, 72
 - QL_MIN_INT, 72
- LinearInterpolation
 - QuantLib::Math::LinearInterpolation, 258
- linearinterpolation.hpp, 543
- linsearch.hpp, 544
- Link
 - QuantLib::Link, 259
- linkTo
 - QuantLib::Link, 259
 - QuantLib::RelinkableHandle, 332
- Log
 - QuantLib, 84
 - QuantLib::Array, 115
- London
 - QuantLib::Calendars::London, 261
- london.cpp, 545
- london.hpp, 546
- lowerBC_

- QuantLib::FiniteDifferences::Tridiagonal-Operator, [384](#)
- lsp_
 - QuantLib::Optimization::LeastSquare-Function, [254](#)
- LTL
 - QuantLib, [86](#)
- LVL
 - QuantLib, [86](#)
- macros
 - QL_DUMMY_RETURN, [70](#)
 - QL_HEX_VERSION, [70](#)
 - QL_TRACE_LEVEL, [70](#)
 - QL_VERSION, [70](#)
- make_combining_iterator
 - QuantLib::Utilities::combining_iterator, [150](#)
- make_coupling_iterator
 - QuantLib::Utilities::coupling_iterator, [156](#)
- make_filtering_iterator
 - QuantLib::Utilities::filtering_iterator, [200](#)
- make_processing_iterator
 - QuantLib::Utilities::processing_iterator, [327](#)
- make_stepping_iterator
 - QuantLib::Utilities::stepping_iterator, [359](#)
- marketelement.hpp, [547](#)
- Math functions, [71](#)
- mathf.cpp, [548](#)
- mathf.hpp, [549](#)
- mathMacros
 - QL_COS, [71](#)
 - QL_EXP, [71](#)
 - QL_FABS, [71](#)
 - QL_LOG, [71](#)
 - QL_MODF, [71](#)
 - QL_POW, [71](#)
 - QL_SIN, [71](#)
 - QL_SQRT, [71](#)
- Matrix
 - QuantLib::Math::Matrix, [264](#)
- matrix.cpp, [550](#)
- matrix.hpp, [551](#)
- matrixSqrt
 - QuantLib::Math, [95](#)
 - QuantLib::Math::Matrix, [265](#)
- maturity
 - QuantLib::Instruments::SimpleSwap, [347](#)
 - QuantLib::TermStructures::DepositRateHelper, [168](#)
 - QuantLib::TermStructures::FraRateHelper, [212](#)
 - QuantLib::TermStructures::FuturesRateHelper, [214](#)
 - QuantLib::TermStructures::RateHelper, [330](#)
 - QuantLib::TermStructures::SwapRateHelper, [364](#)
- max
 - QuantLib::Math::Statistics, [355](#)
 - QuantLib::RiskStatistics, [337](#)
- MAX_FUNCTION_EVALUATIONS
 - solver1d.hpp, [634](#)
- MaxBasketPathPricer
 - QuantLib::MonteCarlo::MaxBasketPathPricer, [267](#)
- maxbasketpathpricer.cpp, [552](#)
- maxbasketpathpricer.hpp, [553](#)
- maxDate
 - QuantLib::Date, [163](#)
 - QuantLib::ForwardSpreadedTermStructure, [209](#)
 - QuantLib::ImpliedTermStructure, [238](#)
 - QuantLib::TermStructure, [376](#)
 - QuantLib::TermStructures::PiecewiseFlatForward, [316](#)
 - QuantLib::ZeroSpreadedTermStructure, [392](#)
- maxEvaluations_
 - QuantLib::Solver1D, [351](#)
- maxTime
 - QuantLib::ForwardSpreadedTermStructure, [209](#)
 - QuantLib::ImpliedTermStructure, [238](#)
 - QuantLib::TermStructure, [376](#)
 - QuantLib::TermStructures::PiecewiseFlatForward, [316](#)
 - QuantLib::ZeroSpreadedTermStructure, [392](#)
- McBasket
 - QuantLib::Pricers::McBasket, [268](#)
- mcbasket.cpp, [554](#)
- McDiscreteArithmeticAPO
 - QuantLib::Pricers::McDiscreteArithmeticAPO, [269](#)
- mcdiscretearithmeticapo.cpp, [555](#)
- mcdiscretearithmeticapo.hpp, [556](#)
- McDiscreteArithmeticASO
 - QuantLib::Pricers::McDiscreteArithmeticASO, [270](#)
- mcdiscretearithmeticaso.cpp, [557](#)
- mcdiscretearithmeticaso.hpp, [558](#)
- McEuropean
 - QuantLib::Pricers::McEuropean, [271](#)
- mceuropean.cpp, [559](#)
- mceuropean.hpp, [560](#)

- McEverest
 - QuantLib::Pricers::McEverest, 272
- mceverest.cpp, 561
- mceverest.hpp, 562
- McHimalaya
 - QuantLib::Pricers::McHimalaya, 273
- mchimalaya.cpp, 563
- McMaxBasket
 - QuantLib::Pricers::McMaxBasket, 274
- mcmaxbasket.cpp, 564
- mcModel_
 - QuantLib::Pricers::McPricer, 276
- McPagoda
 - QuantLib::Pricers::McPagoda, 275
- mcpagoda.cpp, 565
- McPricer
 - QuantLib::Pricers::McPricer, 276
- mcpricer.hpp, 566
- mctypedefs.hpp, 567
- mean
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::Math::Statistics, 354
 - QuantLib::RiskStatistics, 337
- meanVector
 - QuantLib::Math::MultivariateAccumulator, 286
- method_
 - QuantLib::Optimization::Optimization-Problem, 303
- Milan
 - QuantLib::Calendars::Milan, 277
- milan.cpp, 568
- milan.hpp, 569
- min
 - QuantLib::Math::Statistics, 355
 - QuantLib::RiskStatistics, 337
- Min and max functions, 76
- minDate
 - QuantLib::Date, 163
 - QuantLib::ForwardSpreadedTermStructure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::TermStructure, 376
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 316
 - QuantLib::ZeroSpreadedTermStructure, 392
- minimize
 - QuantLib::Optimization::Conjugate-Gradient, 152
 - QuantLib::Optimization::Optimization-Method, 301
 - QuantLib::Optimization::Optimization-Problem, 303
 - QuantLib::Optimization::Simplex, 348
 - QuantLib::Optimization::SteepestDescent, 356
- minimumValue
 - QuantLib::Optimization::Optimization-Problem, 303
- minSample_
 - QuantLib::Pricers::McPricer, 276
- minTime
 - QuantLib::ForwardSpreadedTermStructure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::TermStructure, 376
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 317
 - QuantLib::ZeroSpreadedTermStructure, 392
- MixedScheme
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- mixedscheme.hpp, 570
- Model
 - QuantLib::InterestRateModelling::Model, 280
- model.cpp, 571
- model.hpp, 572
- model_
 - QuantLib::Pricers::CapFloorPricing-Engine, 142
 - QuantLib::Pricers::SwaptionPricing-Engine, 368
- ModelTermStructure
 - QuantLib::InterestRateModelling::Model-TermStructure, 282
- ModifiedFollowing
 - QuantLib, 85
- ModifiedPreceding
 - QuantLib, 85
- MonteCarloModel
 - QuantLib::MonteCarlo::MonteCarlo-Model, 283
- month
 - QuantLib::Date, 162
- MTL
 - QuantLib, 86
- MultiFactorMonteCarloOption
 - QuantLib::MonteCarlo, 97
- MultiPath
 - QuantLib::MonteCarlo::MultiPath, 284
- MultiPathGenerator
 - QuantLib::MonteCarlo::MultiPath-Generator, 285

- multipathgenerator.hpp, 573
- MultivariateAccumulator
 - QuantLib::Math::MultivariateAccumulator, 286
- multivariateaccumulator.cpp, 574
- multivariateaccumulator.hpp, 575
- name
 - QuantLib::Calendar, 135
 - QuantLib::Calendar::CalendarImpl, 137
 - QuantLib::DayCounter, 166
 - QuantLib::DayCounter::DayCounterImpl, 167
 - QuantLib::Index, 240
 - QuantLib::Indexes::Xibor, 388
- newton.cpp, 576
- newton.hpp, 577
- newtonsafe.cpp, 578
- newtonsafe.hpp, 579
- NewYork
 - QuantLib::Calendars::NewYork, 290
- newyork.cpp, 580
- newyork.hpp, 581
- next
 - QuantLib::MonteCarlo::MultiPath-Generator, 285
 - QuantLib::MonteCarlo::PathGenerator, 313
 - QuantLib::RandomNumbers::BoxMuller-GaussianRng, 130
 - QuantLib::RandomNumbers::CLGaussianRng, 147
 - QuantLib::RandomNumbers::ICGaussianRng, 234
 - QuantLib::RandomNumbers::Knuth-UniformRng, 253
 - QuantLib::RandomNumbers::Lecuyer-UniformRng, 255
 - QuantLib::RandomNumbers::Random-ArrayGenerator, 328
- node.hpp, 582
- NOK
 - QuantLib, 86
- nominal
 - QuantLib::CashFlows::Coupon, 157
 - QuantLib::Instruments::SimpleSwap, 347
- nominal_
 - QuantLib::CashFlows::Coupon, 157
- nominals
 - QuantLib::Instruments::CapFloor-Parameters, 141
 - QuantLib::Instruments::Swaption-Parameters, 367
- NonLinearLeastSquare
 - QuantLib::Optimization::NonLinearLeast-Square, 291
- normaldistribution.cpp, 583
- normaldistribution.hpp, 584
- notifyObservers
 - QuantLib::Patterns::Observable, 296
- NPV
 - QuantLib::Instrument, 242
- NPV_
 - QuantLib::Instrument, 243
- Null
 - QuantLib::Null, 293
- null.hpp, 585
- Numeric limits, 72
- numericalmethod.hpp, 586
- NZD
 - QuantLib, 86
- observable.hpp, 587
- Observer
 - QuantLib::Patterns::Observable, 295
 - QuantLib::Patterns::Observer, 297
- OneFactorModel
 - QuantLib::InterestRateModelling::One-FactorModel, 299
- onefactormodel.cpp, 588
- onefactormodel.hpp, 589
- OneFactorMonteCarloOption
 - QuantLib::MonteCarlo, 97
- OneFactorOperator
 - QuantLib::FiniteDifferences::OneFactor-Operator, 300
- onefactoroperator.cpp, 590
- onefactoroperator.hpp, 591
- operator *
 - QuantLib, 84
 - QuantLib::Array, 115
 - QuantLib::FiniteDifferences, 91
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 384
 - QuantLib::Handle, 223
 - QuantLib::History::const_iterator, 230
 - QuantLib::Math, 95
 - QuantLib::Math::Matrix, 265
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::filtering_iterator, 200
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator *=
 - QuantLib::Array, 115
 - QuantLib::Math::Matrix, 264

- QuantLib::Array, 114, 115
- QuantLib::Date, 162
- QuantLib::History::const_iterator, 230
- QuantLib::Math::Matrix, 264
- QuantLib::Utilities::combining_iterator, 149
- QuantLib::Utilities::coupling_iterator, 155
- QuantLib::Utilities::processing_iterator, 326
- QuantLib::Utilities::stepping_iterator, 358
- operator->
 - QuantLib::Handle, 223
 - QuantLib::History::const_iterator, 230
 - QuantLib::RelinkableHandle, 332
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::filtering_iterator, 200
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator/
 - QuantLib, 84
 - QuantLib::Array, 115
 - QuantLib::FiniteDifferences, 91
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 384
 - QuantLib::Math, 95
 - QuantLib::Math::Matrix, 265
- operator/=
 - QuantLib::Array, 115
 - QuantLib::Math::Matrix, 264
- operator<
 - QuantLib, 84
 - QuantLib::Date, 163
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator<<
 - QuantLib, 84
 - QuantLib::Date, 163
- operator<=
 - QuantLib, 84
 - QuantLib::Date, 163
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator=
 - QuantLib::Array, 114
 - QuantLib::Handle, 223
 - QuantLib::Math::Matrix, 264
 - QuantLib::Patterns::Observer, 297
- operator==
 - QuantLib, 84
 - QuantLib::Calendar, 135
 - QuantLib::Date, 163
 - QuantLib::DayCounter, 166
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::filtering_iterator, 200
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator>
 - QuantLib, 84
 - QuantLib::Date, 163
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator>=
 - QuantLib, 84
 - QuantLib::Date, 163
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operator[]
 - QuantLib::Array, 115
 - QuantLib::History, 227
 - QuantLib::History::const_iterator, 230
 - QuantLib::Math::LexicographicalView, 256
 - QuantLib::Math::Matrix, 265
 - QuantLib::MonteCarlo::MultiPath, 284
 - QuantLib::MonteCarlo::Path, 312
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- operatorType
 - QuantLib::FiniteDifferences::Finite-DifferenceModel, 201
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- optimisationMethod
 - QuantLib::Optimization::Optimization-Problem, 303
- OptimizationMethod
 - QuantLib::Optimization::Optimization-Method, 301
- OptimizationProblem

