# RiskAnalysisTool version 1.0.2

Generated by Doxygen 1.8.9.1

Wed May 13 2015 01:08:52

## **Contents**

1	lest	List		1
2	Hiera	archical	I Index	3
	2.1	Class I	Hierarchy	3
3	Clas	s Index	τ	7
	3.1	Class I	List	7
4	File	Index		11
	4.1	File Lis	st	11
5	Clas	s Docu	mentation	15
	5.1	Quantl	Lib::AnalyticESEngine Class Reference	15
		5.1.1	Detailed Description	15
		5.1.2	Member Function Documentation	15
			5.1.2.1 calculate	15
	5.2	Quantl	Lib::AnalyticKouEuropeanEngine Class Reference	16
		5.2.1	Detailed Description	16
	5.3	Quantl	Lib::TVA::arguments Class Reference	16
	5.4	Quantl	Lib::EquitySwap::arguments Class Reference	17
		5.4.1	Detailed Description	17
	5.5	Quantl	Lib::CrossCurrencySwap::arguments Class Reference	17
		5.5.1	Detailed Description	18
	5.6	Quantl	Lib::BVA::arguments Class Reference	18
	5.7	Quantl	Lib::AT1Pmodel Class Reference	19
		5.7.1	Detailed Description	20
	5.8	Quantl	Lib::BVA Class Reference	20
		5.8.1	Detailed Description	21
		5.8.2	Constructor & Destructor Documentation	21
			5.8.2.1 BVA	21
	5.9	Quantl	Lib::BVAEngine < INST > Class Template Reference	21
		5.9.1	Detailed Description	22
		5.9.2	Constructor & Destructor Documentation	22

iv CONTENTS

	5.9.2.1 BVAEngine	22
5.10	QuantLib::BVAPathPricer< ENGINE > Class Template Reference	22
	5.10.1 Detailed Description	23
5.11	QuantLib::CIRBondEngine Class Reference	24
	5.11.1 Detailed Description	24
5.12	QuantLib::CirDefaultModel Class Reference	24
	5.12.1 Detailed Description	25
	5.12.2 Member Function Documentation	25
	5.12.2.1 defaultTime	25
5.13	QuantLib::CIRprocess Class Reference	25
	5.13.1 Detailed Description	26
	5.13.2 Member Enumeration Documentation	26
	5.13.2.1 Discretization	26
	5.13.3 Constructor & Destructor Documentation	27
	5.13.3.1 CIRprocess	27
5.14	QuantLib::Counterparty Class Reference	27
	5.14.1 Detailed Description	28
5.15	QuantLib::CreditVaREngine < INST > Class Template Reference	28
	5.15.1 Detailed Description	29
5.16	QuantLib::CreditVaRPathPricer< ENGINE > Class Template Reference	29
	5.16.1 Detailed Description	30
5.17	QuantLib::CrossCurrencySwap Class Reference	30
	5.17.1 Detailed Description	31
	5.17.2 Member Enumeration Documentation	31
	5.17.2.1 Type	31
5.18	QuantLib::CrossCurrencySwapUCVAPathPricer Class Reference	31
	5.18.1 Detailed Description	32
5.19	QuantLib::DefaultCdsHelper Class Reference	32
	5.19.1 Detailed Description	33
5.20	QuantLib::DefaultModel Class Reference	33
	5.20.1 Detailed Description	33
	5.20.2 Member Function Documentation	33
	5.20.2.1 defaultTime	33
5.21	QuantLib::DeterministicDefaultModel Class Reference	34
	5.21.1 Detailed Description	34
5.22	QuantLib::CrossCurrencySwap::engine Class Reference	34
	5.22.1 Detailed Description	35
	QuantLib::BVA::engine Class Reference	35
5.24	QuantLib::EquitySwap::engine Class Reference	35
	5.24.1 Detailed Description	36

CONTENTS

QuantLib::TVA::engine Class Reference		
QuantLib::EquitySwap Class Reference	. 36	
5.26.1 Detailed Description	. 37	
5.26.2 Member Enumeration Documentation	. 37	
5.26.2.1 Type	. 37	
QuantLib::EquitySwapUCVAPathPricer Class Reference	. 38	
5.27.1 Detailed Description	. 38	
$\label{eq:continuous} \mbox{QuantLib::ExactSimulation} < \mbox{SIM, RNG, S, C} > \mbox{Class Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	. 38	
QuantLib::ExposureModel Class Reference	. 38	
QuantLib::GeneralizedJcirProcess <urng_poisson_type,urng_exp_type> Class Template</urng_poisson_type,urng_exp_type>		
·		
5.31.1 Detailed Description		
5.31.2 Constructor & Destructor Documentation	. 41	
5.31.2.1 GeneralizedKouProcess	. 41	
QuantLib::GenericArrayStatistics < StatisticsType > Class Template Reference	. 41	
5.32.1 Detailed Description	. 43	
QuantLib::KouCalibrationHelper Class Reference	. 43	
QuantLib::KouHelper Class Reference	. 43	
5.34.1 Detailed Description	. 43	
QuantLib::KouModel Class Reference	. 44	
QuantLib::KouProcessCalibrator Class Reference	. 44	
5.36.1 Detailed Description	. 45	
5.36.2 Constructor & Destructor Documentation	. 45	
5.36.2.1 KouProcessCalibrator	. 45	
QuantLib::MCBVAEngine < RNG, S > Class Template Reference	. 45	
5.37.1 Detailed Description	. 46	
5.37.2 Constructor & Destructor Documentation	. 46	
5.37.2.1 MCBVAEngine	. 46	
$\label{eq:continuous} \mbox{QuantLib::MCCreditVaREngine} < \mbox{INST, RNG, S} > \mbox{Class Template Reference}  .  .  .  .  .  .  .  .  .  $	. 47	
5.38.1 Detailed Description	. 47	
QuantLib::MCCrossCurrencySwapUCVAEngine Class Reference	. 48	
5.39.1 Detailed Description	. 48	
QuantLib::MCEquitySwapUCVAEngine Class Reference	. 49	
5.40.1 Detailed Description	. 49	
QuantLib::MCTVACrossCurrencySwapModel Class Reference	. 50	
	OuantLib::EquitySwap Class Reference 5.26.1 Detailed Description 5.26.2.2 Member Enumeration Documentation 5.26.2.1 Type  QuantLib::EquitySwapUCVAPathPricer Class Reference 5.27.1 Detailed Description  QuantLib::ExactSimulation< SIM, RNG, S, C > Class Template Reference QuantLib::ExposureModel Class Reference QuantLib::SeneralizedJoirProcess <urng_poisson_type,urng_exp_type> Class Template Reference  QuantLib::GeneralizedJoirProcess<urng_poisson_type,urng_exp_type> Class Template Reference 5.30.1 Detailed Description 5.30.2 Constructor &amp; Destructor Documentation 5.30.2.1 GeneralizedKouProcess QuantLib::GeneralizedKouProcess<urng_poisson_type,urng_doubleexpdist_type> Class Template Reference 5.31.1 Detailed Description 5.31.2 Constructor &amp; Destructor Documentation 5.31.2.1 GeneralizedKouProcess QuantLib::GenericArrayStatistics&lt; StatisticsType &gt; Class Template Reference 5.32.1 Detailed Description QuantLib::KouCalibrationHelper Class Reference QuantLib::KouCalibrationHelper Class Reference QuantLib::KouProcessCalibrator Class Reference QuantLib::KouProcessCalibrator Class Reference 0.36.1 Detailed Description 0.36.2 Constructor &amp; Destructor Documentation 5.36.2.1 KouProcessCalibrator 0.37.1 Detailed Description 5.37.2 Constructor &amp; Destructor Documentation 5.36.3.1 Detailed Description 5.37.2 Constructor &amp; Destructor Documentation 5.36.2.1 KouProcessCalibrator 0.37.2 Constructor &amp; Destructor Documentation 5.37.3 Detailed Description 0.37.3 Detailed Description 0.37.4 McBVAEngine </urng_poisson_type,urng_doubleexpdist_type></urng_poisson_type,urng_exp_type></urng_poisson_type,urng_exp_type>	

vi CONTENTS

	5.41.1	Detailed Description	50
	5.41.2	Constructor & Destructor Documentation	51
		5.41.2.1 MCTVACrossCurrencySwapModel	51
5.42	QuantL	.ib::MCTVAEngine $<$ RNG, S $>$ Class Template Reference	51
	5.42.1	Detailed Description	52
	5.42.2	Constructor & Destructor Documentation	52
		5.42.2.1 MCTVAEngine	52
5.43	QuantL	ib::MCTVAEquitySwapModel Class Reference	53
	5.43.1	Detailed Description	53
	5.43.2	Constructor & Destructor Documentation	54
		5.43.2.1 MCTVAEquitySwapModel	54
5.44	QuantL	.ib::MCTVAModel Class Reference	54
	5.44.1	Detailed Description	55
5.45	QuantL	.ib::MCTVAVanillaSwapModel Class Reference	55
	5.45.1	Detailed Description	55
	5.45.2	Constructor & Destructor Documentation	56
		5.45.2.1 MCTVAVanillaSwapModel	56
5.46	QuantL	ib::MCUCVAEngine < INST, RNG, S > Class Template Reference	56
	5.46.1	Detailed Description	57
5.47	QuantL	ib::MCVanillaSwapUCVAEngine Class Reference	57
	5.47.1	Detailed Description	58
5.48	QuantL	.ib::MultiValueMultiVariate< RNG > Struct Template Reference	58
	5.48.1	Detailed Description	59
5.49	QuantL	ib::BVA::results Class Reference	59
5.50	QuantL	.ib::EquitySwap::results Class Reference	59
	5.50.1	Detailed Description	59
5.51	QuantL	ib::TVA::results Class Reference	60
5.52	QuantL	ib::CrossCurrencySwap::results Class Reference	60
	5.52.1	Detailed Description	60
5.53	RiskAn	alysisTool::Calculation::Sample::SampleClass Class Reference	60
5.54	RiskAn	alysisTool::Calculation::sealed Class Reference	61
5.55	QuantL	ib::ShortRateTermStructure Class Reference	61
	5.55.1	Detailed Description	61
5.56	QuantL	ib::SimEquitySwapEngine Class Reference	62
5.57	QuantL	ib::TVA Class Reference	62
	5.57.1	Detailed Description	63
5.58	QuantL	ib::TVAEngine < INST > Class Template Reference	63
	5.58.1	Detailed Description	64
	5.58.2	Constructor & Destructor Documentation	64
		5.58.2.1 TVAEngine	64

CONTENTS vii

	5.59	QuantLib::TVAPathPricer< ENGINE > Class Template Reference	64
		5.59.1 Detailed Description	65
	5.60	$\label{eq:QuantLib::UCVAEngine} QuantLib:: UCVAEngine < INST > Class \ Template \ Reference \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	65
		5.60.1 Detailed Description	66
	5.61	$\label{eq:QuantLib::UCVAPathPricer} QuantLib:: UCVAPathPricer < ENGINE > Class \ Template \ Reference \qquad $	66
		5.61.1 Detailed Description	67
	5.62	QuantLib::VanillaSwapUCVAPathPricer Class Reference	67
		5.62.1 Detailed Description	68
		5.62.2 Member Function Documentation	68
		5.62.2.1 defaultNPV	68
	5.63	QuantLib::ZerocouponbondHelper Class Reference	68
		5.63.1 Detailed Description	69
		5.63.2 Member Function Documentation	69
		5.63.2.1 blackPrice	69
		5.63.2.2 modelValue	69
6	File [	Documentation	71
	6.1		71
	6.2		71
		6.2.1 Detailed Description	
	6.3	·	71
	6.4	analytickoueuropeanengine.hpp File Reference	71
		6.4.1 Detailed Description	72
	6.5	at1pdefaultmodel.cpp File Reference	72
	6.6	at1pdefaultmodel.hpp File Reference	72
		6.6.1 Detailed Description	72
	6.7	bva.hpp File Reference	73
		6.7.1 Detailed Description	73
	6.8	bvaengine.hpp File Reference	73
		6.8.1 Detailed Description	73
	6.9	bvapathpricer.hpp File Reference	73
		6.9.1 Detailed Description	74
	6.10	cirbondengine.cpp File Reference	74
	6.11	cirbondengine.hpp File Reference	74
		6.11.1 Detailed Description	74
	6.12	cirdefaultmodel.cpp File Reference	74
	6.13	cirdefaultmodel.hpp File Reference	75
		6.13.1 Detailed Description	75
	6.14	cirprocess.hpp File Reference	75
		6.14.1 Detailed Description	75

viii CONTENTS

6.15	counterparty.cpp File Reference	75
6.16	counterparty.hpp File Reference	75
	6.16.1 Detailed Description	76
6.17	creditvarpathpricer.hpp File Reference	76
	6.17.1 Detailed Description	76
6.18	crosscurrencyswap.hpp File Reference	76
	6.18.1 Detailed Description	77
6.19	defaultcdshelper.hpp File Reference	77
	6.19.1 Detailed Description	77
6.20	defaultmodel.hpp File Reference	77
	6.20.1 Detailed Description	78
6.21	deterministicdefaultmodel.cpp File Reference	78
6.22	deterministicdefaultmodel.hpp File Reference	78
	6.22.1 Detailed Description	78
6.23	equityswap.cpp File Reference	78
6.24	equityswap.hpp File Reference	78
	6.24.1 Detailed Description	79
6.25	exactsimulation.hpp File Reference	79
6.26	jcirprocess.hpp File Reference	79
	6.26.1 Detailed Description	80
6.27	kouprocess.hpp File Reference	80
	6.27.1 Detailed Description	80
6.28	kouprocesscalibrator.cpp File Reference	80
6.29	kouprocesscalibrator.hpp File Reference	81
	6.29.1 Detailed Description	81
6.30	mcbvaengine.hpp File Reference	81
	6.30.1 Detailed Description	81
6.31	mccreditvarengine.hpp File Reference	82
	6.31.1 Detailed Description	82
6.32	mccrosscurrencyswapucvaengine.cpp File Reference	82
6.33	mccrosscurrencyswapucvaengine.hpp File Reference	82
	6.33.1 Detailed Description	83
6.34	mcequityswapucvaengine.cpp File Reference	83
6.35	mcequityswapucvaengine.hpp File Reference	83
	6.35.1 Detailed Description	84
6.36	mctvacrosscurrencyswapmodel.cpp File Reference	84
6.37	mctvacrosscurrencyswapmodel.hpp File Reference	84
	6.37.1 Detailed Description	84
6.38	mctvaengine.hpp File Reference	84
	6.38.1 Detailed Description	85

CONTENTS

6.39	mctvaequityswapmodel.cpp File Reference	85
6.40	mctvaequityswapmodel.hpp File Reference	85
	6.40.1 Detailed Description	86
6.41	mctvamodel.hpp File Reference	86
	6.41.1 Detailed Description	86
6.42	mctvavanillaswapmodel.cpp File Reference	86
6.43	mctvavanillaswapmodel.hpp File Reference	86
	6.43.1 Detailed Description	87
6.44	mcucvaengine.hpp File Reference	87
	6.44.1 Detailed Description	87
6.45	mcvanillaswapucvaengine.cpp File Reference	87
6.46	mcvanillaswapucvaengine.hpp File Reference	88
	6.46.1 Detailed Description	88
6.47	multivaluemctraits.hpp File Reference	88
	6.47.1 Detailed Description	88
6.48	shortratetermstructure.cpp File Reference	88
6.49	shortratetermstructure.hpp File Reference	88
	6.49.1 Detailed Description	89
6.50	tva.hpp File Reference	89
	6.50.1 Detailed Description	89
6.51	tvaengine.hpp File Reference	89
	6.51.1 Detailed Description	90
6.52	tvapathpricer.hpp File Reference	90
	6.52.1 Detailed Description	90
6.53	ucvaengine.hpp File Reference	90
	6.53.1 Detailed Description	90
6.54	ucvapathpricer.hpp File Reference	90
	6.54.1 Detailed Description	91
6.55	zerocouponbondhelper.cpp File Reference	91
6.56	zerocouponbondhelper.hpp File Reference	91
	6.56.1 Detailed Description	91
Index		93

## **Chapter 1**

### **Test List**

Class QuantLib::GenericArrayStatistics< StatisticsType > (p. 41)

the correctness of the returned values is tested by checking them against numerical calculations.

2 **Test List** 

## Chapter 2

### **Hierarchical Index**

### 2.1 Class Hierarchy

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

arguments
QuantLib::CrossCurrencySwap::arguments
arguments
QuantLib::BVA::arguments
QuantLib::TVA::arguments
arguments
QuantLib::EquitySwap::arguments
BlackScholesMertonProcess
$Quant Lib:: Generalized KouProcess < \_URng\_Poisson\_Type, \URng\_Double ExpDist\_Type > \ . \ . \ . \ 4$
CalibratedModel
QuantLib::AT1Pmodel
QuantLib::KouModel
CalibrationHelper
QuantLib::DefaultCdsHelper
QuantLib::KouCalibrationHelper
QuantLib::ZerocouponbondHelper
CostFunction
QuantLib::KouProcessCalibrator
QuantLib::Counterparty
CoxIngersollRoss
QuantLib::CirDefaultModel
QuantLib::DefaultModel
QuantLib::AT1Pmodel
QuantLib::CirDefaultModel
QuantLib::DeterministicDefaultModel
enable shared from this
QuantLib::MCBVAEngine < RNG, S >
QuantLib::MCCrossCurrencySwapUCVAEngine
QuantLib::MCEquitySwapUCVAEngine
QuantLib::MCTVAEngine < RNG, S >
QuantLib::MCVanillaSwapUCVAEngine
engine
QuantLib::UCVAEngine < VanillaSwap >
QuantLib::MCUCVAEngine < VanillaSwap >
QuantLib::MCVanillaSwapUCVAEngine
engine
QuantLib::BVAEngine < INST >
QuantLib::CreditVaREngine < INST >

