

# terreneuve Reference Manual

Generated by Doxygen 1.3.6

Thu Dec 22 23:12:35 2005



# Contents

<b>1</b>	<b>terreneuve Namespace Index</b>	<b>1</b>
1.1	terreneuve Namespace List . . . . .	1
<b>2</b>	<b>terreneuve Hierarchical Index</b>	<b>3</b>
2.1	terreneuve Class Hierarchy . . . . .	3
<b>3</b>	<b>terreneuve Class Index</b>	<b>5</b>
3.1	terreneuve Class List . . . . .	5
<b>4</b>	<b>terreneuve File Index</b>	<b>7</b>
4.1	terreneuve File List . . . . .	7
<b>5</b>	<b>terreneuve Namespace Documentation</b>	<b>11</b>
5.1	std Namespace Reference . . . . .	11
<b>6</b>	<b>terreneuve Class Documentation</b>	<b>13</b>
6.1	asset Class Reference . . . . .	13
6.2	binomialTree Class Reference . . . . .	20
6.3	BlackScholes Class Reference . . . . .	26
6.4	bond Class Reference . . . . .	33
6.5	cachedval Struct Reference . . . . .	39
6.6	cashflow Class Reference . . . . .	40
6.7	CashFlow Class Reference . . . . .	42
6.8	convertiblebond Class Reference . . . . .	44
6.9	creditCurve Class Reference . . . . .	52
6.10	CreditSpreadPoint Class Reference . . . . .	65
6.11	CSVParser Class Reference . . . . .	68
6.12	Date Class Reference . . . . .	71
6.13	Drift Class Reference . . . . .	82
6.14	Exotics Class Reference . . . . .	86

6.15	FileReader Class Reference . . . . .	90
6.16	flowSchedule Class Reference . . . . .	93
6.17	GaussianProcess Class Reference . . . . .	96
6.18	importData Class Reference . . . . .	100
6.19	interpolator Class Reference . . . . .	104
6.20	marketData Struct Reference . . . . .	108
6.21	Matrix Class Reference . . . . .	109
6.22	MCEngine Class Reference . . . . .	127
6.23	MersenneTwister Class Reference . . . . .	137
6.24	OptionStrategy Class Reference . . . . .	140
6.25	ParkMiller Class Reference . . . . .	148
6.26	PayOff Class Reference . . . . .	151
6.27	Portfolio Class Reference . . . . .	158
6.28	RainbowOption Class Reference . . . . .	165
6.29	RandC Class Reference . . . . .	182
6.30	Random Class Reference . . . . .	185
6.31	RandomGenerator Class Reference . . . . .	189
6.32	riskybond Class Reference . . . . .	192
6.33	Sobol Class Reference . . . . .	195
6.34	StringTokenizer Class Reference . . . . .	198
6.35	SwapLeg Class Reference . . . . .	201
6.36	treasurybond Class Reference . . . . .	203
6.37	UsDate Class Reference . . . . .	206
6.38	VanillaSwap Class Reference . . . . .	207
6.39	VarianceSwap Class Reference . . . . .	210
6.40	volsurface Class Reference . . . . .	213
6.41	volsurfaceparams Class Reference . . . . .	219
6.42	yieldCurve Class Reference . . . . .	220
6.43	yieldPoint Class Reference . . . . .	230
<b>7</b>	<b>terreneuve File Documentation</b>	<b>235</b>
7.1	asset.cpp File Reference . . . . .	235
7.2	asset.h File Reference . . . . .	236
7.3	binomialTree.cpp File Reference . . . . .	237
7.4	binomialTree.h File Reference . . . . .	238
7.5	BlackScholes.cpp File Reference . . . . .	240
7.6	BlackScholes.h File Reference . . . . .	241

---

7.7	bond.cpp File Reference . . . . .	242
7.8	bond.h File Reference . . . . .	243
7.9	CashFlow.cpp File Reference . . . . .	244
7.10	CashFlow.h File Reference . . . . .	245
7.11	convertiblebond.cpp File Reference . . . . .	246
7.12	convertiblebond.h File Reference . . . . .	247
7.13	creditCurve.cpp File Reference . . . . .	249
7.14	creditCurve.h File Reference . . . . .	250
7.15	credits.cpp File Reference . . . . .	252
7.16	csvparser.cpp File Reference . . . . .	253
7.17	csvparser.h File Reference . . . . .	254
7.18	date.cpp File Reference . . . . .	255
7.19	date.h File Reference . . . . .	256
7.20	Drift.cpp File Reference . . . . .	259
7.21	Drift.h File Reference . . . . .	260
7.22	Exotics.cpp File Reference . . . . .	261
7.23	Exotics.h File Reference . . . . .	262
7.24	filereader.cpp File Reference . . . . .	264
7.25	filereader.h File Reference . . . . .	265
7.26	GaussianProcess.cpp File Reference . . . . .	266
7.27	GaussianProcess.h File Reference . . . . .	267
7.28	importData.cpp File Reference . . . . .	268
7.29	importData.h File Reference . . . . .	269
7.30	interpolator.cpp File Reference . . . . .	270
7.31	interpolator.h File Reference . . . . .	271
7.32	main.cpp File Reference . . . . .	272
7.33	main.h File Reference . . . . .	273
7.34	mainbinomialtree.cpp File Reference . . . . .	274
7.35	mainbond.cpp File Reference . . . . .	275
7.36	mainconvertiblebond.cpp File Reference . . . . .	276
7.37	maincreditcurves.cpp File Reference . . . . .	277
7.38	maindate.cpp File Reference . . . . .	278
7.39	mainfilereader.cpp File Reference . . . . .	279
7.40	maininterpolator.cpp File Reference . . . . .	280
7.41	mainIRVanillaSwap.cpp File Reference . . . . .	281
7.42	mainmatrix.cpp File Reference . . . . .	282

---

7.43	mainmontecarlo.cpp File Reference . . . . .	283
7.44	mainoptionstrategy.cpp File Reference . . . . .	285
7.45	mainrainbowoptions.cpp File Reference . . . . .	286
7.46	maintestasset.cpp File Reference . . . . .	287
7.47	mainvarianceswap.cpp File Reference . . . . .	288
7.48	mainvolsurface.cpp File Reference . . . . .	289
7.49	mainyieldcurves.cpp File Reference . . . . .	290
7.50	matrix.cpp File Reference . . . . .	291
7.51	matrix.h File Reference . . . . .	292
7.52	MCEngine.cpp File Reference . . . . .	293
7.53	MCEngine.h File Reference . . . . .	294
7.54	MersenneTwister.cpp File Reference . . . . .	295
7.55	MersenneTwister.h File Reference . . . . .	296
7.56	Normals.cpp File Reference . . . . .	297
7.57	Normals.h File Reference . . . . .	299
7.58	OptionStrategy.cpp File Reference . . . . .	301
7.59	OptionStrategy.h File Reference . . . . .	302
7.60	ParkMiller.cpp File Reference . . . . .	303
7.61	ParkMiller.h File Reference . . . . .	304
7.62	PayOff.cpp File Reference . . . . .	306
7.63	PayOff.h File Reference . . . . .	307
7.64	PortFolio.cpp File Reference . . . . .	308
7.65	PortFolio.h File Reference . . . . .	309
7.66	productsCreation.cpp File Reference . . . . .	310
7.67	productsCreation.h File Reference . . . . .	316
7.68	rainbowoption.cpp File Reference . . . . .	322
7.69	rainbowoption.h File Reference . . . . .	323
7.70	RandC.cpp File Reference . . . . .	326
7.71	RandC.h File Reference . . . . .	327
7.72	Random.cpp File Reference . . . . .	328
7.73	Random.h File Reference . . . . .	329
7.74	RandomGenerator.cpp File Reference . . . . .	330
7.75	RandomGenerator.h File Reference . . . . .	331
7.76	Sobol.cpp File Reference . . . . .	332
7.77	Sobol.h File Reference . . . . .	333
7.78	StringTokenizer.cpp File Reference . . . . .	334

---

7.79	StringTokenizer.h File Reference . . . . .	335
7.80	SwapLeg.cpp File Reference . . . . .	336
7.81	SwapLeg.h File Reference . . . . .	337
7.82	testObjects.cpp File Reference . . . . .	338
7.83	testObjects.h File Reference . . . . .	339
7.84	types.h File Reference . . . . .	346
7.85	UsDate.cpp File Reference . . . . .	352
7.86	UsDate.h File Reference . . . . .	353
7.87	utils.cpp File Reference . . . . .	354
7.88	utils.h File Reference . . . . .	357
7.89	VanillaSwap.cpp File Reference . . . . .	360
7.90	VanillaSwap.h File Reference . . . . .	361
7.91	VarianceSwap.cpp File Reference . . . . .	362
7.92	VarianceSwap.h File Reference . . . . .	363
7.93	volsurface.cpp File Reference . . . . .	364
7.94	volsurface.h File Reference . . . . .	365
7.95	yieldCurve.cpp File Reference . . . . .	366
7.96	yieldCurve.h File Reference . . . . .	367





# Chapter 1

## terreneuve Namespace Index

### 1.1 terreneuve Namespace List

Here is a list of all namespaces with brief descriptions:

<b>std</b> .....	11
------------------	----



# Chapter 2

## terreneuve Hierarchical Index

### 2.1 terreneuve Class Hierarchy

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

asset . . . . .	13
binomialTree . . . . .	20
BlackScholes . . . . .	26
bond . . . . .	33
riskybond . . . . .	192
convertiblebond . . . . .	44
treasurybond . . . . .	203
cachedval . . . . .	39
cashflow . . . . .	40
CashFlow . . . . .	42
CreditSpreadPoint . . . . .	65
CSVParser . . . . .	68
Date . . . . .	71
UsDate . . . . .	206
Drift . . . . .	82
Exotics . . . . .	86
FileReader . . . . .	90
flowSchedule . . . . .	93
GaussianProcess . . . . .	96
importData . . . . .	100
interpolator . . . . .	104
marketData . . . . .	108
Matrix . . . . .	109
MCEngine . . . . .	127
OptionStrategy . . . . .	140
PayOff . . . . .	151
Portfolio . . . . .	158
RainbowOption . . . . .	165
Random . . . . .	185
RandomGenerator . . . . .	189
MersenneTwister . . . . .	137
ParkMiller . . . . .	148

RandC . . . . .	182
Sobol . . . . .	195
StringTokenizer . . . . .	198
SwapLeg . . . . .	201
VanillaSwap . . . . .	207
VarianceSwap . . . . .	210
volsurface . . . . .	213
volsurfaceparams . . . . .	219
yieldCurve . . . . .	220
creditCurve . . . . .	52
yieldPoint . . . . .	230

# Chapter 3

## terreneuve Class Index

### 3.1 terreneuve Class List

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

<b>asset</b>	13
<b>binomialTree</b>	20
<b>BlackScholes</b>	26
<b>bond</b>	33
<b>cachedval</b>	39
<b>cashflow</b>	40
<b>CashFlow</b>	42
<b>convertiblebond</b>	44
<b>creditCurve</b>	52
<b>CreditSpreadPoint</b> (Used to encapsulate a spread at a given maturity )	65
<b>CSVParser</b>	68
<b>Date</b>	71
<b>Drift</b>	82
<b>Exotics</b>	86
<b>FileReader</b>	90
<b>flowSchedule</b>	93
<b>GaussianProcess</b>	96
<b>importData</b>	100
<b>interpolator</b>	104
<b>marketData</b>	108
<b>Matrix</b>	109
<b>MCEngine</b>	127
<b>MersenneTwister</b>	137
<b>OptionStrategy</b>	140
<b>ParkMiller</b>	148
<b>PayOff</b>	151
<b>Portfolio</b>	158
<b>RainbowOption</b>	165
<b>RandC</b>	182
<b>Random</b>	185
<b>RandomGenerator</b>	189
<b>riskybond</b>	192
<b>Sobol</b>	195

<b>StringTokenizer</b> . . . . .	198
<b>SwapLeg</b> . . . . .	201
<b>treasurybond</b> . . . . .	203
<b>UsDate</b> . . . . .	206
<b>VanillaSwap</b> . . . . .	207
<b>VarianceSwap</b> . . . . .	210
<b>volsurface</b> . . . . .	213
<b>volsurfaceparams</b> . . . . .	219
<b>yieldCurve</b> . . . . .	220
<b>yieldPoint</b> . . . . .	230

# Chapter 4

## terreneuve File Index

### 4.1 terreneuve File List

Here is a list of all files with brief descriptions:

<b>asset.cpp</b>	235
<b>asset.h</b>	236
<b>binomialTree.cpp</b>	237
<b>binomialTree.h</b>	238
<b>BlackScholes.cpp</b>	240
<b>BlackScholes.h</b>	241
<b>bond.cpp</b>	242
<b>bond.h</b>	243
<b>CashFlow.cpp</b>	244
<b>CashFlow.h</b>	245
<b>convertiblebond.cpp</b>	246
<b>convertiblebond.h</b>	247
<b>creditCurve.cpp</b>	249
<b>creditCurve.h</b>	250
<b>credits.cpp</b>	252
<b>csvparser.cpp</b>	253
<b>csvparser.h</b>	254
<b>date.cpp</b>	255
<b>date.h</b>	256
<b>Drift.cpp</b>	259
<b>Drift.h</b>	260
<b>Exotics.cpp</b>	261
<b>Exotics.h</b>	262
<b>filereader.cpp</b>	264
<b>filereader.h</b>	265
<b>GaussianProcess.cpp</b>	266
<b>GaussianProcess.h</b>	267
<b>importData.cpp</b>	268
<b>importData.h</b>	269
<b>interpolator.cpp</b>	270
<b>interpolator.h</b>	271
<b>main.cpp</b>	272
<b>main.h</b>	273

mainbinomialtree.cpp	274
mainbond.cpp	275
mainconvertiblebond.cpp	276
maincreditcurves.cpp	277
maindate.cpp	278
mainfilereader.cpp	279
maininterpolator.cpp	280
mainIRVanillaSwap.cpp	281
mainmatrix.cpp	282
mainmontecarlo.cpp	283
mainoptionstrategy.cpp	285
mainrainbowoptions.cpp	286
maintestasset.cpp	287
mainvarianceswap.cpp	288
mainvolsurface.cpp	289
mainyieldcurves.cpp	290
matrix.cpp	291
matrix.h	292
MCEngine.cpp	293
MCEngine.h	294
MersenneTwister.cpp	295
MersenneTwister.h	296
Normals.cpp	297
Normals.h	299
OptionStrategy.cpp	301
OptionStrategy.h	302
ParkMiller.cpp	303
ParkMiller.h	304
PayOff.cpp	306
PayOff.h	307
PortFolio.cpp	308
PortFolio.h	309
productsCreation.cpp	310
productsCreation.h	316
rainbowoption.cpp	322
rainbowoption.h	323
RandC.cpp	326
RandC.h	327
Random.cpp	328
Random.h	329
RandomGenerator.cpp	330
RandomGenerator.h	331
Sobol.cpp	332
Sobol.h	333
StringTokenizer.cpp	334
StringTokenizer.h	335
SwapLeg.cpp	336
SwapLeg.h	337
testObjects.cpp	338
testObjects.h	339
types.h	346
UsDate.cpp	352
UsDate.h	353
utils.cpp	354



---

utils.h . . . . .	357
VanillaSwap.cpp . . . . .	360
VanillaSwap.h . . . . .	361
VarianceSwap.cpp . . . . .	362
VarianceSwap.h . . . . .	363
volsurface.cpp . . . . .	364
volsurface.h . . . . .	365
yieldCurve.cpp . . . . .	366
yieldCurve.h . . . . .	367



## Chapter 5

# terreneuve Namespace Documentation

### 5.1 std Namespace Reference



# Chapter 6

## terreneuve Class Documentation

### 6.1 asset Class Reference

```
#include <asset.h>
```

#### Public Member Functions

- **asset** (void)  
*default constructor*
- **asset** (Real price, yieldCurve yc, bool areDividendsAsRate=true, Real divRate=0.0, valarray< flowSchedule > flowsc=valarray< flowSchedule >(), Currency ccy=USD, Real volatility=ASSET\_DEFAULT\_VOL)  
*constructor*
- **asset** (Real price, Real volatility=ASSET\_DEFAULT\_VOL)
- **Real getDelta** ()  
*a stock is delta 1!*
- **Real getRho** (Real T)  
*Rho vs fwd.*
- void **setPrice** (Real p)  
*sets the spot price*
- void **setYieldCurve** (yieldCurve yc)  
*sets the yc*
- void **setDivAsRate** (Real rate=0.0)  
*sets the div rate*
- void **setDivFlows** (flowSchedule fc)  
*sets the flow schedule*
- void **setCcy** (Currency ccy)

*sets the currency*

- void **setVolatility** (**Real** volatility)  
*sets the volatility*
- **Real** **getPrice** () const  
*gets the spot price*
- **yieldCurve** **getYieldCurve** ()  
*gets the spot price*
- bool **areDivAsRate** ()  
*gets the yc*
- **Real** **getRate** ()  
*gets the div rate*
- valarray< **flowSchedule** > **getFlowSchedule** ()  
*gets the div future flow schedule*
- **Currency** **GetCurrencyFormat** ()  
*gets the currency*
- **Real** **GetVolatility** () const  
*gets the volatility*
- virtual **Real** **Price** ()  
*If we have an asset model we can inherit from this class and change the method.*
- **Real** **forwardPrice** (**Date** maturityDate)  
*get the forward price*
- **Real** **forwardPrice** (**Real** T)  
*get the forward price*
- ~**asset** (void)

## Private Attributes

- **Real** **\_currentPrice**
- **yieldCurve** **\_yc**
- bool **\_areDividendsAsGrowingRate**
- **Real** **\_dividendGrowingRate**
- valarray< **flowSchedule** > **\_announcedDividendFlows**
- **Currency** **\_denomCur**
- **Real** **\_volatility**

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 `asset::asset (void)`

default constructor

Definition at line 32 of file `asset.cpp`.

References `_announcedDividendFlows`, `_areDividendsAsGrowingRate`, `_currentPrice`, `_denomCur`, `_dividendGrowingRate`, `_volatility`, `ASSET_DEFAULT_VOL`, and `USD`.

### 6.1.1.2 `asset::asset (Real price, yieldCurve yc, bool areDividendsAsRate = true, Real divRate = 0.0, valarray< flowSchedule > flowsc = valarray< flowSchedule >(), Currency ccy = USD, Real volatility = ASSET_DEFAULT_VOL)`

constructor

#### Parameters:

*price* : current price of the asset

*yc* : mkt yc

*areDividendsAsRate* : tells u if divs are a future schedule (usually useless) or a fixed rate

*divrate*: so called fixed future div rate

*flowsc* : anticipated future div flows

*CCY* : currency denomination

*volatility* : stock's volatility (should be a volsurface but this is a simplification)

Definition at line 43 of file `asset.cpp`.

References `_announcedDividendFlows`, `_areDividendsAsGrowingRate`, `_currentPrice`, `_denomCur`, `_dividendGrowingRate`, `_volatility`, and `Real`.

### 6.1.1.3 `asset::asset (Real price, Real volatility = ASSET_DEFAULT_VOL)`

Definition at line 54 of file `asset.cpp`.

References `_announcedDividendFlows`, `_areDividendsAsGrowingRate`, `_currentPrice`, `_denomCur`, `_dividendGrowingRate`, `_volatility`, `Real`, and `USD`.

### 6.1.1.4 `asset::~~asset (void)`

Definition at line 115 of file `asset.cpp`.

## 6.1.2 Member Function Documentation

### 6.1.2.1 `bool asset::areDivAsRate () [inline]`

gets the yc

Definition at line 112 of file `asset.h`.

References `_areDividendsAsGrowingRate`.

**6.1.2.2 Real asset::forwardPrice (Real  $T$ )**

get the forward price

Definition at line 78 of file asset.cpp.

References forwardPrice(), Natural, Date::plusDays(), Real, and Date::setDateToToday().

**6.1.2.3 Real asset::forwardPrice (Date *maturityDate*)**

get the forward price

Definition at line 97 of file asset.cpp.

References \_announcedDividendFlows, \_areDividendsAsGrowingRate, \_currentPrice, \_dividendGrowingRate, Day30\_360, Date::dayCount(), yieldCurve::discountFactor(), Natural, Real, and Date::setDateToToday().

Referenced by forwardPrice(), getRho(), and mainasset().

**6.1.2.4 Currency asset::GetCurrencyFormat () [inline]**

gets the currency

Definition at line 121 of file asset.h.

References \_denomCur, and Currency.

**6.1.2.5 Real asset::getDelta () [inline]**

a stock is delta 1!

Definition at line 82 of file asset.h.

References Real.

**6.1.2.6 valarray<flowSchedule> asset::getFlowSchedule () [inline]**

gets the div future flow schedule

Definition at line 118 of file asset.h.

References \_announcedDividendFlows.

**6.1.2.7 Real asset::getPrice () const [inline]**

gets the spot price

Definition at line 106 of file asset.h.

References \_currentPrice, and Real.

Referenced by convertiblebond::delta(), convertiblebond::gamma(), operator<<(), and convertiblebond::parity().



**6.1.2.8 Real asset::getRate () [inline]**

gets the div rate

Definition at line 115 of file asset.h.

References `_dividendGrowingRate`, and `Real`.

**6.1.2.9 Real asset::getRho (Real T)**

Rho vs fwd.

Definition at line 67 of file asset.cpp.

References `forwardPrice()`, `Real`, and `yieldCurve::shiftZCBRateCurve()`.

**6.1.2.10 Real asset::GetVolatility () const [inline]**

gets the volatility

Definition at line 124 of file asset.h.

References `_volatility`, and `Real`.

Referenced by `convertiblebond::delta()`, `convertiblebond::gamma()`, and `operator<<()`.

**6.1.2.11 yieldCurve asset::getYieldCurve () [inline]**

gets the spot price

Definition at line 109 of file asset.h.

**6.1.2.12 Real asset::Price () [virtual]**

If we have an asset model we can inherit from this class and change the method.

Definition at line 90 of file asset.cpp.

References `_currentPrice`, and `Real`.

**6.1.2.13 void asset::setCcy (Currency ccy) [inline]**

sets the currency

Definition at line 100 of file asset.h.

References `_denomCur`.

**6.1.2.14 void asset::setDivAsRate (Real rate = 0.0)**

sets the div rate

Definition at line 84 of file asset.cpp.

References `_areDividendsAsGrowingRate`, `_dividendGrowingRate`, and `Real`.

**6.1.2.15 void asset::setDivFlows (flowSchedule *fc*) [inline]**

sets the flow schedule

Definition at line 97 of file asset.h.

References `_announcedDividendFlows`.

**6.1.2.16 void asset::setPrice (Real *p*) [inline]**

sets the spot price

Definition at line 88 of file asset.h.

References `_currentPrice`, and `Real`.

Referenced by `mainconvertiblebond()`.

**6.1.2.17 void asset::setVolatility (Real *volatility*) [inline]**

sets the volatility

Definition at line 103 of file asset.h.

References `_volatility`, and `Real`.

**6.1.2.18 void asset::setYieldCurve (yieldCurve *yc*) [inline]**

sets the yc

Definition at line 91 of file asset.h.

**6.1.3 Member Data Documentation****6.1.3.1 valarray<flowSchedule> asset::\_announcedDividendFlows [private]**

Definition at line 61 of file asset.h.

Referenced by `asset()`, `forwardPrice()`, `getFlowSchedule()`, and `setDivFlows()`.

**6.1.3.2 bool asset::\_areDividendsAsGrowingRate [private]**

Definition at line 59 of file asset.h.

Referenced by `areDivAsRate()`, `asset()`, `forwardPrice()`, and `setDivAsRate()`.

**6.1.3.3 Real asset::\_currentPrice [private]**

Definition at line 57 of file asset.h.

Referenced by `asset()`, `forwardPrice()`, `getPrice()`, `Price()`, and `setPrice()`.

**6.1.3.4 Currency asset::\_denomCur [private]**

Definition at line 62 of file asset.h.

Referenced by `asset()`, `GetCurrencyFormat()`, and `setCcy()`.

#### 6.1.3.5 Real asset::\_dividendGrowingRate [private]

Definition at line 60 of file `asset.h`.

Referenced by `asset()`, `forwardPrice()`, `getRate()`, and `setDivAsRate()`.

#### 6.1.3.6 Real asset::\_volatility [private]

Definition at line 63 of file `asset.h`.

Referenced by `asset()`, `GetVolatility()`, and `setVolatility()`.

#### 6.1.3.7 yieldCurve asset::\_yc [private]

Definition at line 58 of file `asset.h`.

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

- `asset.h`
- `asset.cpp`

## 6.2 binomialTree Class Reference

```
#include <binomialTree.h>
```

### Public Member Functions

- **binomialTree** (void)  
*default constructor*
- **binomialTree** (Real So, Real r, Real sigma, Real T, Natural n)  
*Constructor.*
- **binomialTree** (Real So, Real r, Real sigma, Real T, Natural n, Real u, Real d)
- **binomialTree** (const asset &theAsset, yieldCurve &yc, Real T, Natural n)
- **binomialTree** (const binomialTree &rhs)
- **binomialTree** & operator= (const binomialTree &rhs)
- virtual ~**binomialTree** (void)
- const valarray< Real > \* **getStockProcess** (Natural step)
- const valarray< Real > \* **getClaimProcess** (Natural step)
- **Real** **getPrice** ()
- void **runEngineConvertibleBond** (PayOff thePayoff, Real ConversionRatio, Real Call-Price, Real PutPrice)
- void **runEngineCall** (PayOff thePayoff)

### Protected Member Functions

- **Real** **getSo** () const
- **Real** **getRate** (Natural timestep) const
- **Real** **getSigma** () const
- **Real** **getMaturity** () const
- **Natural** **getSteps** () const
- void **constructStockProcess** ()
- void **setClaimVariables** (Real constantRate)
- void **setClaimVariables** (yieldCurve &yc)

### Private Member Functions

- void **copyObj** (const binomialTree &rhs)

### Private Attributes

- **Real** **\_So**
- **Real** **\_sigma**
- **Real** **\_maturity**
- **Natural** **\_n**
- **Real** **\_dt**
- **Real** **\_u**
- **Real** **\_d**
- valarray< valarray< Real > > **\_stockProcess**

- valarray< valarray< **Real** > > **\_claimProcess**
- valarray< **Real** > **\_discountFactor**
- valarray< **Real** > **\_q**  
*risk neutral probability of an up move*

## Friends

- ostream & **operator**<< (ostream &os, const **binomialTree** &bt)
- ostream & **operator**<< (ostream &os, const **binomialTree** \*bt)

## 6.2.1 Constructor & Destructor Documentation

### 6.2.1.1 binomialTree::binomialTree (void)

default constructor

Definition at line 3 of file binomialTree.cpp.

References **\_d**, **\_dt**, **\_n**, **\_sigma**, **\_u**, **BT\_DEFAULT\_MATURITY**, **BT\_DEFAULT\_RATE**, **BT\_DEFAULT\_SIGMA**, **BT\_DEFAULT\_SO**, **BT\_DEFAULT\_STEPS**, **constructStockProcess()**, and **setClaimVariables()**.

### 6.2.1.2 binomialTree::binomialTree (Real *So*, Real *r*, Real *sigma*, Real *T*, Natural *n*)

Constructor.

Definition at line 20 of file binomialTree.cpp.

References **\_d**, **\_dt**, **\_u**, **constructStockProcess()**, **Natural**, **r**, **Real**, and **setClaimVariables()**.

### 6.2.1.3 binomialTree::binomialTree (Real *So*, Real *r*, Real *sigma*, Real *T*, Natural *n*, Real *u*, Real *d*)

Definition at line 56 of file binomialTree.cpp.

References **\_dt**, **constructStockProcess()**, **Natural**, **r**, **Real**, and **setClaimVariables()**.

### 6.2.1.4 binomialTree::binomialTree (const asset & *theAsset*, yieldCurve & *yc*, Real *T*, Natural *n*)

Definition at line 38 of file binomialTree.cpp.

References **\_d**, **\_dt**, **\_sigma**, **\_u**, **constructStockProcess()**, **Natural**, **Real**, and **setClaimVariables()**.

### 6.2.1.5 binomialTree::binomialTree (const binomialTree & *rhs*)

Definition at line 91 of file binomialTree.cpp.

References **copyObj()**.

### 6.2.1.6 `binomialTree::~~binomialTree (void)` [virtual]

Definition at line 117 of file `binomialTree.cpp`.

## 6.2.2 Member Function Documentation

### 6.2.2.1 `void binomialTree::constructStockProcess ()` [protected]

Definition at line 122 of file `binomialTree.cpp`.

References `_claimProcess`, `_d`, `_n`, `_So`, `_stockProcess`, `_u`, and `Natural`.

Referenced by `binomialTree()`.

### 6.2.2.2 `void binomialTree::copyObj (const binomialTree & rhs)` [private]

Definition at line 102 of file `binomialTree.cpp`.

References `_claimProcess`, `_d`, `_discountFactor`, `_dt`, `_maturity`, `_n`, `_q`, `_sigma`, `_So`, `_stockProcess`, and `_u`.

Referenced by `binomialTree()`, and `operator=()`.

### 6.2.2.3 `const valarray< Real > * binomialTree::getClaimProcess (Natural step)`

Definition at line 149 of file `binomialTree.cpp`.

References `_claimProcess`, `_n`, and `Natural`.

Referenced by `convertiblebond::fairvalue()`.

### 6.2.2.4 `Real binomialTree::getMaturity () const` [inline, protected]

Definition at line 61 of file `binomialTree.h`.

References `Real`.

Referenced by `operator<<()`.

### 6.2.2.5 `Real binomialTree::getPrice ()`

Definition at line 157 of file `binomialTree.cpp`.

References `_claimProcess`, and `_n`.

Referenced by `mainbinomialtree()`.

### 6.2.2.6 `Real binomialTree::getRate (Natural timestep) const` [inline, protected]

Definition at line 57 of file `binomialTree.h`.

References `_discountFactor`, `_n`, `Natural`, and `Real`.

**6.2.2.7 Real binomialTree::getSigma () const [inline, protected]**

Definition at line 60 of file binomialTree.h.

References `_sigma`, and `Real`.

Referenced by `operator<<()`.

**6.2.2.8 Real binomialTree::getSo () const [inline, protected]**

Definition at line 56 of file binomialTree.h.

References `_So`, and `Real`.

Referenced by `operator<<()`.

**6.2.2.9 Natural binomialTree::getSteps () const [inline, protected]**

Definition at line 62 of file binomialTree.h.

References `_n`, and `Natural`.

Referenced by `operator<<()`.

**6.2.2.10 const valarray< Real > \* binomialTree::getStockProcess (Natural step)**

Definition at line 141 of file binomialTree.cpp.

References `_n`, `_stockProcess`, and `Natural`.

Referenced by `mainbinomialtree()`.

**6.2.2.11 binomialTree & binomialTree::operator= (const binomialTree & rhs)**

Definition at line 96 of file binomialTree.cpp.

References `copyObj()`.

**6.2.2.12 void binomialTree::runEngineCall (PayOff thePayoff)**

Definition at line 195 of file binomialTree.cpp.

References `_claimProcess`, `_discountFactor`, `_n`, `_q`, `_stockProcess`, `PayOff::Call()`, `Integer`, and `Natural`.

Referenced by `mainbinomialtree()`.

**6.2.2.13 void binomialTree::runEngineConvertibleBond (PayOff thePayoff, Real ConversionRatio, Real CallPrice, Real PutPrice)**

Definition at line 214 of file binomialTree.cpp.

References `_claimProcess`, `_discountFactor`, `_n`, `_q`, `_stockProcess`, `PayOff::Convertible()`, `Integer`, `Natural`, and `Real`.

Referenced by `convertiblebond::fairvalue()`.

**6.2.2.14 void binomialTree::setClaimVariables (yieldCurve & yc) [protected]**

Definition at line 84 of file binomialTree.cpp.

References `_d`, `_discountFactor`, `_dt`, `_n`, `_q`, `_u`, `yieldCurve::forwardDiscountFactor()`, and `Natural`.

**6.2.2.15 void binomialTree::setClaimVariables (Real constantRate) [protected]**

Definition at line 74 of file binomialTree.cpp.

References `_d`, `_discountFactor`, `_dt`, `_n`, `_q`, `_u`, `Natural`, `q`, and `Real`.

Referenced by `binomialTree()`.

**6.2.3 Friends And Related Function Documentation****6.2.3.1 ostream& operator<< (ostream & os, const binomialTree \* bt) [friend]**

Definition at line 23 of file binomialTree.h.

**6.2.3.2 ostream& operator<< (ostream & os, const binomialTree & bt) [friend]**

Definition at line 162 of file binomialTree.cpp.

**6.2.4 Member Data Documentation****6.2.4.1 valarray<valarray <Real> > binomialTree::\_claimProcess [private]**

Definition at line 77 of file binomialTree.h.

Referenced by `constructStockProcess()`, `copyObj()`, `getClaimProcess()`, `getPrice()`, `operator<<()`, `runEngineCall()`, and `runEngineConvertibleBond()`.

**6.2.4.2 Real binomialTree::\_d [private]**

Definition at line 73 of file binomialTree.h.

Referenced by `binomialTree()`, `constructStockProcess()`, `copyObj()`, `operator<<()`, and `setClaimVariables()`.

**6.2.4.3 valarray<Real> binomialTree::\_discountFactor [private]**

Definition at line 79 of file binomialTree.h.

Referenced by `copyObj()`, `getRate()`, `operator<<()`, `runEngineCall()`, `runEngineConvertibleBond()`, and `setClaimVariables()`.

**6.2.4.4 Real binomialTree::\_dt [private]**

Definition at line 71 of file binomialTree.h.

Referenced by `binomialTree()`, `copyObj()`, and `setClaimVariables()`.



**6.2.4.5 Real binomialTree::\_maturity [private]**

Definition at line 69 of file binomialTree.h.

Referenced by copyObj().

**6.2.4.6 Natural binomialTree::\_n [private]**

Definition at line 70 of file binomialTree.h.

Referenced by binomialTree(), constructStockProcess(), copyObj(), getClaimProcess(), getPrice(), getRate(), getSteps(), getStockProcess(), operator<<(), runEngineCall(), runEngineConvertibleBond(), and setClaimVariables().

**6.2.4.7 valarray<Real> binomialTree::\_q [private]**

risk neutral probability of an up move

Definition at line 82 of file binomialTree.h.

Referenced by copyObj(), operator<<(), runEngineCall(), runEngineConvertibleBond(), and setClaimVariables().

**6.2.4.8 Real binomialTree::\_sigma [private]**

Definition at line 68 of file binomialTree.h.

Referenced by binomialTree(), copyObj(), and getSigma().

**6.2.4.9 Real binomialTree::\_So [private]**

Definition at line 67 of file binomialTree.h.

Referenced by constructStockProcess(), copyObj(), and getSo().

**6.2.4.10 valarray<valarray <Real> > binomialTree::\_stockProcess [private]**

Definition at line 75 of file binomialTree.h.

Referenced by constructStockProcess(), copyObj(), getStockProcess(), operator<<(), runEngineCall(), and runEngineConvertibleBond().

**6.2.4.11 Real binomialTree::\_u [private]**

Definition at line 72 of file binomialTree.h.

Referenced by binomialTree(), constructStockProcess(), copyObj(), operator<<(), and setClaimVariables().

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

- **binomialTree.h**
- **binomialTree.cpp**

## 6.3 BlackScholes Class Reference

```
#include <BlackScholes.h>
```

### Public Member Functions

- **BlackScholes** (**Real** spot, **Real** volOrPrice, **bool** isVol, **Real** r, **Real** K, **Real** T, **Type-OptionBS** typeOption)  
*Default constructor.*
- **BlackScholes** ()
- **virtual** ~**BlackScholes** ()
- **Real** **getPrice** ()  
*Return price of the option.*
- **Real** **getDelta** ()  
*Return the Delta value for the option.*
- **Real** **getGamma** ()  
*Return the Gamma value for the option.*
- **Real** **getVega** ()  
*Return the Vega value for the option.*
- **Real** **getTheta** ()  
*Return the Theta value for the option.*
- **Real** **getRho** ()  
*Return the Rho value for the option.*
- **Real** **getVolatility** () **const**  
*Return the Volatility for the option.*
- **Real** **getStrike** () **const**  
*Return the strike of the option.*
- **Real** **getRate** () **const**  
*Return the risk free rate at maturity of the option.*
- **Real** **getSpot** () **const**  
*Return the spot of the option.*
- **Real** **getMaturity** () **const**  
*Return the maturity of the option.*
- **bool** **isCall** () **const**  
*Return the type of the option.*

## Protected Member Functions

- void **changeRate** (**Real** newRate)
- void **changeVol** (**Real** newVol)
- void **changeMaturity** (**Real** newMat)
- void **changeSpot** (**Real** newSpot)
- void **changeStrike** (**Real** newVol)

## Protected Attributes

- friend **OptionStrategy**  
*Allow to change rate for testing sensibility.*

## Private Member Functions

- void **recalcInformation** ()

## Private Attributes

- **Real** **\_spot**
- **Real** **\_vol**
- **Real** **\_r**
- **Real** **\_K**
- **Real** **\_T**
- **Real** **d1**
- **Real** **d2**
- **Real** **\_price**
- **TypeOptionBS** **\_type**

### 6.3.1 Constructor & Destructor Documentation

#### 6.3.1.1 BlackScholes::BlackScholes (**Real** *spot*, **Real** *volOrPrice*, **bool** *isVol*, **Real** *r*, **Real** *K*, **Real** *T*, **TypeOptionBS** *typeOption*)

Default constructor.

##### Parameters:

***spot***: Spot price of the asset

***volOrPrice***: Parameter given : either vol or price, the other will be computed

***isVol***: Bool allowing the constructor to know if the col or the price has been given

***r***: Spot Rate until maturity of option

***K***: Strike of the option

***T***: Maturity of the option

***typeOption***: Type of the option (Call or Put)

Definition at line 10 of file BlackScholes.cpp.

References **\_K**, **\_price**, **\_r**, **\_spot**, **\_T**, **\_type**, **\_vol**, **absolute()**, **BlackScholes()**, **d1**, **d2**, **getPrice()**, **getVega()**, **r**, and **Real**.

### 6.3.1.2 BlackScholes::BlackScholes ()

Definition at line 43 of file BlackScholes.cpp.

Referenced by BlackScholes().

### 6.3.1.3 BlackScholes::~~BlackScholes () [virtual]

Definition at line 46 of file BlackScholes.cpp.

## 6.3.2 Member Function Documentation

### 6.3.2.1 void BlackScholes::changeMaturity (Real *newMat*) [protected]

Definition at line 154 of file BlackScholes.cpp.

References `_T`, `Real`, and `recalcInformation()`.

Referenced by `OptionStrategy::changeMaturity()`.

### 6.3.2.2 void BlackScholes::changeRate (Real *newRate*) [protected]

Definition at line 144 of file BlackScholes.cpp.

References `_r`, `Real`, and `recalcInformation()`.

Referenced by `OptionStrategy::changeRate()`.

### 6.3.2.3 void BlackScholes::changeSpot (Real *newSpot*) [protected]

Definition at line 159 of file BlackScholes.cpp.

References `_spot`, `Real`, and `recalcInformation()`.

Referenced by `OptionStrategy::changeSpot()`.

### 6.3.2.4 void BlackScholes::changeStrike (Real *newVol*) [protected]

Definition at line 164 of file BlackScholes.cpp.

References `_K`, `Real`, and `recalcInformation()`.

Referenced by `OptionStrategy::changeStrike()`.

### 6.3.2.5 void BlackScholes::changeVol (Real *newVol*) [protected]

Definition at line 149 of file BlackScholes.cpp.

References `_vol`, `Real`, and `recalcInformation()`.

Referenced by `OptionStrategy::changeVol()`.

### 6.3.2.6 Real BlackScholes::getDelta ()

Return the Delta value for the option.

Definition at line 50 of file BlackScholes.cpp.

References `_type`, `Call`, `CumulativeNormal()`, `d1`, `Put`, and `Real`.

Referenced by `inputBSOption()`, and `mainoption()`.

#### 6.3.2.7 Real BlackScholes::getGamma ()

Return the Gamma value for the option.

Definition at line 68 of file BlackScholes.cpp.

References `_spot`, `_T`, `_vol`, `d1`, `NormalDensity()`, and `Real`.

Referenced by `inputBSOption()`, and `mainoption()`.

#### 6.3.2.8 Real BlackScholes::getMaturity () const

Return the maturity of the option.

Definition at line 136 of file BlackScholes.cpp.

References `_T`, and `Real`.

Referenced by `OptionStrategy::changeMaturity()`, and `operator<<()`.

#### 6.3.2.9 Real BlackScholes::getPrice () [inline]

Return price of the option.

Definition at line 72 of file BlackScholes.h.

References `_K`, `_price`, `_r`, `_spot`, `_T`, `_type`, `Call`, `CumulativeNormal()`, `d1`, `d2`, `Put`, and `Real`.

Referenced by `BlackScholes()`, `VarianceSwap::getPrice()`, `inputBSOption()`, `mainbinomialtree()`, `mainoption()`, `RainbowOption::PriceByClosedForm_MinOf2_call()`, `RainbowOption::PriceByClosedForm_MinOf2_put()`, and `RainbowOption::PriceByClosedForm_WorseOf2()`.

#### 6.3.2.10 Real BlackScholes::getRate () const

Return the risk free rate at maturity of the option.

Definition at line 128 of file BlackScholes.cpp.

References `_r`, and `Real`.

Referenced by `OptionStrategy::changeRate()`, and `operator<<()`.

#### 6.3.2.11 Real BlackScholes::getRho ()

Return the Rho value for the option.

Definition at line 102 of file BlackScholes.cpp.

References `_K`, `_r`, `_T`, `_type`, `Call`, `CumulativeNormal()`, `d2`, `Put`, and `Real`.

Referenced by `inputBSOption()`, and `mainoption()`.

**6.3.2.12 Real BlackScholes::getSpot () const**

Return the spot of the option.

Definition at line 132 of file BlackScholes.cpp.

References `_spot`, and `Real`.

Referenced by `OptionStrategy::changeSpot()`, and `operator<<()`.

**6.3.2.13 Real BlackScholes::getStrike () const**

Return the strike of the option.

Definition at line 124 of file BlackScholes.cpp.

References `_K`, and `Real`.

Referenced by `OptionStrategy::changeStrike()`, `VarianceSwap::getPrice()`, and `operator<<()`.

**6.3.2.14 Real BlackScholes::getTheta ()**

Return the Theta value for the option.

Definition at line 84 of file BlackScholes.cpp.

References `_K`, `_r`, `_spot`, `_T`, `_type`, `_vol`, `Call`, `CumulativeNormal()`, `d1`, `d2`, `NormalDensity()`, `Put`, and `Real`.

Referenced by `inputBSOption()`, and `mainoption()`.

**6.3.2.15 Real BlackScholes::getVega ()**

Return the Vega value for the option.

Definition at line 72 of file BlackScholes.cpp.

References `_spot`, `_T`, `d1`, `NormalDensity()`, and `Real`.

Referenced by `BlackScholes()`, `inputBSOption()`, and `mainoption()`.

**6.3.2.16 Real BlackScholes::getVolatility () const**

Return the Volatility for the option.

Definition at line 120 of file BlackScholes.cpp.

References `_vol`, and `Real`.

Referenced by `OptionStrategy::changeVol()`, `volsurface::invertBSformula()`, `mainoption()`, and `operator<<()`.

**6.3.2.17 bool BlackScholes::isCall () const**

Return the type of the option.

Definition at line 140 of file BlackScholes.cpp.

References `_type`, and `Call`.

Referenced by `VarianceSwap::getPrice()`, and `operator<<()`.

#### 6.3.2.18 `void BlackScholes::recalcInformation () [private]`

Definition at line 169 of file `BlackScholes.cpp`.

References `_K`, `_r`, `_spot`, `_T`, `_vol`, `d1`, and `d2`.

Referenced by `changeMaturity()`, `changeRate()`, `changeSpot()`, `changeStrike()`, and `changeVol()`.

### 6.3.3 Member Data Documentation

#### 6.3.3.1 `Real BlackScholes::_K [private]`

Definition at line 66 of file `BlackScholes.h`.

Referenced by `BlackScholes()`, `changeStrike()`, `getPrice()`, `getRho()`, `getStrike()`, `getTheta()`, and `recalcInformation()`.

#### 6.3.3.2 `Real BlackScholes::_price [private]`

Definition at line 68 of file `BlackScholes.h`.

Referenced by `BlackScholes()`, and `getPrice()`.

#### 6.3.3.3 `Real BlackScholes::_r [private]`

Definition at line 66 of file `BlackScholes.h`.

Referenced by `BlackScholes()`, `changeRate()`, `getPrice()`, `getRate()`, `getRho()`, `getTheta()`, and `recalcInformation()`.

#### 6.3.3.4 `Real BlackScholes::_spot [private]`

Definition at line 66 of file `BlackScholes.h`.

Referenced by `BlackScholes()`, `changeSpot()`, `getGamma()`, `getPrice()`, `getSpot()`, `getTheta()`, `getVega()`, and `recalcInformation()`.

#### 6.3.3.5 `Real BlackScholes::_T [private]`

Definition at line 66 of file `BlackScholes.h`.

Referenced by `BlackScholes()`, `changeMaturity()`, `getGamma()`, `getMaturity()`, `getPrice()`, `getRho()`, `getTheta()`, `getVega()`, and `recalcInformation()`.

#### 6.3.3.6 `TypeOptionBS BlackScholes::_type [private]`

Definition at line 69 of file `BlackScholes.h`.

Referenced by `BlackScholes()`, `getDelta()`, `getPrice()`, `getRho()`, `getTheta()`, and `isCall()`.

**6.3.3.7 Real BlackScholes::\_vol** [private]

Definition at line 66 of file BlackScholes.h.

Referenced by BlackScholes(), changeVol(), getGamma(), getTheta(), getVolatility(), and recalcInformation().