- QuantLib::Optimization::Optimization-Problem, 303
- optimizer.hpp, 592
- Option
 - QuantLib::Option, 304
- option.cpp, 593
- option.hpp, 594
- OptionGreeks
 - QuantLib::OptionGreeks, 306
- Option Value
 - QuantLib::Option Value, 308
- OrnsteinUhlenbeckProcess
 - QuantLib::OrnsteinUhlenbeckProcess, 309
- outerProduct
 - QuantLib::Math, 95
 - QuantLib::Math::Matrix, 265
- OutOfMemoryError
 - QuantLib::OutOfMemoryError, 310
- PagodaPathPricer
 - QuantLib::MonteCarlo::PagodaPathPricer, 311
- pagodapathpricer.cpp, 595
- pagodapathpricer.hpp, 596
- parameter.hpp, 597
- parameters
 - QuantLib::OptionPricingEngine, 307
 - QuantLib::Pricers::CapFloorPricingEngine, 142
 - QuantLib::Pricers::PlainOptionEngine, 321
 - QuantLib::Pricers::SwaptionPricingEngine, 368
- parameters_
 - QuantLib::InterestRateModelling::Model, 280
 - QuantLib::Pricers::CapFloorPricingEngine, 142
 - QuantLib::Pricers::PlainOptionEngine, 321
 - QuantLib::Pricers::SwaptionPricingEngine, 368
- Path
 - QuantLib::MonteCarlo::Path, 312
- PathGenerator
 - QuantLib::MonteCarlo::PathGenerator, 313
- PathPricer
 - QuantLib::MonteCarlo::PathPricer, 314
- pathpricer.hpp, 598
- pathSize
 - QuantLib::MonteCarlo::MultiPath, 284
- payFixed
 - QuantLib::Instruments::SwaptionParameters, 367
- payFixedRate
 - QuantLib::Instruments::SimpleSwap, 347
- perform
 - QuantLib::Optimization::NonLinearLeastSquare, 291
- performCalculations
 - QuantLib::Instrument, 243
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::Instruments::Stock, 360
 - QuantLib::Instruments::Swap, 362
 - QuantLib::Instruments::Swaption, 366
 - QuantLib::Option, 304
- Period
 - QuantLib::Period, 315
- phi_
 - QuantLib::InterestRateModelling::ExtendedVasicek, 189
 - QuantLib::InterestRateModelling::GeneralCoxIngersollRoss, 218
- PiecewiseFlatForward
 - QuantLib::TermStructures::PiecewiseFlatForward, 316
- piecewiseflatforward.cpp, 599
- piecewiseflatforward.hpp, 600
- PlainOption
 - QuantLib::Instruments::PlainOption, 319
- plainoption.cpp, 601
- plainoption.hpp, 602
- PlainOptionParameters
 - QuantLib::Instruments::PlainOptionParameters, 322
- plus
 - QuantLib::Date, 163
- plusDays
 - QuantLib::Date, 163
- plusMonths
 - QuantLib::Date, 163
- plusWeeks
 - QuantLib::Date, 163
- plusYears
 - QuantLib::Date, 163
- PLZ
 - QuantLib, 86
- pointer
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::filtering_iterator, 200
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- PostconditionNotSatisfiedError

- QuantLib::PostconditionNotSatisfiedError, 324
- potentialUpside
 - QuantLib::Math::RiskMeasures, 336
 - QuantLib::RiskStatistics, 337
- Preceding
 - QuantLib, 85
- PreconditionNotSatisfiedError
 - QuantLib::PreconditionNotSatisfiedError, 325
- process
 - QuantLib::InterestRate-Modelling::ExtendedVasicek, 189
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, 216
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, 218
 - QuantLib::InterestRateModelling::One-FactorModel, 299
- processing_iterator
 - QuantLib::Utilities::processing_iterator, 326
- processingiterator.hpp, 603
- QL_ALLOW_TEMPLATE_METHOD_CALLS
 - templateMacros, 77
- QL_ASSERT
 - errors.hpp, 485
- QL_CLOCK
 - timeMacros, 73
- QL_COS
 - mathMacros, 71
- QL_DECLARE_TEMPLATE_-SPECIALIZATIONS
 - templateMacros, 77
- QL_DUMMY_RETURN
 - macros, 70
- QL_ENSURE
 - errors.hpp, 486
- QL_EPSILON
 - limitMacros, 72
- QL_EXP
 - mathMacros, 71
- QL_EXPRESSION_TEMPLATES_WORK
 - templateMacros, 77
- QL_FABS
 - mathMacros, 71
- QL_FULL_ITERATOR_SUPPORT
 - iteratorMacros, 78
 - qldefines.hpp, 605
- QL_HEX_VERSION
 - macros, 70
- qldefines.hpp, 604
- QL_ITERATOR
 - iteratorMacros, 78
- QL_ITERATOR_TRAITS
 - iteratorMacros, 78
- QL_LOG
 - mathMacros, 71
- QL_MAX
 - algoMacros, 76
- QL_MAX_DOUBLE
 - limitMacros, 72
- QL_MAX_INT
 - limitMacros, 72
- QL_MAX_VOLATILITY
 - singleassetoption.hpp, 632
- QL_MIN
 - algoMacros, 76
- QL_MIN_DOUBLE
 - limitMacros, 72
- QL_MIN_INT
 - limitMacros, 72
- QL_MIN_POSITIVE_DOUBLE
 - limitMacros, 72
- qldefines.hpp, 604
- QL_MIN_VOLATILITY
 - singleassetoption.hpp, 632
- QL_MODF
 - mathMacros, 71
- QL_POW
 - mathMacros, 71
- QL_REQUIRE
 - errors.hpp, 485
- QL_REVERSE_ITERATOR
 - iteratorMacros, 78
 - QuantLib::Array, 114
 - QuantLib::Math::LexicographicalView, 256
 - QuantLib::Math::Matrix, 264
- QL_SIN
 - mathMacros, 71
- QL_SPECIALIZE_ITERATOR_TRAITS
 - iteratorMacros, 78
- QL_SPRINTF
 - stdioMacros, 75
- QL_SQRT
 - mathMacros, 71
- QL_STRLEN
 - charMacros, 74
- QL_TEMPLATE_METAPROGRAMMING_-WORKS
 - templateMacros, 77
- QL_TIME
 - timeMacros, 73
- QL_TOLOWER

- charMacros, 74
- QL_Toupper
 - charMacros, 74
- QL_TRACE_LEVEL
 - macros, 70
 - qldefines.hpp, 604
- QL_VERSION
 - macros, 70
 - qldefines.hpp, 604
- qldefines.hpp, 604
 - QL_FULL_ITERATOR_SUPPORT, 605
 - QL_HEX_VERSION, 604
 - QL_MIN_POSITIVE_DOUBLE, 604
 - QL_TRACE_LEVEL, 604
 - QL_VERSION, 604
- QuantLib
 - Abs, 84
 - AUD, 85
 - BGL, 85
 - CAD, 85
 - CHF, 85
 - CYP, 85
 - CZK, 85
 - Day, 83
 - DEM, 85
 - DiscountFactor, 83
 - DKK, 85
 - DotProduct, 84
 - EEK, 85
 - EUR, 85
 - Exp, 84
 - Following, 85
 - GBP, 85
 - GRD, 85
 - growthFactor, 84
 - HKD, 85
 - HUF, 85
 - Integer, 83
 - ISK, 85
 - ITL, 85
 - JPY, 85
 - KRW, 85
 - Log, 84
 - LTL, 86
 - LVL, 86
 - ModifiedFollowing, 85
 - ModifiedPreceding, 85
 - MTL, 86
 - NOK, 86
 - NZD, 86
 - operator *, 84
 - operator
 - =, 84
 - operator+, 84
 - operator-, 84
 - operator/, 84
 - operator<, 84
 - operator<=, 84
 - operator<<, 84
 - operator<=, 84
 - operator==, 84
 - operator>, 84
 - operator>=, 84
 - PLZ, 86
 - Preceding, 85
 - Rate, 83
 - Real, 83
 - ROL, 86
 - SEK, 86
 - SGD, 86
 - SIT, 86
 - Size, 83
 - SKK, 86
 - Spread, 83
 - Sqrt, 84
 - Time, 83
 - TRL, 86
 - USD, 86
 - Year, 83
 - ZAR, 86
- QuantLib, 79
 - Currency, 85
 - RollingConvention, 85
 - Weekday, 86
- QuantLib::Arguments
 - ~Arguments, 111
- QuantLib::Arguments, 111
- QuantLib::Array
 - ~Array, 114
 - Abs, 115
 - Array, 114
 - begin, 115
 - const_iterator, 114
 - DotProduct, 115
 - end, 115
 - Exp, 115
 - iterator, 114
 - Log, 115
 - operator *, 115
 - operator *=, 115
 - operator+, 115
 - operator+=, 114
 - operator-, 115
 - operator=, 114, 115
 - operator/, 115
 - operator/=, 115
 - operator=, 114
 - QL_REVERSE_ITERATOR, 114
 - rbegin, 115

- rend, 115
- size, 115
- Sqrt, 115
- QuantLib::Array, 114
- QuantLib::AssertionFailedError
 - AssertionFailedError, 117
- QuantLib::AssertionFailedError, 117
- QuantLib::BlackScholesProcess
 - BlackScholesProcess, 127
 - diffusion, 127
 - drift, 127
 - expectation, 127
 - variance, 127
- QuantLib::BlackScholesProcess, 127
- QuantLib::Calendar
 - operator
 - =, 134
- QuantLib::Calendar, 134
 - advance, 135
 - Calendar, 135
 - isBusinessDay, 135
 - isHoliday, 135
 - name, 135
 - operator==, 135
 - roll, 135
- QuantLib::Calendar::CalendarImpl
 - isBusinessDay, 137
 - name, 137
- QuantLib::Calendar::CalendarImpl, 137
- QuantLib::Calendar::WesternCalendarImpl
 - easterMonday, 138
- QuantLib::Calendar::WesternCalendarImpl, 138
- QuantLib::Calendars, 87
- QuantLib::Calendars::Frankfurt
 - Frankfurt, 211
- QuantLib::Calendars::Frankfurt, 211
- QuantLib::Calendars::Helsinki
 - Helsinki, 225
- QuantLib::Calendars::Helsinki, 225
- QuantLib::Calendars::Johannesburg
 - Johannesburg, 251
- QuantLib::Calendars::Johannesburg, 251
- QuantLib::Calendars::London
 - London, 261
- QuantLib::Calendars::London, 261
- QuantLib::Calendars::Milan
 - Milan, 277
- QuantLib::Calendars::Milan, 277
- QuantLib::Calendars::NewYork
 - NewYork, 290
- QuantLib::Calendars::NewYork, 290
- QuantLib::Calendars::Sydney
 - Sydney, 372
- QuantLib::Calendars::Sydney, 372
- QuantLib::Calendars::TARGET
 - TARGET, 374
- QuantLib::Calendars::TARGET, 374
- QuantLib::Calendars::Tokyo
 - Tokyo, 379
- QuantLib::Calendars::Tokyo, 379
- QuantLib::Calendars::Toronto
 - Toronto, 380
- QuantLib::Calendars::Toronto, 380
- QuantLib::Calendars::Wellington
 - Wellington, 387
- QuantLib::Calendars::Wellington, 387
- QuantLib::Calendars::Zurich
 - Zurich, 396
- QuantLib::Calendars::Zurich, 396
- QuantLib::CapFlatVolatilityStructure
 - ~CapFlatVolatilityStructure, 139
 - dayCounter, 139
 - settlementDate, 139
 - todaysDate, 139
 - volatility, 139
 - volatilityImpl, 139
- QuantLib::CapFlatVolatilityStructure, 139
- QuantLib::CapletForwardVolatilityStructure
 - ~CapletForwardVolatilityStructure, 144
 - dayCounter, 144
 - settlementDate, 144
 - todaysDate, 144
 - volatility, 144
 - volatilityImpl, 144
- QuantLib::CapletForwardVolatilityStructure, 144
- QuantLib::CashFlow
 - ~CashFlow, 145
 - date, 145
- QuantLib::CashFlow, 145
 - amount, 145
- QuantLib::CashFlows, 88
- QuantLib::CashFlows::Coupon
 - accrualDays, 157
 - accrualEndDate, 157
 - accrualPeriod, 157
 - accrualStartDate, 157
 - accruedAmount, 157
 - calendar_, 158
 - Coupon, 157
 - date, 157
 - dayCounter_, 158
 - endDate_, 158
 - nominal, 157
 - nominal_, 157
 - refPeriodEnd_, 158
 - refPeriodStart_, 158
 - rollingConvention_, 158