4 Hierarchical Index

QuantLib::MCCreditVaREngine < INST, RNG, S >	4
QuantLib::CreditVaREngine < INST >	2
QuantLib::TVAEngine < INST >	6
QuantLib::UCVAEngine < INST >	6
QuantLib::MCUCVAEngine < INST, RNG, S >	5
engine	
QuantLib::AnalyticKouEuropeanEngine	1
QuantLib::ExactSimulation < SIM, RNG, S, C >	
QuantLib::ExposureModel	
QuantLib::GenericArrayStatistics< StatisticsType >	
GenericEngine	
QuantLib::BVA::engine	3
QuantLib::BVAEngine < BVA >	2
QuantLib::MCBVAEngine < RNG, S >	
QuantLib::CrossCurrencySwap::engine	
QuantLib::UCVAEngine < CrossCurrencySwap >	
QuantLib::MCUCVAEngine < CrossCurrencySwap >	
QuantLib::MCCrossCurrencySwapUCVAEngine	
QuantLib::TVA::engine	
QuantLib::TVA::engine	
· · · · · · · · · · · · · · · · · · ·	
QuantLib::MCTVAEngine < RNG, S >	
GenericEngine QuantLib::EquitySwap::engine	0
QuantLib::AnalyticESEngine	
QuantLib::SimEquitySwapEngine	
QuantLib::UCVAEngine < EquitySwap >	
QuantLib::MCUCVAEngine < EquitySwap >	
QuantLib::MCEquitySwapUCVAEngine	4
· · · · · · · · · · · · · · · · · · ·	
GenericModelEngine	
GenericModelEngine QuantLib::CIRBondEngine	
GenericModelEngine QuantLib::CIRBondEngine	2
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < RNG, S >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < RNG, S > QuantLib::MCBVAEngine < RNG, S > QuantLib::MCCreditVaREngine < INST, RNG, S >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NoillaSwap > QuantLib::MCUCVAEngine < RoullaSwap > QuantLib::MCUCVAEngine < RoullaSwap > QuantLib::MCUCVAEngine < RoullaSwap > QuantLib::MCCreditVaREngine < INST, RNG, S > QuantLib::MCCreditVaREngine < RNG, S >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCCreditVaREngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCCCREditVaREngine < RNG, S > QuantLib::MCCCREditVaREngine < RNG, S >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCBVAEngine < RNG, S > QuantLib::MCCreditVaREngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCUCVAEngine < RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAModel	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCBVAEngine < RNG, S > QuantLib::MCCreditVaREngine < INST, RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA . QuantLib::TVA  QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCCreditVaREngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA  QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NNG, S > QuantLib::MCUCVAEngine < RNG, S > QuantLib::MCCreditVaRengine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAVanillaSwapModel	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCCreditVaRengine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA  QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine< CrossCurrencySwap > QuantLib::MCUCVAEngine< EquitySwap > QuantLib::MCUCVAEngine< VanillaSwap > QuantLib::MCUCVAEngine< RNG, S > QuantLib::MCCreditVaREngine< INST, RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MUltiValueMultiVariate< RNG > path_pricer_type	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine< CrossCurrencySwap > QuantLib::MCUCVAEngine< EquitySwap > QuantLib::MCUCVAEngine< VanillaSwap > QuantLib::MCBVAEngine< RNG, S > QuantLib::MCCreditVaRengine< INST, RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< RNG > path_pricer_type QuantLib::UCVAPathPricer< MCCrossCurrencySwapUCVAEngine >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA  QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCBVAEngine < Namily	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA. QuantLib::TVA.  QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NG, S > QuantLib::MCEVAEngine < NG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::CrossCurrencySwapUCVAEngine > QuantLib::CrossCurrencySwapUCVAPathPricer path_pricer_type	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine< CrossCurrencySwap > QuantLib::MCUCVAEngine< EquitySwap > QuantLib::MCUCVAEngine< VanillaSwap > QuantLib::MCUCVAEngine< RNG, S > QuantLib::MCCreditVaREngine< INST, RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< RNG, S > QuantLib::MCTVAEngine< INST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::McTVAVanillaSwapModel QuantLib::CrossCurrencySwapUCVAEngine > QuantLib::CrossCurrencySwapUCVAPathPricer path_pricer_type QuantLib::UCVAPathPricer< MCVanillaSwapUCVAEngine >	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NST, RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < RNST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAModel QuantLib::MCTVAModel QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAPathPricer < MCCrossCurrencySwapUCVAEngine > QuantLib::CrossCurrencySwapUCVAPathPricer  path_pricer_type QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > QuantLib::VanillaSwapUCVAPathPricer	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NNG, S > QuantLib::MCEVAEngine < NNG, S > QuantLib::MCTVAEngine < NNST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAPathPricer < MCCrossCurrencySwapUCVAEngine > QuantLib::UCVAPathPricer < MCCrossCurrencySwapUCVAEngine > QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > QuantLib::VanillaSwapUCVAPathPricer	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < RNG, S > QuantLib::MCUCVAEngine < RNG, S > QuantLib::MCCreditVaREngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < INST, RNG, S > QuantLib::MCTVAEngine < RNG, S > QuantLib::MCTVAEquitySwapModel QuantLib::MCTVAEquitySwapModel QuantLib::MUTVAEquitySwapModel QuantLib::MultiValueMultiVariate < RNG > path_pricer_type QuantLib::CrossCurrencySwapUCVAPathPricer path_pricer_type QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > QuantLib::VanillaSwapUCVAPathPricer path_pricer_type QuantLib::VanillaSwapUCVAPathPricer	
GenericModelEngine QuantLib::CIRBondEngine Instrument QuantLib::BVA QuantLib::TVA QuantLib::KouHelper McSimulation QuantLib::MCUCVAEngine < CrossCurrencySwap > QuantLib::MCUCVAEngine < EquitySwap > QuantLib::MCUCVAEngine < VanillaSwap > QuantLib::MCUCVAEngine < NNG, S > QuantLib::MCEVAEngine < NNG, S > QuantLib::MCTVAEngine < NNST, RNG, S > QuantLib::MCTVAModel QuantLib::MCTVAModel QuantLib::MCTVACrossCurrencySwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAVanillaSwapModel QuantLib::MCTVAPathPricer < MCCrossCurrencySwapUCVAEngine > QuantLib::UCVAPathPricer < MCCrossCurrencySwapUCVAEngine > QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > QuantLib::VanillaSwapUCVAPathPricer	

2.1 Class Hierarchy 5

QuantLib::UCVAPathPricer< ENGINE >
path_pricer_type
QuantLib::UCVAPathPricer< MCEquitySwapUCVAEngine >
QuantLib::EquitySwapUCVAPathPricer
results
QuantLib::CrossCurrencySwap::results
QuantLib::EquitySwap::results
results
QuantLib::BVA::results
QuantLib::TVA::results
RiskAnalysisTool::Calculation::Sample::SampleClass
RiskAnalysisTool::Calculation::sealed
QuantLib::ShortRateTermStructure
StochasticProcess1D
QuantLib::CIRprocess
QuantLib::GeneralizedJcirProcess <urng_poisson_type,urng_exp_type></urng_poisson_type,urng_exp_type>
Swap
QuantLib::CrossCurrencySwap
QuantLib::EquitySwap

6 **Hierarchical Index** 

## **Chapter 3**

### **Class Index**

### 3.1 Class List

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

QuantLib::AnalyticESEngine	
Analytic engine of equity swap	15
QuantLib::AnalyticKouEuropeanEngine	
Analytic engine for european option, whose underlying stock process is a kou process	16
QuantLib::TVA::arguments	16
QuantLib::EquitySwap::arguments	
Arguments for equity swap calculation	17
QuantLib::CrossCurrencySwap::arguments	
Arguments for cross currency swap calculation	17
QuantLib::BVA::arguments	18
QuantLib::AT1Pmodel	
Analytic tractable first default model	19
QuantLib::BVA	
Bilateral value adjustment manager class	20
QuantLib::BVAEngine < INST >	
Base class for bva engine	21
QuantLib::BVAPathPricer< ENGINE >	
Path pricer class used to calculate the BVA (p. 20) on each Multi-path for BVAEngine (p. 21)	
class	22
QuantLib::CIRBondEngine	
A CIRBondEngine (p. 24) is a bond pricing engine based on CIR model	24
QuantLib::CirDefaultModel	
Intensity default model based on the CIR process	24
QuantLib::CIRprocess	
CIR process class	25
QuantLib::Counterparty	
Counterparty (p. 27) class	27
QuantLib::CreditVaREngine < INST >	
Base class for credit VaR engine	28
QuantLib::CreditVaRPathPricer< ENGINE >	
Abstract base class used to calculate default value of each path	29
QuantLib::CrossCurrencySwap	
Fixed-fixed cross currency swap class	30
QuantLib::CrossCurrencySwapUCVAPathPricer	
UCVA Path Pricer for fixed-fixed cross currency swap	31
QuantLib::DefaultCdsHelper	
CDS helpers to calibrate default model	32

8 Class Index

QuantLib::DefaultModel	
Abstract base class for default model	33
QuantLib::DeterministicDefaultModel	
Intensity default model based on deterministic default intensity	34
QuantLib::CrossCurrencySwap::engine	
Base class for cross currency swap pricing engine	34
QuantLib::BVA::engine	35
QuantLib::EquitySwap::engine	
Equity option engine base class	35
QuantLib::TVA::engine	36
QuantLib::EquitySwap	
Equity swap class	36
QuantLib::EquitySwapUCVAPathPricer	
UCVA Path Pricer for equity swap	38
$\textbf{QuantLib::ExactSimulation} < \textbf{SIM}, \textbf{RNG}, \textbf{S}, \textbf{C} > \dots $	38
QuantLib::ExposureModel	38
QuantLib::GeneralizedJcirProcess <urng_poisson_type,urng_exp_type></urng_poisson_type,urng_exp_type>	
Jump diffusion CIR process class	39
QuantLib::GeneralizedKouProcess <urng_poisson_type,urng_doubleexpdisttype></urng_poisson_type,urng_doubleexpdisttype>	
Kou process class	40
QuantLib::GenericArrayStatistics< StatisticsType >	
Statistics analysis of N-dimensional (sequence) data	41
QuantLib::KouCalibrationHelper	43
QuantLib::KouHelper	
Kou process helper	43
QuantLib::KouModel	44
QuantLib::KouProcessCalibrator	
A Kou process calibrator class which is used to do calibration using sets of vanilla options	44
QuantLib::MCBVAEngine < RNG, S >	
Pricing engine class for <b>BVA</b> (p. 20) calculation via Monte Carlo Simulation	45
QuantLib::MCCreditVaREngine < INST, RNG, S >	
Abstract base class for Credit VaR calculation via Monte Carlo Simulation	47
QuantLib::MCCrossCurrencySwapUCVAEngine	
Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap	48
QuantLib::MCEquitySwapUCVAEngine	
Monte Carlo pricing engine for UCVA calculation of equity swap	49
QuantLib::MCTVACrossCurrencySwapModel	
TVA (p. 62) model for cross currency swap	50
QuantLib::MCTVAEngine < RNG, S >	
Pricing engine class for TVA (p. 62) calculation via Monte Carlo Simulation	51
QuantLib::MCTVAEquitySwapModel	
TVA (p. 62) model for equity swap	53
QuantLib::MCTVAModel	
Base class for tva model	54
QuantLib::MCTVAVanillaSwapModel	
TVA (p. 62) model for vanilla swap	55
QuantLib::MCUCVAEngine < INST, RNG, S >	
Abstract base class for UCVA calculation via Monte Carlo Simulation	56
QuantLib::MCVanillaSwapUCVAEngine	
Monte Carlo pricing engine for UCVA calculation of vanilla swap	57
QuantLib::MultiValueMultiVariate < RNG >	
Traits class for multi-state Monte Carlo Simulation	58
QuantLib::BVA::results	59
QuantLib::EquitySwap::results	
Results class of equity swap	59
QuantLib::TVA::results	60
QuantLib::CrossCurrencySwap::results	
Results for cross currency swap calculation	60

3.1 Class List

RiskAnalysisTool::Calculation::Sample::SampleClass	
QuantLib::ShortRateTermStructure	01
Interest-rate term structure class based on short-rate model	61
QuantLib::SimEquitySwapEngine	
QuantLib::TVA	
Total value adjustment manager class	62
QuantLib::TVAEngine < INST >	
Base class for bva engine	63
QuantLib::TVAPathPricer< ENGINE >	
Path pricer class used to calculate the TVA (p. 62) on each Multi-path for TVAEngine (p. 63)	
class	64
QuantLib::UCVAEngine < INST >	
Abstract base class for ucva engine	65
QuantLib::UCVAPathPricer< ENGINE >	
Abstract base class used to calculate cash flow at default time of each path under the setting of	
UCVA	66
QuantLib::VanillaSwapUCVAPathPricer	
UCVA Path Pricer for vanilla swap	67
QuantLib::ZerocouponbondHelper	
Zero coupon bond calibration helper	68

10 Class Index

## **Chapter 4**

### File Index

### 4.1 File List

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

analyticequityswapengine.cpp
analyticequityswapengine.hpp
Analytic engine of equity swap
analyticeuropeanucvaengine.hpp?
analytickoueuropeanengine.cpp
analytickoueuropeanengine.hpp
Analytic engine for european option
arraystatistics.hpp
at1pdefaultmodel.cpp
at1pdefaultmodel.hpp
Analytic tractable first passage model
bva.hpp
Bilateral value adjustment manager class
bvaengine.hpp
Base class for bva engine
bvapathpricer.hpp
Path pricer class used to calculate the BVA on each Multi-path for BVAEngine class
Calculation.h
cirbondengine.cpp
cirbondengine.hpp
Bond pricing engine based on CIR model
cirdefaultmodel.cpp
cirdefaultmodel.hpp
Intensity default model based on the CIR process
cirprocess.hpp
CIR process class
Conversions.hpp
counterparty.cpp
counterparty.hpp
Counterparty class
credit/creditvar/creditvarengine.hpp
pricingengines/credit/creditvarengine.hpp
pricingengines/creditvarengine.hpp
creditvarpathpricer.hpp
Abstract base class used to calculate default value of each path
crosscurrencyswap.hpp
Fixed-fixed cross currency swap class

12 File Index

defaultcdshelper.hpp	
CDS helpers to calibrate default model	77
defaultmodel.hpp	
Abstract base class for default model	77
deterministicdefaultmodel.cpp	78
deterministicdefaultmodel.hpp	
Intensity default model based on deterministic default intensity	78
equityswap.cpp	78
equityswap.hpp	
Equity swap class	78
exactsimulation.hpp	79
exposuremodel.hpp	??
calibration/extended.hpp	??
credit/extended.hpp	??
extended.hpp	??
instruments/credit/extended.hpp	??
instruments/extended.hpp	??
••	??
instruments/swap/extended.hpp	
math/extended.hpp	??
math/statistics/extended.hpp	??
methods/extended.hpp	??
methods/montecarlo/extended.hpp	??
pricingengines/extended.hpp	??
pricingengines/swap/extended.hpp	??
pricingengines/ucva/extended.hpp	??
pricingengines/vanilla/extended.hpp	??
processes/extended.hpp	??
jcirprocess.hpp	
JCIR process class	79
koucalibrationhelper.hpp	??
koumodel.hpp	??
kouprocess.hpp	
Kou process class	80
kouprocesscalibrator.cpp	80
kouprocesscalibrator.hpp	
Kou process calibrator class	81
mcbvaengine.hpp	٠.
Pricing engine class for BVA calculation via Monte Carlo Simulation	81
mccreditvarengine.hpp	٥.
Abstract base class for Credit VaR calculation via Monte Carlo Simulation	82
mccrosscurrencyswapucvaengine.cpp	82
mccrosscurrencyswapucvaengine.hpp	02
Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap	82
	83
mcequityswapucvaengine.cpp	03
mcequityswapucvaengine.hpp  Monte Carlo prining angine for LICVA calculation of aguity guen	00
Monte Carlo pricing engine for UCVA calculation of equity swap	83
mctvacrosscurrencyswapmodel.cpp	84
mctvacrosscurrencyswapmodel.hpp	0.4
TVA model for cross currency swap	84
mctvaengine.hpp	0.4
Pricing engine class for TVA calculation via Monte Carlo Simulation	84
mctvaequityswapmodel.cpp	85
mctvaequityswapmodel.hpp	
TVA model for equity swap	85
mctvamodel.hpp	
Base class for tva model	86
mctvavanillaswapmodel.cpp	86

4.1 File List

mctvavanillaswapmodel.hpp	
TVA model for vanilla swap	86
mcucvaengine.hpp	
Abstract base class for UCVA calculation via Monte Carlo Simulation	87
mcvanillaswapucvaengine.cpp	87
mcvanillaswapucvaengine.hpp	
Monte Carlo pricing engine for UCVA calculation of vanilla swap	88
multivaluemctraits.hpp	
Traits class for multi-state Monte Carlo Simulation	88
pch.h	??
SampleClass.hpp	??
shortratetermstructure.cpp	38
shortratetermstructure.hpp	
	38
simequityswapengine.hpp	??
simvanillaswapucvaengine.hpp	??
tva.hpp	
Total value adjustment manager class	89
tvaengine.hpp	
Base class for tva engine	89
tvapathpricer.hpp	
Path pricer class used to calculate the TVA on each Multi-path for TVAEngine class	90
ucvaengine.hpp	
· ·	90
ucvapathpricer.hpp	
Abstract base class used to calculate cash flow at default time of each path under the setting of	
	90
Utilities.hpp	??
r r- r- r-	91
zerocouponbondhelper.hpp	
Zero coupon bond calibration helper	91

14 File Index

### **Chapter 5**

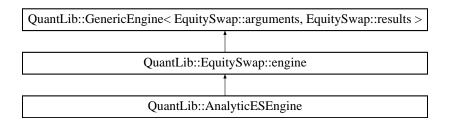
### **Class Documentation**

#### 5.1 QuantLib::AnalyticESEngine Class Reference

Analytic engine of equity swap.

#include <analyticequityswapengine.hpp>

Inheritance diagram for QuantLib::AnalyticESEngine:



**Public Member Functions** 

#### Constructors

- AnalyticESEngine ()
- · void calculate () const

Calculate the npv of the equity swap.

#### 5.1.1 Detailed Description

Analytic engine of equity swap.

This class is the analytic engine of equity swap. The cash expected cash flow from the equity leg and the fixed coupon leg is discounted back to reference date.

#### 5.1.2 Member Function Documentation

5.1.2.1 void AnalyticESEngine::calculate ( ) const

Calculate the npv of the equity swap.

The swap leg is already constructed the calculation is just discounting the cash flow back to the current time.

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

16 Class Documentation

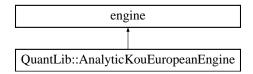
- · analyticequityswapengine.hpp
- · analyticequityswapengine.cpp

#### 5.2 QuantLib::AnalyticKouEuropeanEngine Class Reference

Analytic engine for european option, whose underlying stock process is a kou process.

#include <analytickoueuropeanengine.hpp>

Inheritance diagram for QuantLib::AnalyticKouEuropeanEngine:



#### Constructor

- AnalyticKouEuropeanEngine (const boost::shared\_ptr< KouProcess > &process, const Real &tolerance=0.0001)
- void calculate () const override calculate the european option price

#### 5.2.1 Detailed Description

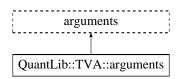
Analytic engine for european option, whose underlying stock process is a kou process.

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

- · analytickoueuropeanengine.hpp
- · analytickoueuropeanengine.cpp

#### 5.3 QuantLib::TVA::arguments Class Reference

Inheritance diagram for QuantLib::TVA::arguments:



#### **Public Member Functions**

· virtual void validate () const

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

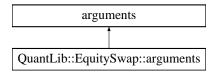
tva.hpp

#### 5.4 QuantLib::EquitySwap::arguments Class Reference

Arguments for equity swap calculation

#include <equityswap.hpp>

Inheritance diagram for QuantLib::EquitySwap::arguments:



#### **Public Member Functions**

· void validate () const

#### **Public Attributes**

- boost::shared\_ptr< QuantLib::YieldTermStructure > discountCurve\_
- Type type
- · QuantLib::Date referenceDate\_
- QuantLib::Date startDate\_
- QuantLib::Real spotPrice\_
- QuantLib::Real startPrice\_
- QuantLib::Integer amount\_
- QuantLib::Rate fixedRate\_
- QuantLib::Rate dividend\_
- QuantLib::Time maturity\_
- QuantLib::Real sigma\_ QuantLib::Real cumulativeCoupoon\_
- QuantLib::Real cumulativeDividend\_
- QuantLib::Frequency freq\_
- boost::shared\_ptr< QuantLib::Schedule > schedule\_
- size t count
- std::vector< QuantLib::Time > yearfraction\_

#### 5.4.1 Detailed Description

Arguments for equity swap calculation

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

· equityswap.hpp

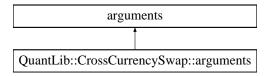
#### 5.5 QuantLib::CrossCurrencySwap::arguments Class Reference

Arguments for cross currency swap calculation

#include <crosscurrencyswap.hpp>

 $Inheritance\ diagram\ for\ QuantLib:: Cross Currency Swap:: arguments:$ 

18 Class Documentation



#### **Public Member Functions**

• void validate () const

#### **Public Attributes**

- Type type
- Date referenceDate
- · Rate fxRate
- Volatility fxVol
- Real payNominal
- Real receiveNominal
- DayCounter payDayCount
- DayCounter receiveDayCount
- std::vector< Date > payDates
- std::vector< Date > receiveDates
- std::vector< Real > payCoupons
- std::vector< Real > receiveCoupons

#### 5.5.1 Detailed Description

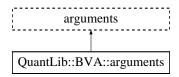
Arguments for cross currency swap calculation

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

- · crosscurrencyswap.hpp
- crosscurrengyswap.cpp

#### 5.6 QuantLib::BVA::arguments Class Reference

Inheritance diagram for QuantLib::BVA::arguments:



#### **Public Member Functions**

• void validate () const

#### **Public Attributes**

- boost::shared ptr< const Counterparty > self
- boost::shared\_ptr< const Counterparty > counterparty
- std::vector< boost::shared\_ptr< const Instrument >> portfolio
- Handle< YieldTermStructure > discountCurve

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

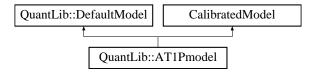
bva.hpp

#### 5.7 QuantLib::AT1Pmodel Class Reference

Analytic tractable first default model.