**6.3.3.8 Real BlackScholes::d1** [private]

Definition at line 67 of file BlackScholes.h.

Referenced by BlackScholes(), getDelta(), getGamma(), getPrice(), getTheta(), getVega(), and recalcInformation().

**6.3.3.9 Real BlackScholes::d2** [private]

Definition at line 67 of file BlackScholes.h.

Referenced by BlackScholes(), getPrice(), getRho(), getTheta(), and recalcInformation().

**6.3.3.10 friend BlackScholes::OptionStrategy** [protected]

Allow to change rate for testing sensibility.

Definition at line 57 of file BlackScholes.h.

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

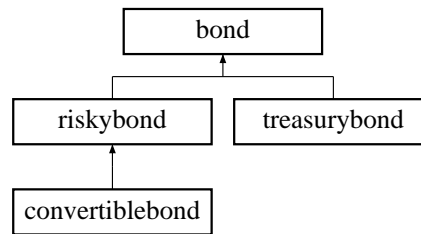
- **BlackScholes.h**
- **BlackScholes.cpp**



## 6.4 bond Class Reference

```
#include <bond.h>
```

Inheritance diagram for bond:



### Public Member Functions

- **bond** (**Date** issue, **Date** maturity, **Date** firstcoupondate, **Real** coupon, **Frequency** freq, **Real** faceamount, **DayCountConvention** daycount, **yieldCurve** yc)

*Constructor.*

- **~bond** (void)

*Destructor.*

- **cashflow** **getCashflow** ()
- virtual **Real** **quotedPrice** (**Date** today)
- virtual **Real** **fairvalue** (**Date** today)
- virtual **Real** **yieldToMaturity** (**Date** today)
- virtual **Real** **duration** (**Date** today)
- virtual **Real** **convexity** (**Date** today)
- virtual **Real** **quotedPrice** ()
- virtual **Real** **fairvalue** ()
- virtual **Real** **yieldToMaturity** ()
- virtual **Real** **duration** ()
- virtual **Real** **convexity** ()
- virtual **Date** **getMaturity** ()
- virtual **Date** **getIssue** ()
- virtual **Real** **getMaturityInYears** ()
- virtual **Real** **getMaturityInYears** (**Date** today)
- virtual **Real** **getFaceAmount** () const

### Protected Attributes

- **Date** **\_issue**
- **Date** **\_maturity**
- **Date** **\_firstcoupondate**
- **Real** **\_coupon**
- **Frequency** **\_freq**
- **Real** **\_faceamount**
- **yieldCurve** **\_yc**
- **DayCountConvention** **\_daycount**

## 6.4.1 Constructor & Destructor Documentation

### 6.4.1.1 `bond::bond` (`Date` *issue*, `Date` *maturity*, `Date` *firstcoupondate*, `Real` *coupon*, `Frequency` *freq*, `Real` *faceamount*, `DayCountConvention` *daycount*, `yieldCurve` *yc*)

Constructor.

#### Parameters:

*issue*: date of issue of the bond

*maturity*: maturity of the bond

*firstcoupondate*: date of the first coupon

*coupon*: coupon of the bond, express as a percentqge of the faceamount

*freq*: frequency of the coupon

*faceamount*: par value

*daycount*: daycount convention

*yc*: yieldcurve

Definition at line 18 of file bond.cpp.

References `Real`.

### 6.4.1.2 `bond::~~bond` (`void`) [`inline`]

Destructor.

Definition at line 55 of file bond.h.

## 6.4.2 Member Function Documentation

### 6.4.2.1 `virtual Real bond::convexity` () [`inline`, `virtual`]

Definition at line 69 of file bond.h.

References `_issue`, and `Real`.

### 6.4.2.2 `Real bond::convexity` (`Date` *today*) [`virtual`]

Definition at line 251 of file bond.cpp.

References `_daycount`, `Date::dayCount()`, `getCashflow()`, `cashflow::getCashflows()`, `cashflow::getDates()`, `Natural`, `Real`, and `yieldToMaturity()`.

Referenced by `inputBond()`, and `mainbond()`.

### 6.4.2.3 `virtual Real bond::duration` () [`inline`, `virtual`]

Definition at line 68 of file bond.h.

References `_issue`, and `Real`.

**6.4.2.4 Real bond::duration (Date *today*) [virtual]**

Definition at line 226 of file bond.cpp.

References `_daycount`, `Date::dayCount()`, `getCashflow()`, `cashflow::getCashflows()`, `cashflow::getDates()`, `Natural`, `Real`, and `yieldToMaturity()`.

Referenced by `inputBond()`, and `mainbond()`.

**6.4.2.5 virtual Real bond::fairvalue () [inline, virtual]**

Reimplemented in `convertiblebond` (p. 46).

Definition at line 66 of file bond.h.

References `_issue`, and `Real`.

Referenced by `riskybond::rho()`, `treasurybond::rho()`, and `yieldToMaturity()`.

**6.4.2.6 Real bond::fairvalue (Date *today*) [virtual]**

Reimplemented in `convertiblebond` (p. 46).

Definition at line 116 of file bond.cpp.

References `_coupon`, `_daycount`, `_faceamount`, `_firstcoupondate`, `_freq`, `_issue`, `ACT_360`, `ACT_365`, `Annual`, `Bimonthly`, `Day30_360`, `Day30_365`, `Date::dayOfMonth()`, `EveryFourthMonth`, `getCashflow()`, `cashflow::getCashflows()`, `cashflow::getDates()`, `Date::month()`, `Monthly`, `Natural`, `NoFrequency`, `Once`, `Quarterly`, `quotedPrice()`, `Real`, `Semiannual`, `Date::serialNumber()`, `TN_REAL`, and `Date::year()`.

Referenced by `inputBond()`, `mainbond()`, `riskybond::rho()`, and `treasurybond::rho()`.

**6.4.2.7 cashflow bond::getCashflow ()**

Definition at line 31 of file bond.cpp.

References `_coupon`, `_daycount`, `_faceamount`, `_firstcoupondate`, `_freq`, `Annual`, `Date::applyConvention()`, `Bimonthly`, `Date::dayCount()`, `EveryFourthMonth`, `Following`, `Integer`, `Monthly`, `Natural`, `NoFrequency`, `Once`, `Date::plusMonths()`, `Quarterly`, `Real`, and `Semiannual`.

Referenced by `convexity()`, `duration()`, `fairvalue()`, `riskybond::quotedPrice()`, `quotedPrice()`, and `yieldToMaturity()`.

**6.4.2.8 virtual Real bond::getFaceAmount () const [inline, virtual]**

Definition at line 75 of file bond.h.

References `_faceamount`, and `Real`.

Referenced by `convertiblebond::adjustedConversionRatio()`, and `convertiblebond::fairvalue()`.

**6.4.2.9 virtual Date bond::getIssue () [inline, virtual]**

Definition at line 72 of file bond.h.

References `_issue`.

**6.4.2.10 virtual Date bond::getMaturity ()** [inline, virtual]

Definition at line 71 of file bond.h.

**6.4.2.11 virtual Real bond::getMaturityInYears (Date *today*)** [inline, virtual]

Definition at line 74 of file bond.h.

References Date::dayCount(), and Real.

**6.4.2.12 virtual Real bond::getMaturityInYears ()** [inline, virtual]

Definition at line 73 of file bond.h.

References \_issue, and Real.

Referenced by convertiblebond::fairvalue().

**6.4.2.13 virtual Real bond::quotedPrice ()** [inline, virtual]

Definition at line 65 of file bond.h.

References \_issue, and Real.

Referenced by fairvalue().

**6.4.2.14 Real bond::quotedPrice (Date *today*)** [virtual]

Reimplemented in **riskybond** (p.193).

Definition at line 200 of file bond.cpp.

References \_daycount, Date::dayCount(), yieldCurve::discountFactor(), getCashflow(), cashflow::getCashflows(), cashflow::getDates(), Natural, and Real.

**6.4.2.15 virtual Real bond::yieldToMaturity ()** [inline, virtual]

Definition at line 67 of file bond.h.

References \_issue, and Real.

Referenced by convexity(), and duration().

**6.4.2.16 Real bond::yieldToMaturity (Date *today*)** [virtual]

Definition at line 276 of file bond.cpp.

References \_daycount, Date::dayCount(), fairvalue(), getCashflow(), cashflow::getCashflows(), cashflow::getDates(), Natural, and Real.

Referenced by inputBond(), and mainbond().

### 6.4.3 Member Data Documentation

#### 6.4.3.1 Real bond::\_coupon [protected]

Definition at line 35 of file bond.h.

Referenced by fairvalue(), and getCashflow().

#### 6.4.3.2 DayCountConvention bond::\_daycount [protected]

Definition at line 40 of file bond.h.

Referenced by convexity(), duration(), fairvalue(), getCashflow(), quotedPrice(), and yieldToMaturity().

#### 6.4.3.3 Real bond::\_faceamount [protected]

Definition at line 37 of file bond.h.

Referenced by fairvalue(), getCashflow(), getFaceAmount(), and operator<<().

#### 6.4.3.4 Date bond::\_firstcoupondate [protected]

Definition at line 34 of file bond.h.

Referenced by fairvalue(), and getCashflow().

#### 6.4.3.5 Frequency bond::\_freq [protected]

Definition at line 36 of file bond.h.

Referenced by fairvalue(), and getCashflow().

#### 6.4.3.6 Date bond::\_issue [protected]

Definition at line 32 of file bond.h.

Referenced by convexity(), duration(), fairvalue(), getIssue(), getMaturityInYears(), operator<<(), quotedPrice(), and yieldToMaturity().

#### 6.4.3.7 Date bond::\_maturity [protected]

Definition at line 33 of file bond.h.

Referenced by operator<<().

#### 6.4.3.8 yieldCurve bond::\_yc [protected]

Definition at line 38 of file bond.h.

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

- bond.h

- `bond.cpp`

## 6.5 cachedval Struct Reference

```
#include <types.h>
```

### Public Attributes

- `bool isCached`
- `Real value`

### 6.5.1 Member Data Documentation

#### 6.5.1.1 `bool cachedval::isCached`

Definition at line 28 of file `types.h`.

#### 6.5.1.2 `Real cachedval::value`

Definition at line 29 of file `types.h`.

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

- `types.h`

## 6.6 cashflow Class Reference

```
#include <bond.h>
```

### Public Member Functions

- **cashflow** (valarray< **Date** > dates, valarray< **Real** > cashflows)
- **~cashflow** ()
- valarray< **Date** > **getDates** ()
- valarray< **Real** > **getCashflows** ()

### Private Attributes

- valarray< **Date** > **\_dates**
- valarray< **Real** > **\_cashflows**

### 6.6.1 Constructor & Destructor Documentation

#### 6.6.1.1 cashflow::cashflow (valarray< **Date** > *dates*, valarray< **Real** > *cashflows*)

Definition at line 3 of file bond.cpp.

#### 6.6.1.2 cashflow::~~cashflow () [inline]

Definition at line 21 of file bond.h.

### 6.6.2 Member Function Documentation

#### 6.6.2.1 valarray< **Real** > cashflow::getCashflows ()

Definition at line 13 of file bond.cpp.

References `_cashflows`.

Referenced by `bond::convexity()`, `bond::duration()`, `bond::fairvalue()`, `riskybond::quotedPrice()`, `bond::quotedPrice()`, and `bond::yieldToMaturity()`.

#### 6.6.2.2 valarray< **Date** > cashflow::getDates ()

Definition at line 9 of file bond.cpp.

References `_dates`.

Referenced by `bond::convexity()`, `bond::duration()`, `bond::fairvalue()`, `riskybond::quotedPrice()`, `bond::quotedPrice()`, and `bond::yieldToMaturity()`.

### 6.6.3 Member Data Documentation

#### 6.6.3.1 valarray<**Real**> cashflow::\_cashflows [private]

Definition at line 17 of file bond.h.



Referenced by `getCashflows()`.

#### 6.6.3.2 `valarray<Date> cashflow::_dates` [private]

Definition at line 16 of file `bond.h`.

Referenced by `getDates()`.

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

- `bond.h`
- `bond.cpp`

## 6.7 CashFlow Class Reference

```
#include <CashFlow.h>
```

### Public Member Functions

- **CashFlow** (**SwapLeg** swapLeg, **Real** fixedRate)  
*Constructor for fixed leg.*
- **CashFlow** (**SwapLeg** swapLeg, **yieldCurve** floatCurve)  
*Constructor for float leg by giving a yield curve object.*
- **Real** getFairValue (**yieldCurve** \*curve)  
*Get the fair value of the swap : discounted value of cash flows.*
- **~CashFlow** (void)

### Private Attributes

- valarray< **Real** > **flowAmount**
- valarray< **Date** > **flowDates**

### 6.7.1 Constructor & Destructor Documentation

#### 6.7.1.1 CashFlow::CashFlow (SwapLeg swapLeg, Real fixedRate)

Constructor for fixed leg.

Definition at line 3 of file CashFlow.cpp.

References flowAmount, flowDates, Natural, Real, SwapLeg::returnAmounts(), SwapLeg::returnDates(), Date::serialNumber(), and Date::setDateToToday().

#### 6.7.1.2 CashFlow::CashFlow (SwapLeg swapLeg, yieldCurve floatCurve)

Constructor for float leg by giving a yield curve object.

Definition at line 15 of file CashFlow.cpp.

References Discrete, flowAmount, flowDates, yieldCurve::forwardRate(), Natural, SwapLeg::returnAmounts(), SwapLeg::returnDates(), Date::serialNumber(), and Date::setDateToToday().

#### 6.7.1.3 CashFlow::~CashFlow (void)

Definition at line 36 of file CashFlow.cpp.

## 6.7.2 Member Function Documentation

### 6.7.2.1 Real CashFlow::getFairValue (yieldCurve \* *curve*)

Get the fair value of the swap : discounted value of cash flows.

Definition at line 27 of file CashFlow.cpp.

References yieldCurve::discountFactor(), Discrete, flowAmount, flowDates, Natural, and Real.

Referenced by VanillaSwap::getFairValue1(), VanillaSwap::getFairValue2(), and mainIRVanillaSwap().

## 6.7.3 Member Data Documentation

### 6.7.3.1 valarray<Real> CashFlow::flowAmount [private]

Definition at line 25 of file CashFlow.h.

Referenced by CashFlow(), and getFairValue().

### 6.7.3.2 valarray<Date> CashFlow::flowDates [private]

Definition at line 26 of file CashFlow.h.

Referenced by CashFlow(), and getFairValue().

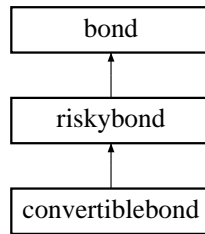
The documentation for this class was generated from the following files:

- CashFlow.h
- CashFlow.cpp

## 6.8 convertiblebond Class Reference

```
#include <convertiblebond.h>
```

Inheritance diagram for convertiblebond::



### Public Member Functions

- **convertiblebond** (**asset** Stock, **riskybond** Bond, **Real** conversionRatio=CB\_DEFAULT\_RATIO, **Natural** nSteps=CB\_DEFAULT\_STEPS, **Real** callPrice=CB\_DEFAULT\_CALLPRICE, **Real** putPrice=CB\_DEFAULT\_PUTPRICE)

*Constructor.*

- virtual **~convertiblebond** (void)
- virtual **Real fairvalue** (**Date** today)
- virtual **Real fairvalue** ()
- **Real adjustedConversionRatio** () const
- **Real parity** (void) const
- **Real delta** (void) const
- **Real delta** (**Date** today) const
- **Real parityDelta** (void) const
- **Real parityDelta** (**Date** today) const
- **convertiblebond shiftedcbond** (**Real** shift)
- **Real rho** (**Date** today)
  - return the derivative of the bond price with respect to interest rates*
- **Real rho** ()
- **Real interestRateDelta** (void) const
- **Real interestRateDelta** (**Date** today) const
- **Real gamma** (void) const
- **Real gamma** (**Date** today) const
- **Real parityGamma** (void) const
- **Real parityGamma** (**Date** today) const
- **Natural getSteps** (void) const

### Protected Attributes

- **binomialTree \* \_bt**
- **bool \_btCached**

## Private Member Functions

- void `copyObj` (const `convertiblebond` &rhs)

## Private Attributes

- `asset` `_stock`
- `riskybond` `_bond`
- `Natural` `_n`
- `Real` `_callPrice`
- `Real` `_putPrice`
- `Real` `_conversionRatio`
- `Real` `_price`
- `bool` `_priceCached`
- `Date` `_datepriceCached`
- `Real` `_delta`
- `bool` `_deltaCached`
- `Date` `_datedeltaCached`
- `Real` `_interestRateDelta`
- `bool` `_interestRateDeltaCached`
- `Date` `_dateinterestRateDeltaCached`
- `Real` `_gamma`
- `bool` `_gammaCached`
- `Date` `_dategammaCached`

## Friends

- `ostream` & `operator<<` (`ostream` &os, `convertiblebond` &cb)
- `ostream` & `operator<<` (`ostream` &os, `convertiblebond` \*cb)

### 6.8.1 Constructor & Destructor Documentation

**6.8.1.1** `convertiblebond::convertiblebond` (`asset` *Stock*, `riskybond` *Bond*, `Real` *conversionRatio* = `CB_DEFAULT_RATIO`, `Natural` *nSteps* = `CB_DEFAULT_STEPS`, `Real` *callPrice* = `CB_DEFAULT_CALLPRICE`, `Real` *putPrice* = `CB_DEFAULT_PUTPRICE`)

Constructor.

Definition at line 5 of file `convertiblebond.cpp`.

References `_btCached`, `_deltaCached`, `_gammaCached`, `_interestRateDeltaCached`, `_priceCached`, `Natural`, and `Real`.

Referenced by `delta()`, `gamma()`, and `shiftedcbond()`.

**6.8.1.2** `convertiblebond::~~convertiblebond` (`void`) [virtual]

Definition at line 29 of file `convertiblebond.cpp`.

## 6.8.2 Member Function Documentation

### 6.8.2.1 Real convertiblebond::adjustedConversionRatio () const [inline]

Definition at line 52 of file convertiblebond.h.

References `_conversionRatio`, `bond::getFaceAmount()`, and `Real`.

Referenced by `parityDelta()`, and `parityGamma()`.

### 6.8.2.2 void convertiblebond::copyObj (const convertiblebond & rhs) [private]

### 6.8.2.3 Real convertiblebond::delta (Date today) const

Definition at line 65 of file convertiblebond.cpp.

References `_bond`, `_callPrice`, `_conversionRatio`, `_datedeltaCached`, `_delta`, `_deltaCached`, `_putPrice`, `_stock`, `convertiblebond()`, `fairvalue()`, `asset::getPrice()`, `asset::GetVolatility()`, and `Real`.

### 6.8.2.4 Real convertiblebond::delta (void) const [inline]

Definition at line 56 of file convertiblebond.h.

References `Real`.

Referenced by `gamma()`, `operator<<()`, and `parityDelta()`.

### 6.8.2.5 virtual Real convertiblebond::fairvalue () [inline, virtual]

Reimplemented from `bond` (p. 35).

Definition at line 50 of file convertiblebond.h.

References `Real`.

### 6.8.2.6 Real convertiblebond::fairvalue (Date today) [virtual]

Reimplemented from `bond` (p. 35).

Definition at line 33 of file convertiblebond.cpp.

References `_bt`, `_btCached`, `_callPrice`, `_conversionRatio`, `_datepriceCached`, `_priceCached`, `_putPrice`, `_stock`, `binomialTree::getClaimProcess()`, `bond::getFaceAmount()`, `bond::getMaturityInYears()`, and `binomialTree::runEngineConvertibleBond()`.

Referenced by `delta()`, `interestRateDelta()`, `mainconvertiblebond()`, and `operator<<()`.

### 6.8.2.7 Real convertiblebond::gamma (Date today) const

Definition at line 123 of file convertiblebond.cpp.

References `_bond`, `_callPrice`, `_conversionRatio`, `_dategammaCached`, `_gamma`, `_gammaCached`, `_putPrice`, `_stock`, `convertiblebond()`, `delta()`, `asset::getPrice()`, `asset::GetVolatility()`, and `Real`.

**6.8.2.8 Real convertiblebond::gamma (void) const [inline]**

Definition at line 65 of file convertiblebond.h.

References Real.

Referenced by operator<<(), and parityGamma().

**6.8.2.9 Natural convertiblebond::getSteps (void) const [inline]**

Definition at line 72 of file convertiblebond.h.

References Natural.

**6.8.2.10 Real convertiblebond::interestRateDelta (Date *today*) const**

Definition at line 107 of file convertiblebond.cpp.

References `_dateinterestRateDeltaCached`, `_interestRateDelta`, `_interestRateDeltaCached`, and `fairvalue()`.

**6.8.2.11 Real convertiblebond::interestRateDelta (void) const [inline]**

Definition at line 63 of file convertiblebond.h.

References Real.

Referenced by operator<<(), and rho().

**6.8.2.12 Real convertiblebond::parity (void) const [inline]**

Definition at line 55 of file convertiblebond.h.

References `_conversionRatio`, `_stock`, `asset::getPrice()`, and Real.

**6.8.2.13 Real convertiblebond::parityDelta (Date *today*) const [inline]**

Definition at line 59 of file convertiblebond.h.

References `adjustedConversionRatio()`, `delta()`, and Real.

**6.8.2.14 Real convertiblebond::parityDelta (void) const [inline]**

Definition at line 58 of file convertiblebond.h.

References Real.

Referenced by `mainconvertiblebond()`, and operator<<().

**6.8.2.15 Real convertiblebond::parityGamma (Date *today*) const [inline]**

Definition at line 68 of file convertiblebond.h.

References `adjustedConversionRatio()`, `gamma()`, and Real.

**6.8.2.16 Real convertiblebond::parityGamma (void) const [inline]**

Definition at line 67 of file convertiblebond.h.

References Real.

Referenced by mainconvertiblebond(), and operator<<().

**6.8.2.17 Real convertiblebond::rho () [inline, virtual]**

Reimplemented from **riskybond** (p.193).

Definition at line 62 of file convertiblebond.h.

References Real.

**6.8.2.18 Real convertiblebond::rho (Date *today*) [inline, virtual]**

return the derivative of the bond price with respect to interest rates

Reimplemented from **riskybond** (p.194).

Definition at line 61 of file convertiblebond.h.

References interestRateDelta(), and Real.

Referenced by mainconvertiblebond().

**6.8.2.19 convertiblebond convertiblebond::shiftedcbond (Real *shift*)**

Definition at line 99 of file convertiblebond.cpp.

References `_bond`, `_callPrice`, `_conversionRatio`, `_putPrice`, `_stock`, `convertiblebond()`, `Real`, `riskybond::shiftedbond()`, and `shiftedcbond()`.

Referenced by `shiftedcbond()`.

**6.8.3 Friends And Related Function Documentation****6.8.3.1 ostream& operator<< (ostream & *os*, convertiblebond \* *cb*) [friend]**

Definition at line 35 of file convertiblebond.h.

**6.8.3.2 ostream& operator<< (ostream & *os*, convertiblebond & *cb*) [friend]**

Definition at line 156 of file convertiblebond.cpp.

**6.8.4 Member Data Documentation****6.8.4.1 riskybond convertiblebond::\_bond [private]**

Definition at line 80 of file convertiblebond.h.

Referenced by `delta()`, `gamma()`, and `shiftedcbond()`.



**6.8.4.2 binomialTree\* convertiblebond::\_bt** [mutable, protected]

Definition at line 75 of file convertiblebond.h.

Referenced by fairvalue(), and operator<<().

**6.8.4.3 bool convertiblebond::\_btCached** [mutable, protected]

Definition at line 76 of file convertiblebond.h.

Referenced by convertiblebond(), fairvalue(), and operator<<().

**6.8.4.4 Real convertiblebond::\_callPrice** [private]

Definition at line 83 of file convertiblebond.h.

Referenced by delta(), fairvalue(), gamma(), operator<<(), and shiftedcbond().

**6.8.4.5 Real convertiblebond::\_conversionRatio** [private]

Definition at line 85 of file convertiblebond.h.

Referenced by adjustedConversionRatio(), delta(), fairvalue(), gamma(), operator<<(), parity(), and shiftedcbond().

**6.8.4.6 Date convertiblebond::\_datedeltaCached** [mutable, private]

Definition at line 94 of file convertiblebond.h.

Referenced by delta().

**6.8.4.7 Date convertiblebond::\_dategammaCached** [mutable, private]

Definition at line 102 of file convertiblebond.h.

Referenced by gamma().

**6.8.4.8 Date convertiblebond::\_dateinterestRateDeltaCached** [mutable, private]

Definition at line 98 of file convertiblebond.h.

Referenced by interestRateDelta().

**6.8.4.9 Date convertiblebond::\_datepriceCached** [mutable, private]

Definition at line 90 of file convertiblebond.h.

Referenced by fairvalue().

**6.8.4.10 Real convertiblebond::\_delta** [mutable, private]

Definition at line 92 of file convertiblebond.h.

Referenced by delta().

**6.8.4.11** `bool convertiblebond::_deltaCached` [mutable, private]

Definition at line 93 of file convertiblebond.h.

Referenced by convertiblebond(), and delta().

**6.8.4.12** `Real convertiblebond::_gamma` [mutable, private]

Definition at line 100 of file convertiblebond.h.

Referenced by gamma().

**6.8.4.13** `bool convertiblebond::_gammaCached` [mutable, private]

Definition at line 101 of file convertiblebond.h.

Referenced by convertiblebond(), and gamma().

**6.8.4.14** `Real convertiblebond::_interestRateDelta` [mutable, private]

Definition at line 96 of file convertiblebond.h.

Referenced by interestRateDelta().

**6.8.4.15** `bool convertiblebond::_interestRateDeltaCached` [mutable, private]

Definition at line 97 of file convertiblebond.h.

Referenced by convertiblebond(), and interestRateDelta().

**6.8.4.16** `Natural convertiblebond::_n` [private]

Definition at line 82 of file convertiblebond.h.

Referenced by operator<<().

**6.8.4.17** `Real convertiblebond::_price` [mutable, private]

Definition at line 88 of file convertiblebond.h.

**6.8.4.18** `bool convertiblebond::_priceCached` [mutable, private]

Definition at line 89 of file convertiblebond.h.

Referenced by convertiblebond(), and fairvalue().

**6.8.4.19** `Real convertiblebond::_putPrice` [private]

Definition at line 84 of file convertiblebond.h.

Referenced by delta(), fairvalue(), gamma(), operator<<(), and shiftedcbond().

**6.8.4.20 asset convertiblebond::\_stock [private]**

Definition at line 79 of file convertiblebond.h.

Referenced by delta(), fairvalue(), gamma(), operator<<(), parity(), and shiftedcbond().

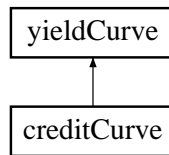
The documentation for this class was generated from the following files:

- **convertiblebond.h**
- **convertiblebond.cpp**

## 6.9 creditCurve Class Reference

```
#include <creditCurve.h>
```

Inheritance diagram for creditCurve::



### Public Member Functions

- **creditCurve** (void)  
*default constructor*
- **creditCurve** (valarray< **yieldPoint** > &yp, valarray< **CreditSpreadPoint** > &cp, char \*name=CC\_DEFAULT\_NAME, **Real** recoveryRate=CC\_DEFAULT\_RECOVERY\_RATE, **Currency** currency=CC\_DEFAULT\_CURRENCY, **Frequency** frequency=CC\_DEFAULT\_FREQUENCY)  
*Constructor.*
- **creditCurve** (**yieldCurve** &yc, valarray< **CreditSpreadPoint** > &cp, char \*name=CC\_DEFAULT\_NAME, **Real** recoveryRate=CC\_DEFAULT\_RECOVERY\_RATE, **Currency** currency=CC\_DEFAULT\_CURRENCY, **Frequency** frequency=CC\_DEFAULT\_FREQUENCY)
- **creditCurve** (**Real** flatRate, **Real** flatSpread, char \*name=CC\_DEFAULT\_NAME, **Real** recoveryRate=CC\_DEFAULT\_RECOVERY\_RATE, **Currency** currency=CC\_DEFAULT\_CURRENCY, **Frequency** frequency=CC\_DEFAULT\_FREQUENCY)
- **creditCurve** (**yieldCurve** &yc, **Real** flatSpread, char \*name=CC\_DEFAULT\_NAME, **Real** recoveryRate=CC\_DEFAULT\_RECOVERY\_RATE, **Currency** currency=CC\_DEFAULT\_CURRENCY, **Frequency** frequency=CC\_DEFAULT\_FREQUENCY)
- **creditCurve** (const **creditCurve** &rhs)
- **creditCurve** & **operator=** (const **creditCurve** &rhs)
- virtual ~**creditCurve** (void)
- **yieldCurve** \* **createSpreadCurve** (**yieldCurve** &underlying, valarray< **CreditSpreadPoint** > &spreads)
- **yieldCurve** \* **combineUnderlyingAndSpreads** (**yieldCurve** &underlying, **yieldCurve** &spreadcurve)
- void **assignFlatSpread** (**Real** r)
- void **resampleSpread** ()
- virtual **Real** **creditSpread** (**Real** maturity) const
- virtual **Real** **creditSpread** (**Date** maturityDate) const
- virtual **Real** **timeOfCurrentSpread** (**Real** maturity) const
- virtual **Natural** **indexOfCurrentSpread** (**Real** maturity) const
- virtual **Real** **timeOfPreviousSpread** (**Real** maturity) const
- virtual **Natural** **indexOfPreviousSpread** (**Real** maturity) const
- virtual **Real** **survivalProbability** (**Real** maturity) const
- virtual **Real** **cumulativeDefaultProbability** (**Real** maturity) const
- virtual **Real** **swapFees** (**Real** maturity) const

- virtual **Real defaultProbability** (**Real** maturity) const  
*returns conditional default probability at a given maturity.*
- virtual **Real hazardRate** (**Real** maturity) const  
*returns hazard rate at a given maturity - this is an alias for defaultProbability i.e.*
- virtual **Real spotRate** (**Real** maturity) const  
*Calculates the spot ZCB rate.*
- virtual **Real spotRate** (**Date** maturityDate) const  
*Calculates the spot ZCB rate.*
- virtual **Real discountFactor** (**Real** maturity, **interestComposition** composition=Continuous)  
*Calculates the discountFactor.*
- virtual **Real riskyDiscountFactor** (**Real** maturity, **interestComposition** composition=Continuous)  
*Calculates the risky discount factor incorporating the hazard rate.*
- virtual **Real discountFactor** (**Date** maturityDate, **interestComposition** composition=Continuous)  
*Calculates the discountFactor.*
- virtual **Real forwardRate** (**Real** forwardStart, **Real** effectiveLengthOfTheContractAfterStart, **interestComposition** composition=Continuous)  
*Calculates the fwd rate.*
- virtual **Real forwardRate** (**Date** forwardStart, **Date** forwardEnd, **interestComposition** composition=Continuous)  
*Calculates the fwd rate.*
- virtual valarray< **Real** > **getMaturitiesInTheZCBCurve** () const  
*Return the maturities present in the market curve, both from the Cash and Swap Pointsvalarray<Real>.*
- virtual char \* **getName** ()
- **Real** **getRecoveryRate** (void) const
- **Currency** **getCurrency** (void) const
- **Frequency** **getFrequency** (void) const

## Protected Member Functions

- **yieldCurve** \* **getUnderlying** (void) const
- **yieldCurve** \* **getCombined** (void) const
- valarray< **CreditSpreadPoint** > **getSpreads** (void) const
- valarray< **cachedval** > **getSurvivalProbability** (void) const
- valarray< **cachedval** > **getDefaultProbability** (void) const
- valarray< **cachedval** > **getSwapFees** (void) const

## Protected Attributes

- `yieldCurve * _underlying`
- `yieldCurve * _combined`

## Private Member Functions

- `void copyObj (const creditCurve &rhs)`

## Private Attributes

- `valarray< CreditSpreadPoint > _spreads`
- `valarray< cachedval > _survivalProbability`
- `valarray< cachedval > _defaultProbability`
- `valarray< cachedval > _swapFees`
- `Real _recoveryRate`
- `Currency _currency`
- `Frequency _frequency`

## Friends

- `ostream & operator<< (ostream &os, const creditCurve &c)`
- `ostream & operator<< (ostream &os, const creditCurve *c)`

## 6.9.1 Constructor & Destructor Documentation

### 6.9.1.1 creditCurve::creditCurve (void)

default constructor

Definition at line 24 of file `creditCurve.cpp`.

References `CC_MAX_NUM_SPREADS`.

### 6.9.1.2 creditCurve::creditCurve (valarray< yieldPoint > & *yp*, valarray< CreditSpreadPoint > & *cp*, char \* *name* = CC\_DEFAULT\_NAME, Real *recoveryRate* = CC\_DEFAULT\_RECOVERY\_RATE, Currency *currency* = CC\_DEFAULT\_CURRENCY, Frequency *frequency* = CC\_DEFAULT\_FREQUENCY)

Constructor.

#### Parameters:

- yp* - as with yield curve, array of yield points
- cp* - array of credit spreads for different maturities
- recoveryRate* - amount of debt collected in case of default
- currency* - that the debt is denominated in
- frequency* - annual, semi-annual, etc.
- name* - a string identifying the curve

Definition at line 100 of file creditCurve.cpp.

References `_combined`, `_underlying`, `CC_MAX_NUM_SPREADS`, `combineUnderlyingAndSpreads()`, `createSpreadCurve()`, `Real`, `resampleSpread()`, and `yieldCurve::yieldCurve()`.

**6.9.1.3** `creditCurve::creditCurve (yieldCurve & yc, valarray< CreditSpreadPoint > & cp, char * name = CC_DEFAULT_NAME, Real recoveryRate = CC_DEFAULT_RECOVERY_RATE, Currency currency = CC_DEFAULT_CURRENCY, Frequency frequency = CC_DEFAULT_FREQUENCY)`

Definition at line 32 of file creditCurve.cpp.

References `_combined`, `_underlying`, `CC_MAX_NUM_SPREADS`, `combineUnderlyingAndSpreads()`, `createSpreadCurve()`, `Real`, `resampleSpread()`, and `yieldCurve::yieldCurve()`.

**6.9.1.4** `creditCurve::creditCurve (Real flatRate, Real flatSpread, char * name = CC_DEFAULT_NAME, Real recoveryRate = CC_DEFAULT_RECOVERY_RATE, Currency currency = CC_DEFAULT_CURRENCY, Frequency frequency = CC_DEFAULT_FREQUENCY)`

Definition at line 55 of file creditCurve.cpp.

References `_combined`, `_underlying`, `assignFlatSpread()`, `CC_MAX_NUM_SPREADS`, `combineUnderlyingAndSpreads()`, `Real`, and `yieldCurve::yieldCurve()`.

**6.9.1.5** `creditCurve::creditCurve (yieldCurve & yc, Real flatSpread, char * name = CC_DEFAULT_NAME, Real recoveryRate = CC_DEFAULT_RECOVERY_RATE, Currency currency = CC_DEFAULT_CURRENCY, Frequency frequency = CC_DEFAULT_FREQUENCY)`

Definition at line 77 of file creditCurve.cpp.

References `_combined`, `_underlying`, `assignFlatSpread()`, `CC_MAX_NUM_SPREADS`, `combineUnderlyingAndSpreads()`, `Real`, and `yieldCurve::yieldCurve()`.

**6.9.1.6** `creditCurve::creditCurve (const creditCurve & rhs)`

Definition at line 129 of file creditCurve.cpp.

References `copyObj()`.

**6.9.1.7** `creditCurve::~creditCurve (void) [virtual]`

Definition at line 243 of file creditCurve.cpp.

References `_combined`, and `_underlying`.

## 6.9.2 Member Function Documentation

### 6.9.2.1 void creditCurve::assignFlatSpread (Real *r*)

Definition at line 214 of file creditCurve.cpp.

References `_defaultProbability`, `_spreads`, `_survivalProbability`, `_swapFees`, `Natural`, `r`, `Real`, and `Relative`.

Referenced by `creditCurve()`.

### 6.9.2.2 yieldCurve \* creditCurve::combineUnderlyingAndSpreads (yieldCurve & *underlying*, yieldCurve & *spreadcurve*)

Definition at line 181 of file creditCurve.cpp.

References `Cash`, `yieldCurve::getMaturitiesInTheZCBCurve()`, `mergeunique()`, `Natural`, `yieldCurve::spotRate()`, and `yieldCurve::yieldCurve()`.

Referenced by `creditCurve()`.

### 6.9.2.3 void creditCurve::copyObj (const creditCurve & *rhs*) [private]

Definition at line 140 of file creditCurve.cpp.

References `_combined`, `_currency`, `_defaultProbability`, `_frequency`, `_recoveryRate`, `_spreads`, `_survivalProbability`, `_swapFees`, `_underlying`, `getCombined()`, `getCurrency()`, `getDefaultProbability()`, `getFrequency()`, `getRecoveryRate()`, `getSpreads()`, `getSurvivalProbability()`, `getSwapFees()`, `getUnderlying()`, and `yieldCurve::yieldCurve()`.

Referenced by `creditCurve()`, and `operator=()`.

### 6.9.2.4 yieldCurve \* creditCurve::createSpreadCurve (yieldCurve & *underlying*, valarray< CreditSpreadPoint > & *spreads*)

Definition at line 153 of file creditCurve.cpp.

References `Absolute`, `Cash`, `Natural`, `yieldCurve::spotRate()`, and `yieldCurve::yieldCurve()`.

Referenced by `creditCurve()`.

### 6.9.2.5 virtual Real creditCurve::creditSpread (Date *maturityDate*) const [inline, virtual]

Definition at line 144 of file creditCurve.h.

References `_combined`, `_underlying`, `Real`, and `yieldCurve::spotRate()`.

### 6.9.2.6 virtual Real creditCurve::creditSpread (Real *maturity*) const [inline, virtual]

Definition at line 139 of file creditCurve.h.

References `_combined`, `_underlying`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `defaultProbability()`, `maincreditcurve()`, and `swapFees()`.



### 6.9.2.7 Real creditCurve::cumulativeDefaultProbability (Real *maturity*) const [virtual]

Definition at line 290 of file creditCurve.cpp.

References Real, and survivalProbability().

Referenced by maincreditcurve().

### 6.9.2.8 Real creditCurve::defaultProbability (Real *maturity*) const [virtual]

returns conditional default probability at a given maturity.

#### Parameters:

*maturity* - time at which to evaluate probability

#### Returns:

conditional default probability - the same probability will be returned for all times between two spreads

Definition at line 357 of file creditCurve.cpp.

References \_defaultProbability, \_recoveryRate, \_underlying, creditSpread(), yieldCurve::discountFactor(), indexOfCurrentSpread(), Natural, Real, survivalProbability(), swapFees(), timeOfCurrentSpread(), and timeOfPreviousSpread().

Referenced by hazardRate(), maincreditcurve(), survivalProbability(), and swapFees().

### 6.9.2.9 virtual Real creditCurve::discountFactor (Date *maturityDate*, interestComposition *composition* = Continuous) [inline, virtual]

Calculates the discountFactor.

#### Parameters:

*maturity* : just after ZCBrates are computed, it it very easy [done at trhe constructor level]

Reimplemented from yieldCurve (p. 224).

Definition at line 206 of file creditCurve.h.

References \_combined, and yieldCurve::discountFactor().

### 6.9.2.10 virtual Real creditCurve::discountFactor (Real *maturity*, interestComposition *composition* = Continuous) [inline, virtual]

Calculates the discountFactor.

#### Parameters:

*maturity* : just after ZCBrates are computed, it it very easy [done at trhe constructor level]

Reimplemented from yieldCurve (p. 224).

Definition at line 189 of file creditCurve.h.

References \_combined, yieldCurve::discountFactor(), and Real.

**6.9.2.11** virtual Real creditCurve::forwardRate (Date *forwardStart*, Date *forwardEnd*, interestComposition *composition* = Continuous) [inline, virtual]

Calculates the fwd rate.

**Parameters:**

*forwardStart* start of the rate

*maturityAfterForward* maturity after the start

Reimplemented from **yieldCurve** (p. 224).

Definition at line 229 of file creditCurve.h.

References `_combined`, and `yieldCurve::forwardRate()`.

**6.9.2.12** virtual Real creditCurve::forwardRate (Real *forwardStart*, Real *effectiveLengthOfTheContractAfterStart*, interestComposition *composition* = Continuous) [inline, virtual]

Calculates the fwd rate.

**Parameters:**

*forwardStart* start of the rate

*maturityAfterForward* maturity after the start

Reimplemented from **yieldCurve** (p. 225).

Definition at line 216 of file creditCurve.h.

References `_combined`, `yieldCurve::forwardRate()`, and `Real`.

**6.9.2.13** yieldCurve\* creditCurve::getCombined (void) const [inline, protected]

Definition at line 248 of file creditCurve.h.

References `_combined`.

Referenced by `copyObj()`.

**6.9.2.14** Currency creditCurve::getCurrency (void) const [inline]

Definition at line 240 of file creditCurve.h.

References `_currency`, and `Currency`.

Referenced by `copyObj()`.

**6.9.2.15** valarray<cachedval> creditCurve::getDefaultProbability (void) const [inline, protected]

Definition at line 251 of file creditCurve.h.

References `_defaultProbability`.

Referenced by `copyObj()`.

**6.9.2.16** Frequency `creditCurve::getFrequency (void) const` [inline]

Definition at line 241 of file `creditCurve.h`.

References `_frequency`, and `Frequency`.

Referenced by `copyObj()`.

**6.9.2.17** `virtual valarray<Real> creditCurve::getMaturitiesInTheZCBCurve () const` [inline, virtual]

Return the maturities present in the market curve, both from the Cash and Swap Points `valarray<Real>`.

Reimplemented from `yieldCurve` (p. 225).

Definition at line 234 of file `creditCurve.h`.

References `_combined`, and `yieldCurve::getMaturitiesInTheZCBCurve()`.

**6.9.2.18** `virtual char* creditCurve::getName ()` [inline, virtual]

Reimplemented from `yieldCurve` (p. 226).

Definition at line 237 of file `creditCurve.h`.

References `_combined`, and `yieldCurve::getName()`.

**6.9.2.19** `Real creditCurve::getRecoveryRate (void) const` [inline]

Definition at line 239 of file `creditCurve.h`.

References `_recoveryRate`, and `Real`.

Referenced by `copyObj()`.

**6.9.2.20** `valarray<CreditSpreadPoint> creditCurve::getSpreads (void) const` [inline, protected]

Definition at line 249 of file `creditCurve.h`.

References `_spreads`.

Referenced by `copyObj()`.

**6.9.2.21** `valarray<cachedval> creditCurve::getSurvivalProbability (void) const` [inline, protected]

Definition at line 250 of file `creditCurve.h`.

References `_survivalProbability`.

Referenced by `copyObj()`.

**6.9.2.22** `valarray<cachedval> creditCurve::getSwapFees (void) const` [inline, protected]

Definition at line 252 of file creditCurve.h.

References `_swapFees`.

Referenced by `copyObj()`.

**6.9.2.23** `yieldCurve* creditCurve::getUnderlying (void) const` [inline, protected]

Definition at line 247 of file creditCurve.h.

References `_underlying`.

Referenced by `copyObj()`.

**6.9.2.24** `virtual Real creditCurve::hazardRate (Real maturity) const` [inline, virtual]

returns hazard rate at a given maturity - this is an alias for `defaultProbability` i.e. probability of default at time  $t$  conditional on no earlier default.

**Parameters:**

*maturity* - time at which to evaluate probability

**Returns:**

conditional default probability

Definition at line 172 of file creditCurve.h.

References `defaultProbability()`, and `Real`.

Referenced by `maincreditcurve()`.

**6.9.2.25** `Natural creditCurve::indexOfCurrentSpread (Real maturity) const` [virtual]

Definition at line 254 of file creditCurve.cpp.

References `_spreads`, `Natural`, and `Real`.

Referenced by `defaultProbability()`, `survivalProbability()`, `swapFees()`, and `timeOfCurrentSpread()`.

**6.9.2.26** `Natural creditCurve::indexOfPreviousSpread (Real maturity) const` [virtual]

Definition at line 272 of file creditCurve.cpp.

References `_spreads`, `Natural`, and `Real`.

Referenced by `timeOfPreviousSpread()`.

**6.9.2.27 creditCurve & creditCurve::operator= (const creditCurve & rhs)**

Definition at line 134 of file creditCurve.cpp.

References copyObj().

**6.9.2.28 void creditCurve::resampleSpread ()**

Definition at line 228 of file creditCurve.cpp.

References `_combined`, `_defaultProbability`, `_spreads`, `_survivalProbability`, `_swapFees`, `_underlying`, `Natural`, `Relative`, and `yieldCurve::spotRate()`.

Referenced by `creditCurve()`.

**6.9.2.29 Real creditCurve::riskyDiscountFactor (Real maturity, interestComposition composition = Continuous) [virtual]**

Calculates the risky discount factor incorporating the hazard rate.

**Parameters:**

*maturity* - maturity to calculate risky discount factor for

risky discount = risk free discount \* survival probability

In the class notes we have  $RF = DF * (1 - Q(T))$   $Q(T)$  is cumulative default probability so it is the complement of  $S(T)$ , the cumulative survival probability.

In discrete time we have the identity:  $(S(n) - S(n+1)) / S(n) = q(n)$  where  $q(n)$  is the default probability for period  $n$  conditional on no earlier default.  $q(n)$  is a discrete time version of hazard rate. In the limit (as  $dt \rightarrow 0$ ) this leads to the expression  $S(t) = \exp(-(\text{integral from } 0 \text{ to } t) * h(t) * dt)$

The risky discount factor is a "discounted" discount factor - the discounting applied is the survival probability. In continuous time we can use the expression above but since we have calculated everything to this point in discrete time and we have an explicit expression for the survival probability we use this as the discount factor rather than the continuous time expression above.

Definition at line 394 of file creditCurve.cpp.

References `_underlying`, `yieldCurve::discountFactor()`, `Real`, and `survivalProbability()`.