- startDate_, 157
- QuantLib::CashFlows::Coupon, 157
- QuantLib::CashFlows::FixedRateCoupon
 - accruedAmount, 202
 - FixedRateCoupon, 202
 - rate, 202
- QuantLib::CashFlows::FixedRateCoupon, 202
 - amount, 202
- QuantLib::CashFlows::FixedRateCouponVector
 - FixedRateCouponVector, 203
- QuantLib::CashFlows::FixedRateCoupon-Vector, 203
- QuantLib::CashFlows::FloatingRateCoupon
 - accruedAmount, 204
 - fixing, 204
 - fixingDays, 204
 - FloatingRateCoupon, 204
 - index, 204
 - spread, 204
- QuantLib::CashFlows::FloatingRateCoupon, 204
 - amount, 205
 - update, 205
- QuantLib::CashFlows::FloatingRateCouponVector
 - FloatingRateCouponVector, 206
- QuantLib::CashFlows::FloatingRateCoupon-Vector, 206
- QuantLib::CashFlows::ShortFloatingRateCoupon
 - accruedAmount, 343
 - fixing, 343
 - index, 343
 - ShortFloatingRateCoupon, 343
 - spread, 343
- QuantLib::CashFlows::ShortFloatingRate-Coupon, 343
 - amount, 344
 - update, 344
- QuantLib::CashFlows::SimpleCashFlow
 - date, 345
 - SimpleCashFlow, 345
- QuantLib::CashFlows::SimpleCashFlow, 345
 - amount, 345
- QuantLib::CompositeMarketElement
 - CompositeMarketElement, 151
 - value, 151
- QuantLib::CompositeMarketElement, 151
 - update, 151
- QuantLib::CurrencyFormatter
 - toString, 161
- QuantLib::CurrencyFormatter, 161
- QuantLib::Date
 - Date, 162
 - dayOfMonth, 162
 - dayOfYear, 162
 - isLeap, 163
 - maxDate, 163
 - minDate, 163
 - month, 162
 - operator
 - =, 163
 - operator+, 162
 - operator++, 162
 - operator+=, 162
 - operator-, 163
 - operator-, 162
 - operator=, 162
 - operator<, 163
 - operator<<, 163
 - operator<=, 163
 - operator==, 163
 - operator>, 163
 - operator>=, 163
 - plus, 163
 - plusDays, 163
 - plusMonths, 163
 - plusWeeks, 163
 - plusYears, 163
 - serialNumber, 162
 - weekday, 162
 - year, 162
- QuantLib::Date, 162
- QuantLib::DateFormatter
 - toString, 164
- QuantLib::DateFormatter, 164
- QuantLib::DayCounter
 - dayCount, 165
 - operator
 - =, 165
 - yearFraction, 165
- QuantLib::DayCounter, 165
 - DayCounter, 165
 - name, 166
 - operator==, 166
- QuantLib::DayCounter::DayCounterImpl
 - dayCount, 167
 - name, 167
 - yearFraction, 167
- QuantLib::DayCounter::DayCounterImpl, 167
- QuantLib::DayCounters, 89
- QuantLib::DayCounters::Actual360
 - Actual360, 107
- QuantLib::DayCounters::Actual360, 107
- QuantLib::DayCounters::Actual365
 - Actual365, 108
- QuantLib::DayCounters::Actual365, 108
- QuantLib::DayCounters::ActualActual
 - ActualActual, 109
- QuantLib::DayCounters::ActualActual, 109

- QuantLib::DayCounters::Thirty360
 - Thirty360, 377
- QuantLib::DayCounters::Thirty360, 377
- QuantLib::DerivedMarketElement
 - DerivedMarketElement, 170
 - value, 170
- QuantLib::DerivedMarketElement, 170
 - update, 170
- QuantLib::DiffusionProcess
 - ~DiffusionProcess, 171
 - diffusion, 171
 - DiffusionProcess, 171
 - drift, 171
 - expectation, 171
 - variance, 171
 - x0, 171
- QuantLib::DiffusionProcess, 171
- QuantLib::DiscountStructure
 - ~DiscountStructure, 172
- QuantLib::DiscountStructure, 172
 - forwardImpl, 172
 - zeroYieldImpl, 172
- QuantLib::DoubleFormatter
 - toString, 176
- QuantLib::DoubleFormatter, 176
- QuantLib::Error
 - ~Error, 180
 - Error, 180
 - what, 180
- QuantLib::Error, 180
- QuantLib::EuroFormatter
 - toString, 182
- QuantLib::EuroFormatter, 182
- QuantLib::Exercise
 - calendar, 187
 - calendar_, 187
 - convention_, 187
 - date, 187
 - dates, 187
 - dates_, 187
 - Exercise, 187
 - rollingConvention, 187
 - settlementDays, 187
 - settlementDays_, 187
 - type, 187
 - type_, 187
- QuantLib::Exercise, 187
- QuantLib::FiniteDifferences
 - firstDerivativeAtCenter, 91
 - operator *, 91
 - operator+, 91
 - operator-, 91
 - operator/, 91
 - secondDerivativeAtCenter, 91
 - StandardFiniteDifferenceModel, 91
 - StandardStepCondition, 91
 - valueAtCenter, 91
- QuantLib::FiniteDifferences, 90
- QuantLib::FiniteDifferences::BoundaryCondition
 - BoundaryCondition, 129
 - type, 129
 - value, 129
- QuantLib::FiniteDifferences::Boundary-Condition, 129
- QuantLib::FiniteDifferences::BSMOperator
 - BSMOperator, 132
- QuantLib::FiniteDifferences::BSMOperator, 132
- QuantLib::FiniteDifferences::CrankNicolson
 - FiniteDifferenceModel< CrankNicolson< Operator > >, 159
- QuantLib::FiniteDifferences::CrankNicolson, 159
- QuantLib::FiniteDifferences::DMinus
 - DMinus, 175
- QuantLib::FiniteDifferences::DMinus, 175
- QuantLib::FiniteDifferences::DPlus
 - DPlus, 177
- QuantLib::FiniteDifferences::DPlus, 177
- QuantLib::FiniteDifferences::DPlusDMinus
 - DPlusDMinus, 178
- QuantLib::FiniteDifferences::DPlusDMinus, 178
- QuantLib::FiniteDifferences::DZero
 - DZero, 179
- QuantLib::FiniteDifferences::DZero, 179
- QuantLib::FiniteDifferences::ExplicitEuler
 - FiniteDifferenceModel< ExplicitEuler< Operator > >, 188
- QuantLib::FiniteDifferences::ExplicitEuler, 188
- QuantLib::FiniteDifferences::FiniteDifferenceModel
 - arrayType, 201
 - FiniteDifferenceModel, 201
 - operatorType, 201
- QuantLib::FiniteDifferences::FiniteDifference-Model, 201
 - rollback, 201
- QuantLib::FiniteDifferences::ImplicitEuler
 - FiniteDifferenceModel< ImplicitEuler< Operator > >, 237
- QuantLib::FiniteDifferences::ImplicitEuler, 237
- QuantLib::FiniteDifferences::MixedScheme
 - arrayType, 278
 - dt_, 278
 - explicitPart_, 278
 - FiniteDifferenceModel< MixedScheme< Operator > >, 278
 - L, 278

- implicitPart_, 278
- L_, 278
- MixedScheme, 278
- operatorType, 278
- setStep, 278
- step, 278
- theta_, 278
- QuantLib::FiniteDifferences::MixedScheme, 278
- QuantLib::FiniteDifferences::OneFactorOperator
 - ~OneFactorOperator, 300
 - OneFactorOperator, 300
- QuantLib::FiniteDifferences::OneFactorOperator, 300
- QuantLib::FiniteDifferences::StepCondition
 - ~StepCondition, 357
 - applyTo, 357
- QuantLib::FiniteDifferences::StepCondition, 357
- QuantLib::FiniteDifferences::TridiagonalOperator
 - aboveDiagonal_, 384
 - applyTo, 383
 - arrayType, 383
 - belowDiagonal_, 384
 - diagonal_, 384
 - identity, 383
 - isTimeDependent, 383
 - lowerBC_, 384
 - operator *, 384
 - operator+, 384
 - operator-, 384
 - operator/, 384
 - setFirstRow, 383
 - setLastRow, 383
 - setLowerBC, 383
 - setMidRow, 383
 - setMidRows, 383
 - setTime, 383
 - setUpperBC, 383
 - size, 383
 - solveFor, 383
 - SOR, 383
 - timeSetter_, 384
 - TridiagonalOperator, 383
 - upperBC_, 384
- QuantLib::FiniteDifferences::TridiagonalOperator, 383
- QuantLib::FiniteDifferences::TridiagonalOperator::TimeSetter
 - ~TimeSetter, 385
 - setTime, 385
- QuantLib::FiniteDifferences::TridiagonalOperator::TimeSetter, 385
- QuantLib::ForwardRateStructure
 - ~ForwardRateStructure, 207
- QuantLib::ForwardRateStructure, 207
 - discountImpl, 207
 - zeroYieldImpl, 207
- QuantLib::ForwardSpreadedTermStructure
 - calendar, 209
 - currency, 209
 - dayCounter, 209
 - forwardImpl, 210
 - ForwardSpreadedTermStructure, 209
 - maxDate, 209
 - maxTime, 209
 - minDate, 209
 - minTime, 209
 - settlementDate, 209
 - settlementDays, 209
 - todaysDate, 209
- QuantLib::ForwardSpreadedTermStructure, 209
 - update, 210
 - zeroYieldImpl, 210
- QuantLib::Grid
 - Grid, 222
- QuantLib::Grid, 222
- QuantLib::Handle
 - ~Handle, 223
 - Handle, 223
 - HandleCopier, 223
 - isNull, 223
 - operator *, 223
 - operator \rightarrow , 223
 - operator=, 223
 - shareSameObject, 223
- QuantLib::Handle, 223
 - Handle, 224
- QuantLib::History
 - begin, 228
 - const_data_iterator, 227
 - const_valid_data_iterator, 227
 - const_valid_iterator, 227
 - data_iterator, 228
 - dbegin, 228
 - dend, 228
 - end, 228
 - firstDate, 227
 - History, 227
 - iterator, 228
 - lastDate, 227
 - size, 227
 - valid_data_iterator, 228
 - valid_iterator, 228
 - vbegin, 228
 - vdbegin, 228
 - vdend, 228
 - vend, 228
- QuantLib::History, 227

- History, 228
- QuantLib::History::const_iterator
 - difference_type, 230
 - History, 230
 - operator *, 230
 - operator
 - =, 230
 - operator+, 230
 - operator++, 230
 - operator+=, 230
 - operator-, 230
 - operator-, 230
 - operator=, 230
 - operator → , 230
 - operator<, 230
 - operator<=, 230
 - operator==, 230
 - operator>, 230
 - operator>=, 230
 - pointer, 230
 - reference, 230
 - value_type, 230
- QuantLib::History::const_iterator, 230
- QuantLib::History::Entry
 - const_iterator, 231
 - date, 231
 - value, 231
- QuantLib::History::Entry, 231
- QuantLib::IllegalArgumentError
 - IllegalArgumentError, 235
- QuantLib::IllegalArgumentError, 235
- QuantLib::IllegalResultError
 - IllegalResultError, 236
- QuantLib::IllegalResultError, 236
- QuantLib::ImpliedTermStructure
 - calendar, 238
 - currency, 238
 - dayCounter, 238
 - discountImpl, 239
 - ImpliedTermStructure, 238
 - maxDate, 238
 - maxTime, 238
 - minDate, 238
 - minTime, 238
 - settlementDate, 238
 - settlementDays, 238
 - todayDate, 238
- QuantLib::ImpliedTermStructure, 238
- QuantLib::Index
 - ~Index, 240
- QuantLib::Index, 240
 - fixing, 240
 - name, 240
- QuantLib::IndexError
 - IndexError, 241
- QuantLib::IndexError, 241
- QuantLib::Indexes, 92
- QuantLib::Indexes::AUDLibor
 - AUDLibor, 118
- QuantLib::Indexes::AUDLibor, 118
- QuantLib::Indexes::CADLibor
 - CADLibor, 133
- QuantLib::Indexes::CADLibor, 133
- QuantLib::Indexes::CHFLibor
 - CHFLibor, 146
- QuantLib::Indexes::CHFLibor, 146
- QuantLib::Indexes::Euribor
 - Euribor, 181
- QuantLib::Indexes::Euribor, 181
- QuantLib::Indexes::GBPLibor
 - GBPLibor, 215
- QuantLib::Indexes::GBPLibor, 215
- QuantLib::Indexes::JPYLibor
 - JPYLibor, 252
- QuantLib::Indexes::JPYLibor, 252
- QuantLib::Indexes::USDLibor
 - USDLibor, 386
- QuantLib::Indexes::USDLibor, 386
- QuantLib::Indexes::Xibor
 - calendar, 388
 - currency, 388
 - dayCounter, 388
 - isAdjusted, 388
 - rollingConvention, 388
 - settlementDays, 388
 - tenor, 388
 - Xibor, 388
- QuantLib::Indexes::Xibor, 388
- QuantLib::Indexes::XiborManager
 - fixing, 388
 - name, 388
- QuantLib::Indexes::XiborManager, 390
 - getHistory, 390
 - hasHistory, 390
 - histories, 390
 - setHistory, 390
- QuantLib::Indexes::XiborManager, 390
- QuantLib::Indexes::ZARLibor
 - ZARLibor, 391
- QuantLib::Indexes::ZARLibor, 391
- QuantLib::Instrument
 - ~Instrument, 242
 - description, 242
 - Instrument, 242
 - isExpired, 242
 - isExpired_, 243
 - isinCode, 242
 - NPV, 242