#include <at1pdefaultmodel.hpp>

Inheritance diagram for QuantLib::AT1Pmodel:



#### **Public Member Functions**

- const QuantLib::Probability defaultProbability (QuantLib::Time T) const override
   Calculate the default probability according to AT1P model.
- const QuantLib::Time defaultTime (const Path &path) const override

Return the default time of the path according to AT1P model.

• boost::shared ptr< StochasticProcess1D > process () const override

Return the process of counterparty after calibration.

void calibrate (const std::vector< boost::shared\_ptr< CalibrationHelper > > &, OptimizationMethod &method, const EndCriteria &endCriteria, const Constraint &constraint=Constraint(), const std::vector<</li>
 Real > &weights=std::vector< Real >(), const std::vector< bool > &fixParameters=std::vector< bool >()) override

Calibrate the at1p model volatility using sets of CDSs as calibration helper.

#### destructors

•  $\sim$ AT1Pmodel ()

#### Constructors

AT1Pmodel (boost::shared\_ptr< const std::vector< QuantLib::Period >> periods, boost::shared\_ptr
 QuantLib::YieldTermStructure > discountcurve, const QuantLib::Real dividend=0, const QuantLib::Real H\_V=0.4, const QuantLib::Real B=0, const QuantLib::Real x0=1, boost::shared\_ptr< QuantLib::Array > volatility\_=nullptr)

20 Class Documentation

#### 5.7.1 Detailed Description

Analytic tractable first default model.

The class support the calculations relating to the analytical tractable first passage time model under the AT1P setting the volatility of the the firm value, is piecewise constant.

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

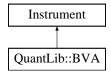
- · at1pdefaultmodel.hpp
- · at1pdefaultmodel.cpp

#### 5.8 QuantLib::BVA Class Reference

Bilateral value adjustment manager class.

```
#include <bva.hpp>
```

Inheritance diagram for QuantLib::BVA:



#### Classes

- · class arguments
- · class engine
- · class results

**Public Member Functions** 

#### **Constructors & Destructors**

• BVA (const boost::shared\_ptr< Counterparty > &self, const boost::shared\_ptr< Counterparty > &counterparty, const std::vector< boost::shared\_ptr< const Instrument >> &portfolio, const Matrix &correlation, const Handle< YieldTermStructure > &discountCurve)

#### Inspectors

- boost::shared ptr< Counterparty > self ()
- boost::shared\_ptr< Counterparty > counterparty ()
- Handle< YieldTermStructure > discountCurve ()

#### **Public interface**

• const Real CVA () const

Calculate the CVA and return the result, or return the result directly if it has calculated.

• const Real DVA () const

Calculate the DVA and return the result, or return the result directly if it has calculated.

- void setupArguments (PricingEngine::arguments \*args) const override
- void **fetchResults** (const PricingEngine::results \*) const
- bool isExpired () const

#### **Protected Attributes**

- std::vector< boost::shared ptr< const Instrument > > portfolio
- const boost::shared\_ptr< Counterparty > self\_
- const boost::shared\_ptr< Counterparty > counterparty\_
- Matrix correlation
- Handle< YieldTermStructure > discountCurve\_
- Real CVA
- Real DVA

#### 5.8.1 Detailed Description

Bilateral value adjustment manager class.

This class inherits from Instrument class, responsible for calculation bilateral value adjustment (includes  $CV \leftarrow A$ , DVA) of a single instrument or portfolio. The arguments are the information of two counterparties (issuer and investor) using the **Counterparty** (p. 27) class and the information of underlying instrument or portfolio using their corresponding model class. The Pricingengine uses tvaengine based on Monte Carlo simulation framework. The results are including CVA, DVA.

#### See also

```
counterparty.hpp (p. 75)
mctvamodel.hpp (p. 86)
mcbvaengine.hpp (p. 81)
```

#### 5.8.2 Constructor & Destructor Documentation

5.8.2.1 QuantLib::BVA::BVA (const boost::shared\_ptr< Counterparty > & self, const boost::shared\_ptr< Counterparty > & counterparty, const std::vector< boost::shared\_ptr< const Instrument >> & portfolio, const Matrix & correlation, const Handle< YieldTermStructure > & discountCurve ) [inline]

Creates an instance of **BVA** (p. 20) manager using two counterparties, instruments (single instrument or portfolio), correlation matrix of the counterparties and instruments, and the yield term structure.

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

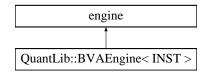
· bva.hpp

#### 5.9 QuantLib::BVAEngine < INST > Class Template Reference

Base class for bva engine.

```
#include <bvaengine.hpp>
```

Inheritance diagram for QuantLib::BVAEngine < INST >:



22 Class Documentation

#### **Public Member Functions**

#### **Constructors & Destructors**

BVAEngine (Time endTime, const std::vector< boost::shared\_ptr< const Counterparty >> &counterparties, const std::vector< boost::shared\_ptr< const MCTVAModel >> &bvaModels, const Handle
 YieldTermStructure > &discountCurve, const boost::shared\_ptr< ShortRateTermStructure > &short RateDynamics)

• virtual  $\sim$ BVAEngine ()

#### Inspector

- std::vector< boost::shared\_ptr< const Counterparty >> counterparties () const
- const Time endTime () const
- const Handle< YieldTermStructure > & discountCurve () const
- $\bullet \ \, \text{const boost::shared\_ptr} < \textbf{ShortRateTermStructure} > \& \ \, \textbf{shortRateDynamics} \ () \ \, \text{const} \\$
- const std::vector< boost::shared ptr< const MCTVAModel > > & models () const

#### **Protected Attributes**

- std::vector< boost::shared\_ptr< const Counterparty >> counterparties\_
- Handle< YieldTermStructure > discountCurve
- boost::shared ptr< ShortRateTermStructure > shortRateDynamics
- std::vector< boost::shared\_ptr< const MCTVAModel > > bvaModels\_
- Time endTime

#### 5.9.1 Detailed Description

template<typename INST>class QuantLib::BVAEngine< INST>

Base class for bva engine.

This class is base class for bva engine, which is used to set some basic information. Each derived class should implement its own method to calculate bva.

#### 5.9.2 Constructor & Destructor Documentation

5.9.2.1 template < typename INST > QuantLib::BVAEngine < INST >::BVAEngine ( Time endTime, const std::vector < boost::shared\_ptr < const Counterparty >> & counterparties, const std::vector < boost::shared\_ptr < const MCTVAModel >> & bvaModels, const Handle < YieldTermStructure > & discountCurve, const boost::shared\_ptr < ShortRateTermStructure > & shortRateDynamics ) [inline]

Create an instance of **BVA** (p. 20) Engine using the information of the two counterparties using **Counterparty** (p. 27) class and underlying instruments using instrument model class.

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

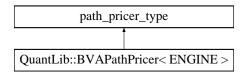
bvaengine.hpp

#### 5.10 QuantLib::BVAPathPricer < ENGINE > Class Template Reference

Path pricer class used to calculate the BVA (p. 20) on each Multi-path for BVAEngine (p. 21) class.

#include <bvapathpricer.hpp>

 $Inheritance\ diagram\ for\ QuantLib:: BVAPathPricer < ENGINE >:$ 



#### **Public Types**

- typedef ENGINE::path\_pricer\_type::argument\_type argument\_type
- typedef ENGINE::path\_pricer\_type::result\_type result\_type

#### **Public Member Functions**

#### Constructors

BVAPathPricer (const boost::shared\_ptr< const ENGINE > &engine)

#### **Destructors**

∼BVAPathPricer ()

#### Operator overload

result\_type operator() (const argument\_type &multiPath) const
 Calculate the TVA (p. 62) (includes CVA, DVA and FVA) given the Multi-path.

#### **Protected Attributes**

const boost::weak ptr< const ENGINE > engine

#### 5.10.1 Detailed Description

template<typename ENGINE>class QuantLib::BVAPathPricer< ENGINE>

Path pricer class used to calculate the BVA (p. 20) on each Multi-path for BVAEngine (p. 21) class.

This class is the core part for **BVAEngine** (p. 21) class, which is used to calculate the **BVA** (p. 20) (includes CVA, DVA) on each Multi-path given the instruments (single instrument or portfolio), based on the definition of **BVA** (p. 20) (includes CVA, DVA) in Brigo's book: "Counterparty Credit Risk, Collateral and Funding".

If the underlying is a portfolio, netting is considered.

See also

#### MCBVAEngine (p. 45)

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

bvapathpricer.hpp

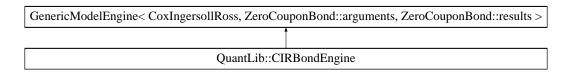
24 Class Documentation

#### 5.11 QuantLib::CIRBondEngine Class Reference

A CIRBondEngine (p. 24) is a bond pricing engine based on CIR model.

#include <cirbondengine.hpp>

Inheritance diagram for QuantLib::CIRBondEngine:



#### **Public Member Functions**

· void calculate () const

Calculates the model value of bond price using CIR model.

#### **Constructors & Destructors**

CIRBondEngine (const boost::shared\_ptr< CoxIngersollRoss > &model, const Handle< YieldTerm
 — Structure > &termStructure)

Creates an instance of CIRBondEngine (p. 24) using a CIR model instance and a yield term structure.

#### 5.11.1 Detailed Description

A CIRBondEngine (p. 24) is a bond pricing engine based on CIR model.

This class is to calculate bond prices using CIR model, which are used by the calibration helpers of CoxIngersoll ← Ross model. Internally, this class calculate the bond price using

$$e^{b(t)-a(t)r_t}$$

See also

CoxIngersollRoss

ZerocouponbondHelper (p. 68)

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

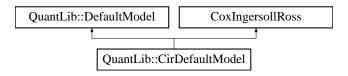
- cirbondengine.hpp
- · cirbondengine.cpp

#### 5.12 QuantLib::CirDefaultModel Class Reference

Intensity default model based on the CIR process.

#include <cirdefaultmodel.hpp>

Inheritance diagram for QuantLib::CirDefaultModel:



**Public Member Functions** 

#### **Constructors & Destructors**

• CirDefaultModel (Rate r0=0.05, Real theta=0.1, Real k=0.1, Real sigma=0.1)

Creates an instance of CIR default model by setting parameters as default initial values.

•  $\sim$ CirDefaultModel ()

#### **Public interface**

· const Probability defaultProbability (Time t) const override

Calculate the default probability according to CIR dedault model.

- const Time defaultTime (const Path &path) const override
- boost::shared\_ptr< StochasticProcess1D > process () const override

Return the process of counterparty after calibration.

void calibrate (const std::vector< boost::shared\_ptr< CalibrationHelper > > &, OptimizationMethod &method, const EndCriteria &endCriteria, const Constraint &constraint=Constraint(), const std::vector<</li>
 Real > &weights=std::vector<</li>
 Real >(), const std::vector< bool > &fixParameters=std::vector< bool >()) override

Calibrate the CIR default model volatility using sets of CDSs as calibration helper.

#### 5.12.1 Detailed Description

Intensity default model based on the CIR process.

The model uses CIR process to describe defalut intensity, it should be calibrated by CDS data.

#### 5.12.2 Member Function Documentation

5.12.2.1 const Time CirDefaultModel::defaultTime ( const Path & path ) const [override], [virtual]

Return the default time of the path according to CIR dedault model, based on Monte Carlo method Implements **QuantLib::DefaultModel** (p. 33).

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

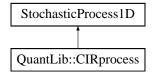
- · cirdefaultmodel.hpp
- · cirdefaultmodel.cpp

#### 5.13 QuantLib::CIRprocess Class Reference

CIR process class.

#include <cirprocess.hpp>

Inheritance diagram for QuantLib::CIRprocess:



26 Class Documentation

#### **Public Types**

enum Discretization { Euler, Milstein, ImplicitMilstein, NonCentralChiSquareVariance }

An enum type for Discretization method.

#### **Public Member Functions**

#### **Constructors & Destructors**

- CIRprocess (Real mean, Real speed, Volatility sigma, Real x0=0.0, Discretization discretization=Implicit
   — Milstein)
- ∼CIRprocess ()

#### Inspectors

- · Real x0 () const
- · Real revertSpeed () const
- Real revertLevel () const
- · Volatility volatility () const

#### **Public interface**

• Real drift (Time t, Real x) const

Return the drift term of the process.

• Real diffusion (Time t, Real x) const

Return the diffusion of the process.

• Real evolve (Time t0, Real x0, Time dt, Real dw) const

Return simulation value of the process at time t0+dts.

#### 5.13.1 Detailed Description

CIR process class.

This class describes a square-root process governed by

$$dx = a(b - x_t)dt + \sigma\sqrt{x_t}dW_t.$$

The process is used to model CIR process which provides more precise evolve method.

#### Remarks

The Implict Milstein discretization scheme is recommended to use for efficient simulation. Implict Milstein is got from classical Milstein, in which we replace the drift term  $a(b-r_t)dt$  with  $a(b-r_{t+dt})dt$ . We then obtain:

$$x_{t+dt} = \frac{x_t + abdt + \sigma\sqrt{x_t}\sqrt{dt}dw + \frac{1}{4}\sigma^2dt(dw^2 - 1)}{1 + adt}$$

#### 5.13.2 Member Enumeration Documentation

#### 5.13.2.1 enum QuantLib::CIRprocess::Discretization

An enum type for Discretization method.

#### **Enumerator**

Euler Discretization Scheme.

Milstein Discretization Scheme.

ImplicitMilstein Implicit Milstein Discretization Scheme.

NonCentralChiSquareVariance Non-Central Chi-Sqaure Distribution.

#### 5.13.3 Constructor & Destructor Documentation

5.13.3.1 CIRprocess::CIRprocess ( Real *mean*, Real *speed*, Volatility *sigma*, Real *x0* = 0 . 0, Discretization *discretization* = ImplicitMilstein )

Creates an instance of CIR process using the four parameters long-term mean, revert speed, volatility and initial value

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

- · cirprocess.hpp
- · cirprocess.cpp

# 5.14 QuantLib::Counterparty Class Reference

Counterparty (p. 27) class.

#include <counterparty.hpp>

## **Public Member Functions**

· double fundingSpread (Time t) const

Return the funding spread at time t.

#### **Constructors & Destructors**

• Counterparty (int settlementDays, double recoveryRate, const boost::shared\_ptr< const std::vector< double >> &cdsSpreads, const boost::shared\_ptr< const std::vector< Period >> &cdsTenors, const boost::shared\_ptr< YieldTermStructure > &discountCurve, const boost::shared\_ptr< Default 

Model > &model\_, const QuantLib::Frequency &freq=QuantLib::Quarterly, const QuantLib::Business 
DayConvention &busDayConvention=QuantLib::Following, const QuantLib::DateGeneration::Rule &dateGenerationRule=QuantLib::DateGeneration::TwentiethIMM, const QuantLib::Date &reference 
Date=QuantLib::Date(), const QuantLib::DayCounter &dayCounter=QuantLib::Actual365Fixed(), const QuantLib::Calendar &calendar=QuantLib::TARGET(), const boost::shared\_ptr< YieldTermStructure > &defaultdiscountCurve=0)

Creates an instance of Counterparty (p. 27) using its CDS information and the default model.

∼Counterparty ()

# **Public interface**

const Real getDefaultProb (const Time &t) const

Return default probability within time t based on the default model.

const Time getDefaultTime (const Path &path) const

Return default time given a path describing the evolution of default state variable.

- const Time defaultTimeGenerator (const Probability &prob, const double &epsilon=1e-2) const Return random number of default time.
- void **modelCalibrate** (OptimizationMethod &method, const EndCriteria &endCriteria, bool need ← Calibrate=true, const Constraint &constraint=Constraint(), const std::vector< Real > &weights=std ← ::vector< Real > (), const std::vector< bool > &fixParameters=std::vector< bool > ())

Calibrate default model.

# Inspectors

- const boost::shared\_ptr< const  $\textbf{DefaultModel} > \& \ \textbf{getModel}$  () const
  - Return shared pointer of default model.
- const double getRecoveryRate () const
- boost::shared\_ptr< YieldTermStructure > discountCurve () const
- const Date referenceDate () const
- $\bullet \ \ boost:: shared\_ptr < Stochastic Process 1D > \textbf{createDefaultProcess} \ () \ const$

Return shared pointer of process describing default state variable.

## 5.14.1 Detailed Description

## Counterparty (p. 27) class.

This class is used to describe defalut behavior of investor and issuer involved in a transaction, based on the default model.

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

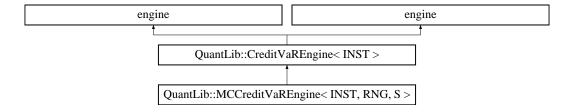
- · counterparty.hpp
- · counterparty.cpp

# 5.15 QuantLib::CreditVaREngine < INST > Class Template Reference

base class for credit VaR engine.

#include <creditvarengine.hpp>

Inheritance diagram for QuantLib::CreditVaREngine < INST >:



#### **Public Member Functions**

- CreditVaREngine (Time endTime, const std::vector< boost::shared\_ptr< const Counterparty >> &counterparties, const std::vector< boost::shared\_ptr< const MCTVAModel >> &creditvarModels, const Handle< YieldTermStructure > &discountCurve, const boost::shared\_ptr< ShortRateTermStructure > &shortRateDynamics)
- std::vector< boost::shared\_ptr< const **Counterparty** >> **counterparties** () const
- const Time endTime () const
- const Handle< YieldTermStructure > & discountCurve () const
- const boost::shared\_ptr< ShortRateTermStructure > & shortRateDynamics () const
- const std::vector< boost::shared\_ptr< const MCTVAModel > >& models () const
- Time endTime () const

# **Constructors & Destructors**

- CreditVaREngine (const boost::shared\_ptr< const INST > instrument, const boost::shared\_ptr< const</li>
   Counterparty > issuer, const Time &endTime, const Handle< YieldTermStructure > &discountCurve)
- virtual ~CreditVaREngine ()

# **Protected Attributes**

- std::vector< boost::shared\_ptr< const Counterparty >> counterparties\_
- Handle < YieldTermStructure > discountCurve
- boost::shared\_ptr< ShortRateTermStructure > shortRateDynamics\_
- std::vector< boost::shared ptr< const MCTVAModel > > CreditVarModels
- Time endTime\_
- const Time endTime\_
- const boost::shared ptr< const INST > instrument
- const boost::shared\_ptr< const Counterparty > issuer\_
- const Handle
   YieldTermStructure > discountCurve\_

# 5.15.1 Detailed Description

template<typename INST>class QuantLib::CreditVaREngine< INST>

base class for credit VaR engine.

This class is base class for credit VaR calculation, which is used to set some basic information. Each derived class should implement its own method to calculate credit VaR.

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

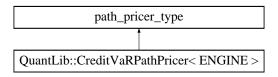
• pricingengines/credit/creditvarengine.hpp

# 5.16 QuantLib::CreditVaRPathPricer < ENGINE > Class Template Reference

Abstract base class used to calculate default value of each path.

#include <creditvarpathpricer.hpp>

Inheritance diagram for QuantLib::CreditVaRPathPricer< ENGINE >:



# **Public Types**

- typedef ENGINE::path\_pricer\_type::argument\_type argument\_type
- typedef ENGINE::path\_pricer\_type::result\_type result\_type

#### **Public Member Functions**

#### **Destructors**

∼CreditVaRPathPricer ()

# Operator overload

• result\_type **operator()** (const argument\_type &multiPath) const

# **Protected Member Functions**

#### **Constructors**

CreditVaRPathPricer (const boost::shared\_ptr< const ENGINE > &engine)

# **Protected interface**

- const boost::weak\_ptr< const ENGINE > engine\_
  - Return weak pointer of pricing engine.
- virtual Real defaultNPV (const MultiPath &path, Time defaultTime) const =0
   Return default NPV at each path, must be implemented in derived class.

# 5.16.1 Detailed Description

template<typename ENGINE>class QuantLib::CreditVaRPathPricer< ENGINE>

Abstract base class used to calculate default value of each path.

This class is used to calculate default value of each path. To use this class, derived class must implement the pure virtual function defaultNPV.