Referenced by `maincreditcurve()`, and `riskybond::quotedPrice()`.

**6.9.2.30 virtual Real creditCurve::spotRate (Date maturityDate) const [inline, virtual]**

Calculates the spot ZCB rate.

**Parameters:**

*maturityDate* : maturityDate of the ZCB

Reimplemented from `yieldCurve` (p. 228).

Definition at line 180 of file creditCurve.h.

References `_combined`, `Real`, and `yieldCurve::spotRate()`.

**6.9.2.31 virtual Real creditCurve::spotRate (Real *maturity*) const** [inline, virtual]

Calculates the spot ZCB rate.

**Parameters:**

*maturity* : if it is exact it just gives the result from a Point, else an interpolated one based on interpolator

Reimplemented from **yieldCurve** (p. 228).

Definition at line 176 of file creditCurve.h.

References `_combined`, `Real`, and `yieldCurve::spotRate()`.

**6.9.2.32 Real creditCurve::survivalProbability (Real *maturity*) const** [virtual]

Definition at line 295 of file creditCurve.cpp.

References `_survivalProbability`, `defaultProbability()`, `indexOfCurrentSpread()`, `Natural`, `Real`, `timeOfCurrentSpread()`, and `timeOfPreviousSpread()`.

Referenced by `cumulativeDefaultProbability()`, `defaultProbability()`, `maincreditcurve()`, `riskyDiscountFactor()`, and `swapFees()`.

**6.9.2.33 Real creditCurve::swapFees (Real *maturity*) const** [virtual]

Definition at line 320 of file creditCurve.cpp.

References `_swapFees`, `_underlying`, `creditSpread()`, `defaultProbability()`, `yieldCurve::discountFactor()`, `indexOfCurrentSpread()`, `Natural`, `Real`, `survivalProbability()`, `timeOfCurrentSpread()`, and `timeOfPreviousSpread()`.

Referenced by `defaultProbability()`.

**6.9.2.34 Real creditCurve::timeOfCurrentSpread (Real *maturity*) const** [virtual]

Definition at line 249 of file creditCurve.cpp.

References `_spreads`, `indexOfCurrentSpread()`, and `Real`.

Referenced by `defaultProbability()`, `survivalProbability()`, and `swapFees()`.

**6.9.2.35 Real creditCurve::timeOfPreviousSpread (Real *maturity*) const** [virtual]

Definition at line 285 of file creditCurve.cpp.

References `_spreads`, `indexOfPreviousSpread()`, and `Real`.

Referenced by `defaultProbability()`, `survivalProbability()`, and `swapFees()`.

## 6.9.3 Friends And Related Function Documentation

**6.9.3.1 ostream& operator<< (ostream & *os*, const creditCurve \* *c*)** [friend]

Definition at line 69 of file creditCurve.h.

**6.9.3.2 ostream& operator<< (ostream & os, const creditCurve & c) [friend]**

Definition at line 423 of file creditCurve.cpp.

**6.9.4 Member Data Documentation****6.9.4.1 yieldCurve\* creditCurve::\_combined [protected]**

Definition at line 245 of file creditCurve.h.

Referenced by copyObj(), creditCurve(), creditSpread(), discountFactor(), forwardRate(), getCombined(), getMaturitiesInTheZCBCurve(), getName(), operator<<(), resampleSpread(), spotRate(), and ~creditCurve().

**6.9.4.2 Currency creditCurve::\_currency [private]**

Definition at line 264 of file creditCurve.h.

Referenced by copyObj(), and getCurrency().

**6.9.4.3 valarray<cachedval> creditCurve::\_defaultProbability [mutable, private]**

Definition at line 260 of file creditCurve.h.

Referenced by assignFlatSpread(), copyObj(), defaultProbability(), getDefaultProbability(), and resampleSpread().

**6.9.4.4 Frequency creditCurve::\_frequency [private]**

Definition at line 265 of file creditCurve.h.

Referenced by copyObj(), and getFrequency().

**6.9.4.5 Real creditCurve::\_recoveryRate [private]**

Definition at line 263 of file creditCurve.h.

Referenced by copyObj(), defaultProbability(), and getRecoveryRate().

**6.9.4.6 valarray<CreditSpreadPoint> creditCurve::\_spreads [private]**

Definition at line 256 of file creditCurve.h.

Referenced by assignFlatSpread(), copyObj(), getSpreads(), indexOfCurrentSpread(), indexOfPreviousSpread(), resampleSpread(), timeOfCurrentSpread(), and timeOfPreviousSpread().

**6.9.4.7 valarray<cachedval> creditCurve::\_survivalProbability [mutable, private]**

Definition at line 259 of file creditCurve.h.

Referenced by assignFlatSpread(), copyObj(), getSurvivalProbability(), resampleSpread(), and survivalProbability().

**6.9.4.8** `valarray<cachedval> creditCurve::_swapFees` [mutable, private]

Definition at line 261 of file `creditCurve.h`.

Referenced by `assignFlatSpread()`, `copyObj()`, `getSwapFees()`, `resampleSpread()`, and `swapFees()`.

**6.9.4.9** `yieldCurve* creditCurve::_underlying` [protected]

Definition at line 244 of file `creditCurve.h`.

Referenced by `copyObj()`, `creditCurve()`, `creditSpread()`, `defaultProbability()`, `getUnderlying()`, `resampleSpread()`, `riskyDiscountFactor()`, `swapFees()`, and `~creditCurve()`.

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

- `creditCurve.h`
- `creditCurve.cpp`



## 6.10 CreditSpreadPoint Class Reference

used to encapsulate a spread at a given maturity

```
#include <creditCurve.h>
```

### Public Member Functions

- **CreditSpreadPoint** (void)  
*Default Constructor.*
- **CreditSpreadPoint** (Real r, Real T, CreditSpreadType t)  
*Constructor.*
- **~CreditSpreadPoint** (void)  
*Destructor.*
- **Real** **getRate** ()  
*Associated rate.*
- **Real** **getMaturity** ()
- **CreditSpreadType** **getSpreadType** ()
- void **setRate** (Real r)
- void **setMaturity** (Real m)
- void **setType** (CreditSpreadType t)

### Static Public Member Functions

- char \* **TypeAsString** (CreditSpreadType t)

### Private Attributes

- **Real** **\_rate**
- **Real** **\_maturity**
- **CreditSpreadType** **\_spreadtype**

#### 6.10.1 Detailed Description

used to encapsulate a spread at a given maturity

Definition at line 30 of file creditCurve.h.

#### 6.10.2 Constructor & Destructor Documentation

##### 6.10.2.1 CreditSpreadPoint::CreditSpreadPoint (void)

Default Constructor.

Definition at line 5 of file creditCurve.cpp.

References Relative.

### 6.10.2.2 `CreditSpreadPoint::CreditSpreadPoint (Real $r$ , Real $T$ , CreditSpreadType $t$ )`

Constructor.

#### Parameters:

$s$ : Real spread between risky asset and risk free rate

$T$ : Real maturity of the spread

$t$ : Absolute or relative spread

Definition at line 7 of file `creditCurve.cpp`.

References `r`, and `Real`.

### 6.10.2.3 `CreditSpreadPoint::~~CreditSpreadPoint (void)`

Destructor.

Definition at line 9 of file `creditCurve.cpp`.

## 6.10.3 Member Function Documentation

### 6.10.3.1 `Real CreditSpreadPoint::getMaturity () [inline]`

Definition at line 50 of file `creditCurve.h`.

References `Real`.

### 6.10.3.2 `Real CreditSpreadPoint::getRate () [inline]`

Associated rate.

Definition at line 48 of file `creditCurve.h`.

References `Real`.

### 6.10.3.3 `CreditSpreadType CreditSpreadPoint::getSpreadType () [inline]`

Definition at line 52 of file `creditCurve.h`.

References `_spreadtype`, and `CreditSpreadType`.

### 6.10.3.4 `void CreditSpreadPoint::setMaturity (Real $m$ ) [inline]`

Definition at line 56 of file `creditCurve.h`.

References `m`, and `Real`.

### 6.10.3.5 `void CreditSpreadPoint::setRate (Real $r$ ) [inline]`

Definition at line 54 of file `creditCurve.h`.

References `r`, and `Real`.

**6.10.3.6 void CreditSpreadPoint::setType (CreditSpreadType t) [inline]**

Definition at line 58 of file creditCurve.h.

References `_spreadtype`.

**6.10.3.7 char \* CreditSpreadPoint::TypeAsString (CreditSpreadType t) [static]**

Definition at line 11 of file creditCurve.cpp.

References `Absolute`, and `Relative`.

Referenced by `CSVParser::operator>>()`.

**6.10.4 Member Data Documentation****6.10.4.1 Real CreditSpreadPoint::\_maturity [private]**

Definition at line 61 of file creditCurve.h.

**6.10.4.2 Real CreditSpreadPoint::\_rate [private]**

Definition at line 60 of file creditCurve.h.

**6.10.4.3 CreditSpreadType CreditSpreadPoint::\_spreadtype [private]**

Definition at line 62 of file creditCurve.h.

Referenced by `getSpreadType()`, and `setType()`.

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

- `creditCurve.h`
- `creditCurve.cpp`

## 6.11 CSVParser Class Reference

```
#include <csvparser.h>
```

### Public Member Functions

- **CSVParser** ()
- **const CSVParser & operator<<** (const string &sIn)
- **const CSVParser & operator<<** (const char \*sIn)
- **CSVParser & operator>>** (int &nOut)
- **CSVParser & operator>>** (**Natural** &nOut)
- **CSVParser & operator>>** (double &nOut)
- **CSVParser & operator>>** (string &sOut)
- **CSVParser & operator>>** (**TypeOfRate** &tOut)
  - used to parse yield curve points - type of rate can be cash or swap*
- **CSVParser & operator>>** (**CreditSpreadType** &tOut)
  - used to parse credit curve spreads - type of rate can be absolute or relative*
- **CSVParser & operator>>** (**Date** &dOut)

### Private Member Functions

- void **SkipSpaces** (void)

### Private Attributes

- string **m\_sData**
- string::size\_type **m\_nPos**

#### 6.11.1 Constructor & Destructor Documentation

##### 6.11.1.1 CSVParser::CSVParser ()

Definition at line 41 of file csvparser.cpp.

References `m_nPos`, and `m_sData`.

#### 6.11.2 Member Function Documentation

##### 6.11.2.1 const CSVParser & CSVParser::operator<< (const char \* sIn)

Definition at line 60 of file csvparser.cpp.

References `m_nPos`, and `m_sData`.

##### 6.11.2.2 const CSVParser & CSVParser::operator<< (const string & sIn)

Definition at line 53 of file csvparser.cpp.

References `m_nPos`, and `m_sData`.

**6.11.2.3 CSVParser & CSVParser::operator>> (Date & dOut)**

Definition at line 162 of file csvparser.cpp.

References Day, m, m\_nPos, m\_sData, Month, StringTokenizer::nextIntToken(), SkipSpaces(), and Year.

**6.11.2.4 CSVParser & CSVParser::operator>> (CreditSpreadType & tOut)**

used to parse credit curve spreads - type of rate can be absolute or relative

Definition at line 111 of file csvparser.cpp.

References Absolute, m\_nPos, m\_sData, Relative, SkipSpaces(), and CreditSpreadPoint::TypeAsString().

**6.11.2.5 CSVParser & CSVParser::operator>> (TypeOfRate & tOut)**

used to parse yield curve points - type of rate can be cash or swap

Definition at line 97 of file csvparser.cpp.

References Cash, m\_nPos, m\_sData, SkipSpaces(), Swap, and yieldPoint::TypeAsString().

**6.11.2.6 CSVParser & CSVParser::operator>> (string & sOut)**

Definition at line 127 of file csvparser.cpp.

References m\_nPos, m\_sData, and SkipSpaces().

**6.11.2.7 CSVParser & CSVParser::operator>> (double & nOut)**

Definition at line 84 of file csvparser.cpp.

References m\_nPos, m\_sData, and SkipSpaces().

**6.11.2.8 CSVParser & CSVParser::operator>> (Natural & nOut)**

Definition at line 79 of file csvparser.cpp.

References Natural, and operator>>().

**6.11.2.9 CSVParser & CSVParser::operator>> (int & nOut)**

Definition at line 67 of file csvparser.cpp.

References m\_nPos, m\_sData, and SkipSpaces().

Referenced by operator>>().

**6.11.2.10 void CSVParser::SkipSpaces (void) [private]**

Definition at line 47 of file csvparser.cpp.

References m\_nPos, and m\_sData.

Referenced by operator>>().

### 6.11.3 Member Data Documentation

#### 6.11.3.1 `string::size_type CSVParser::m_nPos` [private]

Definition at line 46 of file csvparser.h.

Referenced by CSVParser(), operator<<(), operator>>(), and SkipSpaces().

#### 6.11.3.2 `string CSVParser::m_sData` [private]

Definition at line 45 of file csvparser.h.

Referenced by CSVParser(), operator<<(), operator>>(), and SkipSpaces().

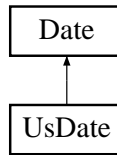
The documentation for this class was generated from the following files:

- `csvparser.h`
- `csvparser.cpp`

## 6.12 Date Class Reference

```
#include <date.h>
```

Inheritance diagram for Date::



### Public Member Functions

- **Date** (void)  
*Default constructor.*
- **Date** (LongInteger serialNumber)  
*Constructor taking a serial number.*
- **Date** (Day d, Month m, Year y)  
*Constructor taking day, month and year.*
- **Date** (Day d, ShortNatural m, Year y)  
*Constructor taking day, month as an integer and year.*
- **~Date** (void)
- **Weekday** weekday () const
- **Day** dayOfMonth () const
- **Day** dayOfYear () const
- **Month** month () const
- **Year** year () const
- **LongInteger** serialNumber () const
- **bool** isEndOfMonth () const
- **Day** lastDayOfMonth () const
- **void** setDateToToday ()  
*Set Date to system today's date.*
- **Date & operator+=** (LongInteger days)  
*increments date by the given number of days*
- **Date & operator-=** (LongInteger days)  
*decrement date by the given number of days*
- **Date & operator++** ()  
*1-day pre-increment*
- **Date** **operator++** (int)  
*1-day post-increment*

- **Date & operator-** ()  
*1-day pre-decrement*
- **Date operator-** (int)  
*1-day post-decrement*
- **Date operator+** (LongInteger days) const  
*returns a new date incremented by the given number of days*
- **Date operator-** (LongInteger days) const  
*returns a new date decremented by the given number of days*
- **bool operator==** (const Date &d2)
- **bool operator!=** (const Date &d2)
- **bool operator<** (const Date &d2)
- **bool operator<=** (const Date &d2)
- **bool operator>** (const Date &d2)
- **bool operator>=** (const Date &d2)
- **Date plusDays** (Integer n) const
- **Date plusWeeks** (Integer n) const
- **Date plusMonths** (Integer n) const
- **Date plusYears** (Integer n) const
- **Date plus** (Integer n, TimeUnit units) const
- **bool isBusinessDay** ()  
*Apply Conventions (use for UsDate(p.206) for example).*
- **void applyConvention** (BusinessDayConvention convention=Following)
- **Date returnDateConvention** (const Date &date, BusinessDayConvention convention=Following)
- **Real dayCount** (const Date &d, DayCountConvention dayconvention=ACT\_365) const  
*DayCount Between Dates.*
- **char \* toString** () const  
*Return char\* version of the date.*

## Static Public Member Functions

- **Date minDate** ()  
*earliest allowed date*
- **Date maxDate** ()  
*latest allowed date*
- **bool isLeap** (Year y)  
*whether the given year is a leap one*
- **Date endOfMonth** (const Date &d)  
*last day of the month to which the given date belongs*



- **bool isEOM** (const **Date** &d)  
*whether a date is the last day of its month*
- **Date nextWeekday** (const **Date** &d, **Weekday**)  
*next given weekday following or equal to the given date*
- **Date nthWeekday** (**ShortInteger** n, **Weekday**, **Month** m, **Year** y)  
*n-th given weekday in the given month and year*

## Static Private Member Functions

- **Date advance** (const **Date** &d, **Integer** units, **TimeUnit**)
- **Integer monthLength** (**Month** m, bool leapYear)
- **Integer monthOffset** (**Month** m, bool leapYear)
- **LongInteger yearOffset** (**Year** y)
- **LongInteger minimumSerialNumber** ()
- **LongInteger maximumSerialNumber** ()

## Private Attributes

- **LongInteger** `_serialNumber`

### 6.12.1 Constructor & Destructor Documentation

#### 6.12.1.1 **Date::Date** (void)

Default constructor.

Definition at line 4 of file date.cpp.

References `LongInteger`.

Referenced by `advance()`, `applyConvention()`, `Date()`, `endOfMonth()`, `nthWeekday()`, `operator+()`, `operator-()`, and `returnDateConvention()`.

#### 6.12.1.2 **Date::Date** (**LongInteger** *serialNumber*)

Constructor taking a serial number.

Definition at line 8 of file date.cpp.

References `LongInteger`.

#### 6.12.1.3 **Date::Date** (**Day** *d*, **Month** *m*, **Year** *y*)

Constructor taking day, month and year.

Definition at line 12 of file date.cpp.

References `_serialNumber`, `Day`, `isLeap()`, `m`, `monthLength()`, `monthOffset()`, `Year`, and `yearOffset()`.

**6.12.1.4 Date::Date (Day *d*, ShortNatural *m*, Year *y*)**

Constructor taking day, month as an integer and year.

Definition at line 18 of file date.cpp.

References April, August, Date(), Day, December, February, January, July, June, m, March, May, November, October, September, ShortNatural, and Year.

**6.12.1.5 Date::~~Date (void)**

Definition at line 418 of file date.cpp.

**6.12.2 Member Function Documentation****6.12.2.1 Date Date::advance (const Date & *d*, Integer *units*, TimeUnit) [static, private]**

Definition at line 175 of file date.cpp.

References Date(), Day, dayOfMonth(), Days, February, Integer, isLeap(), m, Month, month(), monthLength(), Months, Weeks, year(), Year, and Years.

Referenced by plus(), plusDays(), plusMonths(), plusWeeks(), and plusYears().

**6.12.2.2 void Date::applyConvention (BusinessDayConvention *convention* = Following)**

Definition at line 366 of file date.cpp.

References \_serialNumber, Date(), Following, isBusinessDay(), LongInteger, ModifiedFollowing, ModifiedPreceding, month(), and Preceding.

Referenced by bond::getCashflow(), returnDateConvention(), and SwapLeg::SwapLeg().

**6.12.2.3 Real Date::dayCount (const Date & *d*, DayCountConvention *dayconvention* = ACT\_365) const**

DayCount Between Dates.

Definition at line 397 of file date.cpp.

References \_serialNumber, ACT\_360, ACT\_365, Day30\_360, Day30\_365, dayOfMonth(), month(), Real, serialNumber(), TN\_REAL, and year().

Referenced by bond::convexity(), yieldCurve::discountFactor(), bond::duration(), asset::forwardPrice(), yieldCurve::forwardRate(), volsurface::forwardVolatility(), bond::getCashflow(), bond::getMaturityInYears(), riskybond::quotedPrice(), bond::quotedPrice(), volsurface::setvolsurface(), yieldCurve::spotRate(), volsurface::volatility(), and bond::yieldToMaturity().

**6.12.2.4 Day Date::dayOfMonth () const**

Definition at line 53 of file date.cpp.

References Day, dayOfYear(), isLeap(), month(), monthOffset(), and year().

Referenced by `advance()`, `dayCount()`, `bond::fairvalue()`, `UsDate::isBusinessDay()`, `isEOM()`, `lastDayOfMonth()`, and `toString()`.

#### 6.12.2.5 Day Date::dayOfYear () const

Definition at line 57 of file `date.cpp`.

References `_serialNumber`, `Day`, `year()`, and `yearOffset()`.

Referenced by `dayOfMonth()`, and `month()`.

#### 6.12.2.6 Date Date::endOfMonth (const Date & d) [static]

last day of the month to which the given date belongs

Definition at line 263 of file `date.cpp`.

References `Date()`, `isLeap()`, `m`, `month()`, `Month`, `monthLength()`, `year()`, and `Year`.

Referenced by `lastDayOfMonth()`.

#### 6.12.2.7 bool Date::isBusinessDay ()

Apply Conventions (use for `UsDate`(p. 206) for example).

Reimplemented in `UsDate` (p. 206).

Definition at line 362 of file `date.cpp`.

Referenced by `applyConvention()`.

#### 6.12.2.8 bool Date::isEndOfMonth () const

Definition at line 83 of file `date.cpp`.

References `isEOM()`.

#### 6.12.2.9 bool Date::isEOM (const Date & d) [static]

whether a date is the last day of its month

Definition at line 269 of file `date.cpp`.

References `dayOfMonth()`, `isLeap()`, `month()`, `monthLength()`, and `year()`.

Referenced by `isEndOfMonth()`.

#### 6.12.2.10 bool Date::isLeap (Year y) [static]

whether the given year is a leap one

Definition at line 215 of file `date.cpp`.

References `Year`.

Referenced by `advance()`, `Date()`, `dayOfMonth()`, `endOfMonth()`, `isEOM()`, `month()`, and `setDateToToday()`.

**6.12.2.11 Day Date::lastDayOfMonth () const**

Definition at line 87 of file date.cpp.

References Day, dayOfMonth(), and endOfMonth().

Referenced by maindate().

**6.12.2.12 Date Date::maxDate () [static]**

latest allowed date

Definition at line 170 of file date.cpp.

References maximumSerialNumber().

**6.12.2.13 LongInteger Date::maximumSerialNumber () [static, private]**

Definition at line 358 of file date.cpp.

References LongInteger.

Referenced by maxDate().

**6.12.2.14 Date Date::minDate () [static]**

earliest allowed date

Definition at line 165 of file date.cpp.

References minimumSerialNumber().

**6.12.2.15 LongInteger Date::minimumSerialNumber () [static, private]**

Definition at line 354 of file date.cpp.

References LongInteger.

Referenced by minDate().

**6.12.2.16 Month Date::month () const**

Definition at line 61 of file date.cpp.

References Day, dayOfYear(), Integer, isLeap(), m, Month, monthOffset(), and year().

Referenced by advance(), applyConvention(), dayCount(), dayOfMonth(), endOfMonth(), bond::fairvalue(), UsDate::isBusinessDay(), isEOM(), and toString().

**6.12.2.17 Integer Date::monthLength (Month *m*, bool *leapYear*) [static, private]**

Definition at line 285 of file date.cpp.

References Integer, and m.

Referenced by advance(), Date(), endOfMonth(), isEOM(), and setDateToToday().

**6.12.2.18 Integer Date::monthOffset (Month *m*, bool *leapYear*)** [static, private]

Definition at line 295 of file date.cpp.

References Integer, and m.

Referenced by Date(), dayOfMonth(), month(), and setDateToToday().

**6.12.2.19 Date Date::nextWeekday (const Date & *d*, Weekday)** [static]

next given weekday following or equal to the given date

Definition at line 273 of file date.cpp.

References weekday(), and Weekday.

**6.12.2.20 Date Date::nthWeekday (ShortInteger *n*, Weekday, Month *m*, Year *y*)**  
[static]

*n*-th given weekday in the given month and year

Definition at line 278 of file date.cpp.

References Date(), m, ShortInteger, Weekday, and Year.

**6.12.2.21 bool Date::operator!= (const Date & *d2*)** [inline]

Definition at line 124 of file date.h.

References \_serialNumber, and serialNumber().

**6.12.2.22 Date Date::operator+ (LongInteger *days*) const**

returns a new date incremented by the given number of days

Definition at line 137 of file date.cpp.

References \_serialNumber, Date(), and LongInteger.

**6.12.2.23 Date Date::operator++ (int)**

1-day post-increment

Definition at line 117 of file date.cpp.

References \_serialNumber, and LongInteger.

**6.12.2.24 Date & Date::operator++ ()**

1-day pre-increment

Definition at line 111 of file date.cpp.

References \_serialNumber, and LongInteger.

**6.12.2.25 Date & Date::operator+= (LongInteger *days*)**

increments date by the given number of days

Definition at line 99 of file date.cpp.

References `_serialNumber`, and `LongInteger`.

**6.12.2.26 Date Date::operator- (LongInteger *days*) const**

returns a new date decremented by the given number of days

Definition at line 141 of file date.cpp.

References `_serialNumber`, `Date()`, and `LongInteger`.

**6.12.2.27 Date Date::operator- (int)**

1-day post-decrement

Definition at line 130 of file date.cpp.

References `_serialNumber`, and `LongInteger`.

**6.12.2.28 Date & Date::operator- ()**

1-day pre-decrement

Definition at line 124 of file date.cpp.

References `_serialNumber`, and `LongInteger`.

**6.12.2.29 Date & Date::operator-= (LongInteger *days*)**

decrement date by the given number of days

Definition at line 105 of file date.cpp.

References `_serialNumber`, and `LongInteger`.

**6.12.2.30 bool Date::operator< (const Date & *d2*) [inline]**

Definition at line 125 of file date.h.

References `_serialNumber`, and `serialNumber()`.

**6.12.2.31 bool Date::operator<= (const Date & *d2*) [inline]**

Definition at line 126 of file date.h.

References `_serialNumber`, and `serialNumber()`.

**6.12.2.32 bool Date::operator== (const Date & *d2*) [inline]**

Definition at line 123 of file date.h.

References `_serialNumber`, and `serialNumber()`.

#### 6.12.2.33 `bool Date::operator> (const Date & d2) [inline]`

Definition at line 127 of file `date.h`.

References `_serialNumber`, and `serialNumber()`.

#### 6.12.2.34 `bool Date::operator>= (const Date & d2) [inline]`

Definition at line 128 of file `date.h`.

References `_serialNumber`, and `serialNumber()`.

#### 6.12.2.35 `Date Date::plus (Integer n, TimeUnit units) const`

Definition at line 161 of file `date.cpp`.

References `advance()`, and `Integer`.

Referenced by `mainconvertiblebond()`.

#### 6.12.2.36 `Date Date::plusDays (Integer n) const`

Definition at line 145 of file `date.cpp`.

References `advance()`, `Days`, and `Integer`.

Referenced by `Drift::Drift()`, `asset::forwardPrice()`, `Exotics::getTheta()`, `inputBond()`, `inputBSOption()`, `inputButterflySpread()`, `inputCallSpread()`, `inputConvertibleBond()`, `inputPutSpread()`, `inputRatioCallSpread()`, `inputStraddle()`, `inputStrangle()`, `inputVanillaSwap()`, `mainasset()`, `RainbowOption::reassignVolsAtThemoney()`, and `RainbowOption::reassignVolsAtThestrike()`.

#### 6.12.2.37 `Date Date::plusMonths (Integer n) const`

Definition at line 153 of file `date.cpp`.

References `advance()`, `Integer`, and `Months`.

Referenced by `bond::getCashflow()`, `mainasset()`, `mainIRVanillaSwap()`, and `mainyieldcurve()`.

#### 6.12.2.38 `Date Date::plusWeeks (Integer n) const`

Definition at line 149 of file `date.cpp`.

References `advance()`, `Integer`, and `Weeks`.

#### 6.12.2.39 `Date Date::plusYears (Integer n) const`

Definition at line 157 of file `date.cpp`.

References `advance()`, `Integer`, and `Years`.

Referenced by `mainasset()`, and `mainIRVanillaSwap()`.

#### 6.12.2.40 Date Date::returnDateConvention (const Date & date, BusinessDayConvention convention = Following)

Definition at line 391 of file date.cpp.

References applyConvention(), Date(), and serialNumber().

#### 6.12.2.41 LongInteger Date::serialNumber () const

Definition at line 79 of file date.cpp.

References \_serialNumber, and LongInteger.

Referenced by CashFlow::CashFlow(), dayCount(), Drift::Drift(), bond::fairvalue(), operator!=(), operator<(), operator<=(), operator==(), operator>(), operator>=(), returnDateConvention(), and SwapLeg::SwapLeg().

#### 6.12.2.42 void Date::setDateToToday ()

Set Date to system today's date.

Definition at line 91 of file date.cpp.

References \_serialNumber, Day, isLeap(), Month, monthLength(), monthOffset(), and yearOffset().

Referenced by CashFlow::CashFlow(), yieldCurve::discountFactor(), asset::forwardPrice(), yieldCurve::forwardRate(), Exotics::getTheta(), importData::importVolSurface(), inputBond(), inputBSOption(), inputButterflySpread(), inputCallSpread(), inputConvertibleBond(), inputPutSpread(), inputRainbowOption(), inputRatioCallSpread(), inputStraddle(), inputStrangle(), inputVanillaSwap(), mainasset(), mainconvertiblebond(), maindate(), mainIRVanillaSwap(), mainmc(), mainrainbowoptions(), mainyieldcurve(), RainbowOption::RainbowOption(), and yieldCurve::spotRate().

#### 6.12.2.43 char \* Date::toString () const

Return char\* version of the date.

Definition at line 412 of file date.cpp.

References dayOfMonth(), month(), and year().

Referenced by mainasset(), maindate(), mainvolsurface(), and operator<<().

#### 6.12.2.44 Weekday Date::weekday () const

Definition at line 48 of file date.cpp.

References \_serialNumber, Integer, and Weekday.

Referenced by UsDate::isBusinessDay(), and nextWeekday().

#### 6.12.2.45 Year Date::year () const

Definition at line 72 of file date.cpp.

References \_serialNumber, Year, and yearOffset().



Referenced by `advance()`, `dayCount()`, `dayOfMonth()`, `dayOfYear()`, `endOfMonth()`, `bond::fairvalue()`, `isEOM()`, `month()`, and `toString()`.

#### 6.12.2.46 `LongInteger Date::yearOffset (Year y)` [static, private]

Definition at line 306 of file `date.cpp`.

References `LongInteger`, and `Year`.

Referenced by `Date()`, `dayOfYear()`, `setDateToToday()`, and `year()`.

### 6.12.3 Member Data Documentation

#### 6.12.3.1 `LongInteger Date::_serialNumber` [private]

Definition at line 167 of file `date.h`.

Referenced by `applyConvention()`, `Date()`, `dayCount()`, `dayOfYear()`, `operator!=()`, `operator+()`, `operator++()`, `operator+=()`, `operator-()`, `operator--()`, `operator-=()`, `operator<()`, `operator<=()`, `operator==()`, `operator>()`, `operator>=()`, `serialNumber()`, `setDateToToday()`, `weekday()`, and `year()`.

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

- `date.h`
- `date.cpp`

## 6.13 Drift Class Reference

```
#include <Drift.h>
```

### Public Member Functions

- **Drift** (**Date** startDate, **Real** ExpiryInYears, **LongNatural** &nDates, **yieldCurve** \*pyieldCurve, **volsurface** \*pvolsurface, **Real** Strike)  
*Default constructor.*
- **Drift** (void)  
*Default constructor needed.*
- **Drift** (**Date** startDate, **Real** ExpiryInYears, **Real** rateToMaturity, **Real** volToMaturity)  
*Simpler constructor - for 1 date (non path dependant payoffs, not need to have yc and volsurface.*
- **~Drift** ()
- valarray< **Real** > **GetvDrift** (void)  
*Return Drift.*
- **Real** **GetDriftattimei** (**LongNatural** i)  
*Get Drift for time i.*
- valarray< **LongInteger** > **GetvDates** (void)  
*Return serial numbers of dates computed for the drift.*
- **LongInteger** **GetTimeBtwDates** (**LongNatural** i, **LongNatural** j)  
*Return Difference in days between two dates of the drift.*

### Private Attributes

- valarray< **LongInteger** > **vDates**
- valarray< **Real** > **vDrift**
- unsigned long **m\_nDates**

#### 6.13.1 Detailed Description

**Author:**

Simon

Definition at line 11 of file Drift.h.

#### 6.13.2 Constructor & Destructor Documentation

**6.13.2.1** **Drift::Drift** (**Date** *startDate*, **Real** *ExpiryInYears*, **LongNatural** & *nDates*, **yieldCurve** \* *pyieldCurve*, **volsurface** \* *pvolsurface*, **Real** *Strike*)

Default constructor.

**Parameters:**

*startDate*: Start **Date**(p. 71) of the drift

*ExpiryInYears*: Length of the drift to be computed

*nDates*: Number of dates to cut the drift

*pyieldCurve*: Yield Curve to compute the spot rate between each date

*pvolsurface*: Volatility Surface to compute the forward volatility between each date

*Strike*: Strike of the option, used to compute the colatility (function of the strike)

Definition at line 5 of file Drift.cpp.

References `yieldCurve::forwardRate()`, `volsurface::forwardVolatility()`, `LongNatural`, `Natural`, `Date::plusDays()`, `Real`, `Date::serialNumber()`, `vDates`, and `vDrift`.

**6.13.2.2 Drift::Drift (void)**

Default constructor needed.

**Author:**

Yann

Definition at line 26 of file Drift.cpp.

**6.13.2.3 Drift::Drift (Date startDate, Real ExpiryInYears, Real rateToMaturity, Real volToMaturity)**

Simpler constructor - for 1 date (non path dependant payoffs, not need to have yc and volsurface.

**Author:**

Yann

**Parameters:**

*startDate*: Start **Date**(p. 71) of the drift

*ExpiryInYears*: Length of the drift to be computed

*rateToMaturity*: spot rate to maturity - can come from a yc in calling classes

*volToMaturity*: volatility to maturity - can come from a volsurface in calling classes

Definition at line 30 of file Drift.cpp.

References `m_nDates`, `Natural`, `Date::plusDays()`, `Real`, `Date::serialNumber()`, `vDates`, and `vDrift`.

**6.13.2.4 Drift::~~Drift ()**

Definition at line 47 of file Drift.cpp.

### 6.13.3 Member Function Documentation

#### 6.13.3.1 Real Drift::GetDriftatTimei (LongNatural i)

Get Drift for time i.

Definition at line 60 of file Drift.cpp.

References LongNatural, m\_nDates, Real, and vDrift.

#### 6.13.3.2 LongInteger Drift::GetTimeBtwDates (LongNatural i, LongNatural j)

Return Difference in days between two dates of the drift.

Definition at line 78 of file Drift.cpp.

References LongInteger, LongNatural, m\_nDates, and vDates.

#### 6.13.3.3 valarray< LongInteger > Drift::GetvDates (void)

Return serial numbers of dates computed for the drift.

Definition at line 68 of file Drift.cpp.

References m\_nDates, Natural, and vDates.

Referenced by mainmc().

#### 6.13.3.4 valarray< Real > Drift::GetvDrift (void)

Return Drift.

Definition at line 51 of file Drift.cpp.

References m\_nDates, Natural, and vDrift.

Referenced by mainmc().

### 6.13.4 Member Data Documentation

#### 6.13.4.1 unsigned long Drift::m\_nDates [private]

Definition at line 52 of file Drift.h.

Referenced by Drift(), GetDriftatTimei(), GetTimeBtwDates(), GetvDates(), and GetvDrift().

#### 6.13.4.2 valarray<LongInteger> Drift::vDates [private]

Definition at line 50 of file Drift.h.

Referenced by Drift(), GetTimeBtwDates(), and GetvDates().

#### 6.13.4.3 valarray<Real> Drift::vDrift [private]

Definition at line 51 of file Drift.h.

Referenced by `Drift()`, `GetDriftattimei()`, and `GetvDrift()`.

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

- **Drift.h**
- **Drift.cpp**

## 6.14 Exotics Class Reference

```
#include <Exotics.h>
```

### Public Member Functions

- **Exotics** (**exoticsType** type, **yieldCurve** \*curve, **volsurface** \*surface, **Real** spot, **Real** strike, **Real** Expiry, **LongNatural** nDates=10, **Real** strike2=-1., **LongNatural** nPaths=100000)

*Default Constructor.*

- **~Exotics** (void)
- **Real** getPrice ()

*return the price of the option by MC*

- **Real** getRho ()

*return the sensitivity to interest rate of the option by MC*

- **Real** getTheta ()

*return the sensitivity to time of the option by MC (+defaultAdvDays)*

- **Real** getVega ()

*return the sensitivity to volatility of the option by MC*

- **Real** getDelta ()

*return the sensitivity to the spot price of the option by MC*

### Private Attributes

- **exoticsType** \_type
- **Real** \_expiry
- **Real** \_spot
- **Real** \_strike
- **Real** \_strike2
- **volsurface** \* \_volSurface
- **yieldCurve** \* \_yieldCurve
- **LongNatural** \_nPaths
- **LongNatural** \_nDates

#### 6.14.1 Constructor & Destructor Documentation

- ##### 6.14.1.1 Exotics::Exotics (exoticsType type, yieldCurve \* curve, volsurface \* surface, Real spot, Real Expiry, LongNatural nDates = 10, Real strike2 = -1., LongNatural nPaths = 100000)

Default Constructor.

#### Parameters:

*type*: type of exotics product

*curve*: pointer to the yield curve  
*surface*: pointer to a vol surface for the underlying  
*spot*: spot of the underlying  
*strike*: strike1 of the option  
*expiry*: maturity of the option  
*nPaths*: number of paths to be generated by the MC pricer  
*nDates*: number of dates to be generated  
*strike2*: second strike for the option (for collared cliquet here)

Definition at line 3 of file Exotics.cpp.

References LongNatural, and Real.

#### 6.14.1.2 Exotics::~Exotics (void)

Definition at line 17 of file Exotics.cpp.

### 6.14.2 Member Function Documentation

#### 6.14.2.1 Real Exotics::getDelta ()

return the sensitivity to the spot price of the option by MC

Definition at line 46 of file Exotics.cpp.

References getPrice(), and Real.

Referenced by inputExoticOptionOnSingleAsset().

#### 6.14.2.2 Real Exotics::getPrice ()

return the price of the option by MC

Definition at line 21 of file Exotics.cpp.

References \_expiry, \_nDates, \_nPaths, \_strike2, \_volSurface, AsianCall, AsianPut, BarrierCall, BarrierPut, CappedCliquet, CollaredCliquet, FlooredCliquet, mainmc(), Real, RevLookbackCall, and RevLookbackPut.

Referenced by getDelta(), getRho(), getTheta(), getVega(), and inputExoticOptionOnSingleAsset().

#### 6.14.2.3 Real Exotics::getRho ()

return the sensitivity to interest rate of the option by MC

Definition at line 55 of file Exotics.cpp.

References getPrice(), Real, and yieldCurve::shiftZCBRateCurve().

Referenced by inputExoticOptionOnSingleAsset().

#### 6.14.2.4 Real Exotics::getTheta ()

return the sensitivity to time of the option by MC (+defaultAdvDays)

Definition at line 64 of file Exotics.cpp.

References `_volSurface`, `volsurface::forwardvolsurface()`, `yieldCurve::forwardZCBCurve()`, `getPrice()`, `Integer`, `Date::plusDays()`, `Real`, and `Date::setDateToToday()`.

Referenced by `inputExoticOptionOnSingleAsset()`.

#### 6.14.2.5 Real Exotics::getVega ()

return the sensitivity to volatility of the option by MC

Definition at line 80 of file Exotics.cpp.

References `_volSurface`, `getPrice()`, `Real`, and `volsurface::shiftedvolsurface()`.

Referenced by `inputExoticOptionOnSingleAsset()`.

### 6.14.3 Member Data Documentation

#### 6.14.3.1 Real Exotics::\_expiry [private]

Definition at line 64 of file Exotics.h.

Referenced by `getPrice()`.

#### 6.14.3.2 LongNatural Exotics::\_nDates [private]

Definition at line 71 of file Exotics.h.

Referenced by `getPrice()`.

#### 6.14.3.3 LongNatural Exotics::\_nPaths [private]

Definition at line 70 of file Exotics.h.

Referenced by `getPrice()`.

#### 6.14.3.4 Real Exotics::\_spot [private]

Definition at line 65 of file Exotics.h.

#### 6.14.3.5 Real Exotics::\_strike [private]

Definition at line 66 of file Exotics.h.

#### 6.14.3.6 Real Exotics::\_strike2 [private]

Definition at line 67 of file Exotics.h.

Referenced by `getPrice()`.



**6.14.3.7** `exoticsType Exotics::_type` [private]

Definition at line 63 of file Exotics.h.

**6.14.3.8** `volSurface* Exotics::_volSurface` [private]

Definition at line 68 of file Exotics.h.

Referenced by `getPrice()`, `getTheta()`, and `getVega()`.

**6.14.3.9** `yieldCurve* Exotics::_yieldCurve` [private]

Definition at line 69 of file Exotics.h.

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

- `Exotics.h`
- `Exotics.cpp`

## 6.15 FileReader Class Reference

```
#include <filereader.h>
```

### Static Public Member Functions

- bool **fileexists** (const char \*filename)  
*test for file existence*
- bool **fileexists** (string filename)  
*test for file existence - string version*
- bool **setdatadir** (const char \*command)
- char \* **getdatadir** (void)
- string **getdatadirasString** (void)
- valarray< **yieldPoint** > \* **buildYieldPointArray** (string filename)  
*Read data from a file to create an array of yield points which can be used to construct a yield curve object.*
- valarray< **CreditSpreadPoint** > \* **buildCreditSpreadPointArray** (string filename)  
*Read data from a file to create an array of credit spread points which can be used to construct a credit curve object.*
- **volSurfaceParams** \* **buildVolSurfaceParams** (string filename)  
*Read data from a file to create an array of strikes and maturities which can be used to construct a vol surface object.*

### Static Protected Attributes

- char **\_datadir** [128] = ""

#### 6.15.1 Member Function Documentation

##### 6.15.1.1 valarray< CreditSpreadPoint > \* FileReader::buildCreditSpreadPointArray (string *filename*) [static]

Read data from a file to create an array of credit spread points which can be used to construct a credit curve object.

#### Parameters:

*filename* - source of data

#### Returns:

valarray of CreditSpreadPoints or null pointer if file not found

Definition at line 132 of file filereader.cpp.

References CC\_MAX\_NUM\_SPREADS, CreditSpreadType, fileexists(), and Natural.

Referenced by importData::importCreditCurve(), and maincreditcurve().

**6.15.1.2** `volsurfaceparams * FileReader::buildVolSurfaceParams (string filename)`  
[static]

Read data from a file to create an array of strikes and maturities which can be used to construct a vol surface object.

**Parameters:**

*filename* - source of data

**Returns:**

structure containing strikes, maturities, etc.

Definition at line 185 of file filereader.cpp.

References volsurfaceparams::callputprices, fileexists(), volsurfaceparams::iscallputprices, volsurfaceparams::maturities, Natural, and volsurfaceparams::strikes.

Referenced by importData::importVolSurface(), and mainvolsurface().

**6.15.1.3** `valarray< yieldPoint > * FileReader::buildYieldPointArray (string filename)`  
[static]

Read data from a file to create an array of yield points which can be used to construct a yield curve object.

**Parameters:**

*filename* - source of data

**Returns:**

valarray of yieldPoints or null pointer if file not found

Definition at line 79 of file filereader.cpp.

References fileexists(), Natural, TypeOfRate, and YC\_MAX\_NUMBER\_POINTS.

Referenced by importData::importYieldCurve(), mainasset(), maincreditcurve(), mainIRVanillaSwap(), and mainyieldcurve().

**6.15.1.4** `bool FileReader::fileexists (string filename)` [static]

test for file existence - string version

**Parameters:**

*filename* - file to test

**Returns:**

true if file exists, otherwise false

Definition at line 27 of file filereader.cpp.

References fileexists().

**6.15.1.5** `bool FileReader::fileexists (const char * filename)` [static]

test for file existence

**Parameters:**

*filename* - file to test

**Returns:**

true if file exists, otherwise false

Definition at line 12 of file filereader.cpp.

Referenced by buildCreditSpreadPointArray(), buildVolSurfaceParams(), buildYieldPointArray(), fileexists(), fr\_basic(), importData::runUserDefinedInterface(), and setdatadir().

**6.15.1.6** `char* FileReader::getdatadir (void)` [inline, static]

Definition at line 17 of file filereader.h.

References \_datadir.

**6.15.1.7** `string FileReader::getdatadirasString (void)` [inline, static]

Definition at line 18 of file filereader.h.

References \_datadir.

Referenced by fr\_basic(), mainasset(), maincreditcurve(), mainIRVanillaSwap(), mainyield-curve(), importData::runInterface(), and importData::runUserDefinedInterface().

**6.15.1.8** `bool FileReader::setdatadir (const char * command)` [static]

Definition at line 31 of file filereader.cpp.

References \_datadir, and fileexists().

Referenced by importData::importData(), and maintests().

**6.15.2 Member Data Documentation****6.15.2.1** `char FileReader::_datadir = ""` [static, protected]

Definition at line 6 of file filereader.cpp.

Referenced by getdatadir(), getdatadirasString(), and setdatadir().

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

- **filereader.h**
- **filereader.cpp**

## 6.16 flowSchedule Class Reference

```
#include <asset.h>
```

### Public Member Functions

- **flowSchedule** (void)  
*Default Constructor.*
- **~flowSchedule** (void)
- **flowSchedule** (Date date, Real percent, BusinessDayConvention bd=Unadjusted)  
*Constructor.*
- void **setDate** (Date date)  
*Sets a date in the schedule.*
- void **setAmount** (Real percent)  
*Sets an amount in the schedule.*
- void **setBusDayConv** (BusinessDayConvention bd)  
*Sets a bus day convention in the schedule.*
- Date **getDate** ()
- Real **getAmount** ()
- BusinessDayConvention **getBusDayConv** ()

### Private Attributes

- Date **\_dateOfFlowPayment**
- Real **\_FlowAmountInPercent**
- BusinessDayConvention **\_businessDayConventionOnPaymentDate**

#### 6.16.1 Detailed Description

##### Author:

Yann Should have include vol surface / yc, etc to just pass assets and not all parameters into the product classes but this one was not the most important

Definition at line 19 of file asset.h.

#### 6.16.2 Constructor & Destructor Documentation

##### 6.16.2.1 flowSchedule::flowSchedule (void)

Default Constructor.

Definition at line 8 of file asset.cpp.