- NPV_, 243
- QuantLib::Instrument, 242
 - calculate, 243
 - performCalculations, 243
 - recalculate, 243
 - update, 243
- QuantLib::Instruments, 93
- QuantLib::Instruments::CapFloorParameters
 - accrualTimes, 141
 - CapFloorParameters, 141
 - capRates, 141
 - endTimes, 141
 - floorRates, 141
 - forwards, 141
 - nominals, 141
 - startTimes, 141
 - type, 141
 - validate, 141
- QuantLib::Instruments::CapFloorParameters, 141
- QuantLib::Instruments::CapFloorResults, 143
- QuantLib::Instruments::PlainOption
 - delta, 319
 - dividendRho, 319
 - gamma, 319
 - PlainOption, 319
 - rho, 319
 - setupEngine, 319
 - theta, 319
 - vega, 319
- QuantLib::Instruments::PlainOption, 319
 - impliedVolatility, 319
 - performCalculations, 319
- QuantLib::Instruments::PlainOptionParameters
 - dividendYield, 322
 - PlainOptionParameters, 322
 - residualTime, 322
 - riskFreeRate, 322
 - strike, 322
 - type, 322
 - underlying, 322
 - volatility, 322
- QuantLib::Instruments::PlainOptionParameters, 322
- QuantLib::Instruments::PlainOptionResults, 323
- QuantLib::Instruments::SimpleSwap
 - fairRate, 347
 - fixedLeg, 347
 - fixedLegBPS, 347
 - fixedRate, 347
 - floatingLeg, 347
 - floatingLegBPS, 347
 - maturity, 347
 - nominal, 347
 - payFixedRate, 347
 - SimpleSwap, 347
 - spread, 347
- QuantLib::Instruments::SimpleSwap, 347
- QuantLib::Instruments::Stock
 - Stock, 360
- QuantLib::Instruments::Stock, 360
 - performCalculations, 360
- QuantLib::Instruments::Swap
 - firstLeg_, 362
 - firstLegBPS, 362
 - firstLegBPS_, 362
 - secondLeg_, 362
 - secondLegBPS, 362
 - secondLegBPS_, 362
 - Swap, 362
 - termStructure_, 362
- QuantLib::Instruments::Swap, 362
 - performCalculations, 362
- QuantLib::Instruments::Swaption
 - setupEngine, 366
 - Swaption, 366
- QuantLib::Instruments::Swaption, 366
 - performCalculations, 366
- QuantLib::Instruments::SwaptionParameters
 - exerciseTimes, 367
 - exerciseType, 367
 - fairRate, 367
 - fixedBPS, 367
 - fixedCoupons, 367
 - fixedPayTimes, 367
 - fixedRate, 367
 - floatingPayTimes, 367
 - floatingResetTimes, 367
 - nominals, 367
 - payFixed, 367
 - SwaptionParameters, 367
 - validate, 367
- QuantLib::Instruments::SwaptionParameters, 367
- QuantLib::Instruments::SwaptionResults, 369
- QuantLib::IntegerFormatter
 - toString, 245
- QuantLib::IntegerFormatter, 245
- QuantLib::InterestRateModelling::BlackKarasinski
 - ~BlackKarasinski, 125
 - BlackKarasinski, 125
- QuantLib::InterestRateModelling::Black-Karasinski, 125
- QuantLib::InterestRateModelling::BlackModel
 - BlackModel, 126
 - formula, 126
 - termStructure, 126
 - volatility, 126

- QuantLib::InterestRateModelling::BlackModel, 126
 - update, 126
- QuantLib::InterestRateModelling::ExtendedVasicek
 - ~ExtendedVasicek, 189
 - a_, 189
 - ExtendedVasicek, 189
 - generateParameters, 189
 - phi_, 189
 - process, 189
 - sigma_, 189
 - tree, 189
- QuantLib::InterestRateModelling::ExtendedVasicek, 189
- QuantLib::InterestRateModelling::GeneralBlackKarasinski
 - ~GeneralBlackKarasinski, 216
 - a_, 216
 - f_, 216
 - GeneralBlackKarasinski, 216
 - generateParameters, 216
 - process, 216
 - sigma_, 216
 - tree, 216
- QuantLib::InterestRateModelling::GeneralBlackKarasinski, 216
- QuantLib::InterestRateModelling::GeneralCoxIngersollRoss
 - ~GeneralCoxIngersollRoss, 218
 - GeneralCoxIngersollRoss, 218
 - generateParameters, 218
 - k_, 218
 - phi_, 218
 - process, 218
 - sigma_, 218
 - theta_, 218
 - tree, 218
 - x0_, 218
- QuantLib::InterestRateModelling::GeneralCoxIngersollRoss, 218
- QuantLib::InterestRateModelling::HullWhite
 - ~HullWhite, 232
 - discountBond, 232
 - discountBondOption, 232
 - generateParameters, 232
 - HullWhite, 232
 - tree, 232
- QuantLib::InterestRateModelling::HullWhite, 232
- QuantLib::InterestRateModelling::Model
 - ~Model, 280
 - calibrate, 280
 - CalibrationFunction, 280
 - constraint_, 280
 - generateParameters, 280
 - Model, 280
 - parameters_, 280
 - termStructure, 280
- QuantLib::InterestRateModelling::Model, 280
 - update, 280
- QuantLib::InterestRateModelling::ModelTermStructure
 - discountImpl, 282
 - ModelTermStructure, 282
- QuantLib::InterestRateModelling::ModelTermStructure, 282
- QuantLib::InterestRateModelling::OneFactorModel
 - ~OneFactorModel, 299
 - OneFactorModel, 299
 - process, 299
 - tree, 299
- QuantLib::InterestRateModelling::OneFactorModel, 299
- QuantLib::Link
 - currentLink, 259
 - isNull, 259
 - update, 259
- QuantLib::Link, 259
 - Link, 259
 - linkTo, 259
- QuantLib::MarketElement
 - ~MarketElement, 263
 - value, 263
- QuantLib::MarketElement, 263
- QuantLib::Math
 - GaussianDistribution, 94
 - matrixSqrt, 95
 - operator *, 95
 - operator+, 95
 - operator-, 95
 - operator/, 95
 - outerProduct, 95
 - SymmetricEigenvalues, 95
 - SymmetricEigenvectors, 95
 - transpose, 95
- QuantLib::Math, 94
- QuantLib::Math::BilinearInterpolation
 - BilinearInterpolation, 121
 - first_argument_type, 121
 - result_type, 121
 - second_argument_type, 121
- QuantLib::Math::BilinearInterpolation, 121
 - operator(), 121
- QuantLib::Math::CubicSpline
 - ~CubicSpline, 160
 - argument_type, 160
 - CubicSpline, 160
 - result_type, 160
- QuantLib::Math::CubicSpline, 160
 - operator(), 160
- QuantLib::Math::Interpolation

- allowExtrapolation_, 246
- argument_type, 246
- Interpolation, 246
- result_type, 246
- xBegin_, 246
- xEnd_, 246
- yBegin_, 246
- QuantLib::Math::Interpolation, 246
- operator(), 246
- QuantLib::Math::Interpolation2D
 - ~Interpolation2D, 248
 - allowExtrapolation_, 248
 - data_, 248
 - first_argument_type, 248
 - Interpolation2D, 248
 - result_type, 248
 - second_argument_type, 248
 - xBegin_, 248
 - xEnd_, 248
 - yBegin_, 248
 - yEnd_, 248
- QuantLib::Math::Interpolation2D, 248
- operator(), 249
- QuantLib::Math::LexicographicalView
 - LexicographicalView, 256
 - QL_REVERSE_ITERATOR, 256
 - rxbegin, 256
 - rxend, 256
 - rybegin, 256
 - ryend, 256
 - x_iterator, 256
 - xbegin, 256
 - xend, 256
 - xSize, 256
 - y_iterator, 256
 - ybegin, 256
 - yend, 256
 - ySize, 256
- QuantLib::Math::LexicographicalView, 256
- QuantLib::Math::LinearInterpolation
 - argument_type, 258
 - LinearInterpolation, 258
 - result_type, 258
- QuantLib::Math::LinearInterpolation, 258
- operator(), 258
- QuantLib::Math::Matrix
 - ~Matrix, 264
 - begin, 264
 - column_begin, 265
 - column_end, 265
 - column_iterator, 264
 - column_rbegin, 265
 - column_rend, 265
 - columns, 265
 - const_column_iterator, 264
 - const_iterator, 264
 - const_row_iterator, 264
 - diagonal, 265
 - end, 264, 265
 - iterator, 264
 - Matrix, 264
 - matrixSqrt, 265
 - operator *, 265
 - operator *=, 264
 - operator+, 265
 - operator-, 265
 - operator=, 264
 - operator/, 265
 - operator/=, 264
 - operator=, 264
 - outerProduct, 265
 - QL_REVERSE_ITERATOR, 264
 - rbegin, 265
 - rend, 265
 - row_begin, 265
 - row_end, 265
 - row_iterator, 264
 - row_rbegin, 265
 - row_rend, 265
 - rows, 265
 - transpose, 265
- QuantLib::Math::Matrix, 264
- operator+=, 266
- QuantLib::Math::MultivariateAccumulator
 - add, 286
 - addSequence, 286
 - correlation, 286
 - mean, 286
 - meanVector, 286
 - MultivariateAccumulator, 286
 - reset, 286
 - samples, 286
 - size, 286
 - weightSum, 286
- QuantLib::Math::MultivariateAccumulator, 286
- covariance, 287
- QuantLib::Math::RiskMeasures
 - averageShortfall, 336
 - RiskMeasures, 336
 - shortfall, 336
- QuantLib::Math::RiskMeasures, 336
- expectedShortfall, 336
- potentialUpside, 336
- valueAtRisk, 336
- QuantLib::Math::SegmentIntegral
 - operator(), 342
 - SegmentIntegral, 342
- QuantLib::Math::SegmentIntegral, 342

- QuantLib::Math::Statistics
 - addSequence, 353
 - reset, 353
 - samples, 353
 - Statistics, 353
 - weightSum, 353
- QuantLib::Math::Statistics, 353
 - add, 355
 - downsideDeviation, 354
 - downsideVariance, 354
 - errorEstimate, 354
 - kurtosis, 354
 - max, 355
 - mean, 354
 - min, 355
 - skewness, 354
 - standardDeviation, 354
 - variance, 354
- QuantLib::Math::SymmetricSchurDecomposition
 - eigenvalues, 373
 - eigenvectors, 373
 - SymmetricSchurDecomposition, 373
- QuantLib::Math::SymmetricSchurDecomposition, 373
- QuantLib::MonteCarlo
 - GaussianMultiPathGenerator, 97
 - GaussianPathGenerator, 97
 - getCovariance, 97
 - MultiFactorMonteCarloOption, 97
 - OneFactorMonteCarloOption, 97
- QuantLib::MonteCarlo, 96
 - getCovariance, 97
- QuantLib::MonteCarlo::ArithmeticAOPATHPricer
 - ArithmeticAOPATHPricer, 112
 - operator(), 112
- QuantLib::MonteCarlo::ArithmeticAOPATHPricer, 112
- QuantLib::MonteCarlo::ArithmeticASOPATHPricer
 - ArithmeticASOPATHPricer, 113
 - operator(), 113
- QuantLib::MonteCarlo::ArithmeticASOPATHPricer, 113
- QuantLib::MonteCarlo::BasketPathPricer
 - BasketPathPricer, 120
 - operator(), 120
- QuantLib::MonteCarlo::BasketPathPricer, 120
- QuantLib::MonteCarlo::EuropeanPathPricer
 - EuropeanPathPricer, 185
 - operator(), 185
- QuantLib::MonteCarlo::EuropeanPathPricer, 185
- QuantLib::MonteCarlo::EverestPathPricer
 - EverestPathPricer, 186
 - operator(), 186
- QuantLib::MonteCarlo::EverestPathPricer, 186
- QuantLib::MonteCarlo::GeometricAOPATHPricer
 - GeometricAOPATHPricer, 220
 - operator(), 220
- QuantLib::MonteCarlo::GeometricAOPATHPricer, 220
- QuantLib::MonteCarlo::GeometricASOPATHPricer
 - GeometricASOPATHPricer, 221
 - operator(), 221
- QuantLib::MonteCarlo::GeometricASOPATHPricer, 221
- QuantLib::MonteCarlo::HimalayaPathPricer
 - HimalayaPathPricer, 226
 - operator(), 226
- QuantLib::MonteCarlo::HimalayaPathPricer, 226
- QuantLib::MonteCarlo::MaxBasketPathPricer
 - MaxBasketPathPricer, 267
 - operator(), 267
- QuantLib::MonteCarlo::MaxBasketPathPricer, 267
- QuantLib::MonteCarlo::MonteCarloModel
 - addSamples, 283
 - MonteCarloModel, 283
 - result_type, 283
 - sample_type, 283
 - sampleAccumulator, 283
- QuantLib::MonteCarlo::MonteCarloModel, 283
- QuantLib::MonteCarlo::MultiPath
 - assetNumber, 284
 - MultiPath, 284
 - pathSize, 284
- QuantLib::MonteCarlo::MultiPath, 284
- QuantLib::MonteCarlo::MultiPathGenerator
 - MultiPathGenerator, 285
 - next, 285
 - sample_type, 285
- QuantLib::MonteCarlo::MultiPathGenerator, 285
- QuantLib::MonteCarlo::PagodaPathPricer
 - operator(), 311
 - PagodaPathPricer, 311
- QuantLib::MonteCarlo::PagodaPathPricer, 311
- QuantLib::MonteCarlo::Path
 - diffusion, 312
 - drift, 312
 - Path, 312
 - size, 312
 - times, 312
- QuantLib::MonteCarlo::Path, 312
- QuantLib::MonteCarlo::PathGenerator
 - next, 313
 - PathGenerator, 313
 - sample_type, 313