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

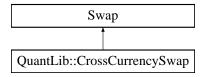
· creditvarpathpricer.hpp

# 5.17 QuantLib::CrossCurrencySwap Class Reference

fixed-fixed cross currency swap class

#include <crosscurrencyswap.hpp>

Inheritance diagram for QuantLib::CrossCurrencySwap:



## **Classes**

· class arguments

Arguments for cross currency swap calculation

· class engine

base class for cross currency swap pricing engine

· class results

Results for cross currency swap calculation

# **Public Types**

enum Type { payDomestic = 0, payForeign }

**Public Member Functions** 

# **Constructors & Destructors**

- CrossCurrencySwap (Type type, Rate fxRate, Volatility fxVol, const Date &referenceDate, Real pay
   — Nominal, Real receiveNominal, const Schedule &paySchedule, Rate payRate, const DayCounter &pay
   — DayCount, const Schedule &receiveSchedule, Rate receiveRate, const DayCounter &receiveDayCount,
   boost::shared\_ptr< Currency > payCurrency, boost::shared\_ptr< Currency > receiveCurrency, boost
   — ::optional< BusinessDayConvention > paymentConvention=boost::none)
- $\bullet \ \sim \! \mathbf{CrossCurrencySwap} \ ()$

# Instrument interface

- void setupArguments (PricingEngine::arguments \*args) const
- void fetchResults (const PricingEngine::results \*r) const

## Inspectors

• Type type () const

Return type of cross currency swap, which should be payDomestic or payForeign.

- · const Rate fxRate () const
- · const Volatility fxVolatility () const
- · const Date & referenceDate () const
- · Real payNominal () const
- Real receiveNominal () const
- · const Schedule & paySchedule () const
- Rate payRate () const
- const DayCounter & payDayCount () const
- const Schedule & receiveSchedule () const
- · Rate receiveRate () const
- const DayCounter & receiveDayCount () const
- BusinessDayConvention paymentConvention () const
- const Leg & payLeg () const
- const Leg & receiveLeg () const

# 5.17.1 Detailed Description

fixed-fixed cross currency swap class

This class is used to describe fixed-fixed cross currency swap

## 5.17.2 Member Enumeration Documentation

5.17.2.1 enum QuantLib::CrossCurrencySwap::Type

An enum type for Swap Type. The documentation block cannot be put after the enum!

Enumerator

payDomestic pay
payForeign receive

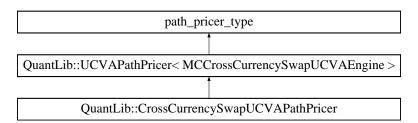
The documentation for this class was generated from the following files:

- crosscurrencyswap.hpp
- · crosscurrengyswap.cpp

# 5.18 QuantLib::CrossCurrencySwapUCVAPathPricer Class Reference

UCVA Path Pricer for fixed-fixed cross currency swap.

Inheritance diagram for QuantLib::CrossCurrencySwapUCVAPathPricer:



#### **Public Member Functions**

## **Constructors & Destructors**

 $\bullet \ \sim \! \textbf{CrossCurrencySwapUCVAPathPricer} \ ()$ 

## **Protected Member Functions**

• virtual Real **defaultNPV** (const MultiPath &path, Time defaultTime) const Calculate the NPV of the remaining cash flow at default time  $\tau$ .

#### **Additional Inherited Members**

# 5.18.1 Detailed Description

UCVA Path Pricer for fixed-fixed cross currency swap.

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

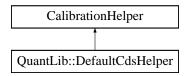
· mccrosscurrencyswapucvaengine.cpp

# 5.19 QuantLib::DefaultCdsHelper Class Reference

CDS helpers to calibrate default model.

#include <defaultcdshelper.hpp>

Inheritance diagram for QuantLib::DefaultCdsHelper:



# **Public Member Functions**

## **Constructors & Destructors**

- DefaultCdsHelper (Time cdsTime, const Handle< Quote > &impliedDefaultProb, const Handle< Yield←
  TermStructure > &termStructure, const boost::shared\_ptr< CalibratedModel > &model)
- ∼DefaultCdsHelper ()

#### **Public interface**

• Real modelValue () const override

Calculate default probability based on default model.

· Real blackPrice (Volatility volatility) const override

Return default probability boostrapped from CDS data.

# 5.19.1 Detailed Description

CDS helpers to calibrate default model.

Calculate default probability based on default and market data, it will be used to calibrate default model.

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

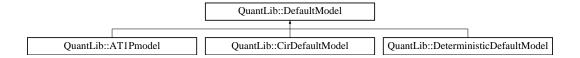
- · defaultcdshelper.hpp
- · defaultcdshelper.cpp

# 5.20 QuantLib::DefaultModel Class Reference

abstract base class for default model

#include <defaultmodel.hpp>

Inheritance diagram for QuantLib::DefaultModel:



**Public Member Functions** 

#### **Constructors & Destructors**

- · DefaultModel ()
- virtual ~DefaultModel ()

# **Publice interface**

- virtual const Probability defaultProbability (Time t) const =0
   Calculate the default probability at time t according to the given dedault model.
- virtual const Time **defaultTime** (const Path &path) const =0
- virtual boost::shared\_ptr< StochasticProcess1D > process () const =0
   Return stochastic process describing default state variable.

#### 5.20.1 Detailed Description

abstract base class for default model

Abstract base class for default model, define pure virtual functions need to be implemented in derived class.

# 5.20.2 Member Function Documentation

5.20.2.1 virtual const Time QuantLib::DefaultModel::defaultTime ( const Path & path ) const [pure virtual]

Return the default time of the path according to the given dedault model, based on Monte Carlo method

Implemented in QuantLib::AT1Pmodel (p. 19), QuantLib::CirDefaultModel (p. 25), and QuantLib:: DeterministicDefaultModel (p. 34).

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

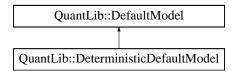
· defaultmodel.hpp

# 5.21 QuantLib::DeterministicDefaultModel Class Reference

Intensity default model based on deterministic default intensity.

#include <deterministicdefaultmodel.hpp>

Inheritance diagram for QuantLib::DeterministicDefaultModel:



**Public Member Functions** 

#### **Constructors & Destructors**

- DeterministicDefaultModel ()
- ∼DeterministicDefaultModel ()

# Inspectors

- virtual const Probability defaultProbability (Time t) const override

Calculate the default probability.

• virtual const Time defaultTime (const Path &path) const override

Return the default time of the path based on Monte Carlo method.

virtual boost::shared\_ptr< StochasticProcess1D > process () const override

Return a geometric brownian motion with zero drift and zero volatility.

## 5.21.1 Detailed Description

Intensity default model based on deterministic default intensity.

The model uses deterministic function to describe defalut intensity, it is boostrapped from CDS data.

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

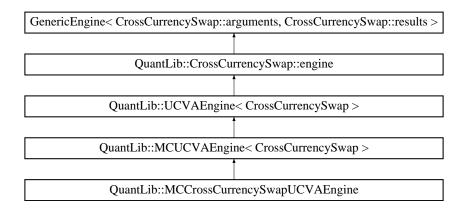
- · deterministicdefaultmodel.hpp
- · deterministicdefaultmodel.cpp

# 5.22 QuantLib::CrossCurrencySwap::engine Class Reference

base class for cross currency swap pricing engine

#include <crosscurrencyswap.hpp>

Inheritance diagram for QuantLib::CrossCurrencySwap::engine:



# 5.22.1 Detailed Description

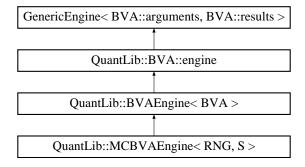
base class for cross currency swap pricing engine

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

· crosscurrencyswap.hpp

# 5.23 QuantLib::BVA::engine Class Reference

Inheritance diagram for QuantLib::BVA::engine:



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

· bva.hpp

# 5.24 QuantLib::EquitySwap::engine Class Reference

equity option engine base class

#include <equityswap.hpp>

Inheritance diagram for QuantLib::EquitySwap::engine:



# 5.24.1 Detailed Description

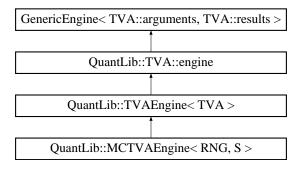
equity option engine base class

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

· equityswap.hpp

# 5.25 QuantLib::TVA::engine Class Reference

Inheritance diagram for QuantLib::TVA::engine:



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

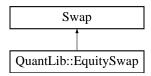
· tva.hpp

# 5.26 QuantLib::EquitySwap Class Reference

equity swap class

#include <equityswap.hpp>

Inheritance diagram for QuantLib::EquitySwap:



## Classes

· class arguments

Arguments for equity swap calculation

class engine

equity option engine base class

• class results

Results class of equity swap

# **Public Types**

• enum Type { Receiver = -1, Payer = 1 }

# **Public Member Functions**

size\_t Count () const

return the number of payoffs

• Real getMaturity () const

return the maturity

Real getDividendYield () const

return the continuous dividend yield of the stock

• Real getSpotPrice () const

return the current spot price of the stock

· Real getVolatility () const

return the constant volatility of the stock

• Calendar getCalendar () const

return the calendar of used in the calculation

#### **Destructor**

∼EquitySwap ()

#### Constructors

- EquitySwap (Type type, QuantLib::Date startdate\_, QuantLib::Date referencedate\_, QuantLib::Real startprice\_, QuantLib::Real spotprice\_, QuantLib::Integer amount\_, QuantLib::Real cumulativecoupoon← \_, QuantLib::Real cumulativedividend\_, QuantLib::Rate fixedrate\_, QuantLib::Rate dividend\_, Quant← Lib::Time maturity, QuantLib::Real sigma\_, boost::shared\_ptr< QuantLib::YieldTermStructure > discountcurve\_, const QuantLib::Frequency &freq=QuantLib::Semiannual, const QuantLib::Business← DayConvention &busDayConvention=QuantLib::Following, const QuantLib::DateGeneration::Rule &rule← \_=QuantLib::DateGeneration::Backward, const QuantLib::Calendar &calendar=TARGET())
- EquitySwap (const EquitySwap &)

# 5.26.1 Detailed Description

equity swap class

This class defines the instrument equity swap Equity swap is a contract between two parties, which one party pays fixed rate coupon on the notional, the other party pays dividend return from the stock. At maturity, the notional is exchanged with the stock. For simplicity, the dividend yield and the volatility of the stock is supposed constant.

See Counterparty (p. 27) Credit Risk, Collateral and Funding(Brigo 2013) page 169 for more detailed descriptions.

The two legs of the equity swap is initialized using the private "initialize" method, whenever an equity swap is constructed.

## 5.26.2 Member Enumeration Documentation

5.26.2.1 enum QuantLib::EquitySwap::Type

An enum type. The documentation block cannot be put after the enum!

Enumerator

Receiver receive fixed coupon

Payer pay fixed coupon

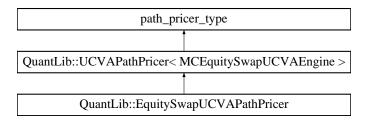
The documentation for this class was generated from the following files:

- equityswap.hpp
- equityswap.cpp

# 5.27 QuantLib::EquitySwapUCVAPathPricer Class Reference

UCVA Path Pricer for equity swap.

Inheritance diagram for QuantLib::EquitySwapUCVAPathPricer:



#### **Public Member Functions**

## **Constructors & Destructors**

- EquitySwapUCVAPathPricer (const boost::shared\_ptr< const MCEquitySwapUCVAEngine > &engine)
- ~EquitySwapUCVAPathPricer ()

# **Protected Member Functions**

• virtual Real **defaultNPV** (const MultiPath &path, Time defaultTime) const Calculate the NPV of the remaining cash flow at default time  $\tau$ .

## **Additional Inherited Members**

# 5.27.1 Detailed Description

UCVA Path Pricer for equity swap.

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

· mcequityswapucvaengine.cpp

# 5.28 QuantLib::ExactSimulation < SIM, RNG, S, C > Class Template Reference

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

· exactsimulation.hpp

# 5.29 QuantLib::ExposureModel Class Reference

# **Public Member Functions**

- virtual Real NPV (const MultiPath &path, Time defaultTime) const =0
- $\bullet \ \ virtual \ std:: vector < boost:: shared\_ptr < \ Stochastic Process 1D >> \textbf{processes} \ () \ const = 0 \\$

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

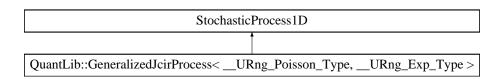
exposuremodel.hpp

# 5.30 QuantLib::GeneralizedJcirProcess< \_\_URng\_Poisson\_Type, \_\_URng\_Exp\_Type > Class Template Reference

Jump diffusion CIR process class.

#include <jcirprocess.hpp>

Inheritance diagram for QuantLib::GeneralizedJcirProcess< URng Poisson Type, URng Exp Type >:



## **Public Member Functions**

#### **Constructors & Destructors**

• GeneralizedJcirProcess (Real mean, Real speed, Real jumpIntensity, Real jumpMean, Volatility sigma, Real x0=0.0, const \_\_URng\_Poisson\_Type &URng\_Poisson=\_\_URng\_Poisson\_Type(Seed ← Generator::instance().get()), const \_\_URng\_Exp\_Type &URng\_Exp=\_\_URng\_Poisson\_Type(Seed ← Generator::instance().get()), const boost::shared\_ptr< discretization > &d=boost::make\_shared< Euler ← Discretization >())

#### **Public interface**

• const Real drift (Time t, Real x) const

Return the drift term of the process.

• const Real diffusion (Time t, Real x) const

Return the diffusion of the process.

• const Real evolve (Time t0, Real x0, Time dt, Real dw) const

Return simulation value of the process at time t0+dt.

- const Real x0 () const
- const Real mean () const
- const Real speed () const
- const Real jumpIntensity () const
- const Real jumpMean () const
- · const Volatility volatility () const

# 5.30.1 Detailed Description

template<typename \_\_URng\_Poisson\_Type, typename \_\_URng\_Exp\_Type>class QuantLib::GeneralizedJcirProcess< \_\_U $\leftarrow$  Rng\_Poisson\_Type, \_\_URng\_Exp\_Type>

Jump diffusion CIR process class.

This class describes a jump diffusion CIR process governed by

$$dx = a(b - x_t)dt + \sigma\sqrt{x_t}dW_t + dJ_t.$$

It is an extension of the classical CIR model. The jumps of the JCIR are introduced with the help of a pure-jump Levy process  $J_t$ .

## 5.30.2 Constructor & Destructor Documentation

5.30.2.1 template < typename \_\_URng\_Poisson\_Type , typename \_\_URng\_Exp\_Type > QuantLib::GeneralizedJcir ←
Process < \_\_URng\_Poisson\_Type, \_\_URng\_Exp\_Type >::GeneralizedJcirProcess ( Real mean, Real speed,
Real jumplntensity, Real jumpMean, Volatility sigma, Real x0 = 0 . 0, const \_\_URng\_Poisson\_Type & URng\_Poisson\_Type (SeedGenerator::instance() .get()), const \_\_URng\_Exp\_Type &
URng\_Exp = \_\_URng\_Poisson\_Type (SeedGenerator::instance() .get()), const
boost::shared ptr < discretization > & d = boost::make shared < EulerDiscretization > () )

Creates an instance of CIR process using the four parameters: long-term mean, revert speed, volatility, initial value and the additional jump information

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

#### · jcirprocess.hpp

# 5.31 QuantLib::GeneralizedKouProcess< \_\_URng\_Poisson\_Type, \_\_URng\_Double← ExpDist\_\_Type > Class Template Reference

Kou process class.

```
#include <kouprocess.hpp>
```

```
BlackScholesMertonProcess

QuantLib::GeneralizedKouProcess< __URng_Poisson_Type, __URng_DoubleExpDist_Type >
```

#### **Public Member Functions**

# **Constructors & Destructors**

• GeneralizedKouProcess (const Handle< Quote > &x0, const Handle< YieldTermStructure > &dividendTS, const Handle< YieldTermStructure > &riskFreeTS, const Handle< BlackVolTermStructure > &blackVolTS, const Real jumpIntensity, const Real posProbability, const Real posJumpMean, const Real negJumpMean, const \_\_URng\_Poisson\_Type &URng\_Poisson=\_URng\_Poisson\_Type(Seed ← Generator::instance().get()), const \_\_URng\_DoubleExpDist\_\_Type &URng\_DoubleExpDist=\_URng\_← Poisson\_Type(SeedGenerator::instance().get()), const boost::shared\_ptr< discretization > &d=boost← ::shared\_ptr< discretization > (new EulerDiscretization))

# **Public interface**

- Real evolve (Time t0, Real x0, Time dt, Real dw) const override
   Return simulation value of the process at time t0+dt.
- const Real jumpIntensity () const
- const Real posProbability () const
- · const Real posJumpMean () const
- const Real negJumpMean () const

## 5.31.1 Detailed Description

Kou process class.

This class describes a double exponential jump process, initiated by Steven KOU, which is a compromise between reality and tractability. It gives an explanation of the two empirical phenomena which received much attention in financial markets: the asymmetric leptokurtic feature and the volatility smile. It permits to obtain analytical solutions to the prices of many derivatives: European call and put options; interest rate derivatives, such as swaptions, caps, floors, and bond options; as well as path-dependant options, such as perpetual American options, barrier, and lookback options.

#### 5.31.2 Constructor & Destructor Documentation

```
5.31.2.1 template < typename __URng_Poisson_Type , typename __URng_DoubleExpDist__Type > QuantLib::GeneralizedKouProcess < __URng_Poisson_Type, __URng_DoubleExpDist__Type > ::GeneralizedKouProcess ( const Handle < Quote > & x0, const Handle < YieldTermStructure > & dividendTS, const Handle < YieldTermStructure > & riskFreeTS, const Handle < BlackVolTermStructure > & blackVolTS, const Real jumpIntensity, const Real posProbability, const Real posJumpMean, const Real negJumpMean, const __URng_Poisson_Type & URng_Poisson = __URng_Poisson_Type (SeedGenerator::instance().get()), const __URng_DoubleExpDist__Type & URng_DoubleExpDist = __URng_Poisson_ ← Type (SeedGenerator::instance().get()), const boost::shared_ptr < discretization > & d = boost::shared_ptr < discretization > (new EulerDiscretization))
```

Creates an instance of Kou process using the four parameters: initial value, risk-free term structure, dividend term structure, black volatility term structure and the additional jump information

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

· kouprocess.hpp

# 5.32 QuantLib::GenericArrayStatistics < StatisticsType > Class Template Reference

Statistics analysis of N-dimensional (sequence) data.

```
#include <arraystatistics.hpp>
```

# **Public Types**

- typedef StatisticsType statistics\_type
- typedef Array value\_type

# **Public Member Functions**

• GenericArrayStatistics (Size dimension=0)

## inspectors

· Size size () const

## covariance and correlation

- Disposable < Matrix > covariance () const
  - returns the covariance Matrix
- Disposable < Matrix > correlation () const

returns the correlation Matrix

## 1-D inspectors lifted from underlying statistics class

- Size samples () const
- Real weightSum () const

# N-D inspectors lifted from underlying statistics class

- · Array mean () const
- · Array variance () const
- Array standardDeviation () const
- Array downsideVariance () const
- · Array downsideDeviation () const
- Array **semiVariance** () const
- · Array semiDeviation () const
- Array errorEstimate () const
- · Array skewness () const
- · Array kurtosis () const
- Array min () const
- Array max () const
- · Array gaussianPercentile (Real y) const
- Array percentile (Real y) const
- Array gaussianPotentialUpside (Real percentile) const
- Array **potentialUpside** (Real percentile) const
- Array gaussianValueAtRisk (Real percentile) const
- Array valueAtRisk (Real percentile) const
- Array gaussianExpectedShortfall (Real percentile) const
- Array expectedShortfall (Real percentile) const
- Array regret (Real target) const
- Array gaussianShortfall (Real target) const
- · Array shortfall (Real target) const
- Array gaussianAverageShortfall (Real target) const
- Array averageShortfall (Real target) const

#### **Modifiers**

- void reset (Size dimension=0)
- template < class Sequence > void add (const Sequence & sample, Real weight=1.0)
- template < class Iterator > void add (Iterator begin, Iterator end, Real weight=1.0)

# **Protected Attributes**

- Size dimension
- std::vector< statistics\_type > stats\_
- Array results\_
- Matrix quadraticSum

# 5.32.1 Detailed Description

template < class StatisticsType > class QuantLib::GenericArrayStatistics < StatisticsType >

Statistics analysis of N-dimensional (sequence) data.