References `_businessDayConventionOnPaymentDate`, `_dateOfFlowPayment`, `_FlowAmountInPercent`, and `Unadjusted`.

### 6.16.2.2 `flowSchedule::~~flowSchedule` (void)

Definition at line 22 of file `asset.cpp`.

### 6.16.2.3 `flowSchedule::flowSchedule` (Date *date*, Real *percent*, BusinessDayConvention *bd* = Unadjusted)

Constructor.

Definition at line 15 of file `asset.cpp`.

References `_businessDayConventionOnPaymentDate`, `_dateOfFlowPayment`, `_FlowAmountInPercent`, and `Real`.

## 6.16.3 Member Function Documentation

### 6.16.3.1 Real `flowSchedule::getAmount` () [inline]

**Returns:**

the amount member of the object

Definition at line 48 of file `asset.h`.

References `_FlowAmountInPercent`, and `Real`.

### 6.16.3.2 BusinessDayConvention `flowSchedule::getBusDayConv` () [inline]

**Returns:**

the bus day convention member of the object

Definition at line 51 of file `asset.h`.

References `_businessDayConventionOnPaymentDate`, and `BusinessDayConvention`.

### 6.16.3.3 Date `flowSchedule::getDate` () [inline]

**Returns:**

the date member of the object

Definition at line 45 of file `asset.h`.

References `_dateOfFlowPayment`.

### 6.16.3.4 void `flowSchedule::setAmount` (Real *percent*) [inline]

Sets an amount in the schedule.

Definition at line 39 of file `asset.h`.

References `_FlowAmountInPercent`, and `Real`.

**6.16.3.5 void flowSchedule::setBusDayConv (BusinessDayConvention *bd*) [inline]**

Sets a bus day convention in the schedule.

Definition at line 42 of file asset.h.

References `_businessDayConventionOnPaymentDate`.

**6.16.3.6 void flowSchedule::setDate (Date *date*) [inline]**

Sets a date in the schedule.

Definition at line 36 of file asset.h.

References `_dateOfFlowPayment`.

**6.16.4 Member Data Documentation****6.16.4.1 BusinessDayConvention flowSchedule::\_businessDayConventionOnPaymentDate [private]**

Definition at line 24 of file asset.h.

Referenced by `flowSchedule()`, `getBusDayConv()`, and `setBusDayConv()`.

**6.16.4.2 Date flowSchedule::\_dateOfFlowPayment [private]**

Definition at line 22 of file asset.h.

Referenced by `flowSchedule()`, `getDate()`, and `setDate()`.

**6.16.4.3 Real flowSchedule::\_FlowAmountInPercent [private]**

Definition at line 23 of file asset.h.

Referenced by `flowSchedule()`, `getAmount()`, and `setAmount()`.

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

- **asset.h**
- **asset.cpp**

## 6.17 GaussianProcess Class Reference

```
#include <GaussianProcess.h>
```

### Public Member Functions

- **GaussianProcess** (const valarray< **LongInteger** > schedule, const **LongNatural** &nDates, const **Real** &initialRate, const valarray< **Real** > drift, const **Real** &meanReversionSpeed, **Real** &vol)
 

*Default constructor.*
- **GaussianProcess** (const valarray< **LongInteger** > schedule, const **LongNatural** &nDates, const **Real** &initialRate, const valarray< **Real** > drift, const **Real** &meanReversionSpeed, **volsurface** \*vol, **Real** strike)
- **GaussianProcess** (void)
- **~GaussianProcess** ()
- valarray< **Real** > **BuildPath** (valarray< **Real** > gaussianShocks)
 

*Build the Path according to given gaussian shocks.*
- **Real BuildTerminalPoint** (**Real** gaussianShock)
 

*For a 1D, only one terminal Point (non path dependant).*
- void **GetStepIncrements** (valarray< **Real** > stepIncrements)
 

*Return Step Increments.*

### Private Attributes

- valarray< **LongInteger** > m\_vDates
- valarray< **Real** > m\_vDrift
- valarray< **Real** > m\_vStepSize
- **Real** m\_dbMeanReversionSpeed
- **Real** m\_dbVol
- **volsurface** \* \_vol
- **Real** \_strike
- **Real** m\_dbInitialRate
- **LongNatural** m\_nDates

#### 6.17.1 Constructor & Destructor Documentation

- 6.17.1.1 **GaussianProcess::GaussianProcess** (const valarray< **LongInteger** > schedule, const **LongNatural** & nDates, const **Real** & initialRate, const valarray< **Real** > drift, const **Real** & meanReversionSpeed, **Real** & vol)

Default constructor.

#### Parameters:

*schedule*: Start **Date**(p. 71) of the drift

*nDates*: Number of dates to simulate the path



***initialRate:***

***pyieldCurve:*** Yield Curve to compute the spot rate between each date

***pvolsurface:*** Volatility Surface to compute the forward volatility between each date

***Strike:*** Strike of the option, used to compute the colatility (function of the strike)

Definition at line 7 of file GaussianProcess.cpp.

References LongNatural, m\_dbInitialRate, m\_dbMeanReversionSpeed, m\_dbVol, m\_vDates, m\_vDrift, m\_vStepSize, Natural, and Real.

**6.17.1.2 GaussianProcess::GaussianProcess (const valarray< LongInteger > schedule, const LongNatural & nDates, const Real & initialRate, const valarray< Real > drift, const Real & meanReversionSpeed, volsurface \* vol, Real strike)**

Definition at line 28 of file GaussianProcess.cpp.

References \_strike, LongNatural, m\_dbInitialRate, m\_dbMeanReversionSpeed, m\_dbVol, m\_vDates, m\_vDrift, m\_vStepSize, Natural, and Real.

**6.17.1.3 GaussianProcess::GaussianProcess (void)**

Definition at line 52 of file GaussianProcess.cpp.

**6.17.1.4 GaussianProcess::~~GaussianProcess ()**

Definition at line 56 of file GaussianProcess.cpp.

## 6.17.2 Member Function Documentation

**6.17.2.1 valarray< Real > GaussianProcess::BuildPath (valarray< Real > gaussianShocks)**

Build the Path according to given gaussian shocks.

Definition at line 60 of file GaussianProcess.cpp.

References \_strike, LongNatural, m\_dbInitialRate, m\_dbMeanReversionSpeed, m\_dbVol, m\_vDates, m\_vDrift, Real, and volsurface::volatility().

Referenced by MCEngine::RunEngineAsianCall(), MCEngine::RunEngineAsianPut(), MCEngine::RunEngineBarrierCall(), MCEngine::RunEngineBarrierPut(), MCEngine::RunEngineCall(), MCEngine::RunEngineCappedCliquet(), MCEngine::RunEngineFlooredCliquet(), MCEngine::RunEnginePut(), MCEngine::RunEngineRevLookbackCall(), and MCEngine::RunEngineRevLookbackPut().

**6.17.2.2 Real GaussianProcess::BuildTerminalPoint (Real gaussianShock)**

For a 1D, only one terminal Point (non path dependant).

**Author:**

Yann

**Parameters:**

*gaussianShock*: the shock that we'll get from the sample given drift, etc

Definition at line 86 of file GaussianProcess.cpp.

References m\_dbInitialRate, m\_dbVol, m\_vDates, m\_vDrift, and Real.

### 6.17.2.3 void GaussianProcess::GetStepIncrements (valarray< Real > *stepIncrements*)

Return Step Increments.

Definition at line 95 of file GaussianProcess.cpp.

References LongNatural, and m\_vStepSize.

Referenced by mainmc().

## 6.17.3 Member Data Documentation

### 6.17.3.1 Real GaussianProcess::\_strike [private]

Definition at line 50 of file GaussianProcess.h.

Referenced by BuildPath(), and GaussianProcess().

### 6.17.3.2 volsurface\* GaussianProcess::\_vol [private]

Definition at line 49 of file GaussianProcess.h.

### 6.17.3.3 Real GaussianProcess::m\_dbInitialRate [private]

Definition at line 51 of file GaussianProcess.h.

Referenced by BuildPath(), BuildTerminalPoint(), and GaussianProcess().

### 6.17.3.4 Real GaussianProcess::m\_dbMeanReversionSpeed [private]

Definition at line 47 of file GaussianProcess.h.

Referenced by BuildPath(), and GaussianProcess().

### 6.17.3.5 Real GaussianProcess::m\_dbVol [private]

Definition at line 48 of file GaussianProcess.h.

Referenced by BuildPath(), BuildTerminalPoint(), and GaussianProcess().

### 6.17.3.6 LongNatural GaussianProcess::m\_nDates [private]

Definition at line 52 of file GaussianProcess.h.

**6.17.3.7** `valarray<LongInteger> GaussianProcess::m_vDates` [private]

Definition at line 43 of file GaussianProcess.h.

Referenced by `BuildPath()`, `BuildTerminalPoint()`, and `GaussianProcess()`.

**6.17.3.8** `valarray<Real> GaussianProcess::m_vDrift` [private]

Definition at line 44 of file GaussianProcess.h.

Referenced by `BuildPath()`, `BuildTerminalPoint()`, and `GaussianProcess()`.

**6.17.3.9** `valarray<Real> GaussianProcess::m_vStepSize` [private]

Definition at line 46 of file GaussianProcess.h.

Referenced by `GaussianProcess()`, and `GetStepIncrements()`.

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

- `GaussianProcess.h`
- `GaussianProcess.cpp`

## 6.18 importData Class Reference

```
#include <importData.h>
```

### Public Member Functions

- **importData** (void)  
*constructor with a reference to local files - need for default data import*
- **importData** (char \*argv)  
*constructor with a reference to local files - need for default data import*
- **bool runInterface** ()  
*default user interface for data import*
- **Natural displayFileFormatsMenu** ()  
*Help output on the file format needed to import.*
- **bool runUserDefinedInterface** ()  
*User interface to input the data files.*
- **void importYieldCurve** (string path="yctest.csv")  
*Import the yield curve from the file.*
- **void importVolSurface** (string path="voltest2.csv", Real spot=2994.0)  
*Import the call/put points from the file.*
- **void importCreditCurve** (string path="ccspread.csv")  
*Import the credit spreads from the file.*
- **marketData getData** ()
- **yieldCurve getYieldCurve** ()
- **creditCurve getCreditCurve** ()
- **volsurface getVolatilitySurface** ()

### Private Member Functions

- **void setMarketData** ()

### Private Attributes

- **string \_datadir**
- **yieldCurve \_yc**
- **creditCurve \_cc**
- **volsurface \_vs**
- **marketData \_marketData**

## 6.18.1 Constructor & Destructor Documentation

### 6.18.1.1 importData::importData (void)

constructor with a reference to local files - need for default data import  
Definition at line 3 of file importData.cpp.

### 6.18.1.2 importData::importData (char \* argv)

constructor with a reference to local files - need for default data import  
Definition at line 7 of file importData.cpp.  
References runInterface(), and FileReader::setdatadir().

## 6.18.2 Member Function Documentation

### 6.18.2.1 Natural importData::displayFileFormatsMenu ()

Help output on the file format needed to import.  
Definition at line 132 of file importData.cpp.  
References Natural.  
Referenced by runInterface().

### 6.18.2.2 creditCurve importData::getCreditCurve () [inline]

Definition at line 57 of file importData.h.  
References \_cc.

### 6.18.2.3 marketData importData::getData () [inline]

Definition at line 55 of file importData.h.  
References \_marketData.  
Referenced by inputExoticOptionOnSingleAsset().

### 6.18.2.4 volsurface importData::getVolatilitySurface () [inline]

Definition at line 58 of file importData.h.  
References \_vs.

### 6.18.2.5 yieldCurve importData::getYieldCurve () [inline]

Definition at line 56 of file importData.h.  
References \_yc.

**6.18.2.6 void importData::importCreditCurve (string *path* = "ccspread.csv")**

Import the credit spreads from the file.

Definition at line 36 of file importData.cpp.

References `_cc`, `_yc`, and `FileReader::buildCreditSpreadPointArray()`.

Referenced by `runInterface()`, and `runUserDefinedInterface()`.

**6.18.2.7 void importData::importVolSurface (string *path* = "voltest2.csv", Real *spot* = 2994.0)**

Import the call/put points from the file.

Definition at line 23 of file importData.cpp.

References `_vs`, `_yc`, `FileReader::buildVolSurfaceParams()`, `Real`, `Date::setDateToToday()`, and `volsurface::setvolsurface()`.

Referenced by `runInterface()`, and `runUserDefinedInterface()`.

**6.18.2.8 void importData::importYieldCurve (string *path* = "yctest.csv")**

Import the yield curve from the file.

Definition at line 13 of file importData.cpp.

References `_yc`, and `FileReader::buildYieldPointArray()`.

Referenced by `runInterface()`, and `runUserDefinedInterface()`.

**6.18.2.9 bool importData::runInterface ()**

default user interface for data import

Definition at line 45 of file importData.cpp.

References `displayFileFormatsMenu()`, `FileReader::getdatadirasString()`, `importCreditCurve()`, `importVolSurface()`, `importYieldCurve()`, `Natural`, `runUserDefinedInterface()`, and `setMarketData()`.

Referenced by `importData()`.

**6.18.2.10 bool importData::runUserDefinedInterface ()**

User interface to input the data files.

Definition at line 80 of file importData.cpp.

References `FileReader::fileexists()`, `FileReader::getdatadirasString()`, `importCreditCurve()`, `importVolSurface()`, `importYieldCurve()`, `Real`, and `setMarketData()`.

Referenced by `inputExoticOptionOnSingleAsset()`, and `runInterface()`.

**6.18.2.11 void importData::setMarketData () [private]**

Definition at line 74 of file importData.cpp.

References `_cc`, `_marketData`, `_vs`, `_yc`, `marketData::creditcurve`, `marketData::vols`, and `marketData::yieldcurve`.

Referenced by `runInterface()`, and `runUserDefinedInterface()`.

### 6.18.3 Member Data Documentation

#### 6.18.3.1 `creditCurve importData::_cc` [private]

Definition at line 32 of file `importData.h`.

Referenced by `getCreditCurve()`, `importCreditCurve()`, and `setMarketData()`.

#### 6.18.3.2 `string importData::_datadir` [private]

Definition at line 30 of file `importData.h`.

#### 6.18.3.3 `marketData importData::_marketData` [private]

Definition at line 34 of file `importData.h`.

Referenced by `getData()`, and `setMarketData()`.

#### 6.18.3.4 `volsurface importData::_vs` [private]

Definition at line 33 of file `importData.h`.

Referenced by `getVolatilitySurface()`, `importVolSurface()`, and `setMarketData()`.

#### 6.18.3.5 `yieldCurve importData::_yc` [private]

Definition at line 31 of file `importData.h`.

Referenced by `getYieldCurve()`, `importCreditCurve()`, `importVolSurface()`, `importYieldCurve()`, and `setMarketData()`.

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

- `importData.h`
- `importData.cpp`

## 6.19 interpolator Class Reference

```
#include <interpolator.h>
```

### Public Member Functions

- **interpolator ()**  
*Default Constructor.*
- **interpolator (valarray< Real > x, valarray< Real > y)**  
*Constructor for 1 dimension.*
- **interpolator (valarray< Real > x1, valarray< Real > x2, valarray< valarray< Real >  
> ymat)**  
*Constructor for 2 dimensions.*
- **Real interpolate (Real x)**  
*Interpolate for point x in dimension 1.*
- **Real interpolate (Real x1, Real x2)**  
*Interpolate for point (x1, x2) in dimension 2.*
- **valarray< Real > interpolate (valarray< Real > vec)**  
*Interpolate for all points x in vec in dimension 1.*
- **valarray< valarray< Real > > interpolate (valarray< Real > vec1, valarray< Real >  
vec2)**  
*Interpolate for all points (x1, x2) in (vec1, vec2) in dimension 2.*
- **Real getInterpolation (valarray< Real > xa, valarray< Real > ya, Real x)**
- **Integer getPlace (Real x)**
- **Integer getPlaceOnXi (Real x, Integer i)**

### Private Attributes

- **valarray< Real > \_x**
- **valarray< Real > \_y**
- **valarray< Real > \_x1**
- **valarray< Real > \_x2**
- **valarray< valarray< Real > > \_ymat**

### 6.19.1 Constructor & Destructor Documentation

#### 6.19.1.1 interpolator::interpolator ()

Default Constructor.

Definition at line 6 of file interpolator.cpp.

Referenced by interpolate().



**6.19.1.2 interpolator::interpolator (valarray< Real > *x*, valarray< Real > *y*)**

Constructor for 1 dimension.

**Parameters:**

*x*: array of points

*y*: array, f(x)

Definition at line 8 of file interpolator.cpp.

**6.19.1.3 interpolator::interpolator (valarray< Real > *x1*, valarray< Real > *x2*, valarray< valarray< Real > > *y*mat)**

Constructor for 2 dimensions.

**Parameters:**

*x1*: array of first components of points

*x2*: array of second components of points

*y*: matrix, f(x1, x2)

Definition at line 15 of file interpolator.cpp.

**6.19.2 Member Function Documentation****6.19.2.1 Real interpolator::getInterpolation (valarray< Real > *xa*, valarray< Real > *ya*, Real *x*)**

Definition at line 22 of file interpolator.cpp.

References Integer, and Real.

Referenced by interpolate().

**6.19.2.2 Integer interpolator::getPlace (Real *x*)**

Definition at line 46 of file interpolator.cpp.

References *\_x*, Integer, and Real.

Referenced by interpolate().

**6.19.2.3 Integer interpolator::getPlaceOnXi (Real *x*, Integer *i*)**

Definition at line 55 of file interpolator.cpp.

References *\_x1*, *\_x2*, Integer, and Real.

Referenced by interpolate().

#### 6.19.2.4 `valarray< valarray< Real > > interpolator::interpolate (valarray< Real > vec1, valarray< Real > vec2)`

Interpolate for all points (x1, x2) in (vec1, vec2) in dimension 2.

Definition at line 179 of file interpolator.cpp.

References Integer, and interpolate().

#### 6.19.2.5 `valarray< Real > interpolator::interpolate (valarray< Real > vec)`

Interpolate for all points x in vec in dimension 1.

Definition at line 35 of file interpolator.cpp.

References Integer, and interpolate().

#### 6.19.2.6 `Real interpolator::interpolate (Real x1, Real x2)`

Interpolate for point (x1, x2) in dimension 2.

Definition at line 103 of file interpolator.cpp.

References `_x1`, `_x2`, `_ymat`, `getInterpolation()`, `getPlaceOnXi()`, Integer, `interpolate()`, `interpolator()`, M, N, and Real.

#### 6.19.2.7 `Real interpolator::interpolate (Real x)`

Interpolate for point x in dimension 1.

Definition at line 75 of file interpolator.cpp.

References `_x`, `_y`, `getInterpolation()`, `getPlace()`, Integer, and Real.

Referenced by `interpolate()`, `interpolatormain()`, `maininterpolator()`, and `volsurface::volatility()`.

### 6.19.3 Member Data Documentation

#### 6.19.3.1 `valarray<Real> interpolator::_x [private]`

Definition at line 26 of file interpolator.h.

Referenced by `getPlace()`, and `interpolate()`.

#### 6.19.3.2 `valarray<Real> interpolator::_x1 [private]`

Definition at line 30 of file interpolator.h.

Referenced by `getPlaceOnXi()`, and `interpolate()`.

#### 6.19.3.3 `valarray<Real> interpolator::_x2 [private]`

Definition at line 31 of file interpolator.h.

Referenced by `getPlaceOnXi()`, and `interpolate()`.

**6.19.3.4** `valarray<Real> interpolator::_y` [private]

Definition at line 27 of file interpolator.h.

Referenced by `interpolate()`.

**6.19.3.5** `valarray<valarray<Real> > interpolator::_ymat` [private]

Definition at line 32 of file interpolator.h.

Referenced by `interpolate()`.

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

- `interpolator.h`
- `interpolator.cpp`

## 6.20 marketData Struct Reference

```
#include <importData.h>
```

### Public Attributes

- **yieldCurve** yieldcurve
- **creditCurve** creditcurve
- **volsurface** vols

### 6.20.1 Member Data Documentation

#### 6.20.1.1 creditCurve marketData::creditcurve

Definition at line 24 of file importData.h.

Referenced by inputBond(), inputConvertibleBond(), and importData::setMarketData().

#### 6.20.1.2 volsurface marketData::vols

Definition at line 25 of file importData.h.

Referenced by inputBSOption(), inputButterflySpread(), inputCallSpread(), inputExoticOptionOnSingleAsset(), inputPutSpread(), inputRainbowOption(), inputRatioCallSpread(), inputStraddle(), inputStrangle(), and importData::setMarketData().

#### 6.20.1.3 yieldCurve marketData::yieldcurve

Definition at line 23 of file importData.h.

Referenced by inputBond(), inputBSOption(), inputButterflySpread(), inputCallSpread(), inputConvertibleBond(), inputExoticOptionOnSingleAsset(), inputPutSpread(), inputRainbowOption(), inputRatioCallSpread(), inputStraddle(), inputStrangle(), inputVanillaSwap(), and importData::setMarketData().

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

- **importData.h**

## 6.21 Matrix Class Reference

```
#include <matrix.h>
```

### Public Member Functions

- **Matrix** ()
- **Matrix** (double InitVal, int Rows, int Cols)
- **Matrix** (double \*Data, int Rows, int Cols)
- **Matrix** (double \*\*Data, int Rows, int Cols)
- **Matrix** (const **Matrix** &obj)
- **~Matrix** ()
- **Matrix** & **operator+** (const **Matrix** &obj) const
- **Matrix** & **operator-** (const **Matrix** &obj) const
- **Matrix** & **operator \*** (const **Matrix** &obj) const
- **Matrix** & **operator \*** (const double \_d) const
- **Matrix** & **operator \*** (const int \_i) const
- **Matrix** & **operator/** (const **Matrix** &obj) const
- **Matrix** & **operator/** (const double \_d) const
- **Matrix** & **operator/** (const int \_i) const
- **Matrix** & **operator+=** (const **Matrix** &obj)
- **Matrix** & **operator-=** (const **Matrix** &obj)
- **Matrix** & **operator \*=** (const **Matrix** &obj)
- **Matrix** & **operator \*=** (const double \_d)
- **Matrix** & **operator \*=** (const int \_i)
- **Matrix** & **operator/=** (const **Matrix** &obj)
- **Matrix** & **operator/=** (const double \_d)
- **Matrix** & **operator/=** (const int \_i)
- **Matrix** & **operator=** (const **Matrix** &obj)
- **Matrix** & **operator~** () const
- bool **operator==** (const **Matrix** &obj) const
- bool **operator!=** (const **Matrix** &obj) const
- double \* **operator[]** (const int \_i) const
- double & **operator()** (const int \_i, const int \_j) const
- bool **IsIdentity** () const
- bool **IsEmpty** () const
- double **Determinant** () const
- double **SumAll** () const
- double **SumAllSquared** () const
- double **SumRow** (const int Row) const
- double **SumColumn** (const int Col) const
- double **SumRowSquared** (const int Row) const
- double **SumColumnSquared** (const int Col) const
- double **GetMax** () const
- double **GetMin** () const
- double **GetRowMax** (const int Row) const
- double **GetRowMin** (const int Row) const
- double **GetColumnMax** (const int Col) const
- double **GetColumnMin** (const int Col) const
- double **GetRange** () const

- double **GetRowRange** (const int Row) const
- double **GetColumnRange** (const int Col) const
- double \* **GetDataOneDimen** () const
- double \*\* **GetDataTwoDimen** () const
- int **GetRows** () const
- int **GetColumns** () const
- **Matrix** & **Clear** ()
- **Matrix** & **ClearRow** (const int Row)
- **Matrix** & **ClearColumn** (const int Col)
- **Matrix** & **SetValue** (int Row, int Col, double \_d)
- **Matrix** & **Fill** (const double \_d)
- **Matrix** & **FillRow** (const int Row, const double \_d)
- **Matrix** & **FillColumn** (const int Col, const double \_d)
- **Matrix** & **GetInverse** () const
- **Matrix** & **Invert** ()
- **Matrix** & **AddRows** (const int SourceRow, const int DestRow, const double factor=1)
- **Matrix** & **MultiplyRow** (const int Row, const double \_d)
- **Matrix** & **DivideRow** (const int Row, const double \_d)
- **Matrix** & **AddColumns** (const int SourceCol, const int DestCol, const double factor=1)
- **Matrix** & **MultiplyColumn** (const int Col, const double \_d)
- **Matrix** & **DivideColumn** (const int Col, const double \_d)
- **Matrix** & **REF** ()
- **Matrix** & **RREF** ()
- **Matrix** & **GetREF** () const
- **Matrix** & **GetRREF** () const
- **Matrix** & **CholeskyDecomposition** ()

*Cholesky decomposition I added.*

- **Matrix** & **GetMinor** (const int RowSpot, const int ColSpot) const
- **Matrix** \* **GetMinorNew** (const int RowSpot, const int ColSpot) const
- **Matrix** & **GetSubMatrix** (const int RowSpot, const int ColSpot, const int RowLen, const int ColLen) const
- **Matrix** & **SetSubMatrix** (const int RowSpot, const int ColSpot, const int RowLen, const int ColLen)
- **Matrix** & **SwapRows** (const int Row1, const int Row2)
- **Matrix** & **SwapCols** (const int Col1, const int Col2)
- **Matrix** & **GetTransposed** () const
- **Matrix** & **Transpose** ()
- **Matrix** & **GetNumericRange** (double &Min, double &Max) const
- **Matrix** & **GetNumericRangeOfRow** (double &Min, double &Max, const int Row) const
- **Matrix** & **GetNumericRangeOfColumn** (double &Min, double &Max, const int Col) const
- **Matrix** & **CMAR** (const **Matrix** &obj)
- **Matrix** & **CMAC** (const **Matrix** &obj)
- **Matrix** & **GetCMAR** (const **Matrix** &obj) const
- **Matrix** & **GetCMAC** (const **Matrix** &obj) const
- **Matrix** & **ConcatenateRow** (const double \*RowData)
- **Matrix** & **ConcatenateColumn** (const double \*ColumnData)
- **Matrix** & **SpliceInRow** (const double \*RowData, const int RowSpot)
- **Matrix** & **SpliceInColumn** (const double \*ColumnData, const int ColumnSpot)

- **Matrix & RemoveRow** (const int Row)
- **Matrix & RemoveColumn** (const int Column)
- **Matrix & SortAscend** ()
- **Matrix & SortDescend** ()
- **Matrix & GetNormalized** (const double Min, const double Max) const
- **Matrix & Normalize** (const double Min, const double Max)
- **Matrix & GetCovariant** () const
- **Matrix & MakeCovariant** ()
- void **Display** () const
- void **Output** (ostream &ostr=cout) const
- void **Input** (istream &istr=cin)
- void **Read** (ifstream &istr)
- void **Write** (ofstream &ostr) const

## Static Public Member Functions

- **Matrix & IdentityMatrix** (int Diagonal)

## Private Member Functions

- **Matrix & RightAppendIdentity** ()
- **Matrix & LeftRemoveIdentity** ()

## Private Attributes

- double \*\* **m\_pData**
- int **m\_nCols**
- int **m\_nRows**

### 6.21.1 Detailed Description

#### Author:

Yann

Definition at line 19 of file matrix.h.

### 6.21.2 Constructor & Destructor Documentation

#### 6.21.2.1 Matrix::Matrix ()

Definition at line 55 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `CholeskyDecomposition()`, `GetMinor()`, `GetMinorNew()`, `GetSubMatrix()`, `GetTransposed()`, `IdentityMatrix()`, `LeftRemoveIdentity()`, `operator*()`, `operator+()`, `operator-()`, `operator/()`, `RightAppendIdentity()`, `SortAscend()`, and `SortDescend()`.

**6.21.2.2 Matrix::Matrix (double *InitVal*, int *Rows*, int *Cols*)**

Definition at line 64 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.2.3 Matrix::Matrix (double \* *Data*, int *Rows*, int *Cols*)**

Definition at line 81 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.2.4 Matrix::Matrix (double \*\* *Data*, int *Rows*, int *Cols*)**

Definition at line 98 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.2.5 Matrix::Matrix (const Matrix & *obj*)**

Definition at line 113 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.2.6 Matrix::~Matrix ()**

Definition at line 128 of file matrix.cpp.

References m\_nRows, and m\_pData.

**6.21.3 Member Function Documentation****6.21.3.1 Matrix & Matrix::AddColumns (const int *SourceCol*, const int *DestCol*, const double *factor* = 1)**

Definition at line 794 of file matrix.cpp.

References m\_nRows, and m\_pData.

**6.21.3.2 Matrix & Matrix::AddRows (const int *SourceRow*, const int *DestRow*, const double *factor* = 1)**

Definition at line 762 of file matrix.cpp.

References m\_nCols, and m\_pData.

Referenced by REF(), and RREF().

**6.21.3.3 Matrix & Matrix::CholeskyDecomposition ()**

Cholesky decomposition I added.

Definition at line 1062 of file matrix.cpp.



References `GetRows()`, `m_pData`, `Matrix()`, and `Real`.

Referenced by `mainmatrix()`, `MCEngine::RunEngineRainbow2AssetsBasketMax()`, `MCEngine::RunEngineRainbow2SpreadOptionMax()`, `MCEngine::RunEngineRainbowBestOf2AssetsCash()`, `MCEngine::RunEngineRainbowMax2AssetsCall()`, `MCEngine::RunEngineRainbowMax2AssetsPut()`, `MCEngine::RunEngineRainbowMin2AssetsCall()`, `MCEngine::RunEngineRainbowMin2AssetsPut()`, and `MCEngine::RunEngineRainbowWorstOf2AssetsCash()`.

#### 6.21.3.4 `Matrix & Matrix::Clear ()`

Definition at line 664 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, and `m_pData`.

#### 6.21.3.5 `Matrix & Matrix::ClearColumn (const int Col)`

Definition at line 686 of file `matrix.cpp`.

References `m_nRows`, and `m_pData`.

#### 6.21.3.6 `Matrix & Matrix::ClearRow (const int Row)`

Definition at line 676 of file `matrix.cpp`.

References `m_nCols`, and `m_pData`.

#### 6.21.3.7 `Matrix & Matrix::CMAC (const Matrix & obj)`

Definition at line 1118 of file `matrix.cpp`.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `GetCMAC()`.

#### 6.21.3.8 `Matrix & Matrix::CMAR (const Matrix & obj)`

Definition at line 1094 of file `matrix.cpp`.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `GetCMAR()`.

#### 6.21.3.9 `Matrix & Matrix::ConcatenateColumn (const double * ColumnData)`

Definition at line 1183 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, and `m_pData`.

#### 6.21.3.10 `Matrix & Matrix::ConcatenateRow (const double * RowData)`

Definition at line 1163 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.11 double Matrix::Determinant () const**

Definition at line 430 of file matrix.cpp.

References `ErrorMsg()`, `GetMinorNew()`, `m_nCols`, `m_nRows`, `m_pData`, and `q`.

**6.21.3.12 void Matrix::Display () const**

Definition at line 1379 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.13 Matrix & Matrix::DivideColumn (const int *Col*, const double *\_d*)**

Definition at line 814 of file matrix.cpp.

References `ErrorMsg()`, `m_nRows`, and `m_pData`.

**6.21.3.14 Matrix & Matrix::DivideRow (const int *Row*, const double *\_d*)**

Definition at line 782 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, and `m_pData`.

Referenced by `REF()`, and `RREF()`.

**6.21.3.15 Matrix & Matrix::Fill (const double *\_d*)**

Definition at line 705 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.16 Matrix & Matrix::FillColumn (const int *Col*, const double *\_d*)**

Definition at line 727 of file matrix.cpp.

References `m_nRows`, and `m_pData`.

**6.21.3.17 Matrix & Matrix::FillRow (const int *Row*, const double *\_d*)**

Definition at line 717 of file matrix.cpp.

References `m_nCols`, and `m_pData`.

**6.21.3.18 Matrix & Matrix::GetCMAC (const Matrix & *obj*) const**

Definition at line 1152 of file matrix.cpp.

References `CMAC()`.

**6.21.3.19 Matrix & Matrix::GetCMAR (const Matrix & *obj*) const**

Definition at line 1140 of file matrix.cpp.

References CMAR().

#### **6.21.3.20 double Matrix::GetColumnMax (const int *Col*) const**

Definition at line 564 of file matrix.cpp.

References m\_nRows, and m\_pData.

#### **6.21.3.21 double Matrix::GetColumnMin (const int *Col*) const**

Definition at line 576 of file matrix.cpp.

References m\_nRows, and m\_pData.

#### **6.21.3.22 double Matrix::GetColumnRange (const int *Col*) const**

Definition at line 608 of file matrix.cpp.

References GetNumericRangeOfColumn().

#### **6.21.3.23 int Matrix::GetColumns () const**

Definition at line 658 of file matrix.cpp.

References m\_nCols.

Referenced by operator<<().

#### **6.21.3.24 Matrix & Matrix::GetCovariant () const**

Definition at line 1359 of file matrix.cpp.

References Transpose().

Referenced by MakeCovariant().

#### **6.21.3.25 double \* Matrix::GetDataOneDimen () const**

Definition at line 619 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

Referenced by SortAscend(), and SortDescend().

#### **6.21.3.26 double \*\* Matrix::GetDataTwoDimen () const**

Definition at line 636 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

#### **6.21.3.27 Matrix & Matrix::GetInverse () const**

Definition at line 737 of file matrix.cpp.

References `ErrorMsg()`, `LeftRemoveIdentity()`, `m_nCols`, `m_nRows`, `RightAppendIdentity()`, and `RREF()`.

Referenced by `Invert()`, `operator/()`, and `operator~()`.

#### **6.21.3.28 double Matrix::GetMax () const**

Definition at line 512 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, and `m_pData`.

#### **6.21.3.29 double Matrix::GetMin () const**

Definition at line 526 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, and `m_pData`.

#### **6.21.3.30 Matrix & Matrix::GetMinor (const int *RowSpot*, const int *ColSpot*) const**

Definition at line 877 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

#### **6.21.3.31 Matrix \* Matrix::GetMinorNew (const int *RowSpot*, const int *ColSpot*) const**

Definition at line 905 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

Referenced by `Determinant()`.

#### **6.21.3.32 Matrix & Matrix::GetNormalized (const double *Min*, const double *Max*) const**

Definition at line 1325 of file `matrix.cpp`.

References `Normalize()`.

#### **6.21.3.33 Matrix & Matrix::GetNumericRange (double & *Min*, double & *Max*) const**

Definition at line 1020 of file `matrix.cpp`.

References `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `GetRange()`, and `Normalize()`.

#### **6.21.3.34 Matrix & Matrix::GetNumericRangeOfColumn (double & *Min*, double & *Max*, const int *Col*) const**

Definition at line 1048 of file `matrix.cpp`.

References `m_nRows`, and `m_pData`.

Referenced by `GetColumnRange()`.

#### 6.21.3.35 `Matrix & Matrix::GetNumericRangeOfRow (double & Min, double & Max, const int Row) const`

Definition at line 1035 of file `matrix.cpp`.

References `m_nCols`, and `m_pData`.

Referenced by `GetRowRange()`.

#### 6.21.3.36 `double Matrix::GetRange () const`

Definition at line 588 of file `matrix.cpp`.

References `GetNumericRange()`.

#### 6.21.3.37 `Matrix & Matrix::GetREF () const`

Definition at line 854 of file `matrix.cpp`.

References `REF()`.

#### 6.21.3.38 `double Matrix::GetRowMax (const int Row) const`

Definition at line 540 of file `matrix.cpp`.

References `m_nCols`, and `m_pData`.

#### 6.21.3.39 `double Matrix::GetRowMin (const int Row) const`

Definition at line 552 of file `matrix.cpp`.

References `m_nCols`, and `m_pData`.

#### 6.21.3.40 `double Matrix::GetRowRange (const int Row) const`

Definition at line 598 of file `matrix.cpp`.

References `GetNumericRangeOfRow()`.

#### 6.21.3.41 `int Matrix::GetRows () const`

Definition at line 652 of file `matrix.cpp`.

References `m_nRows`.

Referenced by `CholeskyDecomposition()`, `RainbowOption::getCorrelRisk()`, `operator<<()`, `RainbowOption::RainbowOption()`, `MCEngine::RunEngineRainbow2AssetsBasketMax()`, `MCEngine::RunEngineRainbow2SpreadOptionMax()`, `MCEngine::RunEngineRainbowBestOf2AssetsCash()`, `MCEngine::RunEngineRainbowMax2AssetsCall()`, `MCEngine::RunEngineRainbowMax2AssetsPut()`, `MCEngine::RunEngineRainbowMin2AssetsCall()`, `MCEngine::RunEngineRainbowMin2AssetsPut()`, and `MCEngine::RunEngineRainbowWorstOf2AssetsCash()`.

**6.21.3.42 Matrix & Matrix::GetRREF () const**

Definition at line 865 of file matrix.cpp.

References RREF().

**6.21.3.43 Matrix & Matrix::GetSubMatrix (const int *RowSpot*, const int *ColSpot*, const int *RowLen*, const int *ColLen*) const**

Definition at line 933 of file matrix.cpp.

References m\_pData, and Matrix().

Referenced by SetSubMatrix().

**6.21.3.44 Matrix & Matrix::GetTransposed () const**

Definition at line 997 of file matrix.cpp.

References m\_nCols, m\_nRows, m\_pData, and Matrix().

Referenced by mainmatrix(), and Transpose().

**6.21.3.45 Matrix & Matrix::IdentityMatrix (int *Diagonal*) [static]**

Definition at line 1452 of file matrix.cpp.

References m\_pData, Matrix(), and q.

Referenced by IdentityMatrix().

**6.21.3.46 void Matrix::Input (istream & *istr* = cin)**

Definition at line 1406 of file matrix.cpp.

References getint(), m\_nCols, m\_nRows, and m\_pData.

**6.21.3.47 Matrix & Matrix::Invert ()**

Definition at line 754 of file matrix.cpp.

References GetInverse().

**6.21.3.48 bool Matrix::IsEmpty () const**

Definition at line 418 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.3.49 bool Matrix::IsIdentity () const**

Definition at line 401 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.3.50 Matrix & Matrix::LeftRemoveIdentity () [private]**

Definition at line 37 of file matrix.cpp.

References `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

Referenced by `GetInverse()`.

**6.21.3.51 Matrix & Matrix::MakeCovariant ()**

Definition at line 1370 of file matrix.cpp.

References `GetCovariant()`.

**6.21.3.52 Matrix & Matrix::MultiplyColumn (const int *Col*, const double *\_d*)**

Definition at line 804 of file matrix.cpp.

References `m_nRows`, and `m_pData`.

**6.21.3.53 Matrix & Matrix::MultiplyRow (const int *Row*, const double *\_d*)**

Definition at line 772 of file matrix.cpp.

References `m_nCols`, and `m_pData`.

**6.21.3.54 Matrix & Matrix::Normalize (const double *Min*, const double *Max*)**

Definition at line 1336 of file matrix.cpp.

References `GetNumericRange()`, `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `GetNormalized()`.

**6.21.3.55 Matrix & Matrix::operator \* (const int *\_i*) const**

Definition at line 216 of file matrix.cpp.

References `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

**6.21.3.56 Matrix & Matrix::operator \* (const double *\_d*) const**

Definition at line 201 of file matrix.cpp.

References `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

**6.21.3.57 Matrix & Matrix::operator \* (const Matrix & *obj*) const**

Definition at line 175 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, `m_pData`, `Matrix()`, and `q`.

**6.21.3.58 Matrix & Matrix::operator \*= (const int *\_i*)**

Definition at line 295 of file matrix.cpp.

**6.21.3.59 Matrix & Matrix::operator \*= (const double *\_d*)**

Definition at line 289 of file matrix.cpp.

**6.21.3.60 Matrix & Matrix::operator \*= (const Matrix & *obj*)**

Definition at line 283 of file matrix.cpp.

**6.21.3.61 bool Matrix::operator!=(const Matrix & *obj*) const**

Definition at line 367 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.62 double & Matrix::operator() (const int *\_i*, const int *\_j*) const**

Definition at line 392 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.63 Matrix & Matrix::operator+ (const Matrix & *obj*) const**

Definition at line 139 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

**6.21.3.64 Matrix & Matrix::operator+= (const Matrix & *obj*)**

Definition at line 271 of file matrix.cpp.

**6.21.3.65 Matrix & Matrix::operator- (const Matrix & *obj*) const**

Definition at line 157 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

**6.21.3.66 Matrix & Matrix::operator-= (const Matrix & *obj*)**

Definition at line 277 of file matrix.cpp.

**6.21.3.67 Matrix & Matrix::operator/ (const int *\_i*) const**

Definition at line 254 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.



**6.21.3.68 Matrix & Matrix::operator/ (const double \_d) const**

Definition at line 237 of file matrix.cpp.

References `ErrorMsg()`, `m_nCols`, `m_nRows`, `m_pData`, and `Matrix()`.

**6.21.3.69 Matrix & Matrix::operator/ (const Matrix & obj) const**

Definition at line 231 of file matrix.cpp.

References `GetInverse()`.

**6.21.3.70 Matrix & Matrix::operator/= (const int \_i)**

Definition at line 313 of file matrix.cpp.

**6.21.3.71 Matrix & Matrix::operator/= (const double \_d)**

Definition at line 307 of file matrix.cpp.

**6.21.3.72 Matrix & Matrix::operator/= (const Matrix & obj)**

Definition at line 301 of file matrix.cpp.

**6.21.3.73 Matrix & Matrix::operator= (const Matrix & obj)**

Definition at line 319 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.74 bool Matrix::operator== (const Matrix & obj) const**

Definition at line 352 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.75 double \* Matrix::operator[] (const int \_i) const**

Definition at line 383 of file matrix.cpp.

References `ErrorMsg()`, `m_nRows`, and `m_pData`.

**6.21.3.76 Matrix & Matrix::operator~ () const**

Definition at line 346 of file matrix.cpp.

References `GetInverse()`.

**6.21.3.77 void Matrix::Output (ostream & *ostr* = cout) const**

Definition at line 1394 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.3.78 void Matrix::Read (ifstream & *istr*)**

Definition at line 1424 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.3.79 Matrix & Matrix::REF ()**

Definition at line 826 of file matrix.cpp.

References AddRows(), DivideRow(), m\_nRows, and m\_pData.

Referenced by GetREF(), and RREF().

**6.21.3.80 Matrix & Matrix::RemoveColumn (const int *Column*)**

Definition at line 1263 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.3.81 Matrix & Matrix::RemoveRow (const int *Row*)**

Definition at line 1246 of file matrix.cpp.

References m\_nCols, m\_nRows, and m\_pData.

**6.21.3.82 Matrix & Matrix::RightAppendIdentity () [private]**

Definition at line 15 of file matrix.cpp.

References m\_nCols, m\_nRows, m\_pData, Matrix(), and q.

Referenced by GetInverse().

**6.21.3.83 Matrix & Matrix::RREF ()**

Definition at line 839 of file matrix.cpp.

References AddRows(), DivideRow(), m\_nRows, m\_pData, and REF().

Referenced by GetInverse(), and GetRREF().

**6.21.3.84 Matrix & Matrix::SetSubMatrix (const int *RowSpot*, const int *ColSpot*,  
const int *RowLen*, const int *ColLen*)**

Definition at line 949 of file matrix.cpp.

References GetSubMatrix().

**6.21.3.85 Matrix & Matrix::SetValue (int *Row*, int *Col*, double *\_d*)****Author:**

Yann - I added setters for (i,j) - not clean but needed

Definition at line 696 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `RainbowOption::getCorrelRisk()`, `RainbowOption::getDelta()`, `RainbowOption::getGamma()`, `RainbowOption::getVega()`, `mainmatrix()`, `RainbowOption::RainbowOption()`, `MCEngine::RunEngineRainbow2AssetsBasketMax()`, `MCEngine::RunEngineRainbow2SpreadOptionMax()`, `MCEngine::RunEngineRainbowBestOf2AssetsCash()`, `MCEngine::RunEngineRainbowMax2AssetsCall()`, `MCEngine::RunEngineRainbowMax2AssetsPut()`, `MCEngine::RunEngineRainbowMin2AssetsCall()`, `MCEngine::RunEngineRainbowMin2AssetsPut()`, `MCEngine::RunEngineRainbowWorstOf2AssetsCash()`, `transform1DvalarrayToColumnMatrix()`, and `transform2DvalarrayToMatrix()`.

**6.21.3.86 Matrix & Matrix::SortAscend ()**

Definition at line 1281 of file matrix.cpp.

References `GetDataOneDimen()`, `m_nCols`, `m_nRows`, and `Matrix()`.

**6.21.3.87 Matrix & Matrix::SortDescend ()**

Definition at line 1303 of file matrix.cpp.

References `GetDataOneDimen()`, `m_nCols`, `m_nRows`, and `Matrix()`.

**6.21.3.88 Matrix & Matrix::SpliceInColumn (const double \* *ColumnData*, const int *ColumnSpot*)**

Definition at line 1226 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.89 Matrix & Matrix::SpliceInRow (const double \* *RowData*, const int *RowSpot*)**

Definition at line 1204 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.3.90 double Matrix::SumAll () const**

Definition at line 450 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

Referenced by `SumAllSquared()`.

**6.21.3.91 double Matrix::SumAllSquared () const**

Definition at line 464 of file matrix.cpp.

References SumAll().

**6.21.3.92 double Matrix::SumColumn (const int *Col*) const**

Definition at line 484 of file matrix.cpp.

References m\_nRows, and m\_pData.

Referenced by RainbowOption::getDelta(), RainbowOption::getGamma(), RainbowOption::getVega(), and SumColumnSquared().

**6.21.3.93 double Matrix::SumColumnSquared (const int *Col*) const**

Definition at line 504 of file matrix.cpp.

References SumColumn().

**6.21.3.94 double Matrix::SumRow (const int *Row*) const**

Definition at line 472 of file matrix.cpp.

References m\_nCols, and m\_pData.

Referenced by SumRowSquared().

**6.21.3.95 double Matrix::SumRowSquared (const int *Row*) const**

Definition at line 496 of file matrix.cpp.