- size, 313
- QuantLib::MonteCarlo::PathGenerator, 313
 - PathGenerator, 313
- QuantLib::MonteCarlo::PathPricer
 - ~PathPricer, 314
 - discount_, 314
 - operator(), 314
 - PathPricer, 314
 - useAntitheticVariance_, 314
- QuantLib::MonteCarlo::PathPricer, 314
- QuantLib::MonteCarlo::Sample
 - Sample, 339
 - value, 339
 - weight, 339
- QuantLib::MonteCarlo::Sample, 339
- QuantLib::Null
 - Null, 293
 - operator Type, 293
- QuantLib::Null, 293
- QuantLib::ObjectiveFunction
 - ~ObjectiveFunction, 294
 - derivative, 294
 - operator(), 294
- QuantLib::ObjectiveFunction, 294
- QuantLib::Optimization::ConjugateGradient
 - ~ConjugateGradient, 152
 - ConjugateGradient, 152
 - minimize, 152
- QuantLib::Optimization::ConjugateGradient, 152
- QuantLib::Optimization::CostFunction
 - finiteDifferenceEpsilon, 154
 - gradient, 154
 - value, 154
 - valueAndGradient, 154
- QuantLib::Optimization::CostFunction, 154
- QuantLib::Optimization::LeastSquareFunction
 - ~LeastSquareFunction, 254
 - gradient, 254
 - LeastSquareFunction, 254
 - lsp_, 254
 - value, 254
 - valueAndGradient, 254
- QuantLib::Optimization::LeastSquareFunction, 254
- QuantLib::Optimization::NonLinearLeastSquare
 - ~NonLinearLeastSquare, 291
 - exitFlag, 291
 - iterationsNumber, 291
 - lastValue, 291
 - NonLinearLeastSquare, 291
 - perform, 291
 - residualNorm, 291
 - results, 291
 - setInitialValue, 291
- QuantLib::Optimization::NonLinearLeast-Square, 291
- QuantLib::Optimization::OptimizationMethod
 - ~OptimizationMethod, 301
 - endCriteria, 301
 - endCriteria_, 302
 - functionEvaluation, 301
 - functionEvaluation_, 302
 - functionValue, 301
 - functionValue_, 302
 - gradientEvaluation, 301
 - gradientEvaluation_, 302
 - gradientNormValue, 301
 - initialValue_, 302
 - iterationNumber, 301
 - iterationNumber_, 302
 - minimize, 301
 - OptimizationMethod, 301
 - searchDirection, 301
 - searchDirection_, 302
 - setEndCriteria, 301
 - setInitialValue, 301
 - squaredNorm_, 302
 - x, 301
 - x_, 302
- QuantLib::Optimization::OptimizationMethod, 301
- QuantLib::Optimization::OptimizationProblem
 - ~OptimizationProblem, 303
 - constraint, 303
 - constraint_, 303
 - costFunction, 303
 - costFunction_, 303
 - gradient, 303
 - method_, 303
 - minimize, 303
 - minimumValue, 303
 - optimisationMethod, 303
 - OptimizationProblem, 303
 - value, 303
 - valueAndGradient, 303
- QuantLib::Optimization::OptimizationProblem, 303
- QuantLib::Optimization::Simplex
 - ~Simplex, 348
 - extrapolate, 348
 - minimize, 348
- QuantLib::Optimization::Simplex, 348
 - Simplex, 348
- QuantLib::Optimization::SteepestDescent
 - ~SteepestDescent, 356
 - minimize, 356
 - SteepestDescent, 356

- QuantLib::Optimization::SteepestDescent, 356
- QuantLib::Option
 - ~Option, 304
 - engine_, 304
 - Option, 304
 - setPricingEngine, 304
 - setupEngine, 304
- QuantLib::Option, 304
 - performCalculations, 304
- QuantLib::OptionGreeks
 - delta, 306
 - dividendRho, 306
 - gamma, 306
 - OptionGreeks, 306
 - rho, 306
 - theta, 306
 - vega, 306
- QuantLib::OptionGreeks, 306
- QuantLib::OptionPricingEngine
 - ~OptionPricingEngine, 307
 - calculate, 307
 - parameters, 307
 - results, 307
 - validateParameters, 307
- QuantLib::OptionPricingEngine, 307
- QuantLib::OptionValue
 - OptionValue, 308
 - value, 308
- QuantLib::OptionValue, 308
- QuantLib::OrnsteinUhlenbeckProcess
 - diffusion, 309
 - drift, 309
 - expectation, 309
 - OrnsteinUhlenbeckProcess, 309
 - variance, 309
- QuantLib::OrnsteinUhlenbeckProcess, 309
- QuantLib::OutOfMemoryError
 - OutOfMemoryError, 310
- QuantLib::OutOfMemoryError, 310
- QuantLib::Patterns, 98
- QuantLib::Patterns::Observable
 - ~Observable, 295
 - Observer, 295
- QuantLib::Patterns::Observable, 295
 - notifyObservers, 296
- QuantLib::Patterns::Observer
 - ~Observer, 297
 - Observer, 297
 - operator=, 297
 - registerWith, 297
 - unregisterWith, 297
- QuantLib::Patterns::Observer, 297
 - update, 298
- QuantLib::Period
 - length, 315
 - Period, 315
 - units, 315
- QuantLib::Period, 315
- QuantLib::PostconditionNotSatisfiedError
 - PostconditionNotSatisfiedError, 324
- QuantLib::PostconditionNotSatisfiedError, 324
- QuantLib::PreconditionNotSatisfiedError
 - PreconditionNotSatisfiedError, 325
- QuantLib::PreconditionNotSatisfiedError, 325
- QuantLib::Pricers
 - ExercisePayoff, 101
- QuantLib::Pricers, 99
- QuantLib::Pricers::AnalyticalCapFloor
 - AnalyticalCapFloor, 110
 - calculate, 110
- QuantLib::Pricers::AnalyticalCapFloor, 110
- QuantLib::Pricers::BarrierOption
 - BarrierOption, 119
 - calculate_, 119
 - clone, 119
 - delta, 119
 - delta_, 119
 - gamma, 119
 - gamma_, 119
 - greeksCalculated_, 119
 - theta, 119
 - theta_, 119
 - value, 119
- QuantLib::Pricers::BarrierOption, 119
- QuantLib::Pricers::BinaryOption
 - BinaryOption, 122
 - clone, 122
 - delta, 122
 - dividendRho, 122
 - gamma, 122
 - rho, 122
 - theta, 122
 - value, 122
 - vega, 122
- QuantLib::Pricers::BinaryOption, 122
- QuantLib::Pricers::BlackCapFloor
 - BlackCapFloor, 124
 - calculate, 124
- QuantLib::Pricers::BlackCapFloor, 124
- QuantLib::Pricers::BlackSwaption
 - BlackSwaption, 128
 - calculate, 128
- QuantLib::Pricers::BlackSwaption, 128
- QuantLib::Pricers::CapFloorPricingEngine
 - CapFloorPricingEngine, 142
 - model_, 142
 - parameters, 142
 - parameters_, 142

- results, 142
- results_, 142
- setModel, 142
- validateParameters, 142
- QuantLib::Pricers::CapFloorPricingEngine, 142
- update, 142
- QuantLib::Pricers::CliquetOption
 - CliquetOption, 148
 - clone, 148
 - delta, 148
 - gamma, 148
 - rho, 148
 - theta, 148
 - value, 148
 - vega, 148
- QuantLib::Pricers::CliquetOption, 148
- QuantLib::Pricers::ContinuousGeometricAPO
 - clone, 153
 - ContinuousGeometricAPO, 153
 - rho, 153
 - vega, 153
- QuantLib::Pricers::ContinuousGeometricAPO, 153
- QuantLib::Pricers::DiscreteGeometricAPO
 - clone, 173
 - delta, 173
 - DiscreteGeometricAPO, 173
 - gamma, 173
 - theta, 173
 - value, 173
- QuantLib::Pricers::DiscreteGeometricAPO, 173
- QuantLib::Pricers::DiscreteGeometricASO
 - clone, 174
 - delta, 174
 - DiscreteGeometricASO, 174
 - gamma, 174
 - theta, 174
 - value, 174
- QuantLib::Pricers::DiscreteGeometricASO, 174
- QuantLib::Pricers::EuropeanEngine
 - calculate, 183
- QuantLib::Pricers::EuropeanEngine, 183
- QuantLib::Pricers::EuropeanOption
 - clone, 184
 - delta, 184
 - dividendRho, 184
 - EuropeanOption, 184
 - gamma, 184
 - rho, 184
 - setDividendYield, 184
 - setRiskFreeRate, 184
 - setVolatility, 184
 - theta, 184
 - value, 184
 - vega, 184
- QuantLib::Pricers::EuropeanOption, 184
- QuantLib::Pricers::FdAmericanOption
 - clone, 192
 - FdAmericanOption, 192
 - initializeStepCondition, 192
- QuantLib::Pricers::FdAmericanOption, 192
- QuantLib::Pricers::FdBermudanOption
 - clone, 193
 - executeIntermediateStep, 193
 - extraTermInBermudan, 193
 - FdBermudanOption, 193
 - initializeStepCondition, 193
- QuantLib::Pricers::FdBermudanOption, 193
- QuantLib::Pricers::FdBsmOption
 - calculate, 194
 - center_, 194
 - delta, 194
 - delta_, 194
 - FdBsmOption, 194
 - finiteDifferenceOperator_, 194
 - gamma, 194
 - gamma_, 194
 - getGrid, 194
 - grid_, 194
 - gridPoints_, 194
 - initializeGrid, 194
 - initializeInitialCondition, 194
 - initializeOperator, 194
 - initialPrices_, 194
 - setGridLimits, 194
 - sMax_, 194
 - sMin_, 194
 - theta, 194
 - theta_, 194
 - value, 194
 - value_, 194
- QuantLib::Pricers::FdBsmOption, 194
- QuantLib::Pricers::FdDividendEuropeanOption
 - clone, 196
 - dividendRho, 196
 - FdDividendEuropeanOption, 196
 - rho, 196
 - theta, 196
- QuantLib::Pricers::FdDividendEuropeanOption, 196
- QuantLib::Pricers::FdDividendShoutOption
 - clone, 197
 - dividendRho, 197
 - FdDividendShoutOption, 197
 - initializeStepCondition, 197
- QuantLib::Pricers::FdDividendShoutOption, 197
- QuantLib::Pricers::FdEuropean

- calculate, 198
- clone, 198
- FdEuropean, 198
- getPrices, 198
- QuantLib::Pricers::FdEuropean, 198
- QuantLib::Pricers::FdStepConditionOption
 - calculate, 199
 - FdStepConditionOption, 199
 - initializeStepCondition, 199
 - stepCondition_, 199
 - timeSteps_, 199
- QuantLib::Pricers::FdStepConditionOption, 199
- QuantLib::Pricers::JamshidianSwaption
 - calculate, 250
 - JamshidianSwaption, 250
 - rStarFinder, 250
- QuantLib::Pricers::JamshidianSwaption, 250
- QuantLib::Pricers::McBasket
 - McBasket, 268
- QuantLib::Pricers::McBasket, 268
- QuantLib::Pricers::McDiscreteArithmeticAPO
 - McDiscreteArithmeticAPO, 269
- QuantLib::Pricers::McDiscreteArithmeticAPO, 269
- QuantLib::Pricers::McDiscreteArithmeticASO
 - McDiscreteArithmeticASO, 270
- QuantLib::Pricers::McDiscreteArithmeticASO, 270
- QuantLib::Pricers::McEuropean
 - McEuropean, 271
- QuantLib::Pricers::McEuropean, 271
- QuantLib::Pricers::McEverest
 - McEverest, 272
- QuantLib::Pricers::McEverest, 272
- QuantLib::Pricers::McHimalaya
 - McHimalaya, 273
- QuantLib::Pricers::McHimalaya, 273
- QuantLib::Pricers::McMaxBasket
 - McMaxBasket, 274
- QuantLib::Pricers::McMaxBasket, 274
- QuantLib::Pricers::McPagoda
 - McPagoda, 275
- QuantLib::Pricers::McPagoda, 275
- QuantLib::Pricers::McPricer
 - ~McPricer, 276
 - errorEstimate, 276
 - mcModel_, 276
 - McPricer, 276
 - minSample_, 276
 - sampleAccumulator, 276
 - value, 276
 - valueWithSamples, 276
- QuantLib::Pricers::McPricer, 276
- QuantLib::Pricers::PlainOptionEngine
 - parameters, 321
 - parameters_, 321
 - results, 321
 - results_, 321
 - validateParameters, 321
- QuantLib::Pricers::PlainOptionEngine, 321
- QuantLib::Pricers::SingleAssetOption
 - ~SingleAssetOption, 349
 - clone, 349
 - delta, 349
 - dividendRho, 349
 - dividendRho_, 349
 - dividendRhoComputed_, 350
 - dividendYield_, 349
 - dRMultiplier_, 350
 - dVolMultiplier_, 350
 - gamma, 349
 - hasBeenCalculated_, 349
 - residualTime_, 349
 - rho, 349
 - rho_, 349
 - rhoComputed_, 350
 - riskFreeRate_, 349
 - setDividendYield, 349
 - setRiskFreeRate, 349
 - setVolatility, 349
 - SingleAssetOption, 349
 - strike_, 349
 - theta, 349
 - type_, 349
 - underlying_, 349
 - value, 349
 - vega, 349
 - vega_, 350
 - vegaComputed_, 350
 - volatility_, 349
 - VolatilityFunction, 350
- QuantLib::Pricers::SingleAssetOption, 349
- impliedVolatility, 350
- QuantLib::Pricers::SwaptionPricingEngine
 - model_, 368
 - parameters, 368
 - parameters_, 368
 - results, 368
 - results_, 368
 - setModel, 368
 - SwaptionPricingEngine, 368
 - validateParameters, 368
- QuantLib::Pricers::SwaptionPricingEngine, 368
- update, 368
- QuantLib::Pricers::TreeCapFloor
 - calculate, 381
 - TreeCapFloor, 381
- QuantLib::Pricers::TreeCapFloor, 381