It provides 1-dimensional statistics as discrepancy plus N-dimensional (sequence) statistics (e.g. mean, variance, skewness, kurtosis, etc.) with one component for each dimension of the sample space.

For most of the statistics this class relies on the StatisticsType underlying class to provide 1-D methods that will be iterated for all the components of the N-D data. These lifted methods are the union of all the methods that might be requested to the 1-D underlying StatisticsType class, with the usual compile-time checks provided by the template approach.

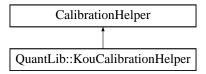
Test the correctness of the returned values is tested by checking them against numerical calculations.

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

· arraystatistics.hpp

# 5.33 QuantLib::KouCalibrationHelper Class Reference

Inheritance diagram for QuantLib::KouCalibrationHelper:



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

· koucalibrationhelper.hpp

# 5.34 QuantLib::KouHelper Class Reference

Kou process helper.

#include <analytickoueuropeanengine.hpp>

#### **Static Public Member Functions**

• static const double **Gamma** (double mu, double sigma, double lambda, double p, double eta1, double eta2, double a, double T, double tolerance)

# 5.34.1 Detailed Description

Kou process helper.

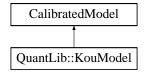
support functions to analytically calculate the option price under kou process assumption

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

- · analytickoueuropeanengine.hpp
- analytickoueuropeanengine.cpp

## 5.35 QuantLib::KouModel Class Reference

Inheritance diagram for QuantLib::KouModel:



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

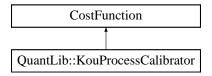
koumodel.hpp

# 5.36 QuantLib::KouProcessCalibrator Class Reference

A Kou process calibrator class which is used to do calibration using sets of vanilla options.

#include <kouprocesscalibrator.hpp>

Inheritance diagram for QuantLib::KouProcessCalibrator:



**Public Member Functions** 

# **Constructors & Destructors**

KouProcessCalibrator (const QuantLib::Date &referenceDate, const QuantLib::Calendar &calendar, const QuantLib::DayCounter &dayCounter, double riskFreeRate, double spotPrice, double dividend, std::shared\_ptr< std::vector< double >> strike, std::shared\_ptr< std::vector< QuantLib::Date >> maturityDate, std::shared\_ptr< std::vector< double >> optionPrice, std::shared\_ptr< QuantLib::End Criteria > endcriteria)

#### **Public interface**

- QuantLib::BlackConstantVol getVol ()
  - Return the black volatility term structure.
- double getPosJumpMean ()
- double getNegJumpMean ()
- double getPosProb ()
- double getJumpIntensity ()
- · void calibrate ()

Calibrate the Kou process.

- QuantLib::Real value (const QuantLib::Array &x) const
  - Sum the all difference term given from values method.
- QuantLib::Disposable < QuantLib::Array > values (const QuantLib::Array &x) const

Caculate the difference between the model value and market value.

## 5.36.1 Detailed Description

A Kou process calibrator class which is used to do calibration using sets of vanilla options.

This class is used to calibrate kou process using sets of vanilla options with different tenors on the underlying stock, it is implemented by inheriting the CostFunction and using the Optimizing framework in QuantLib.

#### 5.36.2 Constructor & Destructor Documentation

5.36.2.1 KouProcessCalibrator::KouProcessCalibrator ( const QuantLib::Date & referenceDate, const QuantLib::Calendar & calendar, const QuantLib::DayCounter & dayCounter, double riskFreeRate, double spotPrice, double dividend, std::shared\_ptr< std::vector< double >> strike, std::shared\_ptr< std::vector< QuantLib::Date >> maturityDate, std::shared\_ptr< std::vector< double >> optionPrice, std::shared\_ptr< QuantLib::EndCriteria > endcriteria )

Creates an instance of Kou process calibrator using sets of vanilla options information and some market observable information of underlying stock

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

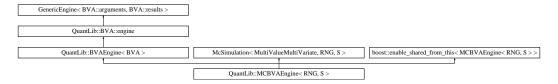
- · kouprocesscalibrator.hpp
- · kouprocesscalibrator.cpp

# 5.37 QuantLib::MCBVAEngine < RNG, S > Class Template Reference

Pricing engine class for BVA (p. 20) calculation via Monte Carlo Simulation.

#include <mcbvaengine.hpp>

Inheritance diagram for QuantLib::MCBVAEngine < RNG, S >:



# **Public Types**

- typedef MultiValueMultiVariate< RNG >::path\_type path\_type
- typedef McSimulation < MultiValueMultiVariate, RNG, S >::stats type stats\_type
- typedef McSimulation < MultiValueMultiVariate, RNG, S >::path pricer type path pricer type
- typedef McSimulation < MultiValueMultiVariate, RNG, S >::path\_generator\_type path\_generator\_type

#### **Public Member Functions**

# **Constructors & Destructors**

- MCBVAEngine (const std::vector< boost::shared\_ptr< const Counterparty >> &counterparties, const std::vector< boost::shared\_ptr< const MCTVAModel >> &instrumentModels, const Matrix &correlationMatrix, const Handle< YieldTermStructure > &discountCurve, const boost::shared\_coptr< ShortRateTermStructure > &shortRateDynamics, Size timeSteps, Size timeStepsPerYear, bool antitheticVariate, Size requiredSamples, Real requiredTolerance, Size maxSamples, Time endTime, BigNatural seed=SeedGenerator::instance().get())
- virtual ∼MCBVAEngine ()

#### **Public interface**

· void calculate () const

calculate bva by monte carlo simulation and store results in pricing engine

boost::shared\_ptr< StochasticProcessArray > process () const

Return shared pointer to stochastic process array containing underlying process as well as default process.

#### **Protected Member Functions**

#### Protected interface

· virtual TimeGrid timeGrid () const

Return time grid used by Monte Carlo Simulation.

- virtual boost::shared\_ptr< path\_generator\_type > pathGenerator () const
  - Return shared pointer to path generator who generates sample pathes in simulation.
- virtual boost::shared\_ptr< path\_pricer\_type > pathPricer () const
- virtual std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const Return shared pointer to stochastic process array containing underlying process.

## **Additional Inherited Members**

## 5.37.1 Detailed Description

template < typename RNG = PseudoRandom, typename S = GenericArrayStatistics < GaussianStatistics >> class QuantLib:: $MC \leftarrow BVAEngine < RNG, S >$ 

Pricing engine class for BVA (p. 20) calculation via Monte Carlo Simulation.

This class is the pricing engine for the "instrument" **BVA** (p. 20), responsible for the calculation of **BVA** (p. 20) (includes CVA, DVA) via Monte Carlo Simulation framework. The information of the two counterparties (issuer and investor) and the information of the underlying instruments (single instrument or portfolio) should be giving.

See also

BVA (p. 20) BVAEngine (p. 21) Counterparty (p. 27) MCTVAModel (p. 54)

## 5.37.2 Constructor & Destructor Documentation

5.37.2.1 template < typename RNG = PseudoRandom, typename S = GenericArrayStatistics < GaussianStatistics >> QuantLib::MCBVAEngine < RNG, S >::MCBVAEngine ( const std::vector < boost::shared\_ptr < const Counterparty >> & counterparties, const std::vector < boost::shared\_ptr < const MCTVAModel >> & instrumentModels, const Matrix & correlationMatrix, const Handle < YieldTermStructure > & discountCurve, const boost::shared\_ptr < ShortRateTermStructure > & shortRateDynamics, Size timeSteps, Size timeStepsPerYear, bool antitheticVariate, Size requiredSamples, Real requiredTolerance, Size maxSamples, Time endTime, BigNatural seed = SeedGenerator::instance().get() ) [inline]

Creates an instance of MC **BVA** (p. 20) engine using two counterparties, instruments (single instrument or portfolio), correlation matrix of the counterparties and instruments, and the yield term structure (deterministic or stochastic), these information may be got from the "instrument" **BVA** (p. 20).

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

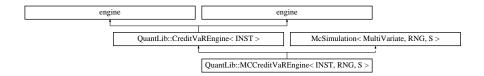
· mcbvaengine.hpp

# 5.38 QuantLib::MCCreditVaREngine < INST, RNG, S > Class Template Reference

Abstract base class for Credit VaR calculation via Monte Carlo Simulation.

#include <mccreditvarengine.hpp>

Inheritance diagram for QuantLib::MCCreditVaREngine < INST, RNG, S >:



# **Public Types**

- typedef MultiVariate< RNG >::path\_type path\_type
- typedef McSimulation < MultiVariate, RNG, S >::stats\_type stats\_type
- typedef McSimulation < MultiVariate, RNG, S >::path\_pricer\_type path\_pricer\_type
- typedef McSimulation< MultiVariate, RNG, S >::path\_generator\_type path\_generator\_type

#### **Public Member Functions**

#### **Constructors & Destructors**

- MCCreditVaREngine (const boost::shared\_ptr< const INST > &instrument, const boost::shared\_ptr< const Counterparty > &issuer, const boost::shared\_ptr< const Counterparty > &investor, const Matrix &correlation, Time endTime, const Handle< YieldTermStructure > &discountCurve, Size timeSteps, Size timeStepsPerYear, bool antitheticVariate, Size requiredSamples, Real requiredTolerance, Size max← Samples, BigNatural seed)
- virtual ~MCCreditVaREngine ()

# **Public interface**

- void calculate () const
  - calculate credit VaR by monte carlo simulation and store results in pricing engine
- boost::shared\_ptr< StochasticProcessArray > process () const

# **Protected Member Functions**

- virtual boost::shared ptr< path pricer type > pathPricer () const =0
- $\bullet \ \ virtual \ std:: vector < boost:: shared\_ptr < Stochastic Process 1D >> instrument Process \ () \ const = 0 \\$

#### Protected interface

- virtual TimeGrid timeGrid () const
  - Return time grid used by Monte Carlo Simulation.
- virtual boost::shared ptr< path generator type > pathGenerator () const

## **Additional Inherited Members**

# 5.38.1 Detailed Description

template<typename INST, typename RNG = PseudoRandom, typename S = Statistics>class QuantLib::MCCreditVaREngine<<br/>INST, RNG, S >

Abstract base class for Credit VaR calculation via Monte Carlo Simulation.

This class is used to calculate Credit VaR via Monte Carlo Simulation.

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

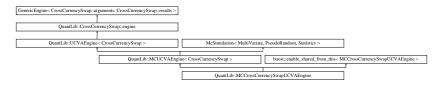
· mccreditvarengine.hpp

# 5.39 QuantLib::MCCrossCurrencySwapUCVAEngine Class Reference

Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap.

#include <mccrosscurrencyswapucvaengine.hpp>

Inheritance diagram for QuantLib::MCCrossCurrencySwapUCVAEngine:



## **Public Member Functions**

## **Constructors & Destructors**

- MCCrossCurrencySwapUCVAEngine (const Calendar &calender, const DayCounter &dayCounter, Date referenceDate, const Handle< YieldTermStructure > &riskFreeTermStructure, const boost::shared 
  \_ptr< const CrossCurrencySwap > &swap, const boost::shared\_ptr< const Counterparty > &issuer, Rate domesticRate, Real domesticRateSpeed, Real domesticRateMean, Volatility domesticRateVol, Rate foreignRate, Real foreignRateSpeed, Real foreignRateMean, Volatility foreignRateVol, const Matrix &corr, Size timeStepsPerYear=360, bool antitheticVariate=true, Size requiredSamples=50000, Real requiredTolerance=0.0001, Size maxSamples=QL\_MAX\_INTEGER, BigNatural seed=SeedGenerator 
  ::instance().get())
- ~MCCrossCurrencySwapUCVAEngine ()

#### **Protected Member Functions**

## **Protected interface**

- virtual std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const Override function in base class. Return the processes of interest, which will be used by path generator.
- virtual boost::shared\_ptr< path\_pricer\_type > pathPricer () const Return shared pointer of path pricer.

# **Additional Inherited Members**

# 5.39.1 Detailed Description

Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap.

This class is used to calculate UCVA of fixed-fixed cross currency swap via Monte Carlo simulation, giving the information of the **Counterparty** (p. 27) and the underlying instrument.

The kernel calculation part for UCVA is implemented in its pathpricer class.

See also

MCUCVAEngine (p. 56) CrossCurrencySwapUCVAPathPricer (p. 31) MCVanillaSwapUCVAEngine (p. 57) MCEquitySwapUCVAEngine (p. 49)

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

- mccrosscurrencyswapucvaengine.hpp
- · mccrosscurrencyswapucvaengine.cpp

# 5.40 QuantLib::MCEquitySwapUCVAEngine Class Reference

Monte Carlo pricing engine for UCVA calculation of equity swap.

#include <mcequityswapucvaengine.hpp>

Inheritance diagram for QuantLib::MCEquitySwapUCVAEngine:



#### **Public Member Functions**

# Constructor

• MCEquitySwapUCVAEngine (Handle< YieldTermStructure > riskFreeTermStructure, const boost⇔ ::shared\_ptr< const EquitySwap > &swap, const boost::shared\_ptr< const Counterparty > &issuer, const Matrix &corr, Size timeStepsPerYear=360, bool antitheticVariate=true, Size required↔ Samples=50000, Real requiredTolerance=0.0001, Size maxSamples=QL\_MAX\_INTEGER, BigNatural seed=QuantLib::SeedGenerator::instance().get())

#### **Destructor**

- $\sim$ MCEquitySwapUCVAEngine ()
- virtual boost::shared\_ptr< path\_pricer\_type > pathPricer () const
- std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const

return the underlying stock process

## **Additional Inherited Members**

# 5.40.1 Detailed Description

Monte Carlo pricing engine for UCVA calculation of equity swap.

This class is used to calculate UCVA of equity swap via Monte Carlo simulation, giving the information of the **Counterparty** (p. 27) and the underlying instrument.

The kernel calculation part for UCVA is implemented in its pathpricer class.

See also

MCUCVAEngine (p. 56)
EquitySwapUCVAPathPricer (p. 38)
MCCrossCurrencySwapUCVAEngine (p. 48)
MCVanillaSwapUCVAEngine (p. 57)

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

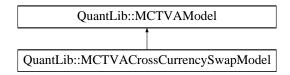
- · mcequityswapucvaengine.hpp
- · mcequityswapucvaengine.cpp

# 5.41 QuantLib::MCTVACrossCurrencySwapModel Class Reference

TVA (p. 62) model for cross currency swap.

#include <mctvacrosscurrencyswapmodel.hpp>

Inheritance diagram for QuantLib::MCTVACrossCurrencySwapModel:



## **Public Member Functions**

#### **Constructors & Destructors**

• MCTVACrossCurrencySwapModel (const Calendar &calendar, const DayCounter &dayCounter, Date referenceDate, Handle< YieldTermStructure > riskFreeTermStructure, const boost::shared\_ptr< const CrossCurrencySwap > &instrument, Rate domesticRate, Real domesticRateSpeed, Real domestic← RateMean, Volatility domesticRateVol, Rate foreignRate, Real foreignRateSpeed, Real foreignRateMean, Volatility foreignRateVol)

## **Public interface**

- std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const Return the underlying instrument processes of the cross currency swap.
- Real **exposure** (const MultiPath &path, const Time issuerDefaultTime, const Time investorDefaultTime, const Handle< YieldTermStructure > disTS) const

Return the exposure of cash flow at given time and paths information.

## **Additional Inherited Members**

# 5.41.1 Detailed Description

TVA (p. 62) model for cross currency swap.

This class implements tva model for cross currency swap, working as tva calculation helper, responsible for providing information of underlying instrument and calculation the exposure of cash flow at given time. It's used in the calculation of CVA, DVA and FVA of underlying instrument under Monte Carlo Simulation framework.

Remarks

The exposure of cross currency swap at given time t, NPV(t), is calculated as the discounted cash flow. The exchange rate  $X_s$ ,  $s \in [t, T]$ , using in calculation of NPV(t), is giving as the expectation

$$X_s = X_t e^{\mu(s-t)}$$

See also

MCTVAVanillaSwapModel (p. 55) MCTVAEquitySwapModel (p. 53)

## 5.41.2 Constructor & Destructor Documentation

5.41.2.1 QuantLib::MCTVACrossCurrencySwapModel::MCTVACrossCurrencySwapModel ( const Calendar & calendar, const DayCounter & dayCounter, Date referenceDate, Handle< YieldTermStructure > riskFreeTermStructure, const boost::shared\_ptr< const CrossCurrencySwap > & instrument, Rate domesticRate, Real domesticRateSpeed, Real domesticRateMean, Volatility domesticRateVol, Rate foreignRate, Real foreignRateSpeed, Real foreignRateWean, Volatility foreignRateVol) [inline]

Creates an instance of cross currency swap model using the information of the contract including the calendar, day counter, reference date, yield term structure, and the instrument.

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

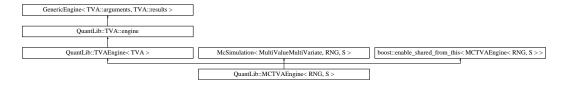
- · mctvacrosscurrencyswapmodel.hpp
- · mctvacrosscurrencyswapmodel.cpp

# 5.42 QuantLib::MCTVAEngine < RNG, S > Class Template Reference

Pricing engine class for TVA (p. 62) calculation via Monte Carlo Simulation.

#include <mctvaengine.hpp>

Inheritance diagram for QuantLib::MCTVAEngine < RNG, S >:



# **Public Types**

- typedef MultiValueMultiVariate< RNG >::path\_type path\_type
- $\bullet \ \ \mathsf{typedef} \ \mathsf{McSimulation} < \mathbf{MultiValueMultiVariate}, \ \mathsf{RNG}, \ \mathsf{S} > ::\mathsf{stats\_type} \ \mathbf{stats\_type}$
- typedef McSimulation < MultiValueMultiVariate, RNG, S >::path\_pricer\_type path\_pricer\_type
- typedef McSimulation < MultiValueMultiVariate, RNG, S >::path\_generator\_type path\_generator\_type

**Public Member Functions** 

# **Constructors & Destructors**

• MCTVAEngine (const std::vector< boost::shared\_ptr< const Counterparty >> &counterparties, const std::vector< boost::shared\_ptr< const MCTVAModel >> &instrumentModels, const Matrix &correlationMatrix, const Handle< YieldTermStructure > &discountCurve, const boost::shared\_cptr< ShortRateTermStructure > &shortRateDynamics, Size timeSteps, Size timeStepsPerYear, bool antitheticVariate, Size requiredSamples, Real requiredTolerance, Size maxSamples, Time endTime, BigNatural seed=SeedGenerator::instance().get())

virtual ∼MCTVAEngine ()

## **Public interface**

· void calculate () const

Calculate bva by monte carlo simulation and store results in pricing engine.

• boost::shared\_ptr< StochasticProcessArray > process () const

Return shared pointer to stochastic procss array containing underlying process as well as default process.

#### **Protected Member Functions**

#### **Protected interface**

• virtual TimeGrid timeGrid () const

Return time grid used by Monte Carlo Simulation.

virtual boost::shared\_ptr< path\_generator\_type > pathGenerator () const

Return shared pointer to path generator who generates sample pathes in simulation.

- virtual boost::shared\_ptr< path\_pricer\_type > pathPricer () const
- virtual std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const Return shared pointer to stochastic process array containing underlying process.

#### **Additional Inherited Members**

# 5.42.1 Detailed Description

template < typename RNG = PseudoRandom, typename S = GenericArrayStatistics < GaussianStatistics >> class QuantLib:: $MC \leftarrow TVAEngine < RNG, S >$ 

Pricing engine class for TVA (p. 62) calculation via Monte Carlo Simulation.

This class is the pricing engine for the "instrument" **TVA** (p. 62), responsible for the calculation of **TVA** (p. 62) (includes CVA, DVA and FVA) via Monte Carlo Simulation framework. The information of the two counterparties (issuer and investor) and the information of the underlying instruments (single instrument or portfolio) should be giving.