References SumRow().

**6.21.3.96 Matrix & Matrix::SwapCols (const int *Col1*, const int *Col2*)**

Definition at line 979 of file matrix.cpp.

References m\_nRows, and m\_pData.

**6.21.3.97 Matrix & Matrix::SwapRows (const int *Row1*, const int *Row2*)**

Definition at line 961 of file matrix.cpp.

References m\_nCols, and m\_pData.

**6.21.3.98 Matrix & Matrix::Transpose ()**

Definition at line 1012 of file matrix.cpp.

References GetTransposed().

Referenced by GetCovariant().

**6.21.3.99 void Matrix::Write (ofstream & ostr) const**

Definition at line 1439 of file matrix.cpp.

References `m_nCols`, `m_nRows`, and `m_pData`.

**6.21.4 Member Data Documentation****6.21.4.1 int Matrix::m\_nCols [private]**

Definition at line 22 of file matrix.h.

Referenced by `AddRows()`, `Clear()`, `ClearRow()`, `CMAC()`, `CMAR()`, `ConcatenateColumn()`, `ConcatenateRow()`, `Determinant()`, `Display()`, `DivideRow()`, `Fill()`, `FillRow()`, `GetColumns()`, `GetDataOneDimen()`, `GetDataTwoDimen()`, `GetInverse()`, `GetMax()`, `GetMin()`, `GetMinor()`, `GetMinorNew()`, `GetNumericRange()`, `GetNumericRangeOfRow()`, `GetRowMax()`, `GetRowMin()`, `GetTransposed()`, `Input()`, `IsEmpty()`, `IsIdentity()`, `LeftRemoveIdentity()`, `Matrix()`, `MultiplyRow()`, `Normalize()`, `operator*()`, `operator!=()`, `operator()`, `operator+()`, `operator-()`, `operator/()`, `operator=()`, `operator==(())`, `Output()`, `Read()`, `RemoveColumn()`, `RemoveRow()`, `RightAppendIdentity()`, `SetValue()`, `SortAscend()`, `SortDescend()`, `SpliceInColumn()`, `SpliceInRow()`, `SumAll()`, `SumRow()`, `SwapRows()`, and `Write()`.

**6.21.4.2 int Matrix::m\_nRows [private]**

Definition at line 23 of file matrix.h.

Referenced by `AddColumns()`, `Clear()`, `ClearColumn()`, `CMAC()`, `CMAR()`, `ConcatenateColumn()`, `ConcatenateRow()`, `Determinant()`, `Display()`, `DivideColumn()`, `Fill()`, `FillColumn()`, `GetColumnMax()`, `GetColumnMin()`, `GetDataOneDimen()`, `GetDataTwoDimen()`, `GetInverse()`, `GetMax()`, `GetMin()`, `GetMinor()`, `GetMinorNew()`, `GetNumericRange()`, `GetNumericRangeOfColumn()`, `GetRows()`, `GetTransposed()`, `Input()`, `IsEmpty()`, `IsIdentity()`, `LeftRemoveIdentity()`, `Matrix()`, `MultiplyColumn()`, `Normalize()`, `operator*()`, `operator!=()`, `operator()`, `operator+()`, `operator-()`, `operator/()`, `operator=()`, `operator==(())`, `operator[]()`, `Output()`, `Read()`, `REF()`, `RemoveColumn()`, `RemoveRow()`, `RightAppendIdentity()`, `RREF()`, `SetValue()`, `SortAscend()`, `SortDescend()`, `SpliceInColumn()`, `SpliceInRow()`, `SumAll()`, `SumColumn()`, `SwapCols()`, `Write()`, and `~Matrix()`.

**6.21.4.3 double\*\* Matrix::m\_pData [private]**

Definition at line 21 of file matrix.h.

Referenced by `AddColumns()`, `AddRows()`, `CholeskyDecomposition()`, `Clear()`, `ClearColumn()`, `ClearRow()`, `CMAC()`, `CMAR()`, `ConcatenateColumn()`, `ConcatenateRow()`, `Determinant()`, `Display()`, `DivideColumn()`, `DivideRow()`, `Fill()`, `FillColumn()`, `FillRow()`, `GetColumnMax()`, `GetColumnMin()`, `GetDataOneDimen()`, `GetDataTwoDimen()`, `GetMax()`, `GetMin()`, `GetMinor()`, `GetMinorNew()`, `GetNumericRange()`, `GetNumericRangeOfColumn()`, `GetNumericRangeOfRow()`, `GetRowMax()`, `GetRowMin()`, `GetSubMatrix()`, `GetTransposed()`, `IdentityMatrix()`, `Input()`, `IsEmpty()`, `IsIdentity()`, `LeftRemoveIdentity()`, `Matrix()`, `MultiplyColumn()`, `MultiplyRow()`, `Normalize()`, `operator*()`, `operator!=()`, `operator()`, `operator+()`, `operator-()`, `operator/()`, `operator=()`, `operator==(())`, `operator[]()`, `Output()`, `Read()`, `REF()`, `RemoveColumn()`, `RemoveRow()`, `RightAppendIdentity()`, `RREF()`, `SetValue()`, `SpliceInColumn()`, `SpliceInRow()`, `SumAll()`, `SumColumn()`, `SumRow()`, `SwapCols()`, `SwapRows()`, `Write()`, and `~Matrix()`.

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

- `matrix.h`
- `matrix.cpp`

## 6.22 MCEngine Class Reference

```
#include <MCEngine.h>
```

### Public Member Functions

- **MCEngine** (**LongNatural** nPaths, **LongNatural** nDates, valarray< **Real** > DiscFactors)  
*Default constructor.*
- **~MCEngine** ()
- **MCEngine** (void)  
*Default constructor.*
- **MCEngine** (**LongNatural** nPaths, **Real** DFToMaturity)  
*for 1 date (non path dependant payOffs)*
- void **RunEngineRainbow2SpreadOptionMax** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation, **Real** Mult)  
*Price Spread option with 2 assets.*
- void **RunEngineRainbow2AssetsBasketMax** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation, **Real** Mult)  
*Price basker option with 2 assets.*
- void **RunEngineRainbowBestOf2AssetsCash** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation)  
*Price best of + cash option with 2 assets.*
- void **RunEngineRainbowWorstOf2AssetsCash** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation)  
*Price worst of + cash option with 2 assets.*
- void **RunEngineRainbowMax2AssetsCall** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation, **Real** Mult)  
*Price max call option with 2 assets.*
- void **RunEngineRainbowMin2AssetsCall** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation, **Real** Mult)  
*Price min call option with 2 assets.*

- void **RunEngineRainbowMax2AssetsPut** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation, **Real** Mult)
 

*Price max put option with 2 assets.*
- void **RunEngineRainbowMin2AssetsPut** (**Random** \*pRandom, valarray< **GaussianProcess** > pHazardRateProcesses, **PayOff** thePayOff, **Real** gaussianSample, valarray< **Real** > TerminalPoints, valarray< **Real** > weights, **Matrix** Correlation, **Real** Mult)
 

*Price min put option with 2 assets.*
- void **RunEngineAsianCall** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price Asian Call.*
- void **RunEngineAsianPut** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price Asian Put.*
- void **RunEngineCall** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price European standard Call.*
- void **RunEnginePut** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price uropean standard Put.*
- void **RunEngineRevLookbackCall** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price Lokback Call.*
- void **RunEngineRevLookbackPut** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price Lokback Put.*
- void **RunEngineBarrierCall** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price Barrier Call.*
- void **RunEngineBarrierPut** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)
 

*Price Barrier Put.*



- void **RunEngineFlooredCliquet** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)

*Price FlooredCliquet.*

- void **RunEngineCappedCliquet** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath)

*Price CappedCliquet.*

- void **RunEngineGeneral** (**Random** \*pRandom, **GaussianProcess** \*pHazardRateProcess, **PayOff** thePayOff, valarray< **Real** > gaussianSample, valarray< **Real** > vHazardRatePath, **Natural** Product)

*Run Monte Carlo Engine with code for the product.*

- **Real** **MCRResult** ()

*Return result of Monte carlo simulation.*

## Private Attributes

- **Real** m\_price
- **Real** m\_DiscFactor
- **LongNatural** m\_nPaths
- **LongNatural** m\_nDates

### 6.22.1 Constructor & Destructor Documentation

#### 6.22.1.1 MCEngine::MCEngine (LongNatural nPaths, LongNatural nDates, valarray< Real > DiscFactors)

Default constructor.

#### Parameters:

*price*: Start **Date**(p. 71) of the drift

*nPaths*: Length of the drift to be computed

*nDates*: Number of dates to cut the drift

*Vol*: Yield Curve to compute the spot rate between each date

*Spot*: Volatility Surface to compute the forward volatility between each date

*Strike*: Strike of the option, used to compute the colatility (function of the strike)

Definition at line 3 of file MCEngine.cpp.

References LongNatural, m\_DiscFactor, and m\_price.

#### 6.22.1.2 MCEngine::~~MCEngine ()

Definition at line 20 of file MCEngine.cpp.

**6.22.1.3 MCEngine::MCEngine (void)**

Default constructor.

**Author:**

Yann

Definition at line 24 of file MCEngine.cpp.

**6.22.1.4 MCEngine::MCEngine (LongNatural *nPaths*, Real *DFToMaturity*)**

for 1 date (non path dependant payOffs)

**Author:**

Yann

**Parameters:**

*nPaths*: to compute the MC

*DFToMaturity*: 1 date so only one Df needed

Definition at line 11 of file MCEngine.cpp.

References LongNatural, m\_price, and Real.

**6.22.2 Member Function Documentation****6.22.2.1 Real MCEngine::MCResult ()**

Return result of Monte carlo simulation.

Definition at line 353 of file MCEngine.cpp.

References m\_nPaths, m\_price, and Real.

Referenced by mainmc(), RainbowOption::PriceByMc\_2AssetsBasketMax(), RainbowOption::PriceByMc\_2SpreadOptionMax(), RainbowOption::PriceByMc\_BestOf2AssetsCash(), RainbowOption::PriceByMc\_Max2AssetsCall(), RainbowOption::PriceByMc\_Max2AssetsPut(), RainbowOption::PriceByMc\_Min2AssetsCall(), RainbowOption::PriceByMc\_Min2AssetsPut(), and RainbowOption::PriceByMc\_WorstOf2AssetsCash().

**6.22.2.2 void MCEngine::RunEngineAsianCall (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)**

Price Asian Call.

Definition at line 213 of file MCEngine.cpp.

References PayOff::AsianCall(), GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.3** void MCEngine::RunEngineAsianPut (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price Asian Put.

Definition at line 224 of file MCEngine.cpp.

References PayOff::AsianPut(), GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.4** void MCEngine::RunEngineBarrierCall (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price Barrier Call.

Definition at line 285 of file MCEngine.cpp.

References PayOff::BarrierCall(), GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.5** void MCEngine::RunEngineBarrierPut (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price Barrier Put.

Definition at line 296 of file MCEngine.cpp.

References PayOff::BarrierPut(), GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.6** void MCEngine::RunEngineCall (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price European standard Call.

Definition at line 235 of file MCEngine.cpp.

References GaussianProcess::BuildPath(), PayOff::Call(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.7** void MCEngine::RunEngineCappedCliquet (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price CappedCliquet.

Definition at line 318 of file MCEngine.cpp.

References GaussianProcess::BuildPath(), PayOff::CappedCliquet(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.8** void MCEngine::RunEngineFlooredCliquet (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price FlooredCliquet.

Definition at line 307 of file MCEngine.cpp.

References GaussianProcess::BuildPath(), PayOff::FlooredCliquet(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, and m\_price.

Referenced by RunEngineGeneral().

**6.22.2.9** void MCEngine::RunEngineGeneral (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*, Natural *Product*)

Run Monte Carlo Engine with code for the product.

Definition at line 329 of file MCEngine.cpp.

References Natural, RunEngineAsianCall(), RunEngineAsianPut(), RunEngineBarrierCall(), RunEngineBarrierPut(), RunEngineCall(), RunEngineCappedCliquet(), RunEngineFlooredCliquet(), RunEnginePut(), RunEngineRevLookbackCall(), and RunEngineRevLookbackPut().

Referenced by mainmc().

**6.22.2.10** void MCEngine::RunEnginePut (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price uropean standard Put.

Definition at line 252 of file MCEngine.cpp.

References GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, and PayOff::Put().

Referenced by RunEngineGeneral().

**6.22.2.11** void MCEngine::RunEngineRainbow2AssetsBasketMax (Random \* *pRandom*, valarray< GaussianProcess > *pHazardRateProcesses*, PayOff *thePayOff*, Real *gaussianSample*, valarray< Real > *TerminalPoints*, valarray< Real > *weights*, Matrix *Correlation*, Real *Mult*)

Price basker option with 2 assets.

Definition at line 51 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(),

LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::Rainbow2AssetsBasketMax(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_2AssetsBasketMax().

**6.22.2.12** void MCEngine::RunEngineRainbow2SpreadOptionMax (Random \* *pRandom*, valarray< GaussianProcess > *pHazardRateProcesses*, PayOff *thePayOff*, Real *gaussianSample*, valarray< Real > *TerminalPoints*, valarray< Real > *weights*, Matrix *Correlation*, Real *Mult*)

Price Spread option with 2 assets.

Definition at line 28 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::Rainbow2SpreadOptionMax(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_2SpreadOptionMax().

**6.22.2.13** void MCEngine::RunEngineRainbowBestOf2AssetsCash (Random \* *pRandom*, valarray< GaussianProcess > *pHazardRateProcesses*, PayOff *thePayOff*, Real *gaussianSample*, valarray< Real > *TerminalPoints*, valarray< Real > *weights*, Matrix *Correlation*)

Price best of + cash option with 2 assets.

Definition at line 75 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::RainbowBestOf2AssetsCash(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_BestOf2AssetsCash().

**6.22.2.14** void MCEngine::RunEngineRainbowMax2AssetsCall (Random \* *pRandom*, valarray< GaussianProcess > *pHazardRateProcesses*, PayOff *thePayOff*, Real *gaussianSample*, valarray< Real > *TerminalPoints*, valarray< Real > *weights*, Matrix *Correlation*, Real *Mult*)

Price max call option with 2 assets.

Definition at line 121 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::RainbowMax2AssetsCall(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_Max2AssetsCall().

**6.22.2.15** void MCEngine::RunEngineRainbowMax2AssetsPut (Random \* *pRandom*, valarray< GaussianProcess > *pHazardRateProcesses*, PayOff *thePayOff*, Real *gaussianSample*, valarray< Real > *TerminalPoints*, valarray< Real > *weights*, Matrix *Correlation*, Real *Mult*)

Price max put option with 2 assets.

Definition at line 167 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::RainbowMax2AssetsPut(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_Max2AssetsPut().

**6.22.2.16 void MCEngine::RunEngineRainbowMin2AssetsCall (Random \* pRandom, valarray< GaussianProcess > pHazardRateProcesses, PayOff thePayOff, Real gaussianSample, valarray< Real > TerminalPoints, valarray< Real > weights, Matrix Correlation, Real Mult)**

Price min call option with 2 assets.

Definition at line 144 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::RainbowMin2AssetsCall(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_Min2AssetsCall().

**6.22.2.17 void MCEngine::RunEngineRainbowMin2AssetsPut (Random \* pRandom, valarray< GaussianProcess > pHazardRateProcesses, PayOff thePayOff, Real gaussianSample, valarray< Real > TerminalPoints, valarray< Real > weights, Matrix Correlation, Real Mult)**

Price min put option with 2 assets.

Definition at line 190 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::RainbowMin2AssetsPut(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_Min2AssetsPut().

**6.22.2.18 void MCEngine::RunEngineRainbowWorstOf2AssetsCash (Random \* pRandom, valarray< GaussianProcess > pHazardRateProcesses, PayOff thePayOff, Real gaussianSample, valarray< Real > TerminalPoints, valarray< Real > weights, Matrix Correlation)**

Price worst of + cash option with 2 assets.

Definition at line 98 of file MCEngine.cpp.

References Matrix::CholeskyDecomposition(), Random::GetGaussian(), Matrix::GetRows(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, Natural, PayOff::RainbowWorstOf2AssetsCash(), Real, and Matrix::SetValue().

Referenced by RainbowOption::PriceByMc\_WorstOf2AssetsCash().

**6.22.2.19** void MCEngine::RunEngineRevLookbackCall (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price Lokback Call.

Definition at line 263 of file MCEngine.cpp.

References GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, and PayOff::RevLookbackCall().

Referenced by RunEngineGeneral().

**6.22.2.20** void MCEngine::RunEngineRevLookbackPut (Random \* *pRandom*, GaussianProcess \* *pHazardRateProcess*, PayOff *thePayOff*, valarray< Real > *gaussianSample*, valarray< Real > *vHazardRatePath*)

Price Lokback Put.

Definition at line 274 of file MCEngine.cpp.

References GaussianProcess::BuildPath(), Random::GetGaussians(), LongNatural, m\_DiscFactor, m\_nPaths, m\_price, and PayOff::RevLookbackPut().

Referenced by RunEngineGeneral().

## 6.22.3 Member Data Documentation

**6.22.3.1** Real MCEngine::m\_DiscFactor [private]

Definition at line 99 of file MCEngine.h.

Referenced by MCEngine(), RunEngineAsianCall(), RunEngineAsianPut(), RunEngineBarrierCall(), RunEngineBarrierPut(), RunEngineCall(), RunEngineCappedCliquet(), RunEngineFlooredCliquet(), RunEnginePut(), RunEngineRainbow2AssetsBasketMax(), RunEngineRainbow2SpreadOptionMax(), RunEngineRainbowBestOf2AssetsCash(), RunEngineRainbowMax2AssetsCall(), RunEngineRainbowMax2AssetsPut(), RunEngineRainbowMin2AssetsCall(), RunEngineRainbowMin2AssetsPut(), RunEngineRainbowWorstOf2AssetsCash(), RunEngineRevLookbackCall(), and RunEngineRevLookbackPut().

**6.22.3.2** LongNatural MCEngine::m\_nDates [private]

Definition at line 101 of file MCEngine.h.

**6.22.3.3** LongNatural MCEngine::m\_nPaths [private]

Definition at line 100 of file MCEngine.h.

Referenced by MCEngine(), RunEngineAsianCall(), RunEngineAsianPut(), RunEngineBarrierCall(), RunEngineBarrierPut(), RunEngineCall(), RunEngineCappedCliquet(), RunEngineFlooredCliquet(), RunEnginePut(), RunEngineRainbow2AssetsBasketMax(), RunEngineRainbow2SpreadOptionMax(), RunEngineRainbowBestOf2AssetsCash(), RunEngineRainbowMax2AssetsCall(), RunEngineRainbowMax2AssetsPut(), RunEngineRainbowMin2AssetsCall(), RunEngineRainbowMin2AssetsPut(), RunEngineRainbowWorstOf2AssetsCash(), RunEngineRevLookbackCall(), and RunEngineRevLookbackPut().

#### 6.22.3.4 Real MCEngine::m\_price [private]

Definition at line 99 of file MCEngine.h.

Referenced by MCEngine(), MCRresult(), RunEngineAsianCall(), RunEngineAsianPut(), RunEngineBarrierCall(), RunEngineBarrierPut(), RunEngineCall(), RunEngineCappedCliquet(), RunEngineFlooredCliquet(), RunEnginePut(), RunEngineRainbow2AssetsBasketMax(), RunEngineRainbow2SpreadOptionMax(), RunEngineRainbowBestOf2AssetsCash(), RunEngineRainbowMax2AssetsCall(), RunEngineRainbowMax2AssetsPut(), RunEngineRainbowMin2AssetsCall(), RunEngineRainbowMin2AssetsPut(), RunEngineRainbowWorstOf2AssetsCash(), RunEngineRevLookbackCall(), and RunEngineRevLookbackPut().

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

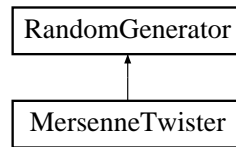
- **MCEngine.h**
- **MCEngine.cpp**



## 6.23 MersenneTwister Class Reference

```
#include <MersenneTwister.h>
```

Inheritance diagram for MersenneTwister::



### Public Member Functions

- **MersenneTwister** (**LongNatural** seed=0)  
*Default constructor : if the given seed is 0, a random seed will be chosen based on clock().*
- **MersenneTwister** (const valarray< **LongNatural** > &seeds)
- **~MersenneTwister** ()
- **Real** **getUniform** ()  
*returns a sample with weight 1.0 containing a random number on (0.0, 1.0)*
- **LongNatural** **GetOneRandomInteger** ()  
*return a random number on [0,0xffffffff]-interval*
- **VeryLongNatural** **Max** ()  
*Return maximum number of random numbers.*
- **LongNatural** **Min** ()  
*Return minimum of numbers generated.*
- void **SetSeed** (**LongNatural** seed)  
*Set seed for generator.*

### Private Attributes

- **LongInteger** seed
- valarray< **LongNatural** > mt
- **LongNatural** mti

#### 6.23.1 Constructor & Destructor Documentation

##### 6.23.1.1 MersenneTwister::MersenneTwister (**LongNatural** seed = 0)

Default constructor : if the given seed is 0, a random seed will be chosen based on clock().

Definition at line 14 of file MersenneTwister.cpp.

References **LongNatural**, **N**, and **SetSeed**().

### 6.23.1.2 MersenneTwister::MersenneTwister (const valarray< LongNatural > & seeds)

Definition at line 42 of file MersenneTwister.cpp.

References LongNatural, mt, N, and SetSeed().

### 6.23.1.3 MersenneTwister::~MersenneTwister ()

Definition at line 20 of file MersenneTwister.cpp.

## 6.23.2 Member Function Documentation

### 6.23.2.1 LongNatural MersenneTwister::GetOneRandomInteger () [virtual]

return a random number on [0,0xffffffff]-interval

Implements **RandomGenerator** (p.190).

Definition at line 65 of file MersenneTwister.cpp.

References LongNatural, LOWER\_MASK, M, MATRIX\_A, mt, mti, N, and UPPER\_MASK.

Referenced by getUniform().

### 6.23.2.2 Real MersenneTwister::getUniform () [inline, virtual]

returns a sample with weight 1.0 containing a random number on (0.0, 1.0)

Implements **RandomGenerator** (p.190).

Definition at line 21 of file MersenneTwister.h.

References GetOneRandomInteger(), and Real.

### 6.23.2.3 VeryLongNatural MersenneTwister::Max () [virtual]

Return maximum number of random numbers.

Implements **RandomGenerator** (p.190).

Definition at line 24 of file MersenneTwister.cpp.

References VeryLongNatural.

### 6.23.2.4 LongNatural MersenneTwister::Min () [virtual]

Return minimum of numbers generated.

Implements **RandomGenerator** (p.190).

Definition at line 28 of file MersenneTwister.cpp.

References LongNatural.

**6.23.2.5 void MersenneTwister::SetSeed (LongNatural *seed*) [virtual]**

Set seed for generator.

Reimplemented from **RandomGenerator** (p. 190).

Definition at line 33 of file MersenneTwister.cpp.

References LongNatural, mt, mti, and N.

Referenced by MersenneTwister().

**6.23.3 Member Data Documentation****6.23.3.1 valarray<LongNatural> MersenneTwister::mt [private]**

Definition at line 31 of file MersenneTwister.h.

Referenced by GetOneRandomInteger(), MersenneTwister(), and SetSeed().

**6.23.3.2 LongNatural MersenneTwister::mti [private]**

Definition at line 32 of file MersenneTwister.h.

Referenced by GetOneRandomInteger(), and SetSeed().

**6.23.3.3 LongInteger MersenneTwister::seed [private]**

Definition at line 30 of file MersenneTwister.h.

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

- **MersenneTwister.h**
- **MersenneTwister.cpp**

## 6.24 OptionStrategy Class Reference

```
#include <OptionStrategy.h>
```

### Public Member Functions

- **OptionStrategy ()**  
*Default constructor: initialize parameters.*
- **~OptionStrategy ()**
- void **addOneOptionToStrategy (Real spot, Real vol, bool isVol, Real r, Real K, Real T, TypeOptionBS type, Real Quantity)**  
*Generic function to add Options to the Strategy.*
- void **addOneBlackScholesObject (BlackScholes \*bs, Real Quantity)**
- void **addLongCallSpread (Real spot, Real volStrike1, bool isVol1, Real volStrike2, bool isVol2, Real r, Real K1, Real K2, Real T, Real Quantity)**  
*Create a long call spread in the portfolio.*
- void **addLongStraddle (Real spot, Real vol, bool isVol, Real r, Real K, Real T, Real Quantity)**  
*Create a long straddle in the portfolio.*
- void **addLongStrangle (Real spot, Real volStrike1, bool isVol1, Real volStrike2, bool isVol2, Real r, Real K1, Real K2, Real T, Real Quantity)**  
*Create a long strangle in the portfolio.*
- void **addLongButterflySpread (Real spot, Real volStrike1, bool isVol1, Real volStrike2, bool isVol2, Real volStrike3, bool isVol3, Real r, Real K1, Real K2, Real T, Real Quantity)**  
*There are two functions for butterfly if you want to specify or not  $K3=(K1+K2)/2$  by default.*
- void **addLongButterflySpread (Real spot, Real volStrike1, bool isVol1, Real volStrike2, bool isVol2, Real volStrike3, bool isVol3, Real r, Real K1, Real K2, Real K3, Real T, Real Quantity)**
- void **addLongRatioCallSpread (Real spot, Real volStrike1, bool isVol1, Real volStrike2, bool isVol2, Real r, Real K1, Real K2, Real T, Real Quantity)**  
*Create a long ratio call spread in the portfolio.*
- void **addLongPutSpread (Real spot, Real volStrike1, bool isVol1, Real volStrike2, bool isVol2, Real r, Real K1, Real K2, Real T, Real Quantity)**  
*Create a long put spread in the portfolio.*
- **Real getGlobalDelta ()**  
*Get Greeks for global Portfolio(p. 158).*
- **Real getGlobalGamma ()**
- **Real getGlobalVega ()**
- **Real getGlobalTheta ()**
- **Real getGlobalRho ()**

- **Real** `returnPrice ()`  
*Return global price of the portfolio.*
- **Real** `recalcPrice ()`  
*Recalculate global price of the portfolio in case of change.*
- **Natural** `returnNbOptions () const`  
*Return number of options in the portfolio.*
- **BlackScholes \* returnOption (Natural i) const**  
*Return pointer on blackscholes object inside, used for variance swaps.*
- **Real** `returnOptionQuantity (Natural i) const`  
*Return quantity on blackscholes object inside.*
- **void** `changeRate (Real addConstant=defaultshiftRate)`  
*Add constant rate to the inside rate of all BlackScholes(p.26) objects.*
- **void** `changeVol (Real addConstant=defaultshiftVol)`  
*Add constant vol to the inside vol of all BlackScholes(p.26) objects.*
- **void** `changeMaturity (Real addConstant=defaultshiftMat)`  
*Add constant maturity to the inside maturity of all BlackScholes(p.26) objects.*
- **void** `changeSpot (Real addConstant=defaultshiftSpot)`  
*Add constant spot to the inside spot of all BlackScholes(p.26) objects.*
- **void** `changeStrike (Real addConstant=defaultshiftStrike)`  
*Add constant strike to the inside strike of all BlackScholes(p.26) objects.*

## Private Attributes

- **Real** `_price`
- **Natural** `_nbOptions`
- **valarray< BlackScholes \* >** `_insideOptions`
- **valarray< Real >** `_insideQuantities`

## Friends

- **ostream & operator<<** (**ostream &os**, **const OptionStrategy &optionStrategy**)  
*display parameters of options in the optionstrategy object*
- **ostream & operator<<** (**ostream &os**, **const OptionStrategy \*optionStrategy**)

## 6.24.1 Constructor & Destructor Documentation

### 6.24.1.1 OptionStrategy::OptionStrategy ()

Default constructor: initialize parameters.

Definition at line 3 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, and `_nbOptions`.

### 6.24.1.2 OptionStrategy::~~OptionStrategy ()

Definition at line 10 of file OptionStrategy.cpp.

## 6.24.2 Member Function Documentation

### 6.24.2.1 void OptionStrategy::addLongButterflySpread (Real *spot*, Real *volStrike1*, bool *isVol1*, Real *volStrike2*, bool *isVol2*, Real *volStrike3*, bool *isVol3*, Real *r*, Real *K1*, Real *K2*, Real *K3*, Real *T*, Real *Quantity*)

Definition at line 78 of file OptionStrategy.cpp.

References `addOneOptionToStrategy()`, `Call`, `r`, and `Real`.

### 6.24.2.2 void OptionStrategy::addLongButterflySpread (Real *spot*, Real *volStrike1*, bool *isVol1*, Real *volStrike2*, bool *isVol2*, Real *volStrike3*, bool *isVol3*, Real *r*, Real *K1*, Real *K2*, Real *T*, Real *Quantity*)

There are two functions for butterfly if you want to specify or not  $K3=(K1+K2)/2$  by default.

Definition at line 72 of file OptionStrategy.cpp.

References `addOneOptionToStrategy()`, `Call`, `r`, and `Real`.

Referenced by `inputButterflySpread()`, `mainoptionstrategy()`, and `mainvarianceswap()`.

### 6.24.2.3 void OptionStrategy::addLongCallSpread (Real *spot*, Real *volStrike1*, bool *isVol1*, Real *volStrike2*, bool *isVol2*, Real *r*, Real *K1*, Real *K2*, Real *T*, Real *Quantity*)

Create a long call spread in the portfolio.

We search to be long the call with smallest strike and short the other

Definition at line 44 of file OptionStrategy.cpp.

References `addOneOptionToStrategy()`, `Call`, `r`, and `Real`.

Referenced by `inputCallSpread()`.

### 6.24.2.4 void OptionStrategy::addLongPutSpread (Real *spot*, Real *volStrike1*, bool *isVol1*, Real *volStrike2*, bool *isVol2*, Real *r*, Real *K1*, Real *K2*, Real *T*, Real *Quantity*)

Create a long put spread in the portfolio.

Definition at line 95 of file OptionStrategy.cpp.

References addOneOptionToStrategy(), Put, r, and Real.

Referenced by inputPutSpread().

#### 6.24.2.5 void OptionStrategy::addLongRatioCallSpread (Real *spot*, Real *volStrike1*, bool *isVol1*, Real *volStrike2*, bool *isVol2*, Real *r*, Real *K1*, Real *K2*, Real *T*, Real *Quantity*)

Create a long ratio call spread in the portfolio.

Definition at line 84 of file OptionStrategy.cpp.

References addOneOptionToStrategy(), Call, r, and Real.

Referenced by inputRatioCallSpread().

#### 6.24.2.6 void OptionStrategy::addLongStraddle (Real *spot*, Real *vol*, bool *isVol*, Real *r*, Real *K*, Real *T*, Real *Quantity*)

Create a long straddle in the portfolio.

Definition at line 56 of file OptionStrategy.cpp.

References addOneOptionToStrategy(), Call, Put, r, and Real.

Referenced by inputStraddle().

#### 6.24.2.7 void OptionStrategy::addLongStrangle (Real *spot*, Real *volStrike1*, bool *isVol1*, Real *volStrike2*, bool *isVol2*, Real *r*, Real *K1*, Real *K2*, Real *T*, Real *Quantity*)

Create a long strangle in the portfolio.

Definition at line 61 of file OptionStrategy.cpp.

References addOneOptionToStrategy(), Call, Put, r, and Real.

Referenced by inputStrangle().

#### 6.24.2.8 void OptionStrategy::addOneBlackScholesObject (BlackScholes \* *bs*, Real *Quantity*)

Definition at line 37 of file OptionStrategy.cpp.

References \_insideOptions, \_insideQuantities, \_nbOptions, and Real.

Referenced by inputOptionStrategy(), and mainvarianceswap().

#### 6.24.2.9 void OptionStrategy::addOneOptionToStrategy (Real *spot*, Real *vol*, bool *isVol*, Real *r*, Real *K*, Real *T*, TypeOptionBS *type*, Real *Quantity*)

Generic function to add Options to the Strategy.

Definition at line 29 of file OptionStrategy.cpp.

References \_insideOptions, \_insideQuantities, \_nbOptions, r, and Real.

Referenced by `addLongButterflySpread()`, `addLongCallSpread()`, `addLongPutSpread()`, `addLongRatioCallSpread()`, `addLongStraddle()`, and `addLongStrangle()`.

#### 6.24.2.10 `void OptionStrategy::changeMaturity (Real addConstant = defaultshiftMat)`

Add constant maturity to the inside maturity of all **BlackScholes**(p. 26) objects.

Definition at line 175 of file `OptionStrategy.cpp`.

References `_insideOptions`, `_nbOptions`, `BlackScholes::changeMaturity()`, `BlackScholes::getMaturity()`, and `Real`.

Referenced by `VarianceSwap::getTheta()`.

#### 6.24.2.11 `void OptionStrategy::changeRate (Real addConstant = defaultshiftRate)`

Add constant rate to the inside rate of all **BlackScholes**(p. 26) objects.

Definition at line 161 of file `OptionStrategy.cpp`.

References `_insideOptions`, `_nbOptions`, `BlackScholes::changeRate()`, `BlackScholes::getRate()`, and `Real`.

Referenced by `VarianceSwap::getRho()`.

#### 6.24.2.12 `void OptionStrategy::changeSpot (Real addConstant = defaultshiftSpot)`

Add constant spot to the inside spot of all **BlackScholes**(p. 26) objects.

Definition at line 182 of file `OptionStrategy.cpp`.

References `_insideOptions`, `_nbOptions`, `BlackScholes::changeSpot()`, `BlackScholes::getSpot()`, and `Real`.

#### 6.24.2.13 `void OptionStrategy::changeStrike (Real addConstant = defaultshiftStrike)`

Add constant strike to the inside strike of all **BlackScholes**(p. 26) objects.

Definition at line 189 of file `OptionStrategy.cpp`.

References `_insideOptions`, `_nbOptions`, `BlackScholes::changeStrike()`, `BlackScholes::getStrike()`, and `Real`.

#### 6.24.2.14 `void OptionStrategy::changeVol (Real addConstant = defaultshiftVol)`

Add constant vol to the inside vol of all **BlackScholes**(p. 26) objects.

Definition at line 168 of file `OptionStrategy.cpp`.

References `_insideOptions`, `_nbOptions`, `BlackScholes::changeVol()`, `BlackScholes::getVolatility()`, and `Real`.

Referenced by `VarianceSwap::getVega()`.



**6.24.2.15 Real OptionStrategy::getGlobalDelta ()**

Get Greeks for global **Portfolio**(p. 158).

Definition at line 106 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, `_nbOptions`, and `Real`.

Referenced by `inputOptionStrategy()`, and `mainoptionstrategy()`.

**6.24.2.16 Real OptionStrategy::getGlobalGamma ()**

Definition at line 114 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, `_nbOptions`, and `Real`.

Referenced by `inputOptionStrategy()`.

**6.24.2.17 Real OptionStrategy::getGlobalRho ()**

Definition at line 138 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, `_nbOptions`, and `Real`.

Referenced by `inputOptionStrategy()`, and `Portfolio::returnSensibilityToRate()`.

**6.24.2.18 Real OptionStrategy::getGlobalTheta ()**

Definition at line 130 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, `_nbOptions`, and `Real`.

Referenced by `inputOptionStrategy()`, and `Portfolio::returnSensibilityToTime()`.

**6.24.2.19 Real OptionStrategy::getGlobalVega ()**

Definition at line 122 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, `_nbOptions`, and `Real`.

Referenced by `inputOptionStrategy()`, and `Portfolio::returnSensibilityToVol()`.

**6.24.2.20 Real OptionStrategy::recalcPrice ()**

Recalculate global price of the portfolio in case of change.

Definition at line 19 of file OptionStrategy.cpp.

References `_insideOptions`, `_insideQuantities`, `_nbOptions`, and `Real`.

**6.24.2.21 Natural OptionStrategy::returnNbOptions () const**

Return number of options in the portfolio.

Definition at line 147 of file OptionStrategy.cpp.

References `_nbOptions`, and `Natural`.

Referenced by `VarianceSwap::getPrice()`, and `operator<<()`.

#### 6.24.2.22 `BlackScholes * OptionStrategy::returnOption (Natural i) const`

Return pointer on blackscholes object inside, used for variance swaps.

Definition at line 151 of file `OptionStrategy.cpp`.

References `_insideOptions`, and `Natural`.

Referenced by `VarianceSwap::getPrice()`, and `operator<<()`.

#### 6.24.2.23 `Real OptionStrategy::returnOptionQuantity (Natural i) const`

Return quantity on blackscholes object inside.

Definition at line 156 of file `OptionStrategy.cpp`.

References `_insideOptions`, `_insideQuantities`, `Natural`, and `Real`.

Referenced by `operator<<()`.

#### 6.24.2.24 `Real OptionStrategy::returnPrice ()`

Return global price of the portfolio.

Definition at line 14 of file `OptionStrategy.cpp`.

References `Real`.

Referenced by `Portfolio::getPrice()`, `inputOptionStrategy()`, and `mainoptionstrategy()`.

### 6.24.3 Friends And Related Function Documentation

#### 6.24.3.1 `ostream& operator<< (ostream & os, const OptionStrategy * optionStrategy) [friend]`

Definition at line 28 of file `OptionStrategy.h`.

#### 6.24.3.2 `ostream& operator<< (ostream & os, const OptionStrategy & optionStrategy) [friend]`

display parameters of options in the `optionstrategy` object

##### Parameters:

*os*: the output stream to direct output to

*optionStrategy*: the option strategy to display

##### Returns:

output stream as is standard for `operator<<`

Definition at line 196 of file `OptionStrategy.cpp`.

## 6.24.4 Member Data Documentation

### 6.24.4.1 `valarray<BlackScholes*> OptionStrategy::_insideOptions` [private]

Definition at line 93 of file OptionStrategy.h.

Referenced by `addOneBlackScholesObject()`, `addOneOptionToStrategy()`, `changeMaturity()`, `changeRate()`, `changeSpot()`, `changeStrike()`, `changeVol()`, `getGlobalDelta()`, `getGlobalGamma()`, `getGlobalRho()`, `getGlobalTheta()`, `getGlobalVega()`, `OptionStrategy()`, `recalcPrice()`, `returnOption()`, and `returnOptionQuantity()`.

### 6.24.4.2 `valarray<Real> OptionStrategy::_insideQuantities` [private]

Definition at line 94 of file OptionStrategy.h.

Referenced by `addOneBlackScholesObject()`, `addOneOptionToStrategy()`, `getGlobalDelta()`, `getGlobalGamma()`, `getGlobalRho()`, `getGlobalTheta()`, `getGlobalVega()`, `OptionStrategy()`, `recalcPrice()`, and `returnOptionQuantity()`.

### 6.24.4.3 `Natural OptionStrategy::_nbOptions` [private]

Definition at line 92 of file OptionStrategy.h.

Referenced by `addOneBlackScholesObject()`, `addOneOptionToStrategy()`, `changeMaturity()`, `changeRate()`, `changeSpot()`, `changeStrike()`, `changeVol()`, `getGlobalDelta()`, `getGlobalGamma()`, `getGlobalRho()`, `getGlobalTheta()`, `getGlobalVega()`, `OptionStrategy()`, `recalcPrice()`, and `returnNbOptions()`.

### 6.24.4.4 `Real OptionStrategy::_price` [private]

Definition at line 91 of file OptionStrategy.h.

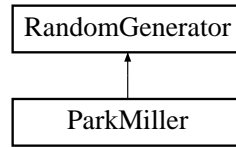
The documentation for this class was generated from the following files:

- `OptionStrategy.h`
- `OptionStrategy.cpp`

## 6.25 ParkMiller Class Reference

```
#include <ParkMiller.h>
```

Inheritance diagram for ParkMiller::



### Public Member Functions

- **ParkMiller (LongNatural Seed\_ =0)**  
*Default constructor: initialize variable.*
- **~ParkMiller ()**
- **LongNatural GetOneRandomInteger ()**  
*Create one random integer.*
- **Real getUniform ()**  
*Creates one uniform number on (0.0,1.0).*
- **void SetSeed (LongNatural Seed)**  
*Set seed for generator.*
- **VeryLongNatural Max ()**  
*Return maximum number of random numbers.*
- **LongNatural Min ()**  
*Return minimum of numbers generated.*

### Private Attributes

- **LongNatural Seed**

#### 6.25.1 Constructor & Destructor Documentation

##### 6.25.1.1 ParkMiller::ParkMiller (LongNatural Seed\_ = 0)

Default constructor: initialize variable.

Definition at line 4 of file ParkMiller.cpp.

References LongNatural, and Seed.

##### 6.25.1.2 ParkMiller::~~ParkMiller ()

Definition at line 9 of file ParkMiller.cpp.

## 6.25.2 Member Function Documentation

### 6.25.2.1 LongNatural ParkMiller::GetOneRandomInteger () [virtual]

Create one random integer.

Implements **RandomGenerator** (p.190).

Definition at line 28 of file ParkMiller.cpp.

References a, LongInteger, LongNatural, m, q, r, and Seed.

Referenced by getUniform().

### 6.25.2.2 Real ParkMiller::getUniform () [virtual]

Creates one uniform number on (0.0,1.0).

Implements **RandomGenerator** (p.190).

Definition at line 42 of file ParkMiller.cpp.

References GetOneRandomInteger(), m, and Real.

### 6.25.2.3 VeryLongNatural ParkMiller::Max () [virtual]

Return maximum number of random numbers.

Implements **RandomGenerator** (p.190).

Definition at line 20 of file ParkMiller.cpp.

References m, and VeryLongNatural.

### 6.25.2.4 LongNatural ParkMiller::Min () [virtual]

Return minimum of numbers generated.

Implements **RandomGenerator** (p.190).

Definition at line 24 of file ParkMiller.cpp.

References LongNatural.

### 6.25.2.5 void ParkMiller::SetSeed (LongNatural Seed) [virtual]

Set seed for generator.

Reimplemented from **RandomGenerator** (p.190).

Definition at line 46 of file ParkMiller.cpp.

References LongNatural, and Seed.

## 6.25.3 Member Data Documentation

### 6.25.3.1 LongNatural ParkMiller::Seed [private]

Reimplemented from **RandomGenerator** (p.191).

Definition at line 36 of file ParkMiller.h.

Referenced by `GetOneRandomInteger()`, `ParkMiller()`, and `SetSeed()`.

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

- **ParkMiller.h**
- **ParkMiller.cpp**

## 6.26 PayOff Class Reference

```
#include <PayOff.h>
```

### Public Member Functions

- **PayOff** (**Real** Strike\_)  
*Default constructor: set Strike.*
- **PayOff** (void)  
*void constructor*
- void **SetStrike** (**Real** Strike\_)
- virtual **Real operator()** (**Real** Spot) const  
*Nice operator for european call options only.*
- virtual **~PayOff** ()
- virtual **Real Call** (**Real** Fwd)  
*Return the payoff of a call.*
- virtual **Real Put** (**Real** Fwd)  
*Return the payoff of a put.*
- virtual **Real AsianCall** (valarray< **Real** > Path, **LongNatural** nDates)  
*Return the payoff of an asian call.*
- virtual **Real AsianPut** (valarray< **Real** > Path, **LongNatural** nDates)  
*Return the payoff of an asian put.*
- virtual **Real RevLookbackCall** (valarray< **Real** > Path, **LongNatural** nDates)  
*Return the payoff of a look back call.*
- virtual **Real RevLookbackPut** (valarray< **Real** > Path, **LongNatural** nDates)  
*Return the payoff of a look back put.*
- virtual **Real BarrierCall** (valarray< **Real** > Path, **LongNatural** nDates)  
*Return the payoff of a barrier call.*
- virtual **Real BarrierPut** (valarray< **Real** > Path, **LongNatural** nDates)  
*Return the payoff of a barrier put.*
- virtual **Real FlooredCliquet** (**Real** Spot, **Real** Fwd)  
*Return the payoff of a floor.*
- virtual **Real CappedCliquet** (**Real** Spot, **Real** Fwd)  
*Return the payoff of a cap.*
- virtual **Real Rainbow2SpreadOptionMax** (**Real** Fwd1, **Real** Fwd2, **Real** W1, **Real** W2, **Real** Mult)

- virtual **Real Rainbow2AssetsBasketMax** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**, **Real Mult**)
- virtual **Real RainbowBestOf2AssetsCash** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**)
- virtual **Real RainbowWorstOf2AssetsCash** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**)
- virtual **Real RainbowMax2AssetsCall** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**, **Real Mult**)
- virtual **Real RainbowMin2AssetsCall** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**, **Real Mult**)
- virtual **Real RainbowMax2AssetsPut** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**, **Real Mult**)
- virtual **Real RainbowMin2AssetsPut** (**Real Fwd1**, **Real Fwd2**, **Real W1**, **Real W2**, **Real Mult**)
- virtual **Real Convertible** (**Real Fwd**, **Real ConversionRatio**, **Real BondPrice**, **Real CallPrice=TN\_INFINITY**, **Real PutPrice=0**)

## Private Attributes

- **Real Strike**

## 6.26.1 Constructor & Destructor Documentation

### 6.26.1.1 PayOff::PayOff (Real *Strike* \_)

Default constructor: set Strike.

Definition at line 5 of file PayOff.cpp.

References Real.

### 6.26.1.2 PayOff::PayOff (void)

void constructor

#### Author:

Yann

Definition at line 9 of file PayOff.cpp.

### 6.26.1.3 virtual PayOff::~PayOff () [inline, virtual]

Definition at line 24 of file PayOff.h.

## 6.26.2 Member Function Documentation

### 6.26.2.1 Real PayOff::AsianCall (valarray< Real > *Path*, LongNatural *nDates*) [virtual]

Return the payoff of an asian call.



Definition at line 34 of file PayOff.cpp.

References Average(), LongNatural, Real, and Strike.

Referenced by MCEngine::RunEngineAsianCall().

#### **6.26.2.2 Real PayOff::AsianPut (valarray< Real > *Path*, LongNatural *nDates*)** [virtual]

Return the payoff of an asian put.