- QuantLib::Pricers::TreeSwaption
 - calculate, 382
 - TreeSwaption, 382
- QuantLib::Pricers::TreeSwaption, 382
- QuantLib::RandomNumbers
 - GaussianArrayGenerator, 102
 - GaussianRandomGenerator, 102
 - UniformRandomGenerator, 102
- QuantLib::RandomNumbers, 102
- QuantLib::RandomNumbers::BoxMullerGaussianRng
 - BoxMullerGaussianRng, 130
 - next, 130
 - sample_type, 130
- QuantLib::RandomNumbers::BoxMuller-
GaussianRng, 130
- QuantLib::RandomNumbers::CLGaussianRng
 - CLGaussianRng, 147
 - next, 147
 - sample_type, 147
- QuantLib::RandomNumbers::CLGaussianRng, 147
- QuantLib::RandomNumbers::ICGaussianRng
 - ICGaussianRng, 234
 - next, 234
 - sample_type, 234
- QuantLib::RandomNumbers::ICGaussianRng, 234
- QuantLib::RandomNumbers::KnuthUniformRng
 - sample_type, 253
- QuantLib::RandomNumbers::KnuthUniform-
Rng, 253
 - KnuthUniformRng, 253
 - next, 253
- QuantLib::RandomNumbers::LecuyerUniformRng
 - sample_type, 255
- QuantLib::RandomNumbers::LecuyerUniform-
Rng, 255
 - LecuyerUniformRng, 255
 - next, 255
- QuantLib::RandomNumbers::RandomArrayGenerator
 - next, 328
 - RandomArrayGenerator, 328
 - sample_type, 328
 - size, 328
- QuantLib::RandomNumbers::RandomArray-
Generator, 328
- QuantLib::RateFormatter
 - toString, 329
- QuantLib::RateFormatter, 329
- QuantLib::RelinkableHandle
 - isNull, 332
 - operator \rightarrow , 332
- QuantLib::RelinkableHandle, 332
 - linkTo, 332
 - RelinkableHandle, 332
- QuantLib::Results
 - ~Results, 334
- QuantLib::Results, 334
- QuantLib::RiskStatistics
 - addSequence, 337
 - averageShortfall, 337
 - errorEstimate, 337
 - expectedShortfall, 337
 - kurtosis, 337
 - max, 337
 - mean, 337
 - min, 337
 - potentialUpside, 337
 - reset, 337
 - samples, 337
 - shortfall, 337
 - skewness, 337
 - standardDeviation, 337
 - valueAtRisk, 337
 - variance, 337
 - weightSum, 337
- QuantLib::RiskStatistics, 337
 - add, 338
- QuantLib::Scheduler
 - begin, 340
 - const_iterator, 340
 - date, 340
 - end, 340
 - isRegular, 340
 - Scheduler, 340
 - size, 340
- QuantLib::Scheduler, 340
- QuantLib::SimpleMarketElement
 - setValue, 346
 - SimpleMarketElement, 346
 - value, 346
- QuantLib::SimpleMarketElement, 346
- QuantLib::Solver1D
 - ~Solver1D, 351
 - evaluationNumber_, 351
 - fxMax_, 351
 - fxMin_, 351
 - maxEvaluations_, 351
 - root_, 351
 - setHiBound, 351
 - setLowBound, 351
 - Solver1D, 351
 - xMax_, 351
 - xMin_, 351
- QuantLib::Solver1D, 351
 - setMaxEvaluations, 352
 - solve, 351
 - solve_, 352

- QuantLib::Solvers1D, 103
- QuantLib::Solvers1D::Bisection, 123
 - solve_, 123
- QuantLib::Solvers1D::Brent, 131
- QuantLib::Solvers1D::FalsePosition, 191
- QuantLib::Solvers1D::Newton, 288
- QuantLib::Solvers1D::NewtonSafe, 289
- QuantLib::Solvers1D::Ridder, 335
- QuantLib::Solvers1D::Secant, 341
- QuantLib::StringFormatter
 - toLowercase, 361
 - toUppercase, 361
- QuantLib::StringFormatter, 361
- QuantLib::SwaptionVolatilityStructure
 - ~SwaptionVolatilityStructure, 371
 - dayCounter, 371
 - todayDate, 371
 - volatility, 371
 - volatilityImpl, 371
- QuantLib::SwaptionVolatilityStructure, 371
- QuantLib::TermStructure
 - ~TermStructure, 375
 - calendar, 375
 - currency, 376
 - dayCounter, 376
 - discount, 375
 - discountImpl, 376
 - forward, 375
 - forwardImpl, 376
 - maxDate, 376
 - maxTime, 376
 - minDate, 376
 - minTime, 376
 - settlementDate, 376
 - settlementDays, 375
 - todayDate, 375
 - zeroYield, 375
 - zeroYieldImpl, 376
- QuantLib::TermStructure, 375
- QuantLib::TermStructures, 104
- QuantLib::TermStructures::DepositRateHelper
 - DepositRateHelper, 168
 - discountGuess, 168
 - impliedQuote, 168
 - maturity, 168
- QuantLib::TermStructures::DepositRateHelper, 168
 - setTermStructure, 168
- QuantLib::TermStructures::FraRateHelper
 - discountGuess, 212
 - FraRateHelper, 212
 - impliedQuote, 212
 - maturity, 212
- QuantLib::TermStructures::FraRateHelper, 212
 - setTermStructure, 212
- QuantLib::TermStructures::FuturesRateHelper
 - discountGuess, 214
 - FuturesRateHelper, 214
 - impliedQuote, 214
 - maturity, 214
- QuantLib::TermStructures::FuturesRateHelper, 214
- QuantLib::TermStructures::PiecewiseFlatForward
 - calendar, 316
 - currency, 316
 - dates, 316
 - dayCounter, 316
 - discountImpl, 317
 - FFObjFunction, 317
 - forwardImpl, 317
 - maxDate, 316
 - maxTime, 316
 - minDate, 316
 - minTime, 317
 - PiecewiseFlatForward, 316
 - settlementDate, 316
 - settlementDays, 316
 - times, 316
 - todayDate, 316
 - zeroYieldImpl, 317
- QuantLib::TermStructures::PiecewiseFlatForward, 316
 - update, 317
- QuantLib::TermStructures::RateHelper
 - ~RateHelper, 330
 - discountGuess, 330
 - impliedQuote, 330
 - maturity, 330
 - quote_, 330
 - quoteError, 330
 - RateHelper, 330
 - termStructure_, 330
- QuantLib::TermStructures::RateHelper, 330
 - setTermStructure, 331
 - update, 331
- QuantLib::TermStructures::SwapRateHelper
 - calendar_, 364
 - convention_, 364
 - fixedDayCount_, 364
 - fixedFrequency_, 364
 - fixedIsAdjusted_, 364
 - floatingFrequency_, 364
 - impliedQuote, 364
 - lengthInYears_, 364
 - maturity, 364
 - settlement_, 364
 - settlementDays_, 364
 - swap_, 364

- SwapRateHelper, 364
- termStructureHandle_, 364
- QuantLib::TermStructures::SwapRateHelper, 364
- setTermStructure, 365
- QuantLib::TimeGrid
 - dt, 378
 - findIndex, 378
 - TimeGrid, 378
- QuantLib::TimeGrid, 378
- QuantLib::Utilities, 105
- QuantLib::Utilities::combining_iterator
 - combining_iterator, 149
 - difference_type, 149
 - make_combining_iterator, 150
 - operator *, 149
 - operator
 - =, 149
 - operator+, 149
 - operator++, 149
 - operator+=, 149
 - operator-, 149
 - operator-, 149
 - operator=, 149
 - operator → , 149
 - operator==, 149
 - pointer, 149
 - reference, 149
 - value_type, 149
- QuantLib::Utilities::combining_iterator, 149
- QuantLib::Utilities::coupling_iterator
 - coupling_iterator, 155
 - difference_type, 155
 - make_coupling_iterator, 156
 - operator *, 155
 - operator
 - =, 155
 - operator+, 155
 - operator++, 155
 - operator+=, 155
 - operator-, 155
 - operator-, 155
 - operator=, 155
 - operator → , 155
 - operator==, 155
 - pointer, 155
 - reference, 155
 - value_type, 155
- QuantLib::Utilities::coupling_iterator, 155
- QuantLib::Utilities::filtering_iterator
 - filtering_iterator, 200
 - make_filtering_iterator, 200
 - operator *, 200
 - operator
 - =, 200
 - operator++, 200
 - operator-, 200
 - operator → , 200
 - operator==, 200
 - pointer, 200
 - reference, 200
- QuantLib::Utilities::filtering_iterator, 200
- QuantLib::Utilities::lowest_category_iterator, 262
- QuantLib::Utilities::processing_iterator
 - difference_type, 326
 - make_processing_iterator, 327
 - operator *, 326
 - operator
 - =, 326
 - operator+, 326
 - operator++, 326
 - operator+=, 326
 - operator-, 326
 - operator-, 326
 - operator=, 326
 - operator → , 326
 - operator<, 326
 - operator<=, 326
 - operator==, 326
 - operator>, 326
 - operator>=, 326
 - pointer, 326
 - processing_iterator, 326
 - reference, 326
 - value_type, 326
- QuantLib::Utilities::processing_iterator, 326
- QuantLib::Utilities::stepping_iterator
 - difference_type, 358
 - make_stepping_iterator, 359
 - operator *, 358
 - operator
 - =, 358
 - operator+, 358
 - operator++, 358
 - operator+=, 358
 - operator-, 358
 - operator-, 358
 - operator=, 358
 - operator → , 358
 - operator<, 358
 - operator<=, 358
 - operator==, 358
 - operator>, 358
 - operator>=, 358
 - pointer, 358
 - reference, 358
 - stepping_iterator, 358

- QuantLib::Utilities::stepping_iterator, 358
- QuantLib::Volatilities::CapFlatVolatilityVector
 - CapFlatVolatilityVector, 140
 - dayCounter, 140
 - settlementDate, 140
 - todayDate, 140
- QuantLib::Volatilities::CapFlatVolatilityVector, 140
- QuantLib::Volatilities::SwaptionVolatilityMatrix
 - dayCounter, 370
 - exerciseDates, 370
 - lengths, 370
 - SwaptionVolatilityMatrix, 370
 - todayDate, 370
- QuantLib::Volatilities::SwaptionVolatilityMatrix, 370
- QuantLib::ZeroSpreadedTermStructure
 - calendar, 392
 - currency, 392
 - dayCounter, 392
 - maxDate, 392
 - maxTime, 392
 - minDate, 392
 - minTime, 392
 - settlementDate, 392
 - settlementDays, 392
 - todayDate, 392
 - ZeroSpreadedTermStructure, 392
 - zeroYieldImpl, 393
- QuantLib::ZeroSpreadedTermStructure, 392
 - forwardImpl, 393
 - update, 393
- QuantLib::ZeroYieldStructure
 - ~ZeroYieldStructure, 394
- QuantLib::ZeroYieldStructure, 394
 - discountImpl, 394
 - forwardImpl, 394
- quote_
 - QuantLib::TermStructures::RateHelper, 330
- quoteError
 - QuantLib::TermStructures::RateHelper, 330
- RandomArrayGenerator
 - QuantLib::RandomNumbers::RandomArrayGenerator, 328
- randomarraygenerator.hpp, 606
- Rate
 - QuantLib, 83
- rate
 - QuantLib::CashFlows::FixedRateCoupon, 202
- RateHelper
 - QuantLib::TermStructures::RateHelper, 330
- ratehelpers.cpp, 607
- ratehelpers.hpp, 608
- rbegin
 - QuantLib::Array, 115
 - QuantLib::Math::Matrix, 265
- Real
 - QuantLib, 83
- recalculate
 - QuantLib::Instrument, 243
- reference
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::filtering_iterator, 200
 - QuantLib::Utilities::processing_iterator, 326
 - QuantLib::Utilities::stepping_iterator, 358
- refPeriodEnd_
 - QuantLib::CashFlows::Coupon, 158
- refPeriodStart_
 - QuantLib::CashFlows::Coupon, 158
- registerWith
 - QuantLib::Patterns::Observer, 297
- RelinkableHandle
 - QuantLib::RelinkableHandle, 332
- relinkablehandle.hpp, 609
- rend
 - QuantLib::Array, 115
 - QuantLib::Math::Matrix, 265
- reset
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::Math::Statistics, 353
 - QuantLib::RiskStatistics, 337
- residualNorm
 - QuantLib::Optimization::NonLinearLeastSquare, 291
- residualTime
 - QuantLib::Instruments::PlainOptionParameters, 322
- residualTime_
 - QuantLib::Pricers::SingleAssetOption, 349
- result_type
 - QuantLib::Math::BilinearInterpolation, 121
 - QuantLib::Math::CubicSpline, 160
 - QuantLib::Math::Interpolation, 246
 - QuantLib::Math::Interpolation2D, 248
 - QuantLib::Math::LinearInterpolation, 258
 - QuantLib::MonteCarlo::MonteCarloModel, 283
- results