See also

TVA (p. 62) Counterparty (p. 27) MCTVAModel (p. 54)

# 5.42.2 Constructor & Destructor Documentation

5.42.2.1 template<typename RNG = PseudoRandom, typename S = GenericArrayStatistics<GaussianStatistics>> QuantLib::MCTVAEngine< RNG, S >::MCTVAEngine ( const std::vector< boost::shared\_ptr< const Counterparty >> & counterparties, const std::vector< boost::shared\_ptr< const MCTVAModel >> & instrumentModels, const Matrix & correlationMatrix, const Handle< YieldTermStructure > & discountCurve, const boost::shared\_ptr< ShortRateTermStructure > & shortRateDynamics, Size timeSteps, Size timeStepsPerYear, bool antitheticVariate, Size requiredSamples, Real requiredTolerance, Size maxSamples, Time endTime, BigNatural seed = SeedGenerator::instance().get() ) [inline]

Creates an instance of MC **TVA** (p. 62) engine using two counterparties, instruments (single instrument or portfolio), correlation matrix of the counterparties and instruments, and the yield term structure (deterministic or stochastic),

these information may be got from the "instrument" TVA (p. 62).

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

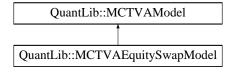
· mctvaengine.hpp

# 5.43 QuantLib::MCTVAEquitySwapModel Class Reference

TVA (p. 62) model for equity swap.

#include <mctvaequityswapmodel.hpp>

Inheritance diagram for QuantLib::MCTVAEquitySwapModel:



#### **Public Member Functions**

#### **Constructors & Destructors**

MCTVAEquitySwapModel (const Calendar &calendar, const DayCounter &dayCounter, Date reference
 — Date, Handle< YieldTermStructure > riskFreeTermStructure, const boost::shared\_ptr< const Equity
 — Swap > &instrument)

#### **Public interface**

- std::vector < boost::shared\_ptr < StochasticProcess1D > > instrumentProcess () const
   Return the underlying instrument processes of the equity swap.
- Real exposure (const MultiPath &path, const Time issuerDefaultTime, const Time investorDefaultTime, const Handle
   YieldTermStructure > disTS) const

Return the exposure of cash flow at given time and paths information.

# **Additional Inherited Members**

# 5.43.1 Detailed Description

TVA (p. 62) model for equity swap.

This class implements tva model for equity swap, working as tva calculation helper, responsible for providing information of underlying instrument and calculation the exposure of cash flow at given time. It's used in the calculation of CVA, DVA and FVA of underlying instrument under Monte Carlo Simulation framework.

## Remarks

The exposure of equity swap at given time t, NPV(t), is calculated as the discounted cash flow. The payoff of floating leg at time  $t_i$  is

$$\frac{q(E(S_{t_i}) - E(S_{t_{i-1}}))}{r - q}$$

where 
$$E(S_{t_i}) = S_0 e^{(r-q)t_i}$$
.

#### See also

MCTVAVanillaSwapModel (p. 55) MCTVACrossCurrencySwapModel (p. 50)

## 5.43.2 Constructor & Destructor Documentation

5.43.2.1 QuantLib::MCTVAEquitySwapModel::MCTVAEquitySwapModel ( const Calendar & calendar, const DayCounter & dayCounter, Date referenceDate, Handle < YieldTermStructure > riskFreeTermStructure, const boost::shared\_ptr < const EquitySwap > & instrument ) [inline]

Creates an instance of equity swap model using the information of the contract including the calendar, day counter, reference date, yield term structure, and the instrument.

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

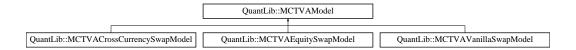
- · mctvaequityswapmodel.hpp
- · mctvaequityswapmodel.cpp

# 5.44 QuantLib::MCTVAModel Class Reference

base class for tva model.

#include <mctvamodel.hpp>

Inheritance diagram for QuantLib::MCTVAModel:



## **Public Member Functions**

## **Constructors & Destructors**

- MCTVAModel (const Calendar &calendar, const DayCounter &dayCounter, Date referenceDate, Handle
   YieldTermStructure > riskFreeTermStructure, const boost::shared\_ptr< const Instrument > &instrument)
- ∼MCTVAModel ()

#### **Public interface**

- boost::shared\_ptr< const Instrument > instrument () const
  - Return shared pointer to underlying instrument.
- virtual std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const =0
   Return shared pointer to stochastic process array containing underlying process.
- virtual Real exposure (const MultiPath &path, const Time issuerDefaultTime, const Time investorDefault
   —
   Time, const Handle < YieldTermStructure > disTS) const =0

Calculate the exposure of cash flow at given time and multi-paths information.

# **Protected Attributes**

- · Calendar calendar\_
- DayCounter dayCounter
- Date referenceDate
- const Handle
   YieldTermStructure > riskFreeTermStructure\_
- const boost::shared\_ptr< const Instrument > instrument\_

## 5.44.1 Detailed Description

base class for tva model.

This class is a base class for tva model, working as tva calculation helper, responsible for providing information of underlying instrument and calculation the exposure of cash flow at given time

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

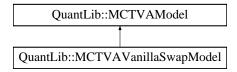
· mctvamodel.hpp

# 5.45 QuantLib::MCTVAVanillaSwapModel Class Reference

TVA (p. 62) model for vanilla swap.

#include <mctvavanillaswapmodel.hpp>

Inheritance diagram for QuantLib::MCTVAVanillaSwapModel:



# **Public Member Functions**

#### **Constructors & Destructors**

• MCTVAVanillaSwapModel (const Calendar &calendar, const DayCounter &dayCounter, Date reference 

Date, Handle 

YieldTermStructure 

riskFreeTermStructure, const boost::shared\_ptr 

const Cox←

IngersollRoss 

&cirModel, const boost::shared\_ptr 

VanillaSwap 

&instrument)

## **Public interface**

- std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const
   Return the underlying instrument processes of the vanilla swap.
- Real exposure (const MultiPath &path, const Time issuerDefaultTime, const Time investorDefaultTime, const Handle
   YieldTermStructure > disTS) const

Return the exposure of cash flow at given time and paths information.

## **Additional Inherited Members**

# 5.45.1 Detailed Description

TVA (p. 62) model for vanilla swap.

This class implements tva model for vanilla swap, working as tva calculation helper, responsible for providing information of underlying instrument and calculation the exposure of cash flow at given time. It's used in the calculation of CVA, DVA and FVA of underlying instrument under Monte Carlo Simulation framework.

#### Remarks

The exposure of vanilla swap at given time t, NPV(t), is calculated as the discounted cash flow.

#### See also

MCTVAEquitySwapModel (p. 53)
MCTVACrossCurrencySwapModel (p. 50)

## 5.45.2 Constructor & Destructor Documentation

5.45.2.1 QuantLib::MCTVAVanillaSwapModel::MCTVAVanillaSwapModel ( const Calendar & calendar, const DayCounter & dayCounter, Date referenceDate, Handle< YieldTermStructure > riskFreeTermStructure, const boost::shared\_ptr< const CoxIngersollRoss > & cirModel, const boost::shared\_ptr< VanillaSwap > & instrument ) [inline]

Creates an instance of vanilla swap model using the information of the contract including the calendar, day counter, reference date, yield term structure, and the instrument.

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

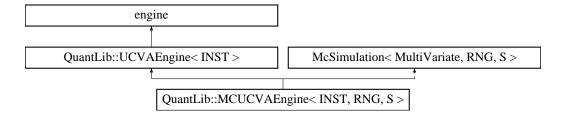
- · mctvavanillaswapmodel.hpp
- · mctvavanillaswapmodel.cpp

# 5.46 QuantLib::MCUCVAEngine < INST, RNG, S > Class Template Reference

Abstract base class for UCVA calculation via Monte Carlo Simulation.

#include <mcucvaengine.hpp>

Inheritance diagram for QuantLib::MCUCVAEngine < INST, RNG, S >:



# **Public Types**

- typedef MultiVariate< RNG >::path\_type path\_type
- typedef McSimulation< MultiVariate, RNG, S >::stats\_type stats\_type
- typedef McSimulation
   MultiVariate, RNG, S >::path\_pricer\_type path\_pricer\_type
- typedef McSimulation< MultiVariate, RNG, S >::path\_generator\_type path\_generator\_type

# **Public Member Functions**

## **Constructors & Destructors**

- MCUCVAEngine (const boost::shared\_ptr< const INST > &instrument, const boost::shared\_ptr< const Counterparty > &issuer, const Matrix &correlation, Time endTime, const Handle< YieldTermStructure > &discountCurve, Size timeSteps, Size timeStepsPerYear, bool antitheticVariate, Size requiredSamples, Real requiredTolerance, Size maxSamples, BigNatural seed)
- virtual ∼MCUCVAEngine ()

#### **Public interface**

• void calculate () const

calculate UCVA by monte carlo simulation and store results in pricing engine

 $\bullet \ \ boost:: shared\_ptr < StochasticProcessArray > \textbf{process} \ () \ const$ 

Return shared pointer to stochastic process array containing underlying process as well as default process.

## **Protected Member Functions**

- virtual boost::shared\_ptr< path\_pricer\_type > pathPricer () const =0
- virtual std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const =0

#### Protected interface

• virtual TimeGrid timeGrid () const

Return time grid used by Monte Carlo Simulation.

• virtual boost::shared\_ptr< path\_generator\_type > pathGenerator () const Return shared pointer to path generator who generates sample pathes in simulation.

#### **Additional Inherited Members**

#### 5.46.1 Detailed Description

 $template < typename\ \ S = Statistics > class\ \ QuantLib::MCUCVAEngine < INST, \ RNG, \ S >$ 

Abstract base class for UCVA calculation via Monte Carlo Simulation.

This abstract base class is the base Monte Carlo pricing engine class of instrument for UCVA calculation.

The kernel calculation part is its path pricer class.

The user who wants to calculate the UCVA of a instrument should implement the UCVA engine class and the path pricer class for the instrument.

See also

# UCVAPathPricer (p. 66)

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

mcucvaengine.hpp

# 5.47 QuantLib::MCVanillaSwapUCVAEngine Class Reference

Monte Carlo pricing engine for UCVA calculation of vanilla swap.

#include <mcvanillaswapucvaengine.hpp>

Inheritance diagram for QuantLib::MCVanillaSwapUCVAEngine:



## **Public Member Functions**

## **Constructors & Destructors**

MCVanillaSwapUCVAEngine (const Calendar &calender, const DayCounter &dayCounter, Date referenceDate, const Handle
 YieldTermStructure > &riskFreeTermStructure, const boost::shared← \_ptr< const VanillaSwap > &swap, const boost::shared\_ptr< const Counterparty > &issuer, const

boost::shared\_ptr< const CoxIngersollRoss > &cirModel, const Matrix &corr, Size timeStepsPer ← Year=360, bool antitheticVariate=true, Size requiredSamples=50000, Real requiredTolerance=0.0001, Size maxSamples=QL\_MAX\_INTEGER, BigNatural seed=SeedGenerator::instance().get())

MCVanillaSwapUCVAEngine::~MCVanillaSwapUCVAEngine ()

#### **Public interface**

 const boost::shared\_ptr< const CoxIngersollRoss > & Model () const Return the cir model.

#### **Protected Member Functions**

## **Protected interface**

- std::vector< boost::shared\_ptr< StochasticProcess1D >> instrumentProcess () const Return the processes of instruments.
- virtual boost::shared\_ptr< path\_pricer\_type > pathPricer () const Return the shared pointer of path\_pricer\_type.

#### **Additional Inherited Members**

## 5.47.1 Detailed Description

Monte Carlo pricing engine for UCVA calculation of vanilla swap.

This class is used to calculate UCVA of vanilla swap via Monte Carlo simulation, giving the information of the **Counterparty** (p. 27) and the underlying instrument.

The kernel calculation part for UCVA is implemented in its pathpricer class.

See also

MCUCVAEngine (p. 56)
VanillaSwapUCVAPathPricer (p. 67)
MCCrossCurrencySwapUCVAEngine (p. 48)
MCEquitySwapUCVAEngine (p. 49)

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

- · mcvanillaswapucvaengine.hpp
- · mcvanillaswapucvaengine.cpp

# 5.48 QuantLib::MultiValueMultiVariate < RNG > Struct Template Reference

Traits class for multi-state Monte Carlo Simulation.

#include <multivaluemctraits.hpp>

## **Public Types**

- enum { allowsErrorEstimate = RNG::allowsErrorEstimate }
- · typedef RNG rng traits
- typedef MultiPath path\_type
- typedef PathPricer< path\_type, Array > path\_pricer\_type
- typedef RNG::rsg\_type rsg\_type
- $\bullet \ \ typedef \ MultiPathGenerator < rsg\_type > \textbf{path\_generator\_type} \\$

# 5.48.1 Detailed Description

 $template < class\ RNG > struct\ QuantLib:: MultiValue MultiVariate < RNG >$ 

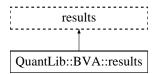
Traits class for multi-state Monte Carlo Simulation.

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

· multivaluemctraits.hpp

# 5.49 QuantLib::BVA::results Class Reference

Inheritance diagram for QuantLib::BVA::results:



## **Public Member Functions**

• void reset ()

## **Public Attributes**

- · Real CVA
- · Real DVA

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

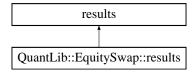
bva.hpp

# 5.50 QuantLib::EquitySwap::results Class Reference

Results class of equity swap

#include <equityswap.hpp>

Inheritance diagram for QuantLib::EquitySwap::results:



# 5.50.1 Detailed Description

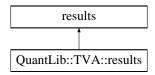
Results class of equity swap

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

equityswap.hpp

# 5.51 QuantLib::TVA::results Class Reference

Inheritance diagram for QuantLib::TVA::results:



## **Public Member Functions**

· virtual void reset ()

#### **Public Attributes**

- · Real CVA
- · Real DVA
- · Real FVA

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

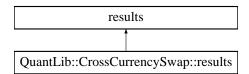
· tva.hpp

# 5.52 QuantLib::CrossCurrencySwap::results Class Reference

Results for cross currency swap calculation

#include <crosscurrencyswap.hpp>

Inheritance diagram for QuantLib::CrossCurrencySwap::results:



# 5.52.1 Detailed Description

Results for cross currency swap calculation

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

· crosscurrencyswap.hpp

# 5.53 RiskAnalysisTool::Calculation::Sample::SampleClass Class Reference

**Public Member Functions** 

double Sum (const std::shared\_ptr< const std::vector< double >> &data)

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

- · SampleClass.hpp
- · SampleClass.cpp

# 5.54 RiskAnalysisTool::Calculation::sealed Class Reference

**Public Member Functions** 

• ComputeEngine ()

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

· ComputeEngine.cpp

# 5.55 QuantLib::ShortRateTermStructure Class Reference

Interest-rate term structure class based on short-rate model.

#include <shortratetermstructure.hpp>

# **Public Member Functions**

#### **Constructors & Destructors**

Creates an instance of yield term structure giving the model

- →ShortRateTermStructure ()

## **Term structure**

These methods return the yield term structure from reference date given max date or max time and the short rate at reference date.

- boost::shared\_ptr< TermStructure > GetTermStructure (Date referenceDate, Date maxDate, Real rate, Frequency freq=Quarterly)
- boost::shared\_ptr< TermStructure > GetTermStructure (Date referenceDate, Time Tmax, Real rate, Frequency freq=Quarterly)

# 5.55.1 Detailed Description

Interest-rate term structure class based on short-rate model.

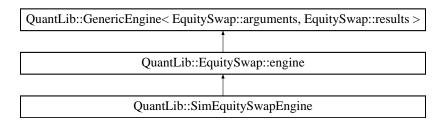
This class is used to generate yield term structure based on the given One factor & Affine short-rate model. The discount factors in the term structure are all expectations and calculated by the corresponding formula of given model.

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

- · shortratetermstructure.hpp
- shortratetermstructure.cpp

# 5.56 QuantLib::SimEquitySwapEngine Class Reference

Inheritance diagram for QuantLib::SimEquitySwapEngine:



#### **Public Member Functions**

- SimEquitySwapEngine (size\_t n\_=1000)
- void calculate () const

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

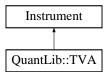
- · simequityswapengine.hpp
- · simequityswapengine.cpp

# 5.57 QuantLib::TVA Class Reference

Total value adjustment manager class.

#include <tva.hpp>

Inheritance diagram for QuantLib::TVA:



# Classes

- · class arguments
- · class engine
- · class results

# **Public Member Functions**

- void **setupArguments** (PricingEngine::arguments \*) const
- void fetchResults (const PricingEngine::results \*) const
- · bool isExpired () const
- · const Real CVA () const
- · const Real DVA () const
- const Real FVA () const

#### **Protected Attributes**

- Real CVA
- Real DVA
- Real FVA

#### 5.57.1 Detailed Description

Total value adjustment manager class.

This class inherits from Instrument class, responsible for calculation total value adjustment (includes CVA, DV ← A, FVA) of a single instrument or portfolio. The arguments are the information of two counterparties (issuer and investor) using the **Counterparty** (p. 27) class and the information of underlying instrument or portfolio using their corresponding model class. The Pricingengine uses tvaengine based on Monte Carlo simulation framework. The results are including CVA, DVA and FVA.

#### See also

```
counterparty.hpp (p. 75) tvamodel.hpp tvaengine.hpp (p. 89)
```

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

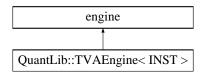
· tva.hpp

### 5.58 QuantLib::TVAEngine < INST > Class Template Reference

Base class for bva engine.

```
#include <tvaengine.hpp>
```

Inheritance diagram for QuantLib::TVAEngine < INST >:



#### **Public Member Functions**

#### **Constructors & Destructors**

- TVAEngine (Time endTime, const std::vector< boost::shared\_ptr< const Counterparty >> &counterparties, const std::vector< boost::shared\_ptr< const MCTVAModel >> &tvaModels, const Handle
   YieldTermStructure > &discountCurve, const boost::shared\_ptr< ShortRateTermStructure > &short RateDynamics)
- virtual ~TVAEngine ()

#### Inspector

- std::vector< boost::shared\_ptr< const Counterparty >> counterparties () const
- const Time endTime () const
- const Handle
   YieldTermStructure > & discountCurve () const
- const boost::shared\_ptr< ShortRateTermStructure > & shortRateDynamics () const
- const std::vector< boost::shared\_ptr< const MCTVAModel > > & models () const

64 Class Documentation

#### **Protected Attributes**

- std::vector< boost::shared\_ptr< const Counterparty > > counterparties\_
- Handle< YieldTermStructure > discountCurve
- boost::shared ptr< ShortRateTermStructure > shortRateDynamics
- std::vector< boost::shared\_ptr< const MCTVAModel > > tvaModels\_
- Time endTime

#### 5.58.1 Detailed Description

template<typename INST>class QuantLib::TVAEngine< INST>

Base class for bva engine.

This class is base class for tva engine, which is used to set some basic information. Each derived class should implement its own method to calculate bva.

#### 5.58.2 Constructor & Destructor Documentation

5.58.2.1 template < typename INST > QuantLib::TVAEngine < INST >::TVAEngine ( Time endTime, const std::vector < boost::shared\_ptr < const Counterparty >> & counterparties, const std::vector < boost::shared\_ptr < const MCTVAModel >> & tvaModels, const Handle < YieldTermStructure > & discountCurve, const boost::shared\_ptr < ShortRateTermStructure > & shortRateDynamics ) [inline]

Create an instance of **TVA** (p. 62) Engine using the information of the two counterparties using **Counterparty** (p. 27) class and underlying instruments using instrument model class.

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

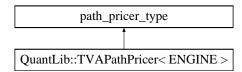
· tvaengine.hpp

### 5.59 QuantLib::TVAPathPricer < ENGINE > Class Template Reference

Path pricer class used to calculate the TVA (p. 62) on each Multi-path for TVAEngine (p. 63) class.

#include <tvapathpricer.hpp>

Inheritance diagram for QuantLib::TVAPathPricer< ENGINE >:



#### **Public Types**

- typedef ENGINE::path\_pricer\_type::argument\_type argument\_type
- typedef ENGINE::path\_pricer\_type::result\_type result\_type

**Public Member Functions** 

### Constructors

• TVAPathPricer (const boost::shared\_ptr< const ENGINE > &engine)

#### **Destructors**

∼TVAPathPricer ()

### Operator overload

result\_type operator() (const argument\_type &multiPath) const
 Calculate the TVA (p. 62) (includes CVA, DVA and FVA) given the Multi-path.