Definition at line 39 of file PayOff.cpp.

References Average(), LongNatural, Real, and Strike.

Referenced by MCEngine::RunEngineAsianPut().

#### **6.26.2.3 Real PayOff::BarrierCall (valarray< Real > *Path*, LongNatural *nDates*)** [virtual]

Return the payoff of a barrier call.

Definition at line 54 of file PayOff.cpp.

References LongNatural, Real, and Strike.

Referenced by MCEngine::RunEngineBarrierCall().

#### **6.26.2.4 Real PayOff::BarrierPut (valarray< Real > *Path*, LongNatural *nDates*)** [virtual]

Return the payoff of a barrier put.

Definition at line 62 of file PayOff.cpp.

References LongNatural, Real, and Strike.

Referenced by MCEngine::RunEngineBarrierPut().

#### **6.26.2.5 Real PayOff::Call (Real *Fwd*)** [virtual]

Return the payoff of a call.

Definition at line 24 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineCall(), and binomialTree::runEngineCall().

#### **6.26.2.6 Real PayOff::CappedCliquet (Real *Spot*, Real *Fwd*)** [virtual]

Return the payoff of a cap.

Definition at line 77 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineCappedCliquet().

**6.26.2.7 Real PayOff::Convertible (Real *Fwd*, Real *ConversionRatio*, Real *BondPrice*, Real *CallPrice* = TN\_INFINITY, Real *PutPrice* = 0) [virtual]**

**Returns:**

the payoff of a Convertible bond

If *CallPrice* and *PutPrice* are omitted they are set to infinity and zero respectively so that they don't factor into the calculation

Definition at line 124 of file `PayOff.cpp`.

References `Real`.

Referenced by `binomialTree::runEngineConvertibleBond()`.

**6.26.2.8 Real PayOff::FlooredCliquet (Real *Spot*, Real *Fwd*) [virtual]**

Return the payoff of a floor.

Definition at line 70 of file `PayOff.cpp`.

References `Real`, and `Strike`.

Referenced by `MCEngine::RunEngineFlooredCliquet()`.

**6.26.2.9 Real PayOff::operator() (Real *Spot*) const [virtual]**

Nice operator for european call options only.

Definition at line 19 of file `PayOff.cpp`.

References `Real`, and `Strike`.

**6.26.2.10 Real PayOff::Put (Real *Fwd*) [virtual]**

Return the payoff of a put.

Definition at line 29 of file `PayOff.cpp`.

References `Real`, and `Strike`.

Referenced by `MCEngine::RunEnginePut()`.

**6.26.2.11 Real PayOff::Rainbow2AssetsBasketMax (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*, Real *Mult*) [virtual]**

**Returns:**

the payoff of a basket option with 2 assets

Definition at line 89 of file `PayOff.cpp`.

References `Real`, and `Strike`.

Referenced by `MCEngine::RunEngineRainbow2AssetsBasketMax()`.

**6.26.2.12 Real PayOff::Rainbow2SpreadOptionMax (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*, Real *Mult*)** [virtual]**Returns:**

the payoff of a Spread option with 2 assets

Definition at line 84 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbow2SpreadOptionMax().

**6.26.2.13 Real PayOff::RainbowBestOf2AssetsCash (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*)** [virtual]**Returns:**

the payoff of a best of + cash option with 2 assets

Definition at line 94 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbowBestOf2AssetsCash().

**6.26.2.14 Real PayOff::RainbowMax2AssetsCall (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*, Real *Mult*)** [virtual]**Returns:**

the payoff of a Max call option with 2 assets

Definition at line 104 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbowMax2AssetsCall().

**6.26.2.15 Real PayOff::RainbowMax2AssetsPut (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*, Real *Mult*)** [virtual]**Returns:**

the payoff of a Max put option with 2 assets

Definition at line 114 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbowMax2AssetsPut().

**6.26.2.16 Real PayOff::RainbowMin2AssetsCall (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*, Real *Mult*)** [virtual]**Returns:**

the payoff of a Min call option with 2 assets

Definition at line 109 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbowMin2AssetsCall().

#### 6.26.2.17 Real PayOff::RainbowMin2AssetsPut (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*, Real *Mult*) [virtual]

##### Returns:

the payoff of a Min put option with 2 assets

Definition at line 119 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbowMin2AssetsPut().

#### 6.26.2.18 Real PayOff::RainbowWorstOf2AssetsCash (Real *Fwd1*, Real *Fwd2*, Real *W1*, Real *W2*) [virtual]

##### Returns:

the payoff of a worst of + cash option with 2 assets

Definition at line 99 of file PayOff.cpp.

References Real, and Strike.

Referenced by MCEngine::RunEngineRainbowWorstOf2AssetsCash().

#### 6.26.2.19 Real PayOff::RevLookbackCall (valarray< Real > *Path*, LongNatural *nDates*) [virtual]

Return the payoff of a look back call.

Definition at line 44 of file PayOff.cpp.

References LongNatural, Maximize(), Real, and Strike.

Referenced by MCEngine::RunEngineRevLookbackCall().

#### 6.26.2.20 Real PayOff::RevLookbackPut (valarray< Real > *Path*, LongNatural *nDates*) [virtual]

Return the payoff of a look back put.

Definition at line 49 of file PayOff.cpp.

References LongNatural, Maximize(), Real, and Strike.

Referenced by MCEngine::RunEngineRevLookbackPut().

#### 6.26.2.21 void PayOff::SetStrike (Real *Strike* \_)

Definition at line 13 of file PayOff.cpp.

References Real, and Strike.

Referenced by `RainbowOption::instanciateMCVariables()`, and `RainbowOption::RainbowOption()`.

### 6.26.3 Member Data Documentation

#### 6.26.3.1 Real PayOff::Strike [private]

Definition at line 88 of file `PayOff.h`.

Referenced by `AsianCall()`, `AsianPut()`, `BarrierCall()`, `BarrierPut()`, `Call()`, `CappedCliquet()`, `FlooredCliquet()`, `operator()()`, `Put()`, `Rainbow2AssetsBasketMax()`, `Rainbow2SpreadOptionMax()`, `RainbowBestOf2AssetsCash()`, `RainbowMax2AssetsCall()`, `RainbowMax2AssetsPut()`, `RainbowMin2AssetsCall()`, `RainbowMin2AssetsPut()`, `RainbowWorstOf2AssetsCash()`, `RevLookbackCall()`, `RevLookbackPut()`, and `SetStrike()`.

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

- `PayOff.h`
- `PayOff.cpp`

## 6.27 Portfolio Class Reference

```
#include <PortFolio.h>
```

### Public Member Functions

- **Portfolio** (char \*name, **Currency** currency)  
*Default Constructor : just initialization of size.*
- **~Portfolio** (void)
- char \* **getName** ()  
*Return name of the portfolio.*
- **Currency** **getCurrency** ()  
*Return currency of the portfolio.*
- char \* **getCurrencyAsString** ()  
*Return currency of the portfolio as a string.*
- void **addOptionStrategy** (**OptionStrategy** \*optionStrategy)  
*Add one option to the Portfolio.*
- void **addRainbowOption** (**RainbowOption** \*rainbowOption, **Real** quantity)  
*Add one rainbow option to the Portfolio.*
- void **addExoticOption** (**Exotics** \*exoticOption, **Real** quantity)  
*Add one exotic option to the Portfolio.*
- void **addVanillaSwap** (**VanillaSwap** \*vanillaSwap, **Real** quantity)  
*Add one vanilla swap to the Portfolio.*
- void **addVarianceSwap** (**VarianceSwap** \*varSwap, **Real** quantity)  
*Add one variance swap to the Portfolio.*
- void **addBond** (**bond** \*oneBond, **Real** quantity)  
*Add one bond to the Portfolio.*
- void **addAsset** (**asset** \*oneAsset, **Real** quantity)  
*Add one asset to the Portfolio.*
- **Real** **getPrice** ()  
*Return Price of the whole Portfolio.*
- **Real** **returnSensibilityToRate** ()  
*Return sensibility to interest rate.*
- **Real** **returnSensibilityToVol** ()  
*Return sensibility to volatility.*

- **Real returnSensibilityToTime ()**

*Return sensibility to time.*

## Private Attributes

- **char \* \_name**
- **Currency \_currency**
- **OptionStrategy \_optionStrategy**
- **valarray< RainbowOption \* > \_rainbowOptions**
- **valarray< Exotics \* > \_exoticsOptions**
- **valarray< VanillaSwap \* > \_vanSwaps**
- **valarray< VarianceSwap \* > \_varSwaps**
- **valarray< bond \* > \_bonds**
- **valarray< asset \* > \_assets**
- **valarray< Real > \_quantityRainbowOptions**
- **valarray< Real > \_quantityExoticsOptions**
- **valarray< Real > \_quantityVanSwaps**
- **valarray< Real > \_quantityVarSwaps**
- **valarray< Real > \_quantityBonds**
- **valarray< Real > \_quantityAssets**
- **Natural \_nbRainbowOptions**
- **Natural \_nbExoticsOptions**
- **Natural \_nbVanSwaps**
- **Natural \_nbVarSwaps**
- **Natural \_nbBonds**
- **Natural \_nbAssets**

### 6.27.1 Detailed Description

#### Author:

Simon

Definition at line 24 of file PortFolio.h.

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 Portfolio::Portfolio (char \* *name*, Currency *currency*)

Default Constructor : just initialization of size.

#### Parameters:

*name*: Name of the portfolio

*currency*: Currency of the portfolio

Definition at line 3 of file PortFolio.cpp.

References `_assets`, `_bonds`, `_nbAssets`, `_nbBonds`, `_nbRainbowOptions`, `_nbVanSwaps`, `_nbVarSwaps`, `_optionStrategy`, `_rainbowOptions`, `_vanSwaps`, `_varSwaps`, `MAX_SIZE`, and `MAX_SIZE_NAME`.

### 6.27.2.2 Portfolio::~~Portfolio (void)

Definition at line 21 of file PortFolio.cpp.

## 6.27.3 Member Function Documentation

### 6.27.3.1 void Portfolio::addAsset (asset \* *oneAsset*, Real *quantity*)

Add one asset to the Portfolio.

Definition at line 85 of file PortFolio.cpp.

References `_assets`, `_nbAssets`, `_quantityAssets`, and `Real`.

### 6.27.3.2 void Portfolio::addBond (bond \* *oneBond*, Real *quantity*)

Add one bond to the Portfolio.

Definition at line 79 of file PortFolio.cpp.

References `_bonds`, `_nbBonds`, `_quantityBonds`, and `Real`.

### 6.27.3.3 void Portfolio::addExoticOption (Exotics \* *exoticOption*, Real *quantity*)

Add one exotic option to the Portfolio.

Definition at line 61 of file PortFolio.cpp.

References `_exoticsOptions`, `_nbExoticsOptions`, `_quantityExoticsOptions`, and `Real`.

### 6.27.3.4 void Portfolio::addOptionStrategy (OptionStrategy \* *optionStrategy*)

Add one option to the Portfolio.

Definition at line 51 of file PortFolio.cpp.

References `_optionStrategy`.

### 6.27.3.5 void Portfolio::addRainbowOption (RainbowOption \* *rainbowOption*, Real *quantity*)

Add one rainbow option to the Portfolio.

Definition at line 55 of file PortFolio.cpp.

References `_nbRainbowOptions`, `_quantityRainbowOptions`, `_rainbowOptions`, and `Real`.

### 6.27.3.6 void Portfolio::addVanillaSwap (VanillaSwap \* *vanillaSwap*, Real *quantity*)

Add one vanilla swap to the Portfolio.

Definition at line 67 of file PortFolio.cpp.

References `_nbVanSwaps`, `_quantityVanSwaps`, `_vanSwaps`, and `Real`.



**6.27.3.7 void Portfolio::addVarianceSwap (VarianceSwap \* *varSwap*, Real *quantity*)**

Add one variance swap to the Portfolio.

Definition at line 73 of file PortFolio.cpp.

References `_nbVarSwaps`, `_quantityVarSwaps`, `_varSwaps`, and `Real`.

**6.27.3.8 Currency Portfolio::getCurrency ()**

Return currency of the portfolio.

Definition at line 29 of file PortFolio.cpp.

References `Currency`.

**6.27.3.9 char \* Portfolio::getCurrencyAsString ()**

Return currency of the portfolio as a string.

Definition at line 33 of file PortFolio.cpp.

References `CAD`, `EUR`, `MAX_SIZE_NAME`, and `USD`.

**6.27.3.10 char \* Portfolio::getName ()**

Return name of the portfolio.

Definition at line 25 of file PortFolio.cpp.

**6.27.3.11 Real Portfolio::getPrice ()**

Return Price of the whole Portfolio.

Definition at line 91 of file PortFolio.cpp.

References `_assets`, `_bonds`, `_exoticsOptions`, `_nbAssets`, `_nbBonds`, `_nbExoticsOptions`, `_nbRainbowOptions`, `_nbVanSwaps`, `_nbVarSwaps`, `_optionStrategy`, `_rainbowOptions`, `_vanSwaps`, `_varSwaps`, `Integer`, `Real`, and `OptionStrategy::returnPrice()`.

**6.27.3.12 Real Portfolio::returnSensibilityToRate ()**

Return sensibility to interest rate.

Definition at line 115 of file PortFolio.cpp.

References `_assets`, `_bonds`, `_exoticsOptions`, `_nbAssets`, `_nbBonds`, `_nbExoticsOptions`, `_nbRainbowOptions`, `_nbVanSwaps`, `_nbVarSwaps`, `_optionStrategy`, `_rainbowOptions`, `_vanSwaps`, `_varSwaps`, `OptionStrategy::getGlobalRho()`, `Integer`, and `Real`.

**6.27.3.13 Real Portfolio::returnSensibilityToTime ()**

Return sensibility to time.

Definition at line 154 of file PortFolio.cpp.

References `_exoticsOptions`, `_nbExoticsOptions`, `_nbRainbowOptions`, `_nbVanSwaps`, `_nbVarSwaps`, `_optionStrategy`, `_rainbowOptions`, `_vanSwaps`, `_varSwaps`, `OptionStrategy::getGlobalTheta()`, `Integer`, and `Real`.

#### 6.27.3.14 Real Portfolio::returnSensibilityToVol ()

Return sensibility to volatility.

Definition at line 139 of file `PortFolio.cpp`.

References `_exoticsOptions`, `_nbExoticsOptions`, `_nbRainbowOptions`, `_nbVarSwaps`, `_optionStrategy`, `_rainbowOptions`, `_varSwaps`, `OptionStrategy::getGlobalVega()`, `Integer`, and `Real`.

### 6.27.4 Member Data Documentation

#### 6.27.4.1 valarray<asset\*> Portfolio::\_assets [private]

Definition at line 88 of file `PortFolio.h`.

Referenced by `addAsset()`, `getPrice()`, `Portfolio()`, and `returnSensibilityToRate()`.

#### 6.27.4.2 valarray<bond\*> Portfolio::\_bonds [private]

Definition at line 87 of file `PortFolio.h`.

Referenced by `addBond()`, `getPrice()`, `Portfolio()`, and `returnSensibilityToRate()`.

#### 6.27.4.3 Currency Portfolio::\_currency [private]

Definition at line 80 of file `PortFolio.h`.

#### 6.27.4.4 valarray<Exotics\*> Portfolio::\_exoticsOptions [private]

Definition at line 84 of file `PortFolio.h`.

Referenced by `addExoticOption()`, `getPrice()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

#### 6.27.4.5 char\* Portfolio::\_name [private]

Definition at line 79 of file `PortFolio.h`.

#### 6.27.4.6 Natural Portfolio::\_nbAssets [private]

Definition at line 102 of file `PortFolio.h`.

Referenced by `addAsset()`, `getPrice()`, `Portfolio()`, and `returnSensibilityToRate()`.

#### 6.27.4.7 Natural Portfolio::\_nbBonds [private]

Definition at line 101 of file `PortFolio.h`.

Referenced by `addBond()`, `getPrice()`, `Portfolio()`, and `returnSensibilityToRate()`.

#### 6.27.4.8 `Natural Portfolio::_nbExoticsOptions` [private]

Definition at line 98 of file `PortFolio.h`.

Referenced by `addExoticOption()`, `getPrice()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

#### 6.27.4.9 `Natural Portfolio::_nbRainbowOptions` [private]

Definition at line 97 of file `PortFolio.h`.

Referenced by `addRainbowOption()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

#### 6.27.4.10 `Natural Portfolio::_nbVanSwaps` [private]

Definition at line 99 of file `PortFolio.h`.

Referenced by `addVanillaSwap()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, and `returnSensibilityToTime()`.

#### 6.27.4.11 `Natural Portfolio::_nbVarSwaps` [private]

Definition at line 100 of file `PortFolio.h`.

Referenced by `addVarianceSwap()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

#### 6.27.4.12 `OptionStrategy Portfolio::_optionStrategy` [private]

Definition at line 82 of file `PortFolio.h`.

Referenced by `addOptionStrategy()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

#### 6.27.4.13 `valarray<Real> Portfolio::_quantityAssets` [private]

Definition at line 95 of file `PortFolio.h`.

Referenced by `addAsset()`.

#### 6.27.4.14 `valarray<Real> Portfolio::_quantityBonds` [private]

Definition at line 94 of file `PortFolio.h`.

Referenced by `addBond()`.

#### 6.27.4.15 `valarray<Real> Portfolio::_quantityExoticsOptions` [private]

Definition at line 91 of file `PortFolio.h`.

Referenced by `addExoticOption()`.

#### **6.27.4.16** `valarray<Real> Portfolio::_quantityRainbowOptions` [private]

Definition at line 90 of file `PortFolio.h`.

Referenced by `addRainbowOption()`.

#### **6.27.4.17** `valarray<Real> Portfolio::_quantityVanSwaps` [private]

Definition at line 92 of file `PortFolio.h`.

Referenced by `addVanillaSwap()`.

#### **6.27.4.18** `valarray<Real> Portfolio::_quantityVarSwaps` [private]

Definition at line 93 of file `PortFolio.h`.

Referenced by `addVarianceSwap()`.

#### **6.27.4.19** `valarray<RainbowOption*> Portfolio::_rainbowOptions` [private]

Definition at line 83 of file `PortFolio.h`.

Referenced by `addRainbowOption()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

#### **6.27.4.20** `valarray<VanillaSwap*> Portfolio::_vanSwaps` [private]

Definition at line 85 of file `PortFolio.h`.

Referenced by `addVanillaSwap()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, and `returnSensibilityToTime()`.

#### **6.27.4.21** `valarray<VarianceSwap*> Portfolio::_varSwaps` [private]

Definition at line 86 of file `PortFolio.h`.

Referenced by `addVarianceSwap()`, `getPrice()`, `Portfolio()`, `returnSensibilityToRate()`, `returnSensibilityToTime()`, and `returnSensibilityToVol()`.

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

- **PortFolio.h**
- **PortFolio.cpp**

## 6.28 RainbowOption Class Reference

```
#include <rainbowoption.h>
```

### Public Member Functions

- **RainbowOption** (void)

*The default constructor will instantiate a rainbow such with : - a non correlated basket - of RO\_DEFAULT\_NB\_ASSETS assets, - equally weighted, and - with RO\_DEFAULT\_MULTIMPLIER.*

- **RainbowOption** (rainbowType type, Date startDate, Real expiry, Real Strike, yieldCurve yc, valarray< volsurface > vols, valarray< Real > spots=valarray< Real >(RO\_DEFAULT\_STRIKE, RO\_DEFAULT\_NB\_ASSETS), Real Multiplier=RO\_DEFAULT\_MULTIMPLIER, Matrix Correl=IdentityMatrix(RO\_DEFAULT\_NB\_ASSETS), valarray< Real > weights=valarray< Real >(1/(Real) RO\_DEFAULT\_NB\_ASSETS, RO\_DEFAULT\_NB\_ASSETS), bool outputMsgs=false)

*Full general constructor with n assets.*

- **RainbowOption** (rainbowType type, Date start, Real exp, Real Strike, yieldCurve yc, valarray< volsurface > vols, Real Spot1, Real Spot2, Real Mult=RO\_DEFAULT\_MULTIMPLIER, Real Correl12=0, Real weight1=0.5, Real weight2=0.5, bool outputMsgs=false)

*For 2 assets.*

- **~RainbowOption** (void)
- **Real getPrice** (priceType priceMethod=ClosedForm, LongNatural nPaths=RO\_NPATHS)
- **Real getPartialDelta** (Natural security, priceType priceMethod=ClosedForm)

*by convention, security 1 is the 0th spot in the array, so user "logical"*

- **Real getPartialGamma** (Natural security, priceType priceMethod=ClosedForm)

*by convention, security 1 is the 0th spot in the array, so user "logical"*

- **Real getPartialVega** (Natural security, priceType priceMethod=ClosedForm)

*by convention, security 1 is the 0th spot in the array, so user "logical"*

- **Real getDelta** (priceType priceMethod=ClosedForm)
- **Real getGamma** (priceType priceMethod=ClosedForm)
- **Real getVega** (priceType priceMethod=ClosedForm)
- **Real getCorrelRisk** (priceType priceMethod=ClosedForm)

*Correl Risk.*

- **Real getRho** (priceType priceMethod=ClosedForm)

*Rho risk.*

- **Real getTheta** (priceType priceMethod=ClosedForm)

*Theata risk.*

- **rainbowType getRainbowType** ()
- **void setRainbowType** (rainbowType newType)

## Private Member Functions

- Real PriceByMc\_2SpreadOptionMax (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_2AssetsBasketMax (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_BestOf2AssetsCash (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_WorstOf2AssetsCash (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_BetterOf2Assets (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_WorseOf2Assets (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_Max2AssetsCall (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_Min2AssetsCall (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_Max2AssetsPut (LongNatural nPaths=RO\_NPATHS)
- Real PriceByMc\_Min2AssetsPut (LongNatural nPaths=RO\_NPATHS)
- Real PriceByClosedForm\_BestOf2\_plusCash ()
- Real PriceByClosedForm\_BetterOf2 ()
- Real PriceByClosedForm\_WorseOf2 ()
- Real PriceByClosedForm\_MaxOf2\_call ()
- Real PriceByClosedForm\_MinOf2\_call ()
- Real PriceByClosedForm\_MaxOf2\_put ()
- Real PriceByClosedForm\_MinOf2\_put ()
- void reassignVolsAtThemoney ()
- void reassignVolsAtThestrike ()
- void instantiateMCVariables ()
- void compute\_sigmaA ()
- void compute\_rho1 ()
- void compute\_rho2 ()
- void compute\_d1 ()
- void compute\_d2 ()
- void compute\_d3 ()
- void compute\_d4 ()
- void compute\_A ()
- void compute\_B ()
- void compute\_C ()
- void compute\_ClosedFormsParameters ()

## Private Attributes

- bool \_outputMsgs
- Date \_startDate
- Real \_expiryInYears
- Natural \_NumberOfAssets
- Real \_Strike
- valarray< Real > \_spots
- Real \_Multiplier
- valarray< Real > \_weights
- Matrix \_CorrelationMatrix
- valarray< Real > \_volatilities
- valarray< volsurface > \_volatilitiesSurfaces
- yieldCurve \_yc
- rainbowType \_type
- PayOff \_thePayOff

- **Real** \_DFTomaturity
- **valarray**< **Drift** > \_Drifts
- **Random** \* \_pRandom
- **MCEngine** \_MCEngine
- **Real** \_gaussianSample
- **valarray**< **Real** > \_TerminalPoints
- **valarray**< **GaussianProcess** > \_pHazardRateProcesses
- **LongNatural** \_seed
- **Real** sigmaA
- **Real** rho1
- **Real** rho2
- **Real** d1
- **Real** d2
- **Real** d3
- **Real** d4
- **Real** A
- **Real** B
- **Real** C
- **bool** haveClosedFormVariablesBeenComputed

## 6.28.1 Constructor & Destructor Documentation

### 6.28.1.1 RainbowOption::RainbowOption (void)

The default constructor will instantiate a rainbow such with : - a non correlated basket - of RO\_DEFAULT\_NB\_ASSETS assets, - equally weighted, and - with RO\_DEFAULT\_MULTIPLIER.

Definition at line 3 of file rainbowoption.cpp.

References \_CorrelationMatrix, \_DFTomaturity, \_expiryInYears, \_Multiplier, \_NumberOfAssets, \_outputMsgs, \_pRandom, \_seed, \_spots, \_startDate, \_Strike, \_volatilities, \_volatilitiesSurfaces, \_weights, BestOf2AssetsCash, yieldCurve::discountFactor(), haveClosedFormVariablesBeenComputed, IdentityMatrix(), LongNatural, Real, RO\_DEFAULT\_MATURITY, RO\_DEFAULT\_NB\_ASSETS, RO\_DEFAULT\_RATE, RO\_DEFAULT\_STRIKE, RO\_DEFAULT\_VOL, RO\_SEED, and Date::setDateToToday().

**6.28.1.2 RainbowOption::RainbowOption (rainbowType *type*, Date *startDate*, Real *expiry*, Real *Strike*, yieldCurve *yc*, valarray< volsurface > *vols*, valarray< Real > *spots* = valarray< Real >(RO\_DEFAULT\_STRIKE, RO\_DEFAULT\_NB\_ASSETS), Real *Multiplier* = RO\_DEFAULT\_MULTIPLIER, Matrix *Correl* = IdentityMatrix(RO\_DEFAULT\_NB\_ASSETS), valarray< Real > *weights* = valarray< Real >(1/(Real)RO\_DEFAULT\_NB\_ASSETS, RO\_DEFAULT\_NB\_ASSETS), bool *outputMsgs* = false)**

Full general constructor with n assets.

Definition at line 29 of file rainbowoption.cpp.

References \_CorrelationMatrix, \_DFTomaturity, \_expiryInYears, \_NumberOfAssets, \_pRandom, \_seed, \_Strike, \_thePayOff, yieldCurve::discountFactor(), Matrix::GetRows(), haveClosedFormVariablesBeenComputed, LongNatural, Real, reassignVolsAtThestrike(), RO\_SEED, and PayOff::SetStrike().

**6.28.1.3 RainbowOption::RainbowOption** (rainbowType *type*, Date *start*, Real *exp*, Real *Strike*, yieldCurve *yc*, valarray< volsurface > *vols*, Real *Spot1*, Real *Spot2*, Real *Mult* = RO\_DEFAULT\_MULTIPLIER, Real *Correl12* = 0, Real *weight1* = 0.5, Real *weight2* = 0.5, bool *outputMsgs* = false)

For 2 assets.

**Parameters:**

Real Correl: the correlation between 1 and 2 - default is 0

Definition at line 52 of file rainbowoption.cpp.

References `_CorrelationMatrix`, `_DFTomaturity`, `_expiryInYears`, `_NumberOfAssets`, `_pRandom`, `_seed`, `_spots`, `_volatilities`, `_weights`, `yieldCurve::discountFactor()`, `haveClosedFormVariablesBeenComputed`, `IdentityMatrix()`, `LongNatural`, `Real`, `reassignVolsAtThestrike()`, `RO_SEED`, and `Matrix::SetValue()`.

**6.28.1.4 RainbowOption::~~RainbowOption** (void)

Definition at line 679 of file rainbowoption.cpp.

## 6.28.2 Member Function Documentation

**6.28.2.1 void RainbowOption::compute\_A** () [private]

Definition at line 541 of file rainbowoption.cpp.

References `_spots`, `A`, `CumulativeBivariateNormal()`, `CumulativeNormal()`, `d3`, `Real`, and `rho1`.

Referenced by `compute_ClosedFormsParameters()`.

**6.28.2.2 void RainbowOption::compute\_B** () [private]

Definition at line 547 of file rainbowoption.cpp.

References `_spots`, `B`, `CumulativeBivariateNormal()`, `CumulativeNormal()`, `d4`, `Real`, and `rho2`.

Referenced by `compute_ClosedFormsParameters()`.

**6.28.2.3 void RainbowOption::compute\_C** () [private]

Definition at line 553 of file rainbowoption.cpp.

References `_CorrelationMatrix`, `_expiryInYears`, `_Strike`, `_volatilities`, `a`, `C`, `CumulativeBivariateNormal()`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `compute_ClosedFormsParameters()`.

**6.28.2.4 void RainbowOption::compute\_ClosedFormsParameters** () [private]

Definition at line 564 of file rainbowoption.cpp.

References `compute_A()`, `compute_B()`, `compute_C()`, `compute_d1()`, `compute_d2()`, `compute_d3()`, `compute_d4()`, `compute_rho1()`, `compute_rho2()`, and `compute_sigmaA()`.



Referenced by `PriceByClosedForm_BestOf2_plusCash()`, `PriceByClosedForm_BetterOf2()`, `PriceByClosedForm_MaxOf2_call()`, and `PriceByClosedForm_MaxOf2_put()`.

#### 6.28.2.5 `void RainbowOption::compute_d1 ()` [private]

Definition at line 507 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_spots`, `_Strike`, `_volatilities`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.6 `void RainbowOption::compute_d2 ()` [private]

Definition at line 516 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_spots`, `_Strike`, `_volatilities`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.7 `void RainbowOption::compute_d3 ()` [private]

Definition at line 525 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_spots`, `d3`, `Real`, and `sigmaA`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.8 `void RainbowOption::compute_d4 ()` [private]

Definition at line 533 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_spots`, `d4`, `Real`, and `sigmaA`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.9 `void RainbowOption::compute_rho1 ()` [private]

Definition at line 491 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_volatilities`, `Real`, `rho1`, and `sigmaA`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.10 `void RainbowOption::compute_rho2 ()` [private]

Definition at line 499 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_volatilities`, `Real`, `rho2`, and `sigmaA`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.11 `void RainbowOption::compute_sigmaA ()` [private]

Definition at line 483 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_volatilities`, `Real`, and `sigmaA`.

Referenced by `compute_ClosedFormsParameters()`.

#### 6.28.2.12 Real RainbowOption::getCorrelRisk (priceType *priceMethod* = ClosedForm)

Correl Risk.

Definition at line 264 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_outputMsgs`, `getPrice()`, `Matrix::GetRows()`, `GREEKAPPROX`, `haveClosedFormVariablesBeenComputed`, `Natural`, `Real`, and `Matrix::SetValue()`.

Referenced by `inputRainbowOption()`, and `mainrainbowoptions()`.

#### 6.28.2.13 Real RainbowOption::getDelta (priceType *priceMethod* = ClosedForm)

##### Returns:

overall delta (if all the market shifts : sum of deltas)

Definition at line 240 of file `rainbowoption.cpp`.

References `_NumberOfAssets`, `getPartialDelta()`, `Natural`, `Real`, `Matrix::SetValue()`, and `Matrix::SumColumn()`.

#### 6.28.2.14 Real RainbowOption::getGamma (priceType *priceMethod* = ClosedForm)

##### Returns:

overall gamma (if all the market shifts : sum of gamma)

Definition at line 248 of file `rainbowoption.cpp`.

References `_NumberOfAssets`, `getPartialGamma()`, `Natural`, `Real`, `Matrix::SetValue()`, and `Matrix::SumColumn()`.

#### 6.28.2.15 Real RainbowOption::getPartialDelta (Natural *security*, priceType *priceMethod* = ClosedForm)

by convention, security 1 is the 0th spot in the array, so user "logical"

Definition at line 177 of file `rainbowoption.cpp`.

References `_NumberOfAssets`, `_outputMsgs`, `_spots`, `getPrice()`, `GREEKAPPROX`, `haveClosedFormVariablesBeenComputed`, `Natural`, and `Real`.

Referenced by `getDelta()`, `getPartialGamma()`, `inputRainbowOption()`, and `mainrainbowoptions()`.

#### 6.28.2.16 Real RainbowOption::getPartialGamma (Natural *security*, priceType *priceMethod* = ClosedForm)

by convention, security 1 is the 0th spot in the array, so user "logical"

Definition at line 198 of file `rainbowoption.cpp`.

References `_NumberOfAssets`, `_outputMsgs`, `_spots`, `getPartialDelta()`, `GREEKAPPROX`, `haveClosedFormVariablesBeenComputed`, `Natural`, and `Real`.

Referenced by `getGamma()`, `inputRainbowOption()`, and `mainrainbowoptions()`.

#### 6.28.2.17 Real RainbowOption::getPartialVega (Natural *security*, priceType *priceMethod* = ClosedForm)

by convention, security 1 is the 0th spot in the array, so user "logical"

Definition at line 219 of file `rainbowoption.cpp`.

References `_NumberOfAssets`, `_outputMsgs`, `_volatilities`, `getPrice()`, `GREEKAPPROX`, `haveClosedFormVariablesBeenComputed`, `Natural`, and `Real`.

Referenced by `getVega()`, `inputRainbowOption()`, and `mainrainbowoptions()`.

#### 6.28.2.18 Real RainbowOption::getPrice (priceType *priceMethod* = ClosedForm, LongNatural *nPaths* = RO\_NPATHS)

##### Parameters:

*price* type : MC (default as for this one we have all prices) - or CF

*nPaths* - only be used for MC, so default

Definition at line 85 of file `rainbowoption.cpp`.

References `_outputMsgs`, `AssetsBasketMax`, `BestOf2AssetsCash`, `BetterOf2Assets`, `ClosedForm`, `LongNatural`, `Max2AssetsCall`, `Max2AssetsPut`, `Min2AssetsCall`, `Min2AssetsPut`, `MonteCarlo`, `PriceByClosedForm_BestOf2_plusCash()`, `PriceByClosedForm_BetterOf2()`, `PriceByClosedForm_MaxOf2_call()`, `PriceByClosedForm_MaxOf2_put()`, `PriceByClosedForm_MinOf2_call()`, `PriceByClosedForm_MinOf2_put()`, `PriceByClosedForm_WorseOf2()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_BetterOf2Assets()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorseOf2Assets()`, `PriceByMc_WorstOf2AssetsCash()`, `Real`, `SpreadOptionMax`, `WorseOf2Assets`, and `WorstOf2AssetsCash`.

Referenced by `getCorrelRisk()`, `getPartialDelta()`, `getPartialVega()`, `getRho()`, `getTheta()`, `inputRainbowOption()`, and `mainrainbowoptions()`.

#### 6.28.2.19 rainbowType RainbowOption::getRainbowType () [inline]

Definition at line 116 of file `rainbowoption.h`.

References `rainbowType`.

#### 6.28.2.20 Real RainbowOption::getRho (priceType *priceMethod* = ClosedForm)

Rho risk.

Definition at line 310 of file `rainbowoption.cpp`.

References `_outputMsgs`, `getPrice()`, `GREEKAPPROX`, `haveClosedFormVariablesBeenComputed`, `Real`, and `yieldCurve::shiftZCBRateCurve()`.

Referenced by `inputRainbowOption()`, and `mainrainbowoptions()`.

**6.28.2.21 Real RainbowOption::getTheta (priceType *priceMethod* = ClosedForm)**

Theata risk.

Definition at line 331 of file rainbowoption.cpp.

References `_expiryInYears`, `_outputMsgs`, `getPrice()`, `haveClosedFormVariablesBeenComputed`, and `Real`.

**6.28.2.22 Real RainbowOption::getVega (priceType *priceMethod* = ClosedForm)****Returns:**

overall vega (if all the market shifts : sum of Vega)

Definition at line 256 of file rainbowoption.cpp.

References `_NumberOfAssets`, `getPartialVega()`, `Natural`, `Real`, `Matrix::SetValue()`, and `Matrix::SumColumn()`.

**6.28.2.23 void RainbowOption::instanciateMCVariables () [private]**

Definition at line 354 of file rainbowoption.cpp.

References `_Drifts`, `_expiryInYears`, `_gaussianSample`, `_NumberOfAssets`, `_pHazardRateProcesses`, `_pRandom`, `_spots`, `_startDate`, `_Strike`, `_TerminalPoints`, `_thePayOff`, `_volatilities`, `Natural`, `RO_SEED`, `Random::SetSeed()`, `PayOff::SetStrike()`, and `yieldCurve::spotRate()`.

Referenced by `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_BetterOf2Assets()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorseOf2Assets()`, and `PriceByMc_WorstOf2AssetsCash()`.

**6.28.2.24 Real RainbowOption::PriceByClosedForm\_BestOf2\_plusCash () [private]**

Definition at line 577 of file rainbowoption.cpp.

References `_outputMsgs`, `A`, `B`, `C`, `compute_ClosedFormsParameters()`, `haveClosedFormVariablesBeenComputed`, and `Real`.

Referenced by `getPrice()`.

**6.28.2.25 Real RainbowOption::PriceByClosedForm\_BetterOf2 () [private]**

Definition at line 587 of file rainbowoption.cpp.

References `_outputMsgs`, `_Strike`, `A`, `B`, `C`, `compute_ClosedFormsParameters()`, `EPSILON`, `haveClosedFormVariablesBeenComputed`, and `Real`.

Referenced by `getPrice()`, `PriceByClosedForm_MaxOf2_put()`, and `PriceByClosedForm_WorseOf2()`.

**6.28.2.26 Real RainbowOption::PriceByClosedForm\_MaxOf2\_call () [private]**

Definition at line 620 of file rainbowoption.cpp.

References `_expiryInYears`, `_outputMsgs`, `_Strike`, `A`, `B`, `C`, `compute_ClosedFormsParameters()`, `haveClosedFormVariablesBeenComputed`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `getPrice()`, `PriceByClosedForm_MaxOf2_put()`, and `PriceByClosedForm_MinOf2_call()`.

#### 6.28.2.27 Real RainbowOption::PriceByClosedForm\_MaxOf2\_put () [private]

Definition at line 651 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_outputMsgs`, `_Strike`, `compute_ClosedFormsParameters()`, `haveClosedFormVariablesBeenComputed`, `PriceByClosedForm_BetterOf2()`, `PriceByClosedForm_MaxOf2_call()`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `getPrice()`, and `PriceByClosedForm_MinOf2_put()`.

#### 6.28.2.28 Real RainbowOption::PriceByClosedForm\_MinOf2\_call () [private]

Definition at line 632 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_outputMsgs`, `_spots`, `_Strike`, `_volatilities`, `Call`, `BlackScholes::getPrice()`, `PriceByClosedForm_MaxOf2_call()`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `getPrice()`.

#### 6.28.2.29 Real RainbowOption::PriceByClosedForm\_MinOf2\_put () [private]

Definition at line 662 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_outputMsgs`, `_spots`, `_Strike`, `_volatilities`, `BlackScholes::getPrice()`, `PriceByClosedForm_MaxOf2_put()`, `Put`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `getPrice()`.

#### 6.28.2.30 Real RainbowOption::PriceByClosedForm\_WorseOf2 () [private]

Definition at line 601 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_outputMsgs`, `_spots`, `_volatilities`, `Call`, `EPSILON`, `BlackScholes::getPrice()`, `PriceByClosedForm_BetterOf2()`, `Real`, and `yieldCurve::spotRate()`.

Referenced by `getPrice()`.

#### 6.28.2.31 Real RainbowOption::PriceByMc\_2AssetsBasketMax (LongNatural *nPaths* = RO\_NPATHS) [private]

Definition at line 395 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_Multiplier`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbow2AssetsBasketMax()`.

Referenced by `getPrice()`.

**6.28.2.32 Real RainbowOption::PriceByMc\_2SpreadOptionMax (LongNatural *nPaths* = RO\_NPATHS) [private]**

Definition at line 383 of file rainbowoption.cpp.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_Multiplier`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbow2SpreadOptionMax()`.

Referenced by `getPrice()`.

**6.28.2.33 Real RainbowOption::PriceByMc\_BestOf2AssetsCash (LongNatural *nPaths* = RO\_NPATHS) [private]**

Definition at line 404 of file rainbowoption.cpp.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbowBestOf2AssetsCash()`.

Referenced by `getPrice()`.

**6.28.2.34 Real RainbowOption::PriceByMc\_BetterOf2Assets (LongNatural *nPaths* = RO\_NPATHS) [private]**

Definition at line 422 of file rainbowoption.cpp.

References `_Strike`, `EPSILON`, `instanciateMCVariables()`, `LongNatural`, `PriceByMc_Max2AssetsCall()`, and `Real`.

Referenced by `getPrice()`.

**6.28.2.35 Real RainbowOption::PriceByMc\_Max2AssetsCall (LongNatural *nPaths* = RO\_NPATHS) [private]**

Definition at line 447 of file rainbowoption.cpp.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_Multiplier`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbowMax2AssetsCall()`.

Referenced by `getPrice()`, and `PriceByMc_BetterOf2Assets()`.

**6.28.2.36 Real RainbowOption::PriceByMc\_Max2AssetsPut (LongNatural *nPaths* = RO\_NPATHS) [private]**

Definition at line 465 of file rainbowoption.cpp.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_Multiplier`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbowMax2AssetsPut()`.

Referenced by `getPrice()`.

#### 6.28.2.37 Real RainbowOption::PriceByMc\_Min2AssetsCall (LongNatural *nPaths* = RO\_NPATHS) [private]

Definition at line 456 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_Multiplier`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbowMin2AssetsCall()`.

Referenced by `getPrice()`, and `PriceByMc_WorseOf2Assets()`.

#### 6.28.2.38 Real RainbowOption::PriceByMc\_Min2AssetsPut (LongNatural *nPaths* = RO\_NPATHS) [private]

Definition at line 474 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_Multiplier`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbowMin2AssetsPut()`.

Referenced by `getPrice()`.

#### 6.28.2.39 Real RainbowOption::PriceByMc\_WorseOf2Assets (LongNatural *nPaths* = RO\_NPATHS) [private]

Definition at line 435 of file `rainbowoption.cpp`.

References `_Strike`, `EPSILON`, `instanciateMCVariables()`, `LongNatural`, `PriceByMc_Min2AssetsCall()`, and `Real`.

Referenced by `getPrice()`.

#### 6.28.2.40 Real RainbowOption::PriceByMc\_WorstOf2AssetsCash (LongNatural *nPaths* = RO\_NPATHS) [private]

Definition at line 413 of file `rainbowoption.cpp`.

References `_CorrelationMatrix`, `_DFTomaturity`, `_gaussianSample`, `_MCEngine`, `_pHazardRateProcesses`, `_pRandom`, `_TerminalPoints`, `_thePayOff`, `_weights`, `instanciateMCVariables()`, `LongNatural`, `MCEngine::MCResult()`, `Real`, and `MCEngine::RunEngineRainbowWorstOf2AssetsCash()`.

Referenced by `getPrice()`.

#### 6.28.2.41 void RainbowOption::reassignVolsAtThemoney () [private]

Definition at line 378 of file `rainbowoption.cpp`.

References `_expiryInYears`, `_NumberOfAssets`, `_spots`, `_startDate`, `_volatilities`, `_volatilitiesSurfaces`, `Natural`, and `Date::plusDays()`.

**6.28.2.42 void RainbowOption::reassignVolsAtThestrike () [private]**

Definition at line 372 of file rainbowoption.cpp.

References `_expiryInYears`, `_NumberOfAssets`, `_startDate`, `_Strike`, `_volatilities`, `_volatilitiesSurfaces`, `Natural`, and `Date::plusDays()`.

Referenced by `RainbowOption()`.

**6.28.2.43 void RainbowOption::setRainbowType (rainbowType *newType*) [inline]**

Definition at line 117 of file rainbowoption.h.

Referenced by `mainrainbowoptions()`.

**6.28.3 Member Data Documentation****6.28.3.1 Matrix RainbowOption::\_CorrelationMatrix [private]**

Definition at line 158 of file rainbowoption.h.

Referenced by `compute_C()`, `compute_rho1()`, `compute_rho2()`, `compute_sigmaA()`, `getCorrelRisk()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorstOf2AssetsCash()`, and `RainbowOption()`.

**6.28.3.2 Real RainbowOption::\_DFTomaturity [private]**

Definition at line 171 of file rainbowoption.h.

Referenced by `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorstOf2AssetsCash()`, and `RainbowOption()`.

**6.28.3.3 valarray<Drift> RainbowOption::\_Drifts [private]**

Definition at line 172 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`.

**6.28.3.4 Real RainbowOption::\_expiryInYears [private]**

Definition at line 152 of file rainbowoption.h.

Referenced by `compute_C()`, `compute_d1()`, `compute_d2()`, `compute_d3()`, `compute_d4()`, `getTheta()`, `instanciateMCVariables()`, `PriceByClosedForm_MaxOf2_call()`, `PriceByClosedForm_MaxOf2_put()`, `PriceByClosedForm_MinOf2_call()`, `PriceByClosedForm_MinOf2_put()`, `PriceByClosedForm_WorseOf2()`, `RainbowOption()`, `reassignVolsAtThemoney()`, and `reassignVolsAtThestrike()`.



**6.28.3.5 Real RainbowOption::\_gaussianSample [private]**

Definition at line 175 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, and `PriceByMc_WorstOf2AssetsCash()`.

**6.28.3.6 MCEngine RainbowOption::\_MCEngine [private]**

Definition at line 174 of file rainbowoption.h.

Referenced by `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, and `PriceByMc_WorstOf2AssetsCash()`.

**6.28.3.7 Real RainbowOption::\_Multiplier [private]**

Definition at line 156 of file rainbowoption.h.

Referenced by `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, and `RainbowOption()`.

**6.28.3.8 Natural RainbowOption::\_NumberOfAssets [private]**

Definition at line 153 of file rainbowoption.h.

Referenced by `getDelta()`, `getGamma()`, `getPartialDelta()`, `getPartialGamma()`, `getPartialVega()`, `getVega()`, `instanciateMCVariables()`, `RainbowOption()`, `reassignVolsAtThemoney()`, and `reassignVolsAtThestrike()`.

**6.28.3.9 bool RainbowOption::\_outputMsgs [private]**

Definition at line 148 of file rainbowoption.h.

Referenced by `getCorrelRisk()`, `getPartialDelta()`, `getPartialGamma()`, `getPartialVega()`, `getPrice()`, `getRho()`, `getTheta()`, `PriceByClosedForm_BestOf2_plusCash()`, `PriceByClosedForm_BetterOf2()`, `PriceByClosedForm_MaxOf2_call()`, `PriceByClosedForm_MaxOf2_put()`, `PriceByClosedForm_MinOf2_call()`, `PriceByClosedForm_MinOf2_put()`, `PriceByClosedForm_WorseOf2()`, and `RainbowOption()`.