- QuantLib::Optimization::NonLinearLeast-Square, 291
- QuantLib::OptionPricingEngine, 307
- QuantLib::Pricers::CapFloorPricing-Engine, 142
- QuantLib::Pricers::PlainOptionEngine, 321
- QuantLib::Pricers::SwaptionPricing-Engine, 368
- results_
 - QuantLib::Pricers::CapFloorPricing-Engine, 142
 - QuantLib::Pricers::PlainOptionEngine, 321
 - QuantLib::Pricers::SwaptionPricing-Engine, 368
- rho
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::OptionGreeks, 306
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::ContinuousGeometric-APO, 153
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdDividendEuropean-Option, 196
 - QuantLib::Pricers::SingleAssetOption, 349
- rho_
 - QuantLib::Pricers::SingleAssetOption, 349
- rhoComputed_
 - QuantLib::Pricers::SingleAssetOption, 350
- ridder.cpp, 610
- SIGN, 610
- ridder.hpp, 611
- riskFreeRate
 - QuantLib::Instruments::PlainOption-Parameters, 322
- riskFreeRate_
 - QuantLib::Pricers::SingleAssetOption, 349
- RiskMeasures
 - QuantLib::Math::RiskMeasures, 336
- riskmeasures.hpp, 612
- riskstatistics.hpp, 613
- rngtypedefs.hpp, 614
- ROL
 - QuantLib, 86
- roll
 - QuantLib::Calendar, 135
- rollback
 - QuantLib::FiniteDifferences::Finite-DifferenceModel, 201
- RollingConvention
 - QuantLib, 85
- rollingConvention
 - QuantLib::Exercise, 187
 - QuantLib::Indexes::Xibor, 388
 - rollingConvention_
 - QuantLib::CashFlows::Coupon, 158
- root_
 - QuantLib::Solver1D, 351
- row_begin
 - QuantLib::Math::Matrix, 265
- row_end
 - QuantLib::Math::Matrix, 265
- row_iterator
 - QuantLib::Math::Matrix, 264
- row_rbegin
 - QuantLib::Math::Matrix, 265
- row_rend
 - QuantLib::Math::Matrix, 265
- rows
 - QuantLib::Math::Matrix, 265
- rStarFinder
 - QuantLib::Pricers::JamshidianSwaption, 250
- rxbegin
 - QuantLib::Math::LexicographicalView, 256
- rxend
 - QuantLib::Math::LexicographicalView, 256
- rybegin
 - QuantLib::Math::LexicographicalView, 256
- ryend
 - QuantLib::Math::LexicographicalView, 256
- Sample
 - QuantLib::MonteCarlo::Sample, 339
- sample.hpp, 615
- sample_type
 - QuantLib::MonteCarlo::MonteCarlo-Model, 283
 - QuantLib::MonteCarlo::MultiPath-Generator, 285
 - QuantLib::MonteCarlo::PathGenerator, 313
 - QuantLib::RandomNumbers::BoxMuller-GaussianRng, 130
 - QuantLib::RandomNumbers::CLGaussian-Rng, 147
 - QuantLib::RandomNumbers::ICGaussian-Rng, 234
 - QuantLib::RandomNumbers::Knuth-UniformRng, 253
 - QuantLib::RandomNumbers::Lecuyer-UniformRng, 255
 - QuantLib::RandomNumbers::Random-ArrayGenerator, 328

- sampleAccumulator
 - QuantLib::MonteCarlo::MonteCarlo-Model, 283
 - QuantLib::Pricers::McPricer, 276
- samples
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::Math::Statistics, 353
 - QuantLib::RiskStatistics, 337
- Scheduler
 - QuantLib::Scheduler, 340
- scheduler.cpp, 616
- scheduler.hpp, 617
- searchDirection
 - QuantLib::Optimization::Optimization-Method, 301
- searchDirection_
 - QuantLib::Optimization::Optimization-Method, 302
- secant.cpp, 618
- secant.hpp, 619
- second_argument_type
 - QuantLib::Math::BilinearInterpolation, 121
 - QuantLib::Math::Interpolation2D, 248
- secondDerivativeAtCenter
 - QuantLib::FiniteDifferences, 91
- secondLeg_
 - QuantLib::Instruments::Swap, 362
- secondLegBPS
 - QuantLib::Instruments::Swap, 362
- secondLegBPS_
 - QuantLib::Instruments::Swap, 362
- SegmentIntegral
 - QuantLib::Math::SegmentIntegral, 342
- segmentintegral.cpp, 620
- segmentintegral.hpp, 621
- SEK
 - QuantLib, 86
- serialNumber
 - QuantLib::Date, 162
- setDividendYield
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::SingleAssetOption, 349
- setEndCriteria
 - QuantLib::Optimization::Optimization-Method, 301
- setFirstRow
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- setGridLimits
 - QuantLib::Pricers::FdBsmOption, 194
- setHiBound
 - QuantLib::Solver1D, 351
- setHistory
 - QuantLib::Indexes::XiborManager, 390
- setInitialValue
 - QuantLib::Optimization::NonLinearLeast-Square, 291
 - QuantLib::Optimization::Optimization-Method, 301
- setLastRow
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- setLowBound
 - QuantLib::Solver1D, 351
- setLowerBC
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- setMaxEvaluations
 - QuantLib::Solver1D, 352
- setMidRow
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- setMidRows
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- setModel
 - QuantLib::Pricers::CapFloorPricing-Engine, 142
 - QuantLib::Pricers::SwaptionPricing-Engine, 368
- setPricingEngine
 - QuantLib::Option, 304
- setRiskFreeRate
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::SingleAssetOption, 349
- setStep
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- setTermStructure
 - QuantLib::TermStructures::DepositRate-Helper, 168
 - QuantLib::TermStructures::FraRateHelper, 212
 - QuantLib::TermStructures::RateHelper, 331
 - QuantLib::TermStructures::SwapRate-Helper, 365
- setTime
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
 - QuantLib::FiniteDifferences::Tridiagonal-Operator::TimeSetter, 385
- settlement_
 - QuantLib::TermStructures::SwapRate-Helper, 364
- settlementDate
 - QuantLib::CapFlatVolatilityStructure, 139

- QuantLib::CapletForwardVolatility-Structure, 144
- QuantLib::ForwardSpreadedTerm-Structure, 209
- QuantLib::ImpliedTermStructure, 238
- QuantLib::TermStructure, 376
- QuantLib::TermStructures::PiecewiseFlat-Forward, 316
- QuantLib::Volatilities::CapFlatVolatility-Vector, 140
- QuantLib::ZeroSpreadedTermStructure, 392
- settlementDays
 - QuantLib::Exercise, 187
 - QuantLib::ForwardSpreadedTerm-Structure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::Indexes::Xibor, 388
 - QuantLib::TermStructure, 375
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 316
 - QuantLib::ZeroSpreadedTermStructure, 392
- settlementDays_
 - QuantLib::Exercise, 187
 - QuantLib::TermStructures::SwapRate-Helper, 364
- setupEngine
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::Instruments::Swaption, 366
 - QuantLib::Option, 304
- setUpUpperBC
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- setValue
 - QuantLib::SimpleMarketElement, 346
- setVolatility
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::SingleAssetOption, 349
- SGD
 - QuantLib, 86
- shareSameObject
 - QuantLib::Handle, 223
- shortfall
 - QuantLib::Math::RiskMeasures, 336
 - QuantLib::RiskStatistics, 337
- shortfloatingcoupon.cpp, 622
- shortfloatingcoupon.hpp, 623
- ShortFloatingRateCoupon
 - QuantLib::CashFlows::ShortFloatingRate-Coupon, 343
- shortrateprocess.hpp, 624
- shoutcondition.hpp, 625
- sigma_
 - QuantLib::InterestRate-Modelling::ExtendedVasicek, 189
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, 216
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, 218
- SIGN
 - brent.cpp, 433
 - ridder.cpp, 610
- SimpleCashFlow
 - QuantLib::CashFlows::SimpleCashFlow, 345
- simplecashflow.hpp, 626
- SimpleMarketElement
 - QuantLib::SimpleMarketElement, 346
- SimpleSwap
 - QuantLib::Instruments::SimpleSwap, 347
- swaption.cpp, 627
- swaption.hpp, 628
- Simplex
 - QuantLib::Optimization::Simplex, 348
- simplex.cpp, 629
- simplex.hpp, 630
- SingleAssetOption
 - QuantLib::Pricers::SingleAssetOption, 349
- singleassetoption.cpp, 631
- singleassetoption.hpp, 632
- QL_MAX_VOLATILITY, 632
- QL_MIN_VOLATILITY, 632
- SIT
 - QuantLib, 86
- Size
 - QuantLib, 83
- size
 - QuantLib::Array, 115
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
 - QuantLib::History, 227
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::MonteCarlo::Path, 312
 - QuantLib::MonteCarlo::PathGenerator, 313
 - QuantLib::RandomNumbers::Random-ArrayGenerator, 328
 - QuantLib::Scheduler, 340
- skewness
 - QuantLib::Math::Statistics, 354
 - QuantLib::RiskStatistics, 337
- SKK
 - QuantLib, 86
- sMax_
 - QuantLib::InterestRate-Modelling::ExtendedVasicek, 189
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, 216
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, 218

- QuantLib::Pricers::FdBsmOption, 194
- sMin_
 - QuantLib::Pricers::FdBsmOption, 194
- solve
 - QuantLib::Solver1D, 351
- solve_
 - QuantLib::Solver1D, 352
 - QuantLib::Solvers1D::Bisection, 123
- solveFor
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- Solver1D
 - QuantLib::Solver1D, 351
- solver1d.cpp, 633
- solver1d.hpp, 634
 - MAX_FUNCTION_EVALUATIONS, 634
- SOR
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- Spread
 - QuantLib, 83
- spread
 - QuantLib::CashFlows::FloatingRate-Coupon, 204
 - QuantLib::CashFlows::ShortFloatingRate-Coupon, 343
 - QuantLib::Instruments::SimpleSwap, 347
- Sqrt
 - QuantLib, 84
 - QuantLib::Array, 115
- squaredNorm_
 - QuantLib::Optimization::Optimization-Method, 302
- standardDeviation
 - QuantLib::Math::Statistics, 354
 - QuantLib::RiskStatistics, 337
- StandardFiniteDifferenceModel
 - QuantLib::FiniteDifferences, 91
- StandardStepCondition
 - QuantLib::FiniteDifferences, 91
- startDate_
 - QuantLib::CashFlows::Coupon, 157
- startTimes
 - QuantLib::Instruments::CapFloor-Parameters, 141
- Statistics
 - QuantLib::Math::Statistics, 353
- statistics.cpp, 635
- statistics.hpp, 636
- stdioMacros
 - QL_PRINTF, 75
- SteepestDescent
 - QuantLib::Optimization::SteepestDescent, 356
- steepestdescent.cpp, 637
- steepestdescent.hpp, 638
- step
 - QuantLib::FiniteDifferences::Mixed-Scheme, 278
- stepcondition.hpp, 639
- stepCondition_
 - QuantLib::Pricers::FdStepCondition-Option, 199
- stepping_iterator
 - QuantLib::Utilities::stepping_iterator, 358
- steppingiterator.hpp, 640
- Stock
 - QuantLib::Instruments::Stock, 360
- stock.cpp, 641
- stock.hpp, 642
- strike
 - QuantLib::Instruments::PlainOption-Parameters, 322
- strike_
 - QuantLib::Pricers::SingleAssetOption, 349
- Swap
 - QuantLib::Instruments::Swap, 362
- swap.cpp, 643
- swap.hpp, 644
- swap_
 - QuantLib::TermStructures::SwapRate-Helper, 364
- SwapRateHelper
 - QuantLib::TermStructures::SwapRate-Helper, 364
- Swaption
 - QuantLib::Instruments::Swaption, 366
- swaption.cpp, 645
- swaption.hpp, 646
- swaptionhelper.cpp, 647
- swaptionhelper.hpp, 648
- SwaptionParameters
 - QuantLib::Instruments::Swaption-Parameters, 367
- SwaptionPricingEngine
 - QuantLib::Pricers::SwaptionPricing-Engine, 368
- SwaptionVolatilityMatrix
 - QuantLib::Volatilities::SwaptionVolatility-Matrix, 370
- swaptionvolmatrix.hpp, 649
- swaptionvolstructure.hpp, 650
- Sydney
 - QuantLib::Calendars::Sydney, 372
- sydney.cpp, 651
- sydney.hpp, 652
- SymmetricEigenvalues
 - QuantLib::Math, 95