#### **Protected Attributes**

const boost::weak\_ptr< const ENGINE > engine\_

### 5.59.1 Detailed Description

template<typename ENGINE>class QuantLib::TVAPathPricer< ENGINE>

Path pricer class used to calculate the TVA (p. 62) on each Multi-path for TVAEngine (p. 63) class.

This class is the core part for **TVAEngine** (p. 63) class, which is used to calculate the **TVA** (p. 62) (includes CVA, DVA and FVA) on each Multi-path given the instruments (single instrument or portfolio), based on the definition of **TVA** (p. 62) (includes CVA, DVA and FVA) in Brigo's book: "Counterparty Credit Risk, Collateral and Funding".

If the underlying is a portfolio, netting is considered.

See also

#### MCTVAEngine (p. 51)

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

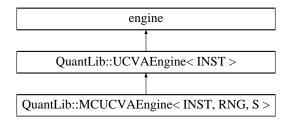
· tvapathpricer.hpp

### 5.60 QuantLib::UCVAEngine < INST > Class Template Reference

Abstract base class for ucva engine.

#include <ucvaengine.hpp>

Inheritance diagram for QuantLib::UCVAEngine < INST >:



**Public Member Functions** 

### **Constructors & Destructors**

66 Class Documentation

UCVAEngine (const boost::shared\_ptr< const INST > instrument, const boost::shared\_ptr< const Counterparty > issuer, const Time &endTime, const Handle< YieldTermStructure > &discountCurve)

Create an instance of UCVA Engine using the information of counterparty and underlying instrument.

virtual ∼UCVAEngine ()

### Inspector

- const Handle< YieldTermStructure > & discountCurve () const
- Time endTime () const
- const boost::shared\_ptr< const Counterparty > & issuer () const
- const boost::shared\_ptr< const INST > & instrument () const

#### **Protected Attributes**

- const Time endTime\_
- const boost::shared ptr< const INST > instrument
- const boost::shared\_ptr< const Counterparty > issuer\_
- const Handle
   YieldTermStructure
   discountCurve\_

### 5.60.1 Detailed Description

template<typename INST>class QuantLib::UCVAEngine< INST>

Abstract base class for ucva engine.

This class is base class for ucva engine, which just contains information as least as possible. Each derived class should implement its own method to calculate ucva.

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

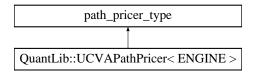
ucvaengine.hpp

### 5.61 QuantLib::UCVAPathPricer < ENGINE > Class Template Reference

Abstract base class used to calculate cash flow at default time of each path under the setting of UCVA.

#include <ucvapathpricer.hpp>

Inheritance diagram for QuantLib::UCVAPathPricer< ENGINE >:



### **Public Types**

- typedef ENGINE::path\_pricer\_type::argument\_type argument\_type
- typedef ENGINE::path\_pricer\_type::result\_type result\_type

### **Public Member Functions**

#### **Destructors**

∼UCVAPathPricer ()

#### Operator overload

result\_type operator() (const argument\_type &multiPath) const
 Calculate the UCVA given the Multi-path.

### **Protected Member Functions**

#### **Constructors**

UCVAPathPricer (const boost::shared\_ptr< const ENGINE > &engine)

#### **Protected interface**

- const boost::weak\_ptr< const ENGINE > engine\_
   Return weak pointer of pricing engine.
- virtual Real defaultNPV (const MultiPath &path, Time defaultTime) const =0
   Return default NPV at each path, must be implemented in derived class.

### 5.61.1 Detailed Description

 $template < typename \ ENGINE > class \ QuantLib:: UCVAPathPricer < ENGINE >$ 

Abstract base class used to calculate cash flow at default time of each path under the setting of UCVA.

This class is used to calculate cash flow at default time of each path. To use this class, derived class must implement the pure virtual function defaultNPV.

See also

### MCUCVAEngine (p. 56)

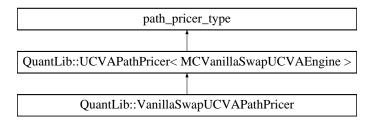
The documentation for this class was generated from the following file:

· ucvapathpricer.hpp

### 5.62 QuantLib::VanillaSwapUCVAPathPricer Class Reference

UCVA Path Pricer for vanilla swap.

Inheritance diagram for QuantLib::VanillaSwapUCVAPathPricer:



68 Class Documentation

#### **Public Member Functions**

#### **Constructors & Destructors**

VanillaSwapUCVAPathPricer (const boost::shared\_ptr< const MCVanillaSwapUCVAEngine > &engine)

#### **Protected Member Functions**

• virtual Real **defaultNPV** (const MultiPath &path, Time defaultTime) const Calculate the NPV of the remaining cash flow at default time  $\tau$ .

#### **Additional Inherited Members**

### 5.62.1 Detailed Description

UCVA Path Pricer for vanilla swap.

#### 5.62.2 Member Function Documentation

5.62.2.1 virtual Real QuantLib::VanillaSwapUCVAPathPricer::defaultNPV (const MultiPath & path, Time defaultTime) const [inline], [protected], [virtual]

Calculate the NPV of the remaining cash flow at default time  $\tau$ .

The calculation method used here is based on CIR model.

Implements QuantLib::UCVAPathPricer < MCVanillaSwapUCVAEngine > (p. 67).

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

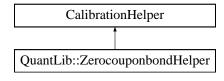
· mcvanillaswapucvaengine.cpp

### 5.63 QuantLib::ZerocouponbondHelper Class Reference

Zero coupon bond calibration helper.

#include <zerocouponbondhelper.hpp>

Inheritance diagram for QuantLib::ZerocouponbondHelper:



### **Public Member Functions**

### **Constructors & Destructors**

ZerocouponbondHelper (Natural settlementDays, const Calendar &calendar, Real faceAmount, const Date &maturityDate, BusinessDayConvention paymentConvention, Real redemption, const Date &issueDate, const Handle
 Quote > &volatility, const Handle
 YieldTermStructure > &termStructure, CalibrationErrorType = RelativePriceError)

Creates an instance of zero coupon bond helper using the information of the bond.

#### **Public interface**

- virtual void addTimesTo (std::list< Time > &times) const
   This method is to be used with tree-based methods.
- virtual Real modelValue () const
- virtual Real blackPrice (Volatility volatility) const

### 5.63.1 Detailed Description

Zero coupon bond calibration helper.

One factor affine short rate model calibration helper using zero coupon bond. It can be use to calibrate CIR model and vasicek model, etc.

#### 5.63.2 Member Function Documentation

5.63.2.1 Real ZerocouponbondHelper::blackPrice ( Volatility volatility ) const [virtual]

Calculate black value of zero coupon bond

5.63.2.2 Real ZerocouponbondHelper::modelValue()const [virtual]

Calculate model value of zero coupon bond

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

- · zerocouponbondhelper.hpp
- · zerocouponbondhelper.cpp

70 **Class Documentation** 

# **Chapter 6**

# **File Documentation**

# 6.1 analyticequityswapengine.cpp File Reference

```
#include <pch.h>
#include <ql/pricingengines/swap/analyticequityswapengine.hpp>
```

# 6.2 analyticequityswapengine.hpp File Reference

### Analytic engine of equity swap.

```
#include <Calculation/Calculation.h>
#include <ql/instruments/swap/equityswap.hpp>
```

### Classes

· class QuantLib::AnalyticESEngine

Analytic engine of equity swap.

### 6.2.1 Detailed Description

Analytic engine of equity swap.

# 6.3 analytickoueuropeanengine.cpp File Reference

```
#include <pch.h>
#include <boost/math/distributions/normal.hpp>
#include "analytickoueuropeanengine.hpp"
```

### 6.4 analytickoueuropeanengine.hpp File Reference

Analytic engine for european option.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <ql/instruments/vanillaoption.hpp>
#include <ql/pricingengines/vanilla/mcvanillaengine.hpp>
#include <ql/processes/kouprocess.hpp>
```

### Classes

• class QuantLib::AnalyticKouEuropeanEngine

Analytic engine for european option, whose underlying stock process is a kou process.

· class QuantLib::KouHelper

Kou process helper.

### 6.4.1 Detailed Description

Analytic engine for european option.

# 6.5 at1pdefaultmodel.cpp File Reference

```
#include <pch.h>
#include <ql/models/default/at1pdefaultmodel.hpp>
```

### 6.6 at1pdefaultmodel.hpp File Reference

### Analytic tractable first passage model.

```
#include <Calculation/Calculation.h>
#include <ql/models/defaultmodel.hpp>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/termstructures/volatility/equityfx/blackvoltermstructure.hpp>
#include <ql/termstructures/defaulttermstructure.hpp>
#include <ql/methods/montecarlo/path.hpp>
#include <ql/processes/blackscholesprocess.hpp>
```

#### Classes

· class QuantLib::AT1Pmodel

Analytic tractable first default model.

### 6.6.1 Detailed Description

Analytic tractable first passage model.

# 6.7 bva.hpp File Reference

#### Bilateral value adjustment manager class.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <ql/types.hpp>
#include <ql/math/matrix.hpp>
#include <ql/instrument.hpp>
#include <ql/pricingengine.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/credit/counterparty.hpp>
```

#### **Classes**

· class QuantLib::BVA

Bilateral value adjustment manager class.

class QuantLib::BVA::arguments
 class QuantLib::BVA::results
 class QuantLib::BVA::engine

### 6.7.1 Detailed Description

Bilateral value adjustment manager class.

# 6.8 bvaengine.hpp File Reference

### Base class for bva engine.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/types.hpp>
#include <ql/instrument.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/models/shortratetermstructure.hpp>
```

### Classes

class QuantLib::BVAEngine < INST >

Base class for bva engine.

### 6.8.1 Detailed Description

Base class for bva engine.

### 6.9 byapathpricer.hpp File Reference

Path pricer class used to calculate the BVA on each Multi-path for BVAEngine class.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <ql/credit/counterparty.hpp>
```

#### **Classes**

class QuantLib::BVAPathPricer< ENGINE >

Path pricer class used to calculate the BVA (p. 20) on each Multi-path for BVAEngine (p. 21) class.

### 6.9.1 Detailed Description

Path pricer class used to calculate the BVA on each Multi-path for BVAEngine class.

# 6.10 cirbondengine.cpp File Reference

```
#include <pch.h>
#include "cirbondengine.hpp"
```

# 6.11 cirbondengine.hpp File Reference

Bond pricing engine based on CIR model.

```
#include <Calculation/Calculation.h>
#include <ql/instruments/bonds/zerocouponbond.hpp>
#include <ql/pricingengines/genericmodelengine.hpp>
#include <ql/models/shortrate/onefactormodels/coxingersollross.hpp>
```

### Classes

• class QuantLib::CIRBondEngine

A CIRBondEngine (p. 24) is a bond pricing engine based on CIR model.

### 6.11.1 Detailed Description

Bond pricing engine based on CIR model.

### 6.12 cirdefaultmodel.cpp File Reference

```
#include <pch.h>
#include "cirdefaultmodel.hpp"
```

### 6.13 cirdefaultmodel.hpp File Reference

Intensity default model based on the CIR process.

```
#include <Calculation/Calculation.h>
#include <ql/models/defaultmodel.hpp>
```

#### **Classes**

· class QuantLib::CirDefaultModel

Intensity default model based on the CIR process.

### 6.13.1 Detailed Description

Intensity default model based on the CIR process.

# 6.14 cirprocess.hpp File Reference

#### CIR process class.

```
#include <Calculation/Calculation.h>
#include <ql/processes/squarerootprocess.hpp>
```

#### **Classes**

• class QuantLib::CIRprocess

CIR process class.

### 6.14.1 Detailed Description

CIR process class.

# 6.15 counterparty.cpp File Reference

```
#include <pch.h>
#include <ql/credit/counterparty.hpp>
#include <ql/models/default/deterministicdefaultmodel.hpp>
#include <ql/errors.hpp>
#include <boost/cast.hpp>
```

### 6.16 counterparty.hpp File Reference

Counterparty class.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <boost/cast.hpp>
#include <ql/termstructures/credit/piecewisedefaultcurve.hpp>
#include <ql/math/interpolations/backwardflatinterpolation.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/time/dategenerationrule.hpp>
#include <ql/time/calendars/target.hpp>
#include <ql/models/defaultmodel.hpp>
#include <ql/models/defaultcdshelper.hpp>
```

#### **Classes**

· class QuantLib::Counterparty

Counterparty (p. 27) class.

### 6.16.1 Detailed Description

Counterparty class.

# 6.17 creditvarpathpricer.hpp File Reference

Abstract base class used to calculate default value of each path.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/methods/montecarlo/multipath.hpp>
```

### Classes

class QuantLib::CreditVaRPathPricer< ENGINE >

Abstract base class used to calculate default value of each path.

### 6.17.1 Detailed Description

Abstract base class used to calculate default value of each path.

This class is used to calculate default value of each path. To use this class, derived class must implement the pure virtual function defaultNPV.

Date

2015-04-15

### 6.18 crosscurrencyswap.hpp File Reference

fixed-fixed cross currency swap class

```
#include <Calculation/Calculation.h>
#include <ql/instruments/swap.hpp>
#include <ql/currency.hpp>
```

class QuantLib::CrossCurrencySwap

fixed-fixed cross currency swap class

class QuantLib::CrossCurrencySwap::arguments

Arguments for cross currency swap calculation

· class QuantLib::CrossCurrencySwap::results

Results for cross currency swap calculation

class QuantLib::CrossCurrencySwap::engine

base class for cross currency swap pricing engine

### 6.18.1 Detailed Description

fixed-fixed cross currency swap class

### 6.19 defaultcdshelper.hpp File Reference

CDS helpers to calibrate default model.

```
#include <Calculation\Calculation.h>
#include <boost/shared_ptr.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/models/calibrationhelper.hpp>
```

### **Classes**

class QuantLib::DefaultCdsHelper

CDS helpers to calibrate default model.

### 6.19.1 Detailed Description

CDS helpers to calibrate default model.

### 6.20 defaultmodel.hpp File Reference

Abstract base class for default model.

```
#include <Calculation/Calculation.h>
#include <ql/models/model.hpp>
#include <ql/stochasticprocess.hpp>
```

#### Classes

· class QuantLib::DefaultModel

abstract base class for default model

### 6.20.1 Detailed Description

Abstract base class for default model.

# 6.21 deterministicdefaultmodel.cpp File Reference

```
#include <pch.h>
#include "deterministicdefaultmodel.hpp"
```

# 6.22 deterministicdefaultmodel.hpp File Reference

intensity default model based on deterministic default intensity

```
#include <ql/models/defaultmodel.hpp>
#include <Calculation/Calculation.h>
```

### Classes

· class QuantLib::DeterministicDefaultModel

Intensity default model based on deterministic default intensity.

### 6.22.1 Detailed Description

intensity default model based on deterministic default intensity

# 6.23 equityswap.cpp File Reference

```
#include <pch.h>
#include "equityswap.hpp"
```

# 6.24 equityswap.hpp File Reference

equity swap class

```
#include <Calculation/Calculation.h>
#include <ql/instruments/swap.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/time/date.hpp>
#include <ql/time/daycounter.hpp>
#include <ql/time/schedule.hpp>
#include <ql/pricingengine.hpp>
#include <boost/shared_ptr.hpp>
#include <ql/processes/blackscholesprocess.hpp>
```

· class QuantLib::EquitySwap

equity swap class

class QuantLib::EquitySwap::arguments

Arguments for equity swap calculation

class QuantLib::EquitySwap::results

Results class of equity swap

class QuantLib::EquitySwap::engine

equity option engine base class

### 6.24.1 Detailed Description

equity swap class

### 6.25 exactsimulation.hpp File Reference

```
#include <ql/types.hpp>
#include <ql/math/copulas/gaussiancopula.hpp>
#include <ql/math/statistics/statistics.hpp>
```

### Classes

class QuantLib::ExactSimulation< SIM, RNG, S, C >

### 6.26 jcirprocess.hpp File Reference

#### JCIR process class.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/processes/eulerdiscretization.hpp>
#include <ql/math/distributions/poissondistribution.hpp>
#include <ql/math/randomnumbers/mt19937uniformrng.hpp>
#include <ql/math/randomnumbers/seedgenerator.hpp>
```

### Classes

class QuantLib::GeneralizedJcirProcess< \_\_URng\_Poisson\_Type, \_\_URng\_Exp\_Type >
 Jump diffusion CIR process class.

### **Typedefs**

typedef GeneralizedJcirProcess
 MersenneTwisterUniformRng, MersenneTwisterUniformRng > Quant ←
 Lib::JcirProcess

### 6.26.1 Detailed Description

JCIR process class.

### 6.27 kouprocess.hpp File Reference

#### Kou process class.

```
#include <Calculation/Calculation.h>
#include <ql/math/randomnumbers/seedgenerator.hpp>
#include <ql/processes/blackscholesprocess.hpp>
#include <ql/processes/eulerdiscretization.hpp>
#include <ql/math/randomnumbers/mt19937uniformrng.hpp>
```

### **Classes**

class QuantLib::GeneralizedKouProcess < \_\_URng\_Poisson\_Type, \_\_URng\_DoubleExpDist\_\_Type >
 Kou process class.

#### **Typedefs**

typedef GeneralizedKouProcess
 MersenneTwisterUniformRng, MersenneTwisterUniformRng > Quant ← Lib::KouProcess

### 6.27.1 Detailed Description

Kou process class.

### 6.28 kouprocesscalibrator.cpp File Reference

```
#include <pch.h>
#include <ql/calibration/kouprocesscalibrator.hpp>
#include <ql/pricingengines/vanilla/analytickoueuropeanengine.hpp>
```

### 6.29 kouprocesscalibrator.hpp File Reference

### Kou process calibrator class.

```
#include <Calculation/Calculation.h>
#include <memory>
#include <vector>
#include <ql/math/optimization/all.hpp>
#include <ql/termstructures/volatility/equityfx/blackconstantvol.hpp>
```

#### **Classes**

class QuantLib::KouProcessCalibrator

A Kou process calibrator class which is used to do calibration using sets of vanilla options.

#### 6.29.1 Detailed Description

Kou process calibrator class.

### 6.30 mcbvaengine.hpp File Reference

Pricing engine class for BVA calculation via Monte Carlo Simulation.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <ql/qldefines.hpp>
#include <ql/types.hpp>
#include <ql/math/matrix.hpp>
#include <ql/methods/montecarlo/mctraits.hpp>
#include <ql/math/statistics/arraystatistics.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/mcsimulation.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/methods/montecarlo/multivaluemctraits.hpp>
#include <ql/pricingengines/credit/mctvamodel.hpp>
#include <ql/instruments/credit/bva.hpp>
#include <ql/pricingengines/credit/bvaengine.hpp>
#include <ql/pricingengines/credit/bvapathpricer.hpp>
#include <boost/enable_shared_from_this.hpp>
```

#### Classes

class QuantLib::MCBVAEngine< RNG, S >

Pricing engine class for BVA (p. 20) calculation via Monte Carlo Simulation.

### 6.30.1 Detailed Description

Pricing engine class for BVA calculation via Monte Carlo Simulation.

### 6.31 mccreditvarengine.hpp File Reference

Abstract base class for Credit VaR calculation via Monte Carlo Simulation.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <dalculation.hpp>
#include <ql/qldefines.hpp>
#include <ql/types.hpp>
#include <ql/math/matrix.hpp>
#include <ql/methods/montecarlo/mctraits.hpp>
#include <ql/math/statistics/arraystatistics.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/mcsimulation.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/pricingengines/bvaengine.hpp>
#include <ql/pricingengines/bvaengine.hpp>
#include <ql/methods/montecarlo/multivaluemctraits.hpp>
#include <ql/pricingengines/credit/creditvarengine.hpp>
#include <ql/pricingengines/credit/creditvarengine.hpp>
#include <ql/pricingengines/credit/creditvarengine.hpp>
```

#### Classes

class QuantLib::MCCreditVaREngine< INST, RNG, S >

Abstract base class for Credit VaR calculation via Monte Carlo Simulation.

### 6.31.1 Detailed Description

Abstract base class for Credit VaR calculation via Monte Carlo Simulation.

This class is used to calculate Credit VaR via Monte Carlo Simulation.

Date

2015-04-15

### 6.32 mccrosscurrencyswapucvaengine.cpp File Reference

```
#include <pch.h>
#include "mccrosscurrencyswapucvaengine.hpp"
#include "ucvapathpricer.hpp"
```

### **Classes**

class QuantLib::CrossCurrencySwapUCVAPathPricer

UCVA Path Pricer for fixed-fixed cross currency swap.

### 6.33 mccrosscurrencyswapucvaengine.hpp File Reference

Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap.