**6.28.3.10 valarray<GaussianProcess> RainbowOption::\_pHazardRateProcesses [private]**

Definition at line 177 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, and `PriceByMc_WorstOf2AssetsCash()`.

**6.28.3.11 Random\* RainbowOption::\_pRandom** [private]

Definition at line 173 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorstOf2AssetsCash()`, and `RainbowOption()`.

**6.28.3.12 LongNatural RainbowOption::\_seed** [private]

Definition at line 180 of file rainbowoption.h.

Referenced by `RainbowOption()`.

**6.28.3.13 valarray<Real> RainbowOption::\_spots** [private]

Definition at line 155 of file rainbowoption.h.

Referenced by `compute_A()`, `compute_B()`, `compute_d1()`, `compute_d2()`, `compute_d3()`, `compute_d4()`, `getPartialDelta()`, `getPartialGamma()`, `instanciateMCVariables()`, `PriceByClosedForm_MinOf2_call()`, `PriceByClosedForm_MinOf2_put()`, `PriceByClosedForm_WorseOf2()`, `RainbowOption()`, and `reassignVolsAtThemoney()`.

**6.28.3.14 Date RainbowOption::\_startDate** [private]

Definition at line 151 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`, `RainbowOption()`, `reassignVolsAtThemoney()`, and `reassignVolsAtThestrike()`.

**6.28.3.15 Real RainbowOption::\_Strike** [private]

Definition at line 154 of file rainbowoption.h.

Referenced by `compute_C()`, `compute_d1()`, `compute_d2()`, `instanciateMCVariables()`, `PriceByClosedForm_BetterOf2()`, `PriceByClosedForm_MaxOf2_call()`, `PriceByClosedForm_MaxOf2_put()`, `PriceByClosedForm_MinOf2_call()`, `PriceByClosedForm_MinOf2_put()`, `PriceByMc_BetterOf2Assets()`, `PriceByMc_WorseOf2Assets()`, `RainbowOption()`, and `reassignVolsAtThestrike()`.

**6.28.3.16 valarray<Real> RainbowOption::\_TerminalPoints** [private]

Definition at line 176 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, and `PriceByMc_WorstOf2AssetsCash()`.

**6.28.3.17 PayOff RainbowOption::\_thePayOff** [private]

Definition at line 170 of file rainbowoption.h.

Referenced by `instanciateMCVariables()`, `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorstOf2AssetsCash()`, and `RainbowOption()`.

#### 6.28.3.18 `rainbowType RainbowOption::_type` [private]

Definition at line 163 of file `rainbowoption.h`.

#### 6.28.3.19 `valarray<Real> RainbowOption::_volatilities` [private]

Definition at line 159 of file `rainbowoption.h`.

Referenced by `compute_C()`, `compute_d1()`, `compute_d2()`, `compute_rho1()`, `compute_rho2()`, `compute_sigmaA()`, `getPartialVega()`, `instanciateMCVariables()`, `PriceByClosedForm_MinOf2_call()`, `PriceByClosedForm_MinOf2_put()`, `PriceByClosedForm_WorseOf2()`, `RainbowOption()`, `reassignVolsAtThemoney()`, and `reassignVolsAtThestrike()`.

#### 6.28.3.20 `valarray<volsurface> RainbowOption::_volatilitiesSurfaces` [private]

Definition at line 160 of file `rainbowoption.h`.

Referenced by `RainbowOption()`, `reassignVolsAtThemoney()`, and `reassignVolsAtThestrike()`.

#### 6.28.3.21 `valarray<Real> RainbowOption::_weights` [private]

Definition at line 157 of file `rainbowoption.h`.

Referenced by `PriceByMc_2AssetsBasketMax()`, `PriceByMc_2SpreadOptionMax()`, `PriceByMc_BestOf2AssetsCash()`, `PriceByMc_Max2AssetsCall()`, `PriceByMc_Max2AssetsPut()`, `PriceByMc_Min2AssetsCall()`, `PriceByMc_Min2AssetsPut()`, `PriceByMc_WorstOf2AssetsCash()`, and `RainbowOption()`.

#### 6.28.3.22 `yieldCurve RainbowOption::_yc` [private]

Definition at line 161 of file `rainbowoption.h`.

#### 6.28.3.23 `Real RainbowOption::A` [private]

Definition at line 197 of file `rainbowoption.h`.

Referenced by `compute_A()`, `PriceByClosedForm_BestOf2_plusCash()`, `PriceByClosedForm_BetterOf2()`, and `PriceByClosedForm_MaxOf2_call()`.

#### 6.28.3.24 `Real RainbowOption::B` [private]

Definition at line 199 of file `rainbowoption.h`.

Referenced by `compute_B()`, `PriceByClosedForm_BestOf2_plusCash()`, `PriceByClosedForm_BetterOf2()`, and `PriceByClosedForm_MaxOf2_call()`.

**6.28.3.25 Real RainbowOption::C** [private]

Definition at line 201 of file rainbowoption.h.

Referenced by compute\_C(), PriceByClosedForm\_BestOf2\_plusCash(), PriceByClosedForm\_BetterOf2(), and PriceByClosedForm\_MaxOf2\_call().

**6.28.3.26 Real RainbowOption::d1** [private]

Definition at line 189 of file rainbowoption.h.

**6.28.3.27 Real RainbowOption::d2** [private]

Definition at line 191 of file rainbowoption.h.

**6.28.3.28 Real RainbowOption::d3** [private]

Definition at line 193 of file rainbowoption.h.

Referenced by compute\_A(), and compute\_d3().

**6.28.3.29 Real RainbowOption::d4** [private]

Definition at line 195 of file rainbowoption.h.

Referenced by compute\_B(), and compute\_d4().

**6.28.3.30 bool RainbowOption::haveClosedFormVariablesBeenComputed**  
[private]

Definition at line 204 of file rainbowoption.h.

Referenced by getCorrelRisk(), getPartialDelta(), getPartialGamma(), getPartialVega(), getRho(), getTheta(), PriceByClosedForm\_BestOf2\_plusCash(), PriceByClosedForm\_BetterOf2(), PriceByClosedForm\_MaxOf2\_call(), PriceByClosedForm\_MaxOf2\_put(), and RainbowOption().

**6.28.3.31 Real RainbowOption::rho1** [private]

Definition at line 185 of file rainbowoption.h.

Referenced by compute\_A(), and compute\_rho1().

**6.28.3.32 Real RainbowOption::rho2** [private]

Definition at line 187 of file rainbowoption.h.

Referenced by compute\_B(), and compute\_rho2().

**6.28.3.33 Real RainbowOption::sigmaA** [private]

Definition at line 183 of file rainbowoption.h.

Referenced by `compute_d3()`, `compute_d4()`, `compute_rho1()`, `compute_rho2()`, and `compute_sigmaA()`.

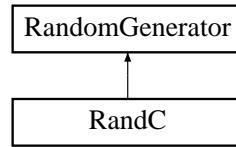
The documentation for this class was generated from the following files:

- `rainbowoption.h`
- `rainbowoption.cpp`

## 6.29 RandC Class Reference

```
#include <RandC.h>
```

Inheritance diagram for RandC::



### Public Member Functions

- **RandC** (**LongNatural** Seed\_<sub>=0</sub>)  
*Default constructor: initialize variable.*
- **~RandC** (void)
- **LongNatural GetOneRandomInteger** ()  
*Create one random integer.*
- **Real getUniform** ()  
*Creates one uniform number on (0.0,1.0).*
- void **SetSeed** (**LongNatural** Seed)  
*Set seed for generator.*
- **VeryLongNatural Max** ()  
*Return maximum number of random numbers.*
- **LongNatural Min** ()  
*Return minimum of numbers generated.*

### Private Attributes

- **LongNatural** Seed

#### 6.29.1 Constructor & Destructor Documentation

##### 6.29.1.1 RandC::RandC (LongNatural Seed\_ = 0)

Default constructor: initialize variable.

Definition at line 8 of file RandC.cpp.

References LongNatural.

##### 6.29.1.2 RandC::~~RandC (void)

Definition at line 14 of file RandC.cpp.

## 6.29.2 Member Function Documentation

### 6.29.2.1 LongNatural RandC::GetOneRandomInteger () [virtual]

Create one random integer.

Implements **RandomGenerator** (p.190).

Definition at line 26 of file RandC.cpp.

References LongNatural, and Maxim.

Referenced by getUniform().

### 6.29.2.2 Real RandC::getUniform () [virtual]

Creates one uniform number on (0.0,1.0).

Implements **RandomGenerator** (p.190).

Definition at line 30 of file RandC.cpp.

References GetOneRandomInteger(), Max(), and Real.

### 6.29.2.3 VeryLongNatural RandC::Max () [virtual]

Return maximum number of random numbers.

Implements **RandomGenerator** (p.190).

Definition at line 18 of file RandC.cpp.

References Maxim, and VeryLongNatural.

Referenced by getUniform().

### 6.29.2.4 LongNatural RandC::Min () [virtual]

Return minimum of numbers generated.

Implements **RandomGenerator** (p.190).

Definition at line 22 of file RandC.cpp.

References LongNatural.

### 6.29.2.5 void RandC::SetSeed (LongNatural *Seed*) [virtual]

Set seed for generator.

Reimplemented from **RandomGenerator** (p.190).

Definition at line 34 of file RandC.cpp.

References LongNatural.

### 6.29.3 Member Data Documentation

#### 6.29.3.1 LongNatural RandC::Seed [private]

Reimplemented from **RandomGenerator** (p. 191).

Definition at line 28 of file RandC.h.

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

- **RandC.h**
- **RandC.cpp**



## 6.30 Random Class Reference

```
#include <Random.h>
```

### Public Member Functions

- **Random** (**LongNatural** Dimensionality, **RandomGenerator** \*rndGen)  
*Default constructor: set dimension.*
- **Random** (**RandomGenerator** \*rndGen)
- **LongNatural** **GetDimensionality** () const  
*Return dimension.*
- **Random** \* **clone** () const  
*Clone function.*
- void **GetUniforms** (valarray< **Real** > &variates)  
*Get uniforms Real.*
- void **GetUniform** (**Real** &variate)
- void **Skip** (**LongNatural** numberOfPaths)
- void **SetSeed** (**LongNatural** Seed)
- void **Reset** ()
- void **GetGaussians** (valarray< **Real** > &variates)  
*Get gaussian random numbers.*
- void **GetGaussian** (**Real** &variate)
- void **ResetDimensionality** (**LongNatural** NewDimensionality)

### Private Attributes

- **RandomGenerator** \* **InnerGenerator**
- **LongNatural** **Dimensionality**
- **LongNatural** **InitialSeed**
- **Real** **Reciprocal**

#### 6.30.1 Constructor & Destructor Documentation

##### 6.30.1.1 **Random::Random** (**LongNatural** *Dimensionality*, **RandomGenerator** \**rndGen*)

Default constructor: set dimension.

Definition at line 3 of file Random.cpp.

References [InnerGenerator](#), and [LongNatural](#).

Referenced by [clone\(\)](#).

### 6.30.1.2 Random::Random (RandomGenerator \* *rndGen*)

**Author:**

Yann if dimension 1 - ne need to specify dimensionality

Definition at line 10 of file Random.cpp.

References Dimensionality, and InnerGenerator.

## 6.30.2 Member Function Documentation

### 6.30.2.1 Random \* Random::clone () const

Clone function.

Definition at line 15 of file Random.cpp.

References Random().

### 6.30.2.2 LongNatural Random::GetDimensionality () const [inline]

Return dimension.

Definition at line 50 of file Random.h.

References Dimensionality, and LongNatural.

Referenced by GetUniforms(), and Skip().

### 6.30.2.3 void Random::GetGaussian (Real & *variate*) [inline]

**Author:**

Yann - Get gaussian random number in a Real to avoid array

Definition at line 66 of file Random.h.

References GetUniform(), InverseCumulativeNormal(), and Real.

Referenced by MCEngine::RunEngineRainbow2AssetsBasketMax(), MCEngine::RunEngineRainbow2SpreadOptionMax(), MCEngine::RunEngineRainbowBestOf2AssetsCash(), MCEngine::RunEngineRainbowMax2AssetsCall(), MCEngine::RunEngineRainbowMax2AssetsPut(), MCEngine::RunEngineRainbowMin2AssetsCall(), MCEngine::RunEngineRainbowMin2AssetsPut(), and MCEngine::RunEngineRainbowWorstOf2AssetsCash().

### 6.30.2.4 void Random::GetGaussians (valarray< Real > & *variates*) [inline]

Get gaussian random numbers.

Definition at line 55 of file Random.h.

References Dimensionality, GetUniforms(), InverseCumulativeNormal(), LongNatural, and Real.

Referenced by MCEngine::RunEngineAsianCall(), MCEngine::RunEngineAsianPut(), MCEngine::RunEngineBarrierCall(), MCEngine::RunEngineBarrierPut(), MCEngine::RunEngineCall(), MCEngine::RunEngineCappedCliquet(), MCEngine::RunEngineFlooredCliquet(), MCEngine::RunEnginePut(), MCEngine::RunEngineRevLookbackCall(), and MCEngine::RunEngineRevLookbackPut().

**6.30.2.5 void Random::GetUniform (Real & *variate*)****Author:**

Yann - Get uniform in a Real to avoid arrays

Definition at line 26 of file Random.cpp.

References RandomGenerator::getUniform(), InnerGenerator, and Real.

Referenced by GetGaussian().

**6.30.2.6 void Random::GetUniforms (valarray< Real > & *variates*) [inline]**

Get uniforms Real.

Definition at line 20 of file Random.cpp.

References GetDimensionality(), RandomGenerator::getUniform(), InnerGenerator, and LongNatural.

Referenced by GetGaussians(), and Skip().

**6.30.2.7 void Random::Reset ()**

Definition at line 44 of file Random.cpp.

References InitialSeed, InnerGenerator, and RandomGenerator::SetSeed().

**6.30.2.8 void Random::ResetDimensionality (LongNatural *NewDimensionality*)**

Definition at line 50 of file Random.cpp.

References Dimensionality, InitialSeed, InnerGenerator, LongNatural, and RandomGenerator::SetSeed().

**6.30.2.9 void Random::SetSeed (LongNatural *Seed*)**

Definition at line 38 of file Random.cpp.

References InitialSeed, InnerGenerator, LongNatural, and RandomGenerator::SetSeed().

Referenced by RainbowOption::instanciateMCVariables(), and mainmc().

**6.30.2.10 void Random::Skip (LongNatural *numberOfPaths*)**

Definition at line 31 of file Random.cpp.

References GetDimensionality(), GetUniforms(), and LongNatural.

**6.30.3 Member Data Documentation****6.30.3.1 LongNatural Random::Dimensionality [private]**

Definition at line 45 of file Random.h.

Referenced by GetDimensionality(), GetGaussians(), Random(), and ResetDimensionality().

**6.30.3.2 LongNatural Random::InitialSeed** [private]

Definition at line 46 of file Random.h.

Referenced by Reset(), ResetDimensionality(), and SetSeed().

**6.30.3.3 RandomGenerator\* Random::InnerGenerator** [private]

Definition at line 44 of file Random.h.

Referenced by GetUniform(), GetUniforms(), Random(), Reset(), ResetDimensionality(), and SetSeed().

**6.30.3.4 Real Random::Reciprocal** [private]

Definition at line 47 of file Random.h.

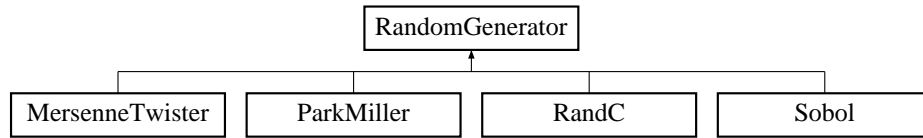
The documentation for this class was generated from the following files:

- **Random.h**
- **Random.cpp**

## 6.31 RandomGenerator Class Reference

```
#include <RandomGenerator.h>
```

Inheritance diagram for RandomGenerator::



### Public Member Functions

- **RandomGenerator** (**LongNatural** Seed\_=0)  
*Default constructor: initialize variable.*
- virtual **~RandomGenerator** (void)
- virtual **LongNatural** **GetOneRandomInteger** ()=0  
*Create one random integer.*
- virtual **Real** **getUniform** ()=0  
*Creates one uniform number on (0.0,1.0).*
- virtual void **SetSeed** (**LongNatural** Seed)  
*Set seed for generator.*
- virtual **VeryLongNatural** **Max** ()=0  
*Return maximum number of random numbers.*
- virtual **LongNatural** **Min** ()=0  
*Return minimum of numbers generated.*

### Private Attributes

- **LongNatural** Seed

#### 6.31.1 Constructor & Destructor Documentation

##### 6.31.1.1 RandomGenerator::RandomGenerator (LongNatural Seed\_ = 0)

Default constructor: initialize variable.

Definition at line 3 of file RandomGenerator.cpp.

References LongNatural, and SetSeed().

##### 6.31.1.2 RandomGenerator::~~RandomGenerator (void) [virtual]

Definition at line 8 of file RandomGenerator.cpp.

## 6.31.2 Member Function Documentation

### 6.31.2.1 virtual LongNatural RandomGenerator::GetOneRandomInteger () [pure virtual]

Create one random integer.

Implemented in **MersenneTwister** (p. 138), **ParkMiller** (p. 149), **RandC** (p. 183), and **Sobol** (p. 196).

Referenced by `getUniform()`.

### 6.31.2.2 Real RandomGenerator::getUniform () [pure virtual]

Creates one uniform number on (0.0,1.0).

Implemented in **MersenneTwister** (p. 138), **ParkMiller** (p. 149), **RandC** (p. 183), and **Sobol** (p. 196).

Definition at line 19 of file `RandomGenerator.cpp`.

References `GetOneRandomInteger()`, `Max()`, and `Real`.

Referenced by `Random::GetUniform()`, and `Random::GetUniforms()`.

### 6.31.2.3 virtual VeryLongNatural RandomGenerator::Max () [pure virtual]

Return maximum number of random numbers.

Implemented in **MersenneTwister** (p. 138), **ParkMiller** (p. 149), **RandC** (p. 183), and **Sobol** (p. 196).

Referenced by `getUniform()`.

### 6.31.2.4 virtual LongNatural RandomGenerator::Min () [pure virtual]

Return minimum of numbers generated.

Implemented in **MersenneTwister** (p. 138), **ParkMiller** (p. 149), **RandC** (p. 183), and **Sobol** (p. 196).

### 6.31.2.5 void RandomGenerator::SetSeed (LongNatural *Seed*) [virtual]

Set seed for generator.

Reimplemented in **MersenneTwister** (p. 139), **ParkMiller** (p. 149), **RandC** (p. 183), and **Sobol** (p. 196).

Definition at line 12 of file `RandomGenerator.cpp`.

References `LongNatural`.

Referenced by `RandomGenerator()`, `Random::Reset()`, `Random::ResetDimensionality()`, and `Random::SetSeed()`.

### 6.31.3 Member Data Documentation

#### 6.31.3.1 LongNatural RandomGenerator::Seed [private]

Reimplemented in **ParkMiller** (p. 149), **RandC** (p. 184), and **Sobol** (p. 197).

Definition at line 30 of file RandomGenerator.h.

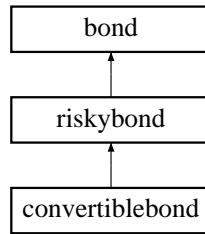
The documentation for this class was generated from the following files:

- **RandomGenerator.h**
- **RandomGenerator.cpp**

## 6.32 riskybond Class Reference

```
#include <bond.h>
```

Inheritance diagram for riskybond::



### Public Member Functions

- **riskybond** (**Date** issue, **Date** maturity, **Date** firstcoupondate, **Real** coupon, **Frequency** freq, **Real** faceamount, **DayCountConvention** daycount, **yieldCurve** yc, **creditCurve** cc)

*Constructor.*

- **riskybond** (**Date** issue, **Date** maturity, **Real** faceamount, **DayCountConvention** daycount, **yieldCurve** yc, **creditCurve** cc)

*Constructor : ZCbond.*

- **~riskybond** (void)

*Destructor.*

- virtual **Real** **quotedPrice** (**Date** today)
- virtual **riskybond** **shiftedbond** (**Real** shift)

*bond with same parameters and a shifted yieldCurve(p. 220)*

- virtual **Real** **rho** (**Date** today)

*return the derivative of the bond price with respect to interest rates*

- virtual **Real** **rho** ()

### Private Attributes

- **creditCurve** \_cc

#### 6.32.1 Constructor & Destructor Documentation

- ##### 6.32.1.1 riskybond::riskybond (**Date** *issue*, **Date** *maturity*, **Date** *firstcoupondate*, **Real** *coupon*, **Frequency** *freq*, **Real** *faceamount*, **DayCountConvention** *daycount*, **yieldCurve** *yc*, **creditCurve** *cc*)

Constructor.



**Parameters:***issue*: date of issue of the bond*maturity*: maturity of the bond*firstcoupondate*: date of the first coupon*coupon*: coupon of the bond, express as a percentqge of the faceamount*freq*: frequency of the coupon*faceamount*: par value*daycount*: daycount convention*yc*: yieldcurve*cc*: creditcurve

Definition at line 336 of file bond.cpp.

References Real.

**6.32.1.2 riskybond::riskybond (Date *issue*, Date *maturity*, Real *faceamount*, DayCountConvention *daycount*, yieldCurve *yc*, creditCurve *cc*)**

Constructor : ZCbond.

Definition at line 342 of file bond.cpp.

References Once, and Real.

**6.32.1.3 riskybond::~~riskybond (void) [inline]**

Destructor.

Definition at line 135 of file bond.h.

**6.32.2 Member Function Documentation****6.32.2.1 Real riskybond::quotedPrice (Date *today*) [virtual]**Reimplemented from **bond** (p. 36).

Definition at line 348 of file bond.cpp.

References Date::dayCount(), bond::getCashflow(), cashflow::getCashflows(), cashflow::getDates(), Natural, Real, and creditCurve::riskyDiscountFactor().

**6.32.2.2 virtual Real riskybond::rho () [inline, virtual]**Reimplemented in **convertiblebond** (p. 48).

Definition at line 144 of file bond.h.

References Real.

### 6.32.2.3 Real riskybond::rho (Date *today*) [virtual]

return the derivative of the bond price with respect to interest rates

Reimplemented in **convertiblebond** (p. 48).

Definition at line 380 of file bond.cpp.

References bond::fairvalue(), Real, and shiftedbond().

### 6.32.2.4 riskybond riskybond::shiftedbond (Real *shift*) [virtual]

bond with same parameters and a shifted **yieldCurve**(p. 220)

Definition at line 374 of file bond.cpp.

References Real, and yieldCurve::shiftZCBRateCurve().

Referenced by rho(), and convertiblebond::shiftedcbond().

## 6.32.3 Member Data Documentation

### 6.32.3.1 creditCurve riskybond::\_cc [private]

Definition at line 114 of file bond.h.

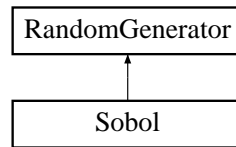
The documentation for this class was generated from the following files:

- **bond.h**
- **bond.cpp**

## 6.33 Sobol Class Reference

```
#include <Sobol.h>
```

Inheritance diagram for Sobol::



### Public Member Functions

- **Sobol** (**LongNatural** Seed\_ = 0)  
*Default constructor.*
- **~Sobol** ()
- void **sobseq** (**Integer** \*n, **Real** x[])
- **LongNatural** **GetOneRandomInteger** ()  
*Create one random integer.*
- **Real** **getUniform** ()  
*Creates one uniform number on (0.0,1.0).*
- void **SetSeed** (**LongNatural** Seed)
- **VeryLongNatural** **Max** ()  
*Return maximum number of random numbers.*
- **LongNatural** **Min** ()  
*Return minimum of numbers generated.*

### Private Attributes

- **LongNatural** Seed
- **Integer** n\_
- **Real** x\_ [MAXDIM+1]

#### 6.33.1 Constructor & Destructor Documentation

##### 6.33.1.1 Sobol::Sobol (**LongNatural** Seed\_ = 0)

Default constructor.

Definition at line 4 of file Sobol.cpp.

References `LongNatural`, `n_`, `sobseq()`, and `x_`.

### 6.33.1.2 Sobol::~~Sobol ()

Definition at line 12 of file Sobol.cpp.

## 6.33.2 Member Function Documentation

### 6.33.2.1 LongNatural Sobol::GetOneRandomInteger () [virtual]

Create one random integer.

Implements **RandomGenerator** (p.190).

Definition at line 79 of file Sobol.cpp.

References LongNatural.

### 6.33.2.2 Real Sobol::getUniform () [virtual]

Creates one uniform number on (0.0,1.0).

Implements **RandomGenerator** (p.190).

Definition at line 84 of file Sobol.cpp.

References n\_, Real, sobseq(), and x\_.

### 6.33.2.3 VeryLongNatural Sobol::Max () [virtual]

Return maximum number of random numbers.

Implements **RandomGenerator** (p.190).

Definition at line 71 of file Sobol.cpp.

References VeryLongNatural.

### 6.33.2.4 LongNatural Sobol::Min () [virtual]

Return minimum of numbers generated.

Implements **RandomGenerator** (p.190).

Definition at line 75 of file Sobol.cpp.

References LongNatural.

### 6.33.2.5 void Sobol::SetSeed (LongNatural *Seed*) [virtual]

Set seed for generator.

Reimplemented from **RandomGenerator** (p.190).

Definition at line 16 of file Sobol.cpp.

References LongNatural, n\_, sobseq(), and x\_.

### 6.33.2.6 void Sobol::sobseq (Integer \* n, Real x[])

Definition at line 23 of file Sobol.cpp.

References Integer, LongInteger, LongNatural, MAXBIT, Natural, and Real.

Referenced by getUniform(), SetSeed(), and Sobol().

## 6.33.3 Member Data Documentation

### 6.33.3.1 Integer Sobol::n\_ [private]

Definition at line 38 of file Sobol.h.

Referenced by getUniform(), SetSeed(), and Sobol().

### 6.33.3.2 LongNatural Sobol::Seed [private]

Reimplemented from **RandomGenerator** (p. 191).

Definition at line 37 of file Sobol.h.

### 6.33.3.3 Real Sobol::x\_[MAXDIM+1] [private]

Definition at line 39 of file Sobol.h.

Referenced by getUniform(), SetSeed(), and Sobol().

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

- **Sobol.h**
- **Sobol.cpp**

## 6.34 StringTokenizer Class Reference

```
#include <StringTokenizer.h>
```

### Public Member Functions

- **StringTokenizer** (const std::string &\_str, const std::string &\_delim)
- **~StringTokenizer** ()
- int **countTokens** ()
- bool **hasMoreTokens** ()
- std::string **nextToken** ()
- int **nextIntToken** ()
- double **nextFloatToken** ()
- std::string **nextToken** (const std::string &delim)
- std::string **remainingString** ()
- std::string **filterNextToken** (const std::string &filterStr)

### Private Attributes

- std::string **token\_str**
- std::string **delim**

### 6.34.1 Constructor & Destructor Documentation

#### 6.34.1.1 StringTokenizer::StringTokenizer (const std::string & \_str, const std::string & \_delim)

Definition at line 3 of file StringTokenizer.cpp.

References `delim`, and `token_str`.

#### 6.34.1.2 StringTokenizer::~StringTokenizer () [inline]

Definition at line 34 of file StringTokenizer.h.

### 6.34.2 Member Function Documentation

#### 6.34.2.1 int StringTokenizer::countTokens ()

Definition at line 52 of file StringTokenizer.cpp.

References `delim`, and `token_str`.

#### 6.34.2.2 std::string StringTokenizer::filterNextToken (const std::string & filterStr)

Definition at line 155 of file StringTokenizer.cpp.

References `nextToken()`.

**6.34.2.3 bool StringTokenizer::hasMoreTokens ()**

Definition at line 84 of file StringTokenizer.cpp.

References token\_str.

**6.34.2.4 double StringTokenizer::nextFloatToken ()**

Definition at line 120 of file StringTokenizer.cpp.

References nextToken().

**6.34.2.5 int StringTokenizer::nextIntToken ()**

Definition at line 114 of file StringTokenizer.cpp.

References nextToken().

Referenced by CSVParser::operator>>().

**6.34.2.6 std::string StringTokenizer::nextToken (const std::string & *delim*)**

Definition at line 126 of file StringTokenizer.cpp.

References token\_str.

**6.34.2.7 std::string StringTokenizer::nextToken ()**

Definition at line 90 of file StringTokenizer.cpp.

References delim, and token\_str.

Referenced by filterNextToken(), nextFloatToken(), and nextIntToken().

**6.34.2.8 std::string StringTokenizer::remainingString ()**

Definition at line 149 of file StringTokenizer.cpp.

References token\_str.

**6.34.3 Member Data Documentation****6.34.3.1 std::string StringTokenizer::delim [private]**

Definition at line 48 of file StringTokenizer.h.

Referenced by countTokens(), nextToken(), and StringTokenizer().

**6.34.3.2 std::string StringTokenizer::token\_str [private]**

Definition at line 47 of file StringTokenizer.h.

Referenced by countTokens(), hasMoreTokens(), nextToken(), remainingString(), and StringTokenizer().

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

- **StringTokenizer.h**
- **StringTokenizer.cpp**



## 6.35 SwapLeg Class Reference

```
#include <SwapLeg.h>
```

### Public Member Functions

- **SwapLeg** (**Date** startDate, **Real** Frequency, **Date** endDate, **Real** Notional, **Real** AmortizingConstant, **BusinessDayConvention** convention)  
*Constructor that takes a notional and the constant value of which the notional should be decreasing every time.*
- **SwapLeg** (valarray< **Date** > dates, valarray< **Real** > Notionals)  
*For more general swaps : takes a valarray of notionals for every period and a valarray of dates.*
- **LongInteger** returnSize ()  
*Return size of valarray.*
- valarray< **Date** > returnDates ()  
*Return dates valarray.*
- valarray< **Real** > returnAmounts ()  
*Return dates valarray.*
- ~**SwapLeg** (void)

### Private Attributes

- valarray< **Date** > **\_dateSchedule**
- valarray< **Real** > **\_flowSchedule**

#### 6.35.1 Constructor & Destructor Documentation

##### 6.35.1.1 SwapLeg::SwapLeg (**Date** startDate, **Real** Frequency, **Date** endDate, **Real** Notional, **Real** AmortizingConstant, **BusinessDayConvention** convention)

Constructor that takes a notional and the constant value of which the notional should be decreasing every time.

If endDate!=startDate + multiple of frequency then endDate is reduced until it satisfies the condition

Definition at line 3 of file SwapLeg.cpp.

References `_dateSchedule`, `_flowSchedule`, `Date::applyConvention()`, `Frequency`, `Integer`, `LongInteger`, `Natural`, `Real`, and `Date::serialNumber()`.

##### 6.35.1.2 SwapLeg::SwapLeg (valarray< **Date** > dates, valarray< **Real** > Notionals)

For more general swaps : takes a valarray of notionals for every period and a valarray of dates.

Definition at line 27 of file SwapLeg.cpp.

References `_dateSchedule`, and `_flowSchedule`.

### 6.35.1.3 SwapLeg::~SwapLeg (void)

Definition at line 50 of file SwapLeg.cpp.

## 6.35.2 Member Function Documentation

### 6.35.2.1 valarray< Real > SwapLeg::returnAmounts ()

Return dates valarray.

Definition at line 45 of file SwapLeg.cpp.

References `_flowSchedule`.

Referenced by `CashFlow::CashFlow()`.

### 6.35.2.2 valarray< Date > SwapLeg::returnDates ()

Return dates valarray.

Definition at line 40 of file SwapLeg.cpp.

References `_dateSchedule`.

Referenced by `CashFlow::CashFlow()`.

### 6.35.2.3 LongInteger SwapLeg::returnSize ()

Return size of valarray.

Definition at line 35 of file SwapLeg.cpp.

References `_dateSchedule`, and `LongInteger`.

## 6.35.3 Member Data Documentation

### 6.35.3.1 valarray<Date> SwapLeg::\_dateSchedule [private]

Definition at line 36 of file SwapLeg.h.

Referenced by `returnDates()`, `returnSize()`, and `SwapLeg()`.

### 6.35.3.2 valarray<Real> SwapLeg::\_flowSchedule [private]

Definition at line 37 of file SwapLeg.h.

Referenced by `returnAmounts()`, and `SwapLeg()`.

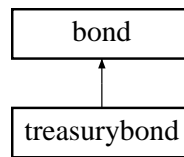
The documentation for this class was generated from the following files:

- `SwapLeg.h`
- `SwapLeg.cpp`

## 6.36 treasurybond Class Reference

```
#include <bond.h>
```

Inheritance diagram for treasurybond:



### Public Member Functions

- **treasurybond** (**Date** issue, **Date** maturity, **Date** firstcoupondate, **Real** coupon, **Frequency** freq, **Real** faceamount, **DayCountConvention** daycount, **yieldCurve** yc)  
*Constructor.*
- **treasurybond** (**Date** issue, **Date** maturity, **Date** firstcoupondate, **Real** coupon, **yieldCurve** yc)  
*Constructor : bond whose freq is semiannual, faceamount=100 and daycount is actual/360.*
- **treasurybond** (**Date** issue, **Date** maturity, **Real** faceamount, **DayCountConvention** daycount, **yieldCurve** yc)  
*Constructor : ZCbond.*
- **~treasurybond** (void)  
*Destructor.*
- **treasurybond shiftedbond** (**Real** shift)  
*bond with same parameters and a shifted yield curve*
- **Real rho** (**Date** today)  
*return the derivative of the bond price with respect to interest rates*
- **Real rho** ()

### 6.36.1 Constructor & Destructor Documentation

#### 6.36.1.1 treasurybond::treasurybond (**Date** *issue*, **Date** *maturity*, **Date** *firstcoupondate*, **Real** *coupon*, **Frequency** *freq*, **Real** *faceamount*, **DayCountConvention** *daycount*, **yieldCurve** *yc*)

Constructor.

#### Parameters:

*issue*: date of issue of the bond

*maturity*: maturity of the bond

*firstcoupondate*: date of the first coupon

*coupon*: coupon of the bond, express as a percentqge of the faceamount

*freq*: frequency of the coupon  
*faceamount*: par value  
*daycount*: daycount convention  
*yc*: yieldcurve

Definition at line 307 of file bond.cpp.

References Real.

### 6.36.1.2 `treasurybond::treasurybond` (`Date issue`, `Date maturity`, `Date firstcoupondate`, `Real coupon`, `yieldCurve yc`)

Constructor : bond whose freq is semiannual, faceamount=100 and daycount is actual/360.

Definition at line 312 of file bond.cpp.

References ACT\_360, Real, and Semiannual.

### 6.36.1.3 `treasurybond::treasurybond` (`Date issue`, `Date maturity`, `Real faceamount`, `DayCountConvention daycount`, `yieldCurve yc`)

Constructor : ZCbond.

Definition at line 317 of file bond.cpp.

References Once, and Real.

### 6.36.1.4 `treasurybond::~~treasurybond` (`void`) [`inline`]

Destructor.

Definition at line 101 of file bond.h.

## 6.36.2 Member Function Documentation

### 6.36.2.1 `Real treasurybond::rho` () [`inline`]

Definition at line 108 of file bond.h.

References Real.

### 6.36.2.2 `Real treasurybond::rho` (`Date today`)

return the derivative of the bond price with respect to interest rates

Definition at line 328 of file bond.cpp.

References `bond::fairvalue()`, Real, and `shiftedbond()`.

### 6.36.2.3 `treasurybond treasurybond::shiftedbond` (`Real shift`)

bond with same parameters and a shifted yield curve

Definition at line 322 of file bond.cpp.

References Real, and yieldCurve::shiftZCRateCurve().

Referenced by rho().

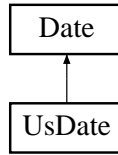
The documentation for this class was generated from the following files:

- **bond.h**
- **bond.cpp**

## 6.37 UsDate Class Reference

```
#include <UsDate.h>
```

Inheritance diagram for UsDate::



### Public Member Functions

- `bool isBusinessDay ()`  
*Apply Conventions (use for UsDate for example).*

#### 6.37.1 Detailed Description

**Author:**

Simon

Definition at line 8 of file UsDate.h.

#### 6.37.2 Member Function Documentation

##### 6.37.2.1 `bool UsDate::isBusinessDay ()`

Apply Conventions (use for UsDate for example).

Reimplemented from **Date** (p. 75).

Definition at line 4 of file UsDate.cpp.

References `Day`, `Date::dayOfMonth()`, `December`, `February`, `Friday`, `January`, `July`, `m`, `May`, `Monday`, `Date::month()`, `Month`, `November`, `October`, `Saturday`, `September`, `Sunday`, `Thursday`, `Date::weekday()`, and `Weekday`.

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

- `UsDate.h`
- `UsDate.cpp`

## 6.38 VanillaSwap Class Reference

```
#include <VanillaSwap.h>
```

### Public Member Functions

- **VanillaSwap** (**CashFlow** cashflowReceived1, **CashFlow** cashflowPaid2, char \*name1, char \*name2, **yieldCurve** \*curve)

*Default constructor : create a vanilla swap with two legs.*

- **~VanillaSwap** (void)
- **Real** getFairValue1 ()

*Return fair value of first leg.*

- **Real** getFairValue2 ()

*Return fair value of second leg.*

- **Real** returnPrice ()

*Return price of the swap.*

- **Real** getRho ()

*Return sensitivity to the interest rate.*

- **Real** getTheta ()

*Return sensitivity to the time.*

### Private Attributes

- **yieldCurve** \* \_curve
- **CashFlow** \_leg1
- **CashFlow** \_leg2
- char \* \_name1
- char \* \_name2

#### 6.38.1 Constructor & Destructor Documentation

##### 6.38.1.1 VanillaSwap::VanillaSwap (**CashFlow** cashflowReceived1, **CashFlow** cashflowPaid2, char \* name1, char \* name2, **yieldCurve** \* curve)

Default constructor : create a vanilla swap with two legs.

Definition at line 3 of file VanillaSwap.cpp.

References \_name1, \_name2, and ShortNatural.

##### 6.38.1.2 VanillaSwap::~VanillaSwap (void)

Definition at line 16 of file VanillaSwap.cpp.

## 6.38.2 Member Function Documentation

### 6.38.2.1 Real VanillaSwap::getFairValue1 ()

Return fair value of first leg.

Definition at line 20 of file VanillaSwap.cpp.

References `_curve`, `_leg1`, `CashFlow::getFairValue()`, and `Real`.

Referenced by `inputVanillaSwap()`, and `returnPrice()`.

### 6.38.2.2 Real VanillaSwap::getFairValue2 ()

Return fair value of second leg.

Definition at line 24 of file VanillaSwap.cpp.

References `_curve`, `_leg2`, `CashFlow::getFairValue()`, and `Real`.

Referenced by `inputVanillaSwap()`, and `returnPrice()`.

### 6.38.2.3 Real VanillaSwap::getRho ()

Return sensitivity to the interest rate.

Definition at line 34 of file VanillaSwap.cpp.

References `_curve`, `Real`, `returnPrice()`, and `yieldCurve::shiftZCBRateCurve()`.

Referenced by `inputVanillaSwap()`.

### 6.38.2.4 Real VanillaSwap::getTheta ()

Return sensitivity to the time.

Definition at line 42 of file VanillaSwap.cpp.

References `_curve`, `yieldCurve::forwardZCBCurve()`, `Real`, and `returnPrice()`.

Referenced by `inputVanillaSwap()`.

### 6.38.2.5 Real VanillaSwap::returnPrice ()

Return price of the swap.

Definition at line 28 of file VanillaSwap.cpp.

References `getFairValue1()`, `getFairValue2()`, and `Real`.

Referenced by `getRho()`, `getTheta()`, `inputVanillaSwap()`, and `mainIRVanillaSwap()`.

## 6.38.3 Member Data Documentation

### 6.38.3.1 yieldCurve\* VanillaSwap::\_curve [private]

Definition at line 31 of file VanillaSwap.h.

Referenced by `getFairValue1()`, `getFairValue2()`, `getRho()`, and `getTheta()`.



**6.38.3.2 CashFlow VanillaSwap::\_leg1 [private]**

Definition at line 32 of file VanillaSwap.h.

Referenced by `getFairValue1()`.

**6.38.3.3 CashFlow VanillaSwap::\_leg2 [private]**

Definition at line 33 of file VanillaSwap.h.

Referenced by `getFairValue2()`.

**6.38.3.4 char\* VanillaSwap::\_name1 [private]**

Definition at line 34 of file VanillaSwap.h.

Referenced by `VanillaSwap()`.

**6.38.3.5 char\* VanillaSwap::\_name2 [private]**

Definition at line 35 of file VanillaSwap.h.

Referenced by `VanillaSwap()`.

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

- **VanillaSwap.h**
- **VanillaSwap.cpp**

## 6.39 VarianceSwap Class Reference

```
#include <VarianceSwap.h>
```

### Public Member Functions

- **VarianceSwap** (**OptionStrategy** \*options, **Real** maturity, **Real** forwardprice)  
*Default Constructor.*
- **~VarianceSwap** (void)
- **Real** getPrice ()  
*Return price.*
- **Real** getRho (**Real** shiftCurve=defaultshiftRate)  
*Return sensitivity to a move in rates.*
- **Real** getVega (**Real** shiftVol=defaultshiftVol)  
*Return sensitivity to a move in volatility.*
- **Real** getTheta (**Real** shiftMat=defaultshiftMat)  
*Return sensitivity to a move in time.*

### Private Attributes

- **OptionStrategy** \* \_options
- **Real** \_maturity
- **Real** \_forward

#### 6.39.1 Detailed Description

**Author:**

Simon

Definition at line 12 of file VarianceSwap.h.

#### 6.39.2 Constructor & Destructor Documentation

##### 6.39.2.1 VarianceSwap::VarianceSwap (**OptionStrategy** \* options, **Real** maturity, **Real** forwardprice)

Default Constructor.

**Parameters:**

*options*: pointer to a basket of options to price the swap

*maturity*: maturity of the swap

*forwardprice*: forward value of the underlying =  $S_0 \cdot \exp(r \cdot T)$

Definition at line 3 of file VarianceSwap.cpp.

References Real.

### 6.39.2.2 VarianceSwap::~~VarianceSwap (void)

Definition at line 10 of file VarianceSwap.cpp.

## 6.39.3 Member Function Documentation

### 6.39.3.1 Real VarianceSwap::getPrice ()

Return price.

Definition at line 14 of file VarianceSwap.cpp.

References `_forward`, `_options`, `BlackScholes::getPrice()`, `BlackScholes::getStrike()`, `Integer`, `BlackScholes::isCall()`, `Natural`, `Real`, `OptionStrategy::returnNbOptions()`, and `OptionStrategy::returnOption()`.

Referenced by `getRho()`, `getTheta()`, `getVega()`, and `mainvarianceswap()`.

### 6.39.3.2 Real VarianceSwap::getRho (Real *shiftCurve* = defaultshiftRate)

Return sensitivity to a move in rates.

Definition at line 72 of file VarianceSwap.cpp.

References `_options`, `OptionStrategy::changeRate()`, `getPrice()`, and `Real`.

### 6.39.3.3 Real VarianceSwap::getTheta (Real *shiftMat* = defaultshiftMat)

Return sensitivity to a move in time.

Definition at line 86 of file VarianceSwap.cpp.

References `_options`, `OptionStrategy::changeMaturity()`, `getPrice()`, and `Real`.

### 6.39.3.4 Real VarianceSwap::getVega (Real *shiftVol* = defaultshiftVol)

Return sensitivity to a move in volatility.

Definition at line 79 of file VarianceSwap.cpp.

References `_options`, `OptionStrategy::changeVol()`, `getPrice()`, and `Real`.

## 6.39.4 Member Data Documentation

### 6.39.4.1 Real VarianceSwap::\_forward [private]

Definition at line 33 of file VarianceSwap.h.

Referenced by `getPrice()`.

### 6.39.4.2 Real VarianceSwap::\_maturity [private]

Definition at line 33 of file VarianceSwap.h.

### 6.39.4.3 OptionStrategy\* VarianceSwap::\_options [private]

Definition at line 32 of file VarianceSwap.h.

Referenced by getPrice(), getRho(), getTheta(), and getVega().

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

- **VarianceSwap.h**
- **VarianceSwap.cpp**

## 6.40 volsurface Class Reference

```
#include <volsurface.h>
```

### Public Member Functions

- **volsurface** (**Real** stockPrice, **Date** today, valarray< **Real** > strikes, valarray< **Date** > maturities, **yieldCurve** yCurve, valarray< valarray< **Real** > > callputprices, valarray< valarray< bool > > iscallputprices)  
*Constructor.*
- **volsurface** (**Real** stockPrice, **Date** today, **yieldCurve** yCurve, **volsurfaceparams** &params)  
*Constructor.*
- **volsurface** (valarray< valarray< **Real** > > volsurf)  
*Constructor.*
- **volsurface** (**Real** constantvol)  
*Default Constructor.*
- **volsurface** (void)  
*Default Constructor.*
- **~volsurface** (void)  
*Destructor.*
- **Real invertBSformula** (**Real** r, **Real** maturity, **Real** stockPrice, **Real** strike, **Real** callputPrice, bool isacall)
- **Real volatility** (**Real** K, **Date** T)
- **Real variance** (**Real** K, **Date** T)
- **Real forwardVolatility** (**Real** K, **Date** t, **Date** T)
- void **setvolsurface** ()
- valarray< valarray< **Real** > > **getvolsurface** ()
- **volsurface forwardvolsurface** (**Date** t)  
*Implied volatility surface seen at time t.*
- **volsurface shiftedYCvolsurface** (**Real** shift)  
*Implied volatility surface when the yield curve is shifted.*
- **volsurface shiftedvolsurface** (**Real** shift)  
*Shift the implied volatility surface.*

### Private Attributes

- **Real** **\_stockPrice**
- **Date** **\_today**
- valarray< **Real** > **\_strikes**

- valarray< **Date** > **\_maturities**
- **yieldCurve** **\_yieldCurve**
- valarray< valarray< **Real** > > **\_callputprices**
- valarray< valarray< **bool** > > **\_iscallputprices**
- valarray< valarray< **Real** > > **\_impliedvolsurface**
- **interpolator** **\_interpolvolsurf**
- **bool** **\_volsurfconst**
- **Real** **\_constantvol**

## 6.40.1 Constructor & Destructor Documentation

6.40.1.1 **volsurface::volsurface** (**Real** *stockPrice*, **Date** *today*, valarray< **Real** > *strikes*, valarray< **Date** > *maturities*, yieldCurve *yCurve*, valarray< valarray< **Real** > > *callputprices*, valarray< valarray< **bool** > > *iscallputprices*)

Constructor.