- symmetriceigenvalues.hpp, 653
- SymmetricEigenvectors
 - QuantLib::Math, 95
- SymmetricSchurDecomposition
 - QuantLib::Math::SymmetricSchurDecomposition, 373
- symmetricschurdecomposition.cpp, 654
- symmetricschurdecomposition.hpp, 655
- TARGET
 - QuantLib::Calendars::TARGET, 374
- target.cpp, 656
- target.hpp, 657
- Template capabilities, 77
- templateMacros
 - QL_ALLOW_TEMPLATE_METHOD_CALLS, 77
 - QL_DECLARE_TEMPLATE_SPECIALIZATIONS, 77
 - QL_EXPRESSION_TEMPLATES_WORK, 77
 - QL_TEMPLATE_METAPROGRAMMING_WORKS, 77
- tenor
 - QuantLib::Indexes::Xibor, 388
- termStructure
 - QuantLib::InterestRateModelling::BlackModel, 126
 - QuantLib::InterestRateModelling::Model, 280
- termstructure.hpp, 658
- termStructure_
 - QuantLib::Instruments::Swap, 362
 - QuantLib::TermStructures::RateHelper, 330
- termStructureHandle_
 - QuantLib::TermStructures::SwapRateHelper, 364
- theta
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::OptionGreeks, 306
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::DiscreteGeometricAPO, 173
 - QuantLib::Pricers::DiscreteGeometricASO, 174
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdBsmOption, 194
 - QuantLib::Pricers::FdDividendEuropeanOption, 196
 - QuantLib::Pricers::SingleAssetOption, 349
- theta_
 - QuantLib::FiniteDifferences::MixedScheme, 278
 - QuantLib::InterestRateModelling::GeneralCoxIngersollRoss, 218
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::FdBsmOption, 194
- Thirty360
 - QuantLib::DayCounters::Thirty360, 377
- thirty360.cpp, 659
- thirty360.hpp, 660
- Time
 - QuantLib, 83
- Time functions, 73
- TimeGrid
 - QuantLib::TimeGrid, 378
- timeMacros
 - QL_CLOCK, 73
 - QL_TIME, 73
- times
 - QuantLib::MonteCarlo::Path, 312
 - QuantLib::TermStructures::PiecewiseFlatForward, 316
- timeSetter_
 - QuantLib::FiniteDifferences::TridiagonalOperator, 384
- timeSteps_
 - QuantLib::Pricers::FdStepConditionOption, 199
- today'sDate
 - QuantLib::CapFlatVolatilityStructure, 139
 - QuantLib::CapletForwardVolatilityStructure, 144
 - QuantLib::ForwardSpreadedTermStructure, 209
 - QuantLib::ImpliedTermStructure, 238
 - QuantLib::SwaptionVolatilityStructure, 371
 - QuantLib::TermStructure, 375
 - QuantLib::TermStructures::PiecewiseFlatForward, 316
 - QuantLib::Volatilities::CapFlatVolatilityVector, 140
 - QuantLib::Volatilities::SwaptionVolatilityMatrix, 370
 - QuantLib::ZeroSpreadedTermStructure, 392
- Tokyo
 - QuantLib::Calendars::Tokyo, 379
- tokyo.cpp, 661
- tokyo.hpp, 662
- toLowerCase
 - QuantLib::StringFormatter, 361

- Toronto
 - QuantLib::Calendars::Toronto, 380
- toronto.cpp, 663
- toronto.hpp, 664
- toString
 - QuantLib::CurrencyFormatter, 161
 - QuantLib::DateFormatter, 164
 - QuantLib::DoubleFormatter, 176
 - QuantLib::EuroFormatter, 182
 - QuantLib::IntegerFormatter, 245
 - QuantLib::RateFormatter, 329
- toUppercase
 - QuantLib::StringFormatter, 361
- transpose
 - QuantLib::Math, 95
 - QuantLib::Math::Matrix, 265
- tree
 - QuantLib::InterestRate-Modelling::ExtendedVasicek, 189
 - QuantLib::InterestRate-Modelling::GeneralBlackKarasinski, 216
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, 218
 - QuantLib::InterestRateModelling::Hull-White, 232
 - QuantLib::InterestRateModelling::One-FactorModel, 299
- tree.cpp, 665
- tree.hpp, 666
- TreeCapFloor
 - QuantLib::Pricers::TreeCapFloor, 381
- treecapfloor.cpp, 667
- treecapfloor.hpp, 668
- TreeSwaption
 - QuantLib::Pricers::TreeSwaption, 382
- treeswaption.cpp, 669
- treeswaption.hpp, 670
- TridiagonalOperator
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 383
- tridiagonaloperator.cpp, 671
- tridiagonaloperator.hpp, 672
- trinomialtree.cpp, 673
- trinomialtree.hpp, 674
- TRL
 - QuantLib, 86
- twofactormodel.hpp, 675
- type
 - QuantLib::Exercise, 187
 - QuantLib::FiniteDifferences::Boundary-Condition, 129
 - QuantLib::Instruments::CapFloor-Parameters, 141
 - QuantLib::Instruments::PlainOption-Parameters, 322
 - QuantLib::Exercise, 187
 - QuantLib::Pricers::SingleAssetOption, 349
- type_
 - QuantLib::Exercise, 187
 - QuantLib::Pricers::SingleAssetOption, 349
- types.hpp, 676
- underlying
 - QuantLib::Instruments::PlainOption-Parameters, 322
- underlying_
 - QuantLib::Pricers::SingleAssetOption, 349
- UniformRandomGenerator
 - QuantLib::RandomNumbers, 102
- units
 - QuantLib::Period, 315
- unregisterWith
 - QuantLib::Patterns::Observer, 297
- update
 - QuantLib::CashFlows::FloatingRate-Coupon, 205
 - QuantLib::CashFlows::ShortFloatingRate-Coupon, 344
 - QuantLib::CompositeMarketElement, 151
 - QuantLib::DerivedMarketElement, 170
 - QuantLib::ForwardSpreadedTerm-Structure, 210
 - QuantLib::ImpliedTermStructure, 239
 - QuantLib::Instrument, 243
 - QuantLib::InterestRateModelling::Black-Model, 126
 - QuantLib::InterestRateModelling::Model, 280
 - QuantLib::Link, 259
 - QuantLib::Patterns::Observer, 298
 - QuantLib::Pricers::CapFloorPricing-Engine, 142
 - QuantLib::Pricers::SwaptionPricing-Engine, 368
 - QuantLib::TermStructures::PiecewiseFlat-Forward, 317
 - QuantLib::TermStructures::RateHelper, 331
 - QuantLib::ZeroSpreadedTermStructure, 393
- upperBC_
 - QuantLib::FiniteDifferences::Tridiagonal-Operator, 384
- USD
 - QuantLib, 86
- USDLibor
 - QuantLib::Indexes::USDLibor, 386

- usdlbor.hpp, 677
- useAntitheticVariance_
 - QuantLib::MonteCarlo::PathPricer, 314
- valid_data_iterator
 - QuantLib::History, 228
- valid_iterator
 - QuantLib::History, 228
- validate
 - QuantLib::Instruments::CapFloorParameters, 141
 - QuantLib::Instruments::SwaptionParameters, 367
- validateParameters
 - QuantLib::OptionPricingEngine, 307
 - QuantLib::Pricers::CapFloorPricingEngine, 142
 - QuantLib::Pricers::PlainOptionEngine, 321
 - QuantLib::Pricers::SwaptionPricingEngine, 368
- value
 - QuantLib::CompositeMarketElement, 151
 - QuantLib::DerivedMarketElement, 170
 - QuantLib::FiniteDifferences::BoundaryCondition, 129
 - QuantLib::History::Entry, 231
 - QuantLib::MarketElement, 263
 - QuantLib::MonteCarlo::Sample, 339
 - QuantLib::Optimization::CostFunction, 154
 - QuantLib::Optimization::LeastSquareFunction, 254
 - QuantLib::Optimization::OptimizationProblem, 303
 - QuantLib::OptionValue, 308
 - QuantLib::Pricers::BarrierOption, 119
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::DiscreteGeometricAPO, 173
 - QuantLib::Pricers::DiscreteGeometricASO, 174
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::FdBsmOption, 194
 - QuantLib::Pricers::McPricer, 276
 - QuantLib::Pricers::SingleAssetOption, 349
 - QuantLib::SimpleMarketElement, 346
- value_
 - QuantLib::Pricers::FdBsmOption, 194
- value_type
 - QuantLib::History::const_iterator, 230
 - QuantLib::Utilities::combining_iterator, 149
 - QuantLib::Utilities::coupling_iterator, 155
 - QuantLib::Utilities::processing_iterator, 326
- valueAndGradient
 - QuantLib::Optimization::CostFunction, 154
 - QuantLib::Optimization::LeastSquareFunction, 254
 - QuantLib::Optimization::OptimizationProblem, 303
- valueAtCenter
 - QuantLib::FiniteDifferences, 91
- valueatcenter.cpp, 678
- valueatcenter.hpp, 679
- valueAtRisk
 - QuantLib::Math::RiskMeasures, 336
 - QuantLib::RiskStatistics, 337
- valueWithSamples
 - QuantLib::Pricers::McPricer, 276
- variance
 - QuantLib::BlackScholesProcess, 127
 - QuantLib::DiffusionProcess, 171
 - QuantLib::Math::Statistics, 354
 - QuantLib::OrnsteinUhlenbeckProcess, 309
 - QuantLib::RiskStatistics, 337
- vbegins
 - QuantLib::History, 228
- vdbegins
 - QuantLib::History, 228
- vdend
 - QuantLib::History, 228
- vega
 - QuantLib::Instruments::PlainOption, 319
 - QuantLib::OptionGreeks, 306
 - QuantLib::Pricers::BinaryOption, 122
 - QuantLib::Pricers::CliquetOption, 148
 - QuantLib::Pricers::ContinuousGeometricAPO, 153
 - QuantLib::Pricers::EuropeanOption, 184
 - QuantLib::Pricers::SingleAssetOption, 349
- vega_
 - QuantLib::Pricers::SingleAssetOption, 350
- vegaComputed_
 - QuantLib::Pricers::SingleAssetOption, 350
- vend
 - QuantLib::History, 228
- volatility
 - QuantLib::CapFlatVolatilityStructure, 139
 - QuantLib::CapletForwardVolatilityStructure, 144
 - QuantLib::Instruments::PlainOptionParameters, 322
 - QuantLib::InterestRateModelling::BlackModel, 126

- QuantLib::SwaptionVolatilityStructure, 371
- volatility_
 - QuantLib::Pricers::SingleAssetOption, 349
- VolatilityFunction
 - QuantLib::Pricers::SingleAssetOption, 350
- volatilityImpl
 - QuantLib::CapFlatVolatilityStructure, 139
 - QuantLib::CapletForwardVolatility-Structure, 144
 - QuantLib::SwaptionVolatilityStructure, 371
- Weekday
 - QuantLib, 86
- weekday
 - QuantLib::Date, 162
- weight
 - QuantLib::MonteCarlo::Sample, 339
- weightSum
 - QuantLib::Math::MultivariateAccumulator, 286
 - QuantLib::Math::Statistics, 353
 - QuantLib::RiskStatistics, 337
- Wellington
 - QuantLib::Calendars::Wellington, 387
- wellington.cpp, 680
- wellington.hpp, 681
- what
 - QuantLib::Error, 180
- x
 - QuantLib::Optimization::Optimization-Method, 301
- x0
 - QuantLib::DiffusionProcess, 171
- x0_
 - QuantLib::InterestRate-Modelling::GeneralCoxIngersollRoss, 218
- x_
 - QuantLib::Optimization::Optimization-Method, 302
- x_iterator
 - QuantLib::Math::LexicographicalView, 256
- xbegin
 - QuantLib::Math::LexicographicalView, 256
- xBegin_
 - QuantLib::Math::Interpolation, 246
 - QuantLib::Math::Interpolation2D, 248
- xend
- QuantLib::Math::LexicographicalView, 256
- xEnd_
 - QuantLib::Math::Interpolation, 246
 - QuantLib::Math::Interpolation2D, 248
- Xibor
 - QuantLib::Indexes::Xibor, 388
- xibor.cpp, 682
- xibor.hpp, 683
- xibormanager.cpp, 684
- xibormanager.hpp, 685
- xMax_
 - QuantLib::Solver1D, 351
- xMin_
 - QuantLib::Solver1D, 351
- xSize
 - QuantLib::Math::LexicographicalView, 256
- y_iterator
 - QuantLib::Math::LexicographicalView, 256
- ybegin
 - QuantLib::Math::LexicographicalView, 256
- yBegin_
 - QuantLib::Math::Interpolation, 246
 - QuantLib::Math::Interpolation2D, 248
- Year
 - QuantLib, 83
- year
 - QuantLib::Date, 162
- yearFraction
 - QuantLib::DayCounter, 165
 - QuantLib::DayCounter::DayCounterImpl, 167
- yend
 - QuantLib::Math::LexicographicalView, 256
- yEnd_
 - QuantLib::Math::Interpolation2D, 248
- ySize
 - QuantLib::Math::LexicographicalView, 256
- ZAR
 - QuantLib, 86
- ZARLibor
 - QuantLib::Indexes::ZARLibor, 391
- zarlibor.hpp, 686
- ZeroSpreadedTermStructure
 - QuantLib::ZeroSpreadedTermStructure, 392
- zeroYield

- QuantLib::TermStructure, [375](#)
- zeroYieldImpl
 - QuantLib::DiscountStructure, [172](#)
 - QuantLib::ForwardRateStructure, [207](#)
 - QuantLib::ForwardSpreadedTermStructure, [210](#)
 - QuantLib::TermStructure, [376](#)
 - QuantLib::TermStructures::PiecewiseFlatForward, [317](#)
 - QuantLib::ZeroSpreadedTermStructure, [393](#)
- Zurich
 - QuantLib::Calendars::Zurich, [396](#)
- zurich.cpp, [687](#)
- zurich.hpp, [688](#)