```
#include <Calculation/Calculation.h>
#include <boost/enable_shared_from_this.hpp>
#include <ql/types.hpp>
#include <ql/time/calendar.hpp>
#include <ql/time/daycounter.hpp>
#include <ql/math/randomnumbers/seedgenerator.hpp>
#include <ql/pricingengines/ucva/mcucvaengine.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/termstructures/voltermstructure.hpp>
#include <ql/instruments/swap/crosscurrencyswap.hpp>
#include <ql/processes/cirprocess.hpp>
```

· class QuantLib::MCCrossCurrencySwapUCVAEngine

Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap.

### 6.33.1 Detailed Description

Monte Carlo pricing engine for UCVA calculation of fixed-fixed cross currency swap.

# 6.34 mcequityswapucvaengine.cpp File Reference

```
#include <pch.h>
#include "mcequityswapucvaengine.hpp"
#include <ql/pricingengines/swap/analyticequityswapengine.hpp>
#include <ql/pricingengines/ucva/ucvapathpricer.hpp>
```

### Classes

· class QuantLib::EquitySwapUCVAPathPricer

UCVA Path Pricer for equity swap.

# 6.35 mcequityswapucvaengine.hpp File Reference

Monte Carlo pricing engine for UCVA calculation of equity swap.

```
#include <Calculation/Calculation.h>
#include <boost/enable_shared_from_this.hpp>
#include <ql/instruments/swap/equityswap.hpp>
#include <ql/pricingengines/ucva/mcucvaengine.hpp>
#include <ql/credit/counterparty.hpp>
```

### Classes

class QuantLib::MCEquitySwapUCVAEngine

Monte Carlo pricing engine for UCVA calculation of equity swap.

### 6.35.1 Detailed Description

Monte Carlo pricing engine for UCVA calculation of equity swap.

# 6.36 mctvacrosscurrencyswapmodel.cpp File Reference

```
#include <pch.h>
#include "mctvacrosscurrencyswapmodel.hpp"
```

# 6.37 mctvacrosscurrencyswapmodel.hpp File Reference

### TVA model for cross currency swap.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/instruments/swap/crosscurrencyswap.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/credit/mctvamodel.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/methods/montecarlo/multipath.hpp>
```

### **Classes**

class QuantLib::MCTVACrossCurrencySwapModel

TVA (p. 62) model for cross currency swap.

### 6.37.1 Detailed Description

TVA model for cross currency swap.

### 6.38 mctvaengine.hpp File Reference

Pricing engine class for TVA calculation via Monte Carlo Simulation.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <ql/qldefines.hpp>
#include <ql/types.hpp>
#include <ql/math/matrix.hpp>
#include <ql/methods/montecarlo/mctraits.hpp>
#include <ql/math/statistics/arraystatistics.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/mcsimulation.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/methods/montecarlo/multivaluemctraits.hpp>
#include <ql/pricingengines/credit/mctvamodel.hpp>
#include <ql/instruments/credit/tva.hpp>
#include <ql/pricingengines/credit/tvaengine.hpp>
#include <ql/pricingengines/credit/tvapathpricer.hpp>
#include <boost/enable shared from this.hpp>
```

class QuantLib::MCTVAEngine< RNG, S >

Pricing engine class for TVA (p. 62) calculation via Monte Carlo Simulation.

### 6.38.1 Detailed Description

Pricing engine class for TVA calculation via Monte Carlo Simulation.

# 6.39 mctvaequityswapmodel.cpp File Reference

```
#include <pch.h>
#include "mctvaequityswapmodel.hpp"
#include <ql/pricingengines/swap/analyticequityswapengine.hpp>
#include <ql/instruments/swap/equityswap.hpp>
```

### 6.40 mctvaequityswapmodel.hpp File Reference

#### TVA model for equity swap.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/instruments/swap/equityswap.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/credit/mctvamodel.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/methods/montecarlo/multipath.hpp>
```

### Classes

• class QuantLib::MCTVAEquitySwapModel

TVA (p. 62) model for equity swap.

### 6.40.1 Detailed Description

TVA model for equity swap.

# 6.41 mctvamodel.hpp File Reference

#### Base class for tva model.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/instrument.hpp>
#include <ql/pricingengine.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/methods/montecarlo/multipath.hpp>
#include <ql/time/all.hpp>
```

#### Classes

· class QuantLib::MCTVAModel

base class for tva model.

### 6.41.1 Detailed Description

Base class for tva model.

### 6.42 mctvavanillaswapmodel.cpp File Reference

```
#include <pch.h>
#include "mctvavanillaswapmodel.hpp"
```

### 6.43 mctvavanillaswapmodel.hpp File Reference

#### TVA model for vanilla swap.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/instruments/vanillaswap.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/credit/mctvamodel.hpp>
#include <ql/termstructures/yieldtermstructure.hpp>
#include <ql/methods/montecarlo/multipath.hpp>
#include <ql/termstructures/volatility/equityfx/blackconstantvol.hpp>
```

class QuantLib::MCTVAVanillaSwapModel

TVA (p. 62) model for vanilla swap.

### 6.43.1 Detailed Description

TVA model for vanilla swap.

### 6.44 mcucvaengine.hpp File Reference

Abstract base class for UCVA calculation via Monte Carlo Simulation.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <ql/qldefines.hpp>
#include <ql/types.hpp>
#include <ql/math/matrix.hpp>
#include <ql/methods/montecarlo/mctraits.hpp>
#include <ql/math/statistics/arraystatistics.hpp>
#include <ql/stochasticprocess.hpp>
#include <ql/pricingengines/mcsimulation.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/pricingengines/ucvaengine.hpp>
#include <ql/pricingengines/ucvaengine.hpp>
#include <ql/methods/montecarlo/multivaluemctraits.hpp>
#include <ql/methods/montecarlo/multivaluemctraits.hpp>
```

### Classes

- class QuantLib::MCUCVAEngine < INST, RNG, S >

Abstract base class for UCVA calculation via Monte Carlo Simulation.

### 6.44.1 Detailed Description

Abstract base class for UCVA calculation via Monte Carlo Simulation.

### 6.45 mcvanillaswapucvaengine.cpp File Reference

```
#include <pch.h>
#include "mcvanillaswapucvaengine.hpp"
#include "ucvapathpricer.hpp"
```

#### **Classes**

· class QuantLib::VanillaSwapUCVAPathPricer

UCVA Path Pricer for vanilla swap.

# 6.46 mcvanillaswapucvaengine.hpp File Reference

Monte Carlo pricing engine for UCVA calculation of vanilla swap.

```
#include <Calculation/Calculation.h>
#include <boost/enable_shared_from_this.hpp>
#include <ql/math/randomnumbers/seedgenerator.hpp>
#include <ql/instruments/vanillaswap.hpp>
#include <ql/pricingengines/ucva/mcucvaengine.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/processes/cirprocess.hpp>
#include <ql/models/shortrate/onefactormodels/coxingersollross.hpp>
```

#### Classes

class QuantLib::MCVanillaSwapUCVAEngine

Monte Carlo pricing engine for UCVA calculation of vanilla swap.

### 6.46.1 Detailed Description

Monte Carlo pricing engine for UCVA calculation of vanilla swap.

# 6.47 multivaluemctraits.hpp File Reference

Traits class for multi-state Monte Carlo Simulation.

```
#include <Calculation/Calculation.h>
#include <ql/types.hpp>
#include <ql/methods/montecarlo/mctraits.hpp>
```

### Classes

struct QuantLib::MultiValueMultiVariate< RNG >

Traits class for multi-state Monte Carlo Simulation.

### 6.47.1 Detailed Description

Traits class for multi-state Monte Carlo Simulation.

# 6.48 shortratetermstructure.cpp File Reference

```
#include <pch.h>
#include "shortratetermstructure.hpp"
```

# 6.49 shortratetermstructure.hpp File Reference

Interest-rate term structure class based on short-rate model.

```
#include <Calculation/Calculation.h>
#include <ql/models/shortrate/onefactormodel.hpp>
#include <ql/termstructure.hpp>
```

· class QuantLib::ShortRateTermStructure

Interest-rate term structure class based on short-rate model.

#### 6.49.1 Detailed Description

Interest-rate term structure class based on short-rate model.

# 6.50 tva.hpp File Reference

Total value adjustment manager class.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/types.hpp>
#include <ql/instrument.hpp>
#include <ql/pricingengine.hpp>
#include <ql/credit/counterparty.hpp>
```

#### Classes

· class QuantLib::TVA

Total value adjustment manager class.

class QuantLib::TVA::arguments
 class QuantLib::TVA::results
 class QuantLib::TVA::engine

### 6.50.1 Detailed Description

Total value adjustment manager class.

# 6.51 tvaengine.hpp File Reference

### Base class for tva engine.

```
#include <Calculation/Calculation.h>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ql/types.hpp>
#include <ql/instrument.hpp>
#include <ql/credit/counterparty.hpp>
#include <ql/models/shortratetermstructure.hpp>
```

#### Classes

class QuantLib::TVAEngine < INST >

Base class for bva engine.

### 6.51.1 Detailed Description

Base class for tva engine.

# 6.52 tvapathpricer.hpp File Reference

Path pricer class used to calculate the TVA on each Multi-path for TVAEngine class.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
```

#### **Classes**

• class QuantLib::TVAPathPricer< ENGINE >

Path pricer class used to calculate the TVA (p. 62) on each Multi-path for TVAEngine (p. 63) class.

### 6.52.1 Detailed Description

Path pricer class used to calculate the TVA on each Multi-path for TVAEngine class.

### 6.53 ucvaengine.hpp File Reference

Abstract base class for ucva engine.

```
#include <Calculation/Calculation.h>
#include <ql/pricingengine.hpp>
```

### Classes

class QuantLib::UCVAEngine < INST >

Abstract base class for ucva engine.

### 6.53.1 Detailed Description

Abstract base class for ucva engine.

### 6.54 ucvapathpricer.hpp File Reference

Abstract base class used to calculate cash flow at default time of each path under the setting of UCVA.

```
#include <Calculation/Calculation.h>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
```

class QuantLib::UCVAPathPricer< ENGINE >

Abstract base class used to calculate cash flow at default time of each path under the setting of UCVA.

### 6.54.1 Detailed Description

Abstract base class used to calculate cash flow at default time of each path under the setting of UCVA.

# 6.55 zerocouponbondhelper.cpp File Reference

```
#include <pch.h>
#include <ql/pricingengines/bond/discountingbondengine.hpp>
#include "zerocouponbondhelper.hpp"
```

### 6.56 zerocouponbondhelper.hpp File Reference

Zero coupon bond calibration helper.

```
#include <Calculation/Calculation.h>
#include <ql/models/calibrationhelper.hpp>
#include <ql/instruments/bonds/zerocouponbond.hpp>
```

### **Classes**

· class QuantLib::ZerocouponbondHelper

Zero coupon bond calibration helper.

### 6.56.1 Detailed Description

Zero coupon bond calibration helper.

# Index

GeneralizedJcirProcess

QuantLib::GeneralizedJcirProcess, 40

analyticequityswapengine.cpp, 71 GeneralizedKouProcess analyticequityswapengine.hpp, 71 QuantLib::GeneralizedKouProcess, 41 analytickoueuropeanengine.cpp, 71 ImplicitMilstein analytickoueuropeanengine.hpp, 71 QuantLib::CIRprocess, 26 at1pdefaultmodel.cpp, 72 at1pdefaultmodel.hpp, 72 jcirprocess.hpp, 79 **BVA** KouProcessCalibrator QuantLib::BVA, 21 QuantLib::KouProcessCalibrator, 45 **BVAEngine** kouprocess.hpp, 80 QuantLib::BVAEngine, 22 kouprocesscalibrator.cpp, 80 blackPrice kouprocesscalibrator.hpp, 81 QuantLib::ZerocouponbondHelper, 69 bva.hpp, 73 **MCBVAEngine** byaengine.hpp, 73 QuantLib::MCBVAEngine, 46 bvapathpricer.hpp, 73 **MCTVACrossCurrencySwapModel** QuantLib::MCTVACrossCurrencySwapModel, 51 **CIRprocess MCTVAEngine** QuantLib::CIRprocess, 27 QuantLib::MCTVAEngine, 52 calculate MCTVAEquitySwapModel QuantLib::AnalyticESEngine, 15 QuantLib::MCTVAEquitySwapModel, 54 cirbondengine.cpp, 74 **MCTVAVanillaSwapModel** cirbondengine.hpp, 74 QuantLib::MCTVAVanillaSwapModel, 56 cirdefaultmodel.cpp, 74 mcbvaengine.hpp, 81 cirdefaultmodel.hpp, 75 mccreditvarengine.hpp, 82 cirprocess.hpp, 75 mccrosscurrencyswapucvaengine.cpp, 82 counterparty.cpp, 75 mccrosscurrencyswapucvaengine.hpp, 82 counterparty.hpp, 75 mcequityswapucvaengine.cpp, 83 creditvarpathpricer.hpp, 76 mcequityswapucvaengine.hpp, 83 crosscurrencyswap.hpp, 76 mctvacrosscurrencyswapmodel.cpp, 84 mctvacrosscurrencyswapmodel.hpp, 84 defaultNPV QuantLib::VanillaSwapUCVAPathPricer, 68 mctvaengine.hpp, 84 mctvaequityswapmodel.cpp, 85 defaultTime mctvaequityswapmodel.hpp, 85 QuantLib::CirDefaultModel, 25 mctvamodel.hpp, 86 QuantLib::DefaultModel, 33 mctvavanillaswapmodel.cpp, 86 defaultcdshelper.hpp, 77 mctvavanillaswapmodel.hpp, 86 defaultmodel.hpp, 77 mcucvaengine.hpp, 87 deterministicdefaultmodel.cpp, 78 mcvanillaswapucvaengine.cpp, 87 deterministicdefaultmodel.hpp, 78 mcvanillaswapucvaengine.hpp, 88 Discretization QuantLib::CIRprocess, 26 Milstein QuantLib::CIRprocess, 26 equityswap.cpp, 78 modelValue equityswap.hpp, 78 QuantLib::ZerocouponbondHelper, 69 Euler multivaluemctraits.hpp, 88 QuantLib::CIRprocess, 26 exactsimulation.hpp, 79 NonCentralChiSquareVariance

QuantLib::CIRprocess, 26

payDomestic

94 INDEX

QuantLib::CrossCurrencySwap, 31	GeneralizedKouProcess, 41
payForeign	QuantLib::GeneralizedKouProcess <urng_←< td=""></urng_←<>
QuantLib::CrossCurrencySwap, 31	Poisson_Type,URng_DoubleExpDist_←
Payer	_Type >, 40
QuantLib::EquitySwap, 37	QuantLib::GenericArrayStatistics< StatisticsType >, 41
0 11 47 0	QuantLib::KouCalibrationHelper, 43
QuantLib::AT1Pmodel, 19	QuantLib::KouHelper, 43
QuantLib::AnalyticESEngine, 15	QuantLib::KouModel, 44
calculate, 15	QuantLib::KouProcessCalibrator, 44
QuantLib::AnalyticKouEuropeanEngine, 16	KouProcessCalibrator, 45
QuantLib::BVA, 20	QuantLib::MCBVAEngine
BVA, 21	MCBVAEngine, 46
QuantLib::BVA::arguments, 18	QuantLib::MCBVAEngine < RNG, S >, 45
QuantLib::BVA::engine, 35	QuantLib::MCCreditVaREngine < INST, RNG, S >, 47
QuantLib::BVA::results, 59	QuantLib::MCCrossCurrencySwapUCVAEngine, 48
QuantLib::BVAEngine	QuantLib::MCEquitySwapUCVAEngine, 49
BVAEngine, 22	QuantLib::MCTVACrossCurrencySwapModel, 50
QuantLib::BVAEngine < INST >, 21	MCTVACrossCurrencySwapModel, 51
QuantLib::BVAPathPricer< ENGINE >, 22	QuantLib::MCTVAEngine
QuantLib::CIRBondEngine, 24	MCTVAEngine, 52
QuantLib::CIRprocess, 25	QuantLib::MCTVAEngine < RNG, S >, 51
CIRprocess, 27	QuantLib::MCTVAEquitySwapModel, 53
Discretization, 26	MCTVAEquitySwapModel, 54
Euler, 26	QuantLib::MCTVAModel, 54
ImplicitMilstein, 26	QuantLib::MCTVAVanillaSwapModel, 55
Milstein, 26	MCTVAVanillaSwapModel, 56
NonCentralChiSquareVariance, 26	QuantLib::MCUCVAEngine < INST, RNG, S >, 56
QuantLib::CirDefaultModel, 24	QuantLib::MCVanillaSwapUCVAEngine, 57
defaultTime, 25	•
QuantLib::Counterparty, 27	QuantLib::MultiValueMultiVariate< RNG >, 58 QuantLib::ShortRateTermStructure, 61
QuantLib::CreditVaREngine < INST >, 28	
QuantLib::CreditVaRPathPricer< ENGINE >, 29	QuantLib::SimEquitySwapEngine, 62
QuantLib::CrossCurrencySwap, 30	QuantLib::TVA, 62
payDomestic, 31	QuantLib::TVA::arguments, 16
payForeign, 31	QuantLib::TVA::engine, 36 QuantLib::TVA::results, 60
Type, 31	QuantLib::TVAE.results, 60  QuantLib::TVAEngine
QuantLib::CrossCurrencySwap::arguments, 17	<u> </u>
QuantLib::CrossCurrencySwap::engine, 34	TVAEngine, 64
QuantLib::CrossCurrencySwap::results, 60	QuantLib::TVAEngine < INST >, 63
QuantLib::CrossCurrencySwapUCVAPathPricer, 31	QuantLib::TVAPathPricer< ENGINE >, 64
QuantLib::DefaultCdsHelper, 32	QuantLib::UCVAEngine < INST >, 65
QuantLib::DefaultModel, 33	QuantLib::UCVAPathPricer< ENGINE >, 66
defaultTime, 33	QuantLib::VanillaSwapUCVAPathPricer, 67
QuantLib::DeterministicDefaultModel, 34	defaultNPV, 68
QuantLib::EquitySwap, 36	QuantLib::ZerocouponbondHelper, 68
Payer, 37	blackPrice, 69
Receiver, 37	modelValue, 69
Type, 37	December
QuantLib::EquitySwap::arguments, 17	Receiver
QuantLib::EquitySwap::engine, 35	QuantLib::EquitySwap, 37
QuantLib::EquitySwap::erigine, 33	RiskAnalysisTool::Calculation::Sample::SampleClass,
QuantLib::EquitySwapUCVAPathPricer, 38	60
QuantLib::ExactSimulation< SIM, RNG, S, C >, 38	RiskAnalysisTool::Calculation::sealed, 61
	ah autustata umaatuu atuu a arra 20
QuantLib::ExposureModel, 38	shortratetermstructure.cpp, 88
QuantLib::GeneralizedJcirProcess	shortratetermstructure.hpp, 88
GeneralizedJcirProcess, 40	TVA Engine
QuantLib::GeneralizedJcirProcess <urng_←< td=""><td>TVAEngine Ought ib::TVAEnging 64</td></urng_←<>	TVAEngine Ought ib::TVAEnging 64
Poisson_Type,URng_Exp_Type >, 39	QuantLib::TVAEngine, 64
QuantLib::GeneralizedKouProcess	tva.hpp, 89

INDEX 95

tvaengine.hpp, 89 tvapathpricer.hpp, 90 Type

QuantLib::CrossCurrencySwap, 31 QuantLib::EquitySwap, 37

ucvaengine.hpp, 90 ucvapathpricer.hpp, 90

zerocouponbondhelper.cpp, 91 zerocouponbondhelper.hpp, 91