### Parameters:

- stockPrice*: Current stock price
- today*: date of today
- strikes*: vector of strikes
- maturities*: vector of maturities
- yCurve*: the yield curve
- callputprices*: matrix of option prices
- iscallputprices*: matrix indicating if we have call or put prices

Definition at line 7 of file volsurface.cpp.

References **Real**.

6.40.1.2 **volsurface::volsurface** (**Real** *stockPrice*, **Date** *today*, yieldCurve *yCurve*, volsurfaceparams & *params*)

Constructor.

### Parameters:

- stockPrice*: Current stock price
- today*: date of today
- yCurve*: the yield curve
- params*: strikes, maturities, prices, etc.

Definition at line 19 of file volsurface.cpp.

References **Real**.

**6.40.1.3 volsurface::volsurface (valarray< valarray< Real > > volsurf)**

Constructor.

**Parameters:**

*volsurf*: volsurface

Definition at line 44 of file volsurface.cpp.

**6.40.1.4 volsurface::volsurface (Real constantvol)**

Default Constructor.

Definition at line 32 of file volsurface.cpp.

References Real.

**6.40.1.5 volsurface::volsurface (void)**

Default Constructor.

Definition at line 38 of file volsurface.cpp.

**6.40.1.6 volsurface::~~volsurface (void)**

Destructor.

Definition at line 50 of file volsurface.cpp.

**6.40.2 Member Function Documentation****6.40.2.1 Real volsurface::forwardVolatility (Real K, Date t, Date T)**

Definition at line 139 of file volsurface.cpp.

References `_today`, `Date::dayCount()`, `Real`, and `variance()`.

Referenced by `Drift::Drift()`, and `forwardvolsurface()`.

**6.40.2.2 volsurface volsurface::forwardvolsurface (Date t)**

Implied volatility surface seen at time t.

Definition at line 148 of file volsurface.cpp.

References `_impliedvolsurface`, `_maturities`, `_strikes`, `forwardVolatility()`, `Integer`, `M`, and `N`.

Referenced by `Exotics::getTheta()`.

**6.40.2.3 valarray< valarray< Real > > volsurface::getvolsurface ()**

Definition at line 118 of file volsurface.cpp.

References `_impliedvolsurface`.

#### 6.40.2.4 Real volsurface::invertBSformula (Real *r*, Real *maturity*, Real *stockPrice*, Real *strike*, Real *callputPrice*, bool *isacall*)

Definition at line 55 of file volsurface.cpp.

References Call, BlackScholes::getVolatility(), Put, *r*, and Real.

Referenced by setvolsurface().

#### 6.40.2.5 void volsurface::setvolsurface ()

Definition at line 68 of file volsurface.cpp.

References \_callputprices, \_constantvol, \_impliedvolsurface, \_interpolvolsurf, \_iscallputprices, \_maturities, \_stockPrice, \_strikes, \_today, \_volsurfconst, \_yieldCurve, Date::dayCount(), invertBSformula(), *N*, *r*, Real, and yieldCurve::spotRate().

Referenced by importData::importVolSurface(), mainvolsurface(), and shiftedYCvolsurface().

#### 6.40.2.6 volsurface volsurface::shiftedvolsurface (Real *shift*)

Shift the implied volatility surface.

Definition at line 194 of file volsurface.cpp.

References \_impliedvolsurface, \_maturities, \_strikes, Integer, *M*, *N*, and Real.

Referenced by Exotics::getVega().

#### 6.40.2.7 volsurface volsurface::shiftedYCvolsurface (Real *shift*)

Implied volatility surface when the yield curve is shifted.

Definition at line 185 of file volsurface.cpp.

References \_callputprices, \_iscallputprices, \_maturities, \_stockPrice, \_strikes, \_today, \_yieldCurve, Real, setvolsurface(), and yieldCurve::shiftZCBRateCurve().

#### 6.40.2.8 Real volsurface::variance (Real *K*, Date *T*)

Definition at line 131 of file volsurface.cpp.

References \_constantvol, \_volsurfconst, Real, and volatility().

Referenced by forwardVolatility().

#### 6.40.2.9 Real volsurface::volatility (Real *K*, Date *T*)

Definition at line 122 of file volsurface.cpp.

References \_constantvol, \_interpolvolsurf, \_today, \_volsurfconst, Date::dayCount(), interpolator::interpolate(), and Real.

Referenced by GaussianProcess::BuildPath(), inputBSOption(), inputButterflySpread(), inputCallSpread(), inputPutSpread(), inputRatioCallSpread(), inputStraddle(), inputStrangle(), mainvolsurface(), and variance().



### 6.40.3 Member Data Documentation

#### 6.40.3.1 `valarray<valarray<Real> > volsurface::_callputprices` [private]

Definition at line 35 of file volsurface.h.

Referenced by `setvolsurface()`, and `shiftedYCvolsurface()`.

#### 6.40.3.2 `Real volsurface::_constantvol` [private]

Definition at line 40 of file volsurface.h.

Referenced by `setvolsurface()`, `variance()`, and `volatility()`.

#### 6.40.3.3 `valarray<valarray<Real> > volsurface::_impliedvolsurface` [private]

Definition at line 37 of file volsurface.h.

Referenced by `forwardvolsurface()`, `getvolsurface()`, `setvolsurface()`, and `shiftedvolsurface()`.

#### 6.40.3.4 `interpolator volsurface::_interpolvolsurf` [private]

Definition at line 38 of file volsurface.h.

Referenced by `setvolsurface()`, and `volatility()`.

#### 6.40.3.5 `valarray<valarray<bool> > volsurface::_iscallputprices` [private]

Definition at line 36 of file volsurface.h.

Referenced by `setvolsurface()`, and `shiftedYCvolsurface()`.

#### 6.40.3.6 `valarray<Date> volsurface::_maturities` [private]

Definition at line 33 of file volsurface.h.

Referenced by `forwardvolsurface()`, `setvolsurface()`, `shiftedvolsurface()`, and `shiftedYCvolsurface()`.

#### 6.40.3.7 `Real volsurface::_stockPrice` [private]

Definition at line 30 of file volsurface.h.

Referenced by `setvolsurface()`, and `shiftedYCvolsurface()`.

#### 6.40.3.8 `valarray<Real> volsurface::_strikes` [private]

Definition at line 32 of file volsurface.h.

Referenced by `forwardvolsurface()`, `setvolsurface()`, `shiftedvolsurface()`, and `shiftedYCvolsurface()`.

**6.40.3.9** `Date volsurface::_today` [private]

Definition at line 31 of file volsurface.h.

Referenced by `forwardVolatility()`, `setvolsurface()`, `shiftedYCVolsurface()`, and `volatility()`.

**6.40.3.10** `bool volsurface::_volsurfconst` [private]

Definition at line 39 of file volsurface.h.

Referenced by `setvolsurface()`, `variance()`, and `volatility()`.

**6.40.3.11** `yieldCurve volsurface::_yieldCurve` [private]

Definition at line 34 of file volsurface.h.

Referenced by `setvolsurface()`, and `shiftedYCVolsurface()`.

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

- `volsurface.h`
- `volsurface.cpp`

## 6.41 volsurfaceparams Class Reference

```
#include <volsurface.h>
```

### Public Attributes

- valarray< **Real** > **strikes**
- valarray< **Date** > **maturities**
- valarray< valarray< **Real** > > **callputprices**
- valarray< valarray< bool > > **iscallputprices**

### 6.41.1 Member Data Documentation

#### 6.41.1.1 valarray<valarray<Real> > volsurfaceparams::callputprices

Definition at line 23 of file volsurface.h.

Referenced by FileReader::buildVolSurfaceParams().

#### 6.41.1.2 valarray<valarray<bool> > volsurfaceparams::iscallputprices

Definition at line 24 of file volsurface.h.

Referenced by FileReader::buildVolSurfaceParams().

#### 6.41.1.3 valarray<Date> volsurfaceparams::maturities

Definition at line 22 of file volsurface.h.

Referenced by FileReader::buildVolSurfaceParams().

#### 6.41.1.4 valarray<Real> volsurfaceparams::strikes

Definition at line 21 of file volsurface.h.

Referenced by FileReader::buildVolSurfaceParams().

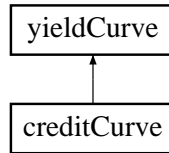
The documentation for this class was generated from the following file:

- **volsurface.h**

## 6.42 yieldCurve Class Reference

```
#include <yieldCurve.h>
```

Inheritance diagram for yieldCurve::



### Public Member Functions

- **yieldCurve** (void)  
*Default void constructor.*
- **yieldCurve** (Real flatRate)  
*FlatCurve.*
- void **assignFlatRate** (Real r=0.0)  
*For flat rate curves, set the flat rate value.*
- void **assignZCBrateAtIndex** (Real rate, Natural i)  
*sets an EXISTING rate at a certain level*
- **yieldCurve shiftZCBrateCurve** (Real shift=defaultshiftfactorForShortRate)  
*for risk mngmt purposes, shifts the yc*
- **yieldCurve rotateZCBrateCurve** (Real moveInShortestRate=defaultshiftfactorForShortRate, Real maturityOfRotation=7)  
*for risk mngmt purposes, rotates the yc with a ref to how you move the shortest rate, around which rate*
- **yieldCurve** (valarray< yieldPoint > yieldPoints, char \*name="unnamed")  
*Constructor.*
- **~yieldCurve** (void)
- virtual Real **spotRate** (Real maturity) const  
*Calculates the spot ZCB rate.*
- virtual valarray< Real > **getMaturitiesInTheMarketCurve** () const  
*Return the maturities present in the market curve, both from the Cash and Swap Pointsvalarray<Real>.*
- virtual valarray< Real > **getMaturitiesInTheZCBCurve** () const  
*Return the maturities present in the market curve, both from the Cash and Swap Pointsvalarray<Real>.*
- virtual Real **spotRate** (Date maturityDate) const

*Calculates the spot ZCB rate.*

- virtual **Real discountFactor** (**Real** maturity, **interestComposition** composition=Continuous)

*Calculates the discountFactor.*

- virtual **Real discountFactor** (**Date** maturityDate, **interestComposition** composition=Continuous)

*Calculates the discountFactor.*

- virtual **Real forwardDiscountFactor** (**Real** forwardstart, **Real** lengthofcontractafterstart, **interestComposition** composition=Continuous)

*Calculates the discountFactor.*

- virtual **Real forwardRate** (**Real** forwardStart, **Real** effectiveLengthOfTheContractAfterStart, **interestComposition** composition=Continuous)

*Calculates the fwd rate.*

- virtual **Real forwardRate** (**Date** forwardStart, **Date** forwardEnd, **interestComposition** composition=Continuous)

*Calculates the fwd rate.*

- virtual **yieldCurve forwardZCBCurve** (**Real** forwardStart)

*Forward curve.*

- char \* **getName** ()
- virtual bool **operator==** (const **yieldCurve** &yours)  
*compares two y curves.*
- virtual bool **operator!=** (const **yieldCurve** &yours)

## Private Member Functions

- **yieldPoint getPointAtMaturity** (**Real** maturity)  
*for know maturities, we can return the market Point as it is*
- void **sortMarketRatesByMaturity** ()  
*Sorting rates by maturity, just in case ...*
- void **sortCashSwap** ()  
*Routine to make sure the short term rates (cash) are before the mid/long term (swap).*
- valarray< **yieldPoint** > **getSwapRates** ()  
*needed in the bootstrap method to be able to back the ZC*
- valarray< **yieldPoint** > **getSequentSwapRates** ()  
*1Y difference in swaps -> ZC's easily backed up The method assumes that rates are sorted by ascending maturity The mkt curve is 1Y by 1Y - we should fill in the gaps by linear interpolation*
- void **computeZCBBRatesBootstrap** ()

Go from the swap rates to the ZC - matrix inversion to be done Swap rates can be annual or semi annual in their most common quotes on the market.

- valarray< Real > **SequentDiscountFactorsByInvertSwapMatrix** ()

Discount Factors corresponding to annual swaps quoted on sequent yrs :  $X^{-1} * (1...1)'$  where  $X$  is as follows: - diag = +1 swap (i) - inferior triang (i,j) = swap (i) - superior diag (i,j) = 0.

## Private Attributes

- valarray< yieldPoint > **\_marketRates**
- valarray< yieldPoint > **\_zcbRates**
- char **\_name** [YC\_NAME\_STRLEN]

## Friends

- ostream & **operator**<< (ostream &os, const yieldCurve &c)  
*display maturities and spotrates in the curve.*
- ostream & **operator**<< (ostream &os, const yieldCurve \*c)

## 6.42.1 Constructor & Destructor Documentation

### 6.42.1.1 yieldCurve::yieldCurve (void)

Default void constructor.

clean ?

Definition at line 42 of file yieldCurve.cpp.

References `_name`, `assignFlatRate()`, `computeZCBRatesBootstrap()`, and `YC_MAX_NUMBER_POINTS`.

Referenced by `creditCurve::combineUnderlyingAndSpreads()`, `creditCurve::copyObj()`, `creditCurve::createSpreadCurve()`, `creditCurve::creditCurve()`, `forwardZCBCurve()`, `rotateZCBRateCurve()`, and `shiftZCBRateCurve()`.

### 6.42.1.2 yieldCurve::yieldCurve (Real flatRate)

FlatCurve.

#### Parameters:

**flatRate** : Real is the flat rate - by default, it will simulate a flat ZCB curve with 15 Points by default

clean ?

Definition at line 64 of file yieldCurve.cpp.

References `_name`, `assignFlatRate()`, `computeZCBRatesBootstrap()`, `Real`, and `YC_MAX_NUMBER_POINTS`.

### 6.42.1.3 yieldCurve::yieldCurve (valarray< yieldPoint > *yieldPoints*, char \* *name* = "unnamed")

Constructor.

#### Parameters:

*yieldPoints* the array of yieldPoints  
*name* the name we give it

clean ?

Definition at line 74 of file yieldCurve.cpp.

References `_name`, `computeZCBRatesBootstrap()`, `sortCashSwap()`, and `sortMarketRatesByMaturity()`.

### 6.42.1.4 yieldCurve::~~yieldCurve (void)

Definition at line 85 of file yieldCurve.cpp.

## 6.42.2 Member Function Documentation

### 6.42.2.1 void yieldCurve::assignFlatRate (Real *r* = 0.0)

For flat rate curves, set the flat rate value.

Definition at line 53 of file yieldCurve.cpp.

References `_marketRates`, `ACT_360`, `Cash`, `Natural`, `r`, and `Real`.

Referenced by `yieldCurve()`.

### 6.42.2.2 void yieldCurve::assignZCBrateAtIndex (Real *rate*, Natural *i*)

sets an EXISTING rate at a certain level

Definition at line 377 of file yieldCurve.cpp.

References `_zcbRates`, `Natural`, and `Real`.

Referenced by `rotateZCBRateCurve()`, and `shiftZCBRateCurve()`.

### 6.42.2.3 void yieldCurve::computeZCBRatesBootstrap () [private]

Go from the swap rates to the ZC - matrix inversion to be done Swap rates can be annual or semi annual in their most common quotes on the market.

Having semi annual swaps with annual maturities makes life hard as to bootstrap the ZC's as we would not be able to back out the semi annual ZC's to move to the next step without market bond prices, etc: it would really be cumbersome On the quote screen provided by Prof Laud, swaps are quoted annually from 2 to 10 -> easy calculation [cf latex Code] as for the 15 - 20 and 30 with frequency annual, we need to find a way to find mid Points 11-14, 16-19 21-29.

Definition at line 354 of file yieldCurve.cpp.

References `_marketRates`, `_zcbRates`, `Cash`, `Natural`, `Real`, and `SequentDiscountFactorsByInvertSwapMatrix()`.

Referenced by `yieldCurve()`.

#### 6.42.2.4 Real `yieldCurve::discountFactor` (`Date maturityDate`, `interestComposition composition = Continuous`) [virtual]

Calculates the `discountFactor`.

**Parameters:**

*maturity* : just after ZCBrates are computed, it it very easy [done at trhe constructor level]

Reimplemented in `creditCurve` (p. 57).

Definition at line 247 of file `yieldCurve.cpp`.

References `Day30_360`, `Date::dayCount()`, `discountFactor()`, `Real`, and `Date::setDateToToday()`.

#### 6.42.2.5 Real `yieldCurve::discountFactor` (`Real maturity`, `interestComposition composition = Continuous`) [virtual]

Calculates the `discountFactor`.

**Parameters:**

*maturity* : just after ZCBrates are computed, it it very easy [done at trhe constructor level]

Reimplemented in `creditCurve` (p. 57).

Definition at line 235 of file `yieldCurve.cpp`.

References `Continuous`, `Discrete`, `Real`, and `spotRate()`.

Referenced by `creditCurve::defaultProbability()`, `discountFactor()`, `creditCurve::discountFactor()`, `asset::forwardPrice()`, `forwardRate()`, `CashFlow::getFairValue()`, `mainmc()`, `bond::quotedPrice()`, `RainbowOption::RainbowOption()`, `creditCurve::riskyDiscountFactor()`, and `creditCurve::swapFees()`.

#### 6.42.2.6 Real `yieldCurve::forwardDiscountFactor` (`Real forwardstart`, `Real lengthofcontractafterstart`, `interestComposition composition = Continuous`) [virtual]

Calculates the `discountFactor`.

**Parameters:**

*maturity* : just after ZCBrates are computed, it it very easy [done at trhe constructor level]

Definition at line 253 of file `yieldCurve.cpp`.

References `Continuous`, `Discrete`, `forwardRate()`, and `Real`.

Referenced by `binomialTree::setClaimVariables()`.

#### 6.42.2.7 Real `yieldCurve::forwardRate` (`Date forwardStart`, `Date forwardEnd`, `interestComposition composition = Continuous`) [virtual]

Calculates the fwd rate.



**Parameters:**

*forwardStart* start of the rate  
*maturityAfterForward* maturity after the start

Reimplemented in **creditCurve** (p. 58).

Definition at line 298 of file yieldCurve.cpp.

References Day30\_360, Date::dayCount(), forwardRate(), Real, and Date::setDateToToday().

**6.42.2.8 Real yieldCurve::forwardRate (Real *forwardStart*, Real *effectiveLengthOfTheContractAfterStart*, interestComposition *composition* = Continuous) [virtual]**

Calculates the fwd rate.

**Parameters:**

*forwardStart* start of the rate  
*maturityAfterForward* maturity after the start

Reimplemented in **creditCurve** (p. 58).

Definition at line 264 of file yieldCurve.cpp.

References Continuous, discountFactor(), Discrete, Real, and spotRate().

Referenced by CashFlow::CashFlow(), Drift::Drift(), forwardDiscountFactor(), forwardRate(), creditCurve::forwardRate(), and forwardZCBCurve().

**6.42.2.9 yieldCurve yieldCurve::forwardZCBCurve (Real *forwardStart*) [virtual]**

Forward curve.

Definition at line 279 of file yieldCurve.cpp.

References *\_zcbRates*, forwardRate(), Natural, Real, and yieldCurve().

Referenced by VanillaSwap::getTheta(), and Exotics::getTheta().

**6.42.2.10 valarray< Real > yieldCurve::getMaturitiesInTheMarketCurve () const [virtual]**

Return the maturities present in the market curve, both from the Cash and Swap Pointsvalarray<Real>.

Definition at line 333 of file yieldCurve.cpp.

References *\_marketRates*, and Natural.

**6.42.2.11 valarray< Real > yieldCurve::getMaturitiesInTheZCBCurve () const [virtual]**

Return the maturities present in the market curve, both from the Cash and Swap Pointsvalarray<Real>.

Reimplemented in **creditCurve** (p. 59).

Definition at line 343 of file yieldCurve.cpp.

References `_zcbRates`, and `Natural`.

Referenced by `creditCurve::combineUnderlyingAndSpreads()`, `creditCurve::getMaturitiesInTheZCBCurve()`, `mainyieldcurve()`, `operator<<()`, and `operator==()`.

#### 6.42.2.12 `char* yieldCurve::getName () [inline]`

Reimplemented in `creditCurve` (p. 59).

Definition at line 192 of file yieldCurve.h.

References `_name`.

Referenced by `creditCurve::getName()`.

#### 6.42.2.13 `yieldPoint yieldCurve::getPointAtMaturity (Real maturity) [private]`

for know maturities, we can return the market Point as it is

##### Parameters:

*maturity* : Be careful with this method as if it does not know the maturity it will give a blank Point

Definition at line 89 of file yieldCurve.cpp.

References `_marketRates`, `Natural`, and `Real`.

Referenced by `sortMarketRatesByMaturity()`.

#### 6.42.2.14 `valarray< yieldPoint > yieldCurve::getSequentSwapRates () [private]`

1Y difference in swaps -> ZC's easily backed up The method assumes that rates are sorted by ascending maturity The mkt curve is 1Y by 1Y - we should fill in the gaps by linear interpolation Swaps are quoted per year sequent then gaps : we need to fill in the gaps with a linear interpolation we know that the first one is in 1Y swap rate (assigned at 1Y cash rate), as for the rest we ahve to Check for 1y sequent swap rates first

Definition at line 172 of file yieldCurve.cpp.

References `getSwapRates()`, `Natural`, `Real`, and `Swap`.

Referenced by `SequentDiscountFactorsByInvertSwapMatrix()`.

#### 6.42.2.15 `valarray< yieldPoint > yieldCurve::getSwapRates () [private]`

needed in the bootstrap method to be able to back the ZC

Definition at line 155 of file yieldCurve.cpp.

References `_marketRates`, `Cash`, `Natural`, `spotRate()`, `Swap`, and `YC_MAX_NUMBER_POINTS`.

Referenced by `getSequentSwapRates()`.

**6.42.2.16** `bool yieldCurve::operator!= (const yieldCurve & yours)` [virtual]

Definition at line 452 of file yieldCurve.cpp.

References `operator==()`.

**6.42.2.17** `bool yieldCurve::operator== (const yieldCurve & yours)` [virtual]

compares two y curves.

Two y curves are equal if they give identical spotrates for all ZCB maturities in each curve

**Parameters:**

*param* - the creditcurve to compare to

**Returns:**

true if all spotrates match, otherwise false

Definition at line 423 of file yieldCurve.cpp.

References `getMaturitiesInTheZCBCurve()`, `m`, `mergeunique()`, `Natural`, `Real`, and `spotRate()`.

Referenced by `operator!=()`.

**6.42.2.18** `yieldCurve yieldCurve::rotateZCBRateCurve (Real moveInShortestRate = defaultshiftfactorForShortRate, Real maturityOfRotation = 7)`

for risk mngmt purposes, rotates the yc with a ref to how you move the shortest rate, around which rate

Definition at line 399 of file yieldCurve.cpp.

References `_zcbRates`, `assignZCBrateAtIndex()`, `Natural`, `Real`, and `yieldCurve()`.

Referenced by `mainyieldcurve()`.

**6.42.2.19** `valarray< Real > yieldCurve::SequentDiscountFactorsByInvertSwapMatrix ()` [private]

Discount Factors corresponding to annual swaps quoted on sequent yrs :  $X^{-1} * (1 \dots 1)'$  where X is as follows: - diag = +1 swap (i) - inferior triang (i,j) = swap (i) - superior diag (i,j) =0.

Definition at line 305 of file yieldCurve.cpp.

References `getSequentSwapRates()`, `M`, and `Natural`.

Referenced by `computeZCBRatesBootstrap()`.

**6.42.2.20** `yieldCurve yieldCurve::shiftZCBRateCurve (Real shift = defaultshiftfactorForShortRate)`

for risk mngmt purposes, shifts the yc

Definition at line 382 of file yieldCurve.cpp.

References `_zcbRates`, `assignZCBrateAtIndex()`, `Natural`, `Real`, and `yieldCurve()`.

Referenced by VanillaSwap::getRho(), RainbowOption::getRho(), Exotics::getRho(), asset::getRho(), mainyieldcurve(), riskybond::shiftedbond(), treasurybond::shiftedbond(), and volsurface::shiftedYCvolsurface().

#### 6.42.2.21 void yieldCurve::sortCashSwap () [private]

Routine to make sure the short term rates (cash) are before the mid/long term (swap).

Definition at line 134 of file yieldCurve.cpp.

References \_marketRates, Cash, Natural, and Swap.

Referenced by yieldCurve().

#### 6.42.2.22 void yieldCurve::sortMarketRatesByMaturity () [private]

Sorting rates by maturity, just in case ...

Definition at line 112 of file yieldCurve.cpp.

References \_marketRates, getPointAtMaturity(), Natural, Real, and YC\_MAX\_NUMBER\_POINTS.

Referenced by yieldCurve().

#### 6.42.2.23 Real yieldCurve::spotRate (Date *maturityDate*) const [virtual]

Calculates the spot ZCB rate.

##### Parameters:

*maturityDate* : maturityDate of the ZCB

Reimplemented in **creditCurve** (p.61).

Definition at line 229 of file yieldCurve.cpp.

References Day30\_360, Date::dayCount(), Real, Date::setDateToToday(), and spotRate().

#### 6.42.2.24 Real yieldCurve::spotRate (Real *maturity*) const [virtual]

Calculates the spot ZCB rate.

##### Parameters:

*maturity* : if it is exact it just gives the result from a Point, else an interpolated one based on interpolator

Reimplemented in **creditCurve** (p.62).

Definition at line 207 of file yieldCurve.cpp.

References \_zcbRates, Natural, and Real.

Referenced by creditCurve::combineUnderlyingAndSpreads(), RainbowOption::compute\_C(), RainbowOption::compute\_d1(), RainbowOption::compute\_d2(), creditCurve::createSpreadCurve(), creditCurve::creditSpread(), discountFactor(), forwardRate(), getSwapRates(), inputBSOption(), inputButterflySpread(), inputCallSpread(), inputPutSpread(),

inputRatioCallSpread(), inputStraddle(), inputStrangle(), RainbowOption::instanciateMCVariables(), mainyieldcurve(), operator<<(), operator==(), RainbowOption::PriceByClosedForm\_MaxOf2\_call(), RainbowOption::PriceByClosedForm\_MaxOf2\_put(), RainbowOption::PriceByClosedForm\_MinOf2\_call(), RainbowOption::PriceByClosedForm\_MinOf2\_put(), RainbowOption::PriceByClosedForm\_WorseOf2(), creditCurve::resampleSpread(), volsurface::setvolsurface(), spotRate(), and creditCurve::spotRate().

### 6.42.3 Friends And Related Function Documentation

#### 6.42.3.1 ostream& operator<< (ostream & os, const yieldCurve \* c) [friend]

Definition at line 100 of file yieldCurve.h.

#### 6.42.3.2 ostream& operator<< (ostream & os, const yieldCurve & c) [friend]

display maturities and spotrates in the curve.

##### Parameters:

- os* - the output stream to direct output to
- c* - the curve to display

##### Returns:

output stream as is standard for operator<<

Definition at line 457 of file yieldCurve.cpp.

### 6.42.4 Member Data Documentation

#### 6.42.4.1 valarray<yieldPoint> yieldCurve::\_marketRates [private]

Definition at line 199 of file yieldCurve.h.

Referenced by assignFlatRate(), computeZCBRatesBootstrap(), getMaturitiesInTheMarketCurve(), getPointAtMaturity(), getSwapRates(), sortCashSwap(), and sortMarketRatesByMaturity().

#### 6.42.4.2 char yieldCurve::\_name[YC\_NAME\_STRLEN] [private]

Definition at line 201 of file yieldCurve.h.

Referenced by getName(), and yieldCurve().

#### 6.42.4.3 valarray<yieldPoint> yieldCurve::\_zcbRates [private]

Definition at line 200 of file yieldCurve.h.

Referenced by assignZCBRateAtIndex(), computeZCBRatesBootstrap(), forwardZCBCurve(), getMaturitiesInTheZCBCurve(), rotateZCBRateCurve(), shiftZCBRateCurve(), and spotRate().

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

- **yieldCurve.h**
- **yieldCurve.cpp**

## 6.43 yieldPoint Class Reference

```
#include <yieldCurve.h>
```

### Public Member Functions

- **yieldPoint** (void)  
*Default constructor.*
- **yieldPoint** (Real r, Real T, TypeOfRate type=Cash, DayCountConvention dayCount=ACT\_360)  
*Constructor.*
- **~yieldPoint** (void)  
*Standard Destructor.*
- **Real getRate** ()  
*Returns the rate associated to the Point.*
- **Real getMaturity** ()  
*Returns the maturity associated to the Point.*
- **TypeOfRate getType** ()  
*Returns the type associated to the Point.*
- **DayCountConvention getDayCount** ()  
*Returns the dayCount associated to the Point.*
- **void setRate** (Real r)  
*Set the rate associated to the Point.*
- **void setMaturity** (Real m)  
*Set the maturity associated to the Point.*
- **void setType** (TypeOfRate t)  
*Set the type associated to the Point.*
- **void setDayCount** (DayCountConvention d)  
*Set the dayCount associated to the Point.*

### Static Public Member Functions

- **char \* TypeAsString** (TypeOfRate t)

## Private Attributes

- **Real** `_rate`
- **Real** `_maturity`
- **TypeOfRate** `_type`
- **DayCountConvention** `_dayCount`

### 6.43.1 Constructor & Destructor Documentation

#### 6.43.1.1 `yieldPoint::yieldPoint (void)`

Default constructor.

Definition at line 7 of file `yieldCurve.cpp`.

References `_dayCount`, `_maturity`, `_rate`, `ACT_360`, and `Cash`.

#### 6.43.1.2 `yieldPoint::yieldPoint (Real r, Real T, TypeOfRate type = Cash, DayCountConvention dayCount = ACT_360)`

Constructor.

##### Parameters:

*r*: Real rate associated to the Point - on an annual basis

*T*: Real maturity of the rate

*Type*: Type of the market Point Type : either cash or swap. Default will be the case where the short term is always cash, long is swap.

Definition at line 14 of file `yieldCurve.cpp`.

References `_dayCount`, `_maturity`, `_rate`, `r`, and `Real`.

#### 6.43.1.3 `yieldPoint::~~yieldPoint (void)`

Standard Destructor.

Definition at line 35 of file `yieldCurve.cpp`.

### 6.43.2 Member Function Documentation

#### 6.43.2.1 `DayCountConvention yieldPoint::getDayCount () [inline]`

Returns the `dayCount` associated to the Point.

Definition at line 65 of file `yieldCurve.h`.

References `_dayCount`, and `DayCountConvention`.

#### 6.43.2.2 `Real yieldPoint::getMaturity () [inline]`

Returns the maturity associated to the Point.

Definition at line 59 of file `yieldCurve.h`.

References `_maturity`, and `Real`.

**6.43.2.3 Real yieldPoint::getRate () [inline]**

Returns the rate associated to the Point.

Definition at line 56 of file yieldCurve.h.

References `_rate`, and `Real`.

**6.43.2.4 TypeOfRate yieldPoint::getType () [inline]**

Returns the type associated to the Point.

Definition at line 62 of file yieldCurve.h.

References `TypeOfRate`.

**6.43.2.5 void yieldPoint::setDayCount (DayCountConvention d) [inline]**

Set the dayCount associated to the Point.

Definition at line 77 of file yieldCurve.h.

References `_dayCount`.

**6.43.2.6 void yieldPoint::setMaturity (Real m) [inline]**

Set the maturity associated to the Point.

Definition at line 71 of file yieldCurve.h.

References `_maturity`, `m`, and `Real`.

**6.43.2.7 void yieldPoint::setRate (Real r) [inline]**

Set the rate associated to the Point.

Definition at line 68 of file yieldCurve.h.

References `_rate`, `r`, and `Real`.

**6.43.2.8 void yieldPoint::setType (TypeOfRate t) [inline]**

Set the type associated to the Point.

Definition at line 74 of file yieldCurve.h.

**6.43.2.9 char \* yieldPoint::TypeAsString (TypeOfRate t) [static]**

Definition at line 23 of file yieldCurve.cpp.

References `Cash`, and `Swap`.

Referenced by `CSVParser::operator>>()`.



### 6.43.3 Member Data Documentation

#### 6.43.3.1 DayCountConvention yieldPoint::\_dayCount [private]

Definition at line 86 of file yieldCurve.h.

Referenced by `getDayCount()`, `setDayCount()`, and `yieldPoint()`.

#### 6.43.3.2 Real yieldPoint::\_maturity [private]

Definition at line 84 of file yieldCurve.h.

Referenced by `getMaturity()`, `setMaturity()`, and `yieldPoint()`.

#### 6.43.3.3 Real yieldPoint::\_rate [private]

Definition at line 83 of file yieldCurve.h.

Referenced by `getRate()`, `setRate()`, and `yieldPoint()`.

#### 6.43.3.4 TypeOfRate yieldPoint::\_type [private]

Definition at line 85 of file yieldCurve.h.

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

- `yieldCurve.h`
- `yieldCurve.cpp`



## Chapter 7

# terreneuve File Documentation

### 7.1 asset.cpp File Reference

```
#include "..\asset.h"
```

## 7.2 asset.h File Reference

```
#include "../common/types.h"
#include "../common/date.h"
#include "../PartB/yieldCurve.h"
```

### Classes

- class `flowSchedule`
- class `asset`

### Defines

- `#define ASSET_DEFAULT_VOL 0.20`

#### 7.2.1 Define Documentation

##### 7.2.1.1 `#define ASSET_DEFAULT_VOL 0.20`

Definition at line 9 of file `asset.h`.

Referenced by `asset::asset()`.

## 7.3 binomialTree.cpp File Reference

```
#include "..\binomialTree.h"
```

### Functions

- ostream & operator<< (ostream &os, const binomialTree &bt)

### 7.3.1 Function Documentation

#### 7.3.1.1 ostream& operator<< (ostream & os, const binomialTree & bt)

Definition at line 162 of file binomialTree.cpp.

References binomialTree::\_claimProcess, binomialTree::\_d, binomialTree::\_discountFactor, binomialTree::\_n, binomialTree::\_q, binomialTree::\_stockProcess, binomialTree::\_u, binomialTree::getMaturity(), binomialTree::getSigma(), binomialTree::getSo(), binomialTree::getSteps(), Natural, and valarrayRealToString().

## 7.4 binomialTree.h File Reference

```
#include "../common/types.h"
#include "../PartA/MonteCarlo1/PayOff.h"
#include "../PartB/yieldCurve.h"
#include "../PartC/asset.h"
#include "../common/utils.h"
#include <iostream>
#include <valarray>
```

### Classes

- class `binomialTree`

### Defines

- `#define BT_DEFAULT_SO 100`
- `#define BT_DEFAULT_RATE 0.05`
- `#define BT_DEFAULT_SIGMA 0.30`
- `#define BT_DEFAULT_MATURITY 1`
- `#define BT_DEFAULT_STEPS 10`

#### 7.4.1 Define Documentation

##### 7.4.1.1 `#define BT_DEFAULT_MATURITY 1`

Definition at line 17 of file `binomialTree.h`.

Referenced by `binomialTree::binomialTree()`.

##### 7.4.1.2 `#define BT_DEFAULT_RATE 0.05`

Definition at line 15 of file `binomialTree.h`.

Referenced by `binomialTree::binomialTree()`.

##### 7.4.1.3 `#define BT_DEFAULT_SIGMA 0.30`

Definition at line 16 of file `binomialTree.h`.

Referenced by `binomialTree::binomialTree()`.

##### 7.4.1.4 `#define BT_DEFAULT_SO 100`

Definition at line 14 of file `binomialTree.h`.

Referenced by `binomialTree::binomialTree()`.

#### 7.4.1.5 `#define BT_DEFAULT_STEPS 10`

Definition at line 18 of file binomialTree.h.

Referenced by binomialTree::binomialTree().

## 7.5 BlackScholes.cpp File Reference

```
#include "BlackScholes.h"
```

### Functions

- **Real absolute** (**Real**  $x$ )

#### 7.5.1 Function Documentation

##### 7.5.1.1 Real absolute (**Real** $x$ )

Definition at line 3 of file BlackScholes.cpp.

References Real.

Referenced by BlackScholes::BlackScholes().



## 7.6 BlackScholes.h File Reference

```
#include "../../common/types.h"
#include "../../common/Normals.h"
#include <cmath>
```

### Classes

- class **BlackScholes**

### Enumerations

- enum **TypeOptionBS** { **Call**, **Put** }

#### 7.6.1 Enumeration Type Documentation

##### 7.6.1.1 enum TypeOptionBS

###### Enumeration values:

*Call*

*Put*

Definition at line 8 of file BlackScholes.h.

Referenced by `inputBSOption()`.

## 7.7 bond.cpp File Reference

```
#include "..\bond.h"
```

## 7.8 bond.h File Reference

```
#include "../common/types.h"
#include "../common/date.h"
#include <valarray>
#include "../PartB/yieldCurve.h"
#include "../PartF/creditCurve.h"
```

### Classes

- class **cashflow**
- class **bond**
- class **treasurybond**
- class **riskybond**

## 7.9 CashFlow.cpp File Reference

```
#include "../cashflow.h"
```

## 7.10 CashFlow.h File Reference

```
#include "../PartB/yieldCurve.h"  
#include "../SwapLeg.h"  
#include "../Common/types.h"  
#include <iostream>
```

### Classes

- class **CashFlow**

## 7.11 convertiblebond.cpp File Reference

```
#include ".\convertiblebond.h"  
#include ".\binomialTree.h"  
#include "../PartA/MonteCarlo1/PayOff.h"
```

### Functions

- ostream & operator<< (ostream &os, convertiblebond &cb)

#### 7.11.1 Function Documentation

##### 7.11.1.1 ostream& operator<< (ostream & os, convertiblebond & cb)

Definition at line 156 of file convertiblebond.cpp.

References convertiblebond::\_bt, convertiblebond::\_btCached, convertiblebond::\_callPrice, convertiblebond::\_conversionRatio, bond::\_faceamount, bond::\_issue, bond::\_maturity, convertiblebond::\_n, convertiblebond::\_putPrice, convertiblebond::\_stock, CB\_DEFAULT\_CALLPRICE, CB\_DEFAULT\_PUTPRICE, convertiblebond::delta(), convertiblebond::fairvalue(), convertiblebond::gamma(), asset::getPrice(), asset::GetVolatility(), convertiblebond::interestRateDelta(), convertiblebond::parityDelta(), convertiblebond::parityGamma(), and Date::toString().

## 7.12 convertiblebond.h File Reference

```
#include "../common/types.h"
#include "../PartH/bond.h"
#include "../PartC/asset.h"
#include "./binomialTree.h"
#include <iostream>
#include <valarray>
```

### Classes

- class convertiblebond

### Defines

- #define CB\_DEFAULT\_FACEAMOUNT 100
- #define CB\_DEFAULT\_MATURITY 1
- #define CB\_DEFAULT\_DAYCOUNT ACT\_360
- #define CB\_DEFAULT\_RATE 0.05
- #define CB\_DEFAULT\_SPREAD 0.02
- #define CB\_DEFAULT\_SO 100
- #define CB\_DEFAULT\_SIGMA 0.30
- #define CB\_DEFAULT\_STEPS 4
- #define CB\_DEFAULT\_RATIO 10
- #define CB\_DEFAULT\_CALLPRICE TN\_INFINITY
- #define CB\_DEFAULT\_PUTPRICE 0

### 7.12.1 Define Documentation

#### 7.12.1.1 #define CB\_DEFAULT\_CALLPRICE TN\_INFINITY

Definition at line 29 of file convertiblebond.h.

Referenced by operator<<().

#### 7.12.1.2 #define CB\_DEFAULT\_DAYCOUNT ACT\_360

Definition at line 16 of file convertiblebond.h.

#### 7.12.1.3 #define CB\_DEFAULT\_FACEAMOUNT 100

Definition at line 14 of file convertiblebond.h.

#### 7.12.1.4 #define CB\_DEFAULT\_MATURITY 1

Definition at line 15 of file convertiblebond.h.

**7.12.1.5 #define CB\_DEFAULT\_PUTPRICE 0**

Definition at line 30 of file convertiblebond.h.

Referenced by operator<<().

**7.12.1.6 #define CB\_DEFAULT\_RATE 0.05**

Definition at line 17 of file convertiblebond.h.

**7.12.1.7 #define CB\_DEFAULT\_RATIO 10**

Definition at line 28 of file convertiblebond.h.

**7.12.1.8 #define CB\_DEFAULT\_SIGMA 0.30**

Definition at line 22 of file convertiblebond.h.

**7.12.1.9 #define CB\_DEFAULT\_SO 100**

Definition at line 21 of file convertiblebond.h.

**7.12.1.10 #define CB\_DEFAULT\_SPREAD 0.02**

Definition at line 18 of file convertiblebond.h.

**7.12.1.11 #define CB\_DEFAULT\_STEPS 4**

Definition at line 25 of file convertiblebond.h.



## 7.13 creditCurve.cpp File Reference

```
#include ".\creditCurve.h"  
#include "..\common\utils.h"  
#include <cmath>
```

### Functions

- ostream & operator<< (ostream &os, const creditCurve &c)

#### 7.13.1 Function Documentation

##### 7.13.1.1 ostream& operator<< (ostream & os, const creditCurve & c)

Definition at line 423 of file creditCurve.cpp.

References creditCurve::\_combined.

## 7.14 creditCurve.h File Reference

```
#include "../common/types.h"
#include "../common/date.h"
#include "../PartB/yieldCurve.h"
#include <string.h>
#include <math.h>
#include <valarray>
```

### Classes

- class **CreditSpreadPoint**  
*used to encapsulate a spread at a given maturity*
- class **creditCurve**

### Defines

- **#define CC\_MAX\_NUM\_SPREADS 30**
- **#define CC\_DEFAULT\_RECOVERY\_RATE 0.40**
- **#define CC\_DEFAULT\_FREQUENCY Annual**
- **#define CC\_DEFAULT\_CURRENCY USD**
- **#define CC\_DEFAULT\_NAME "creditCurve"**

### Enumerations

- enum **CreditSpreadType { Absolute, Relative }**

#### 7.14.1 Define Documentation

##### 7.14.1.1 **#define CC\_DEFAULT\_CURRENCY USD**

Definition at line 19 of file creditCurve.h.

##### 7.14.1.2 **#define CC\_DEFAULT\_FREQUENCY Annual**

Definition at line 18 of file creditCurve.h.

##### 7.14.1.3 **#define CC\_DEFAULT\_NAME "creditCurve"**

Definition at line 20 of file creditCurve.h.

##### 7.14.1.4 **#define CC\_DEFAULT\_RECOVERY\_RATE 0.40**

Definition at line 17 of file creditCurve.h.

### 7.14.1.5 `#define CC_MAX_NUM_SPREADS 30`

Definition at line 14 of file creditCurve.h.

Referenced by `FileReader::buildCreditSpreadPointArray()`, and `creditCurve::creditCurve()`.

## 7.14.2 Enumeration Type Documentation

### 7.14.2.1 `enum CreditSpreadType`

Enumeration values:

*Absolute*

*Relative*

Definition at line 22 of file creditCurve.h.

Referenced by `FileReader::buildCreditSpreadPointArray()`, and `CreditSpreadPoint::getSpreadType()`.

## 7.15 credits.cpp File Reference

```
#include "main.h"
```

### Functions

- void `credits` ()

#### 7.15.1 Function Documentation

##### 7.15.1.1 void `credits` ()

Definition at line 3 of file `credits.cpp`.

Referenced by `main()`.

## 7.16 csvparser.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include "csvparser.h"
#include "../PartB/yieldCurve.h"
#include "../PartF/creditCurve.h"
#include "StringTokenizer.h"
```

### Namespaces

- namespace **std**

## 7.17 csvparser.h File Reference

```
#include <string>
#include "../PartB/yieldCurve.h"
#include "../PartF/creditCurve.h"
```

### Classes

- class `CSVParser`

## 7.18 date.cpp File Reference

```
#include ".\date.h"  
#include <time.h>
```

## 7.19 date.h File Reference

```
#include ".\types.h"  
#include <stdio.h>
```

### Classes

- class **Date**

### Typedefs

- typedef **ShortNatural Day**
- typedef **ShortNatural Year**

### Enumerations

- enum **Weekday** {  
    **Sunday** = 1, **Monday** = 2, **Tuesday** = 3, **Wednesday** = 4,  
    **Thursday** = 5, **Friday** = 6, **Saturday** = 7 }
- enum **Month** {  
    **January** = 1, **February** = 2, **March** = 3, **April** = 4,  
    **May** = 5, **June** = 6, **July** = 7, **August** = 8,  
    **September** = 9, **October** = 10, **November** = 11, **December** = 12 }
- enum **TimeUnit** { **Days**, **Weeks**, **Months**, **Years** }
- enum **Frequency** {  
    **NoFrequency** = -1, **Once** = 0, **Annual** = 1, **Semiannual** = 2,  
    **EveryFourthMonth** = 3, **Quarterly** = 4, **Bimonthly** = 6, **Monthly** = 12 }
- enum **BusinessDayConvention** {  
    **Unadjusted**, **Preceding**, **ModifiedPreceding**, **Following**,  
    **ModifiedFollowing**, **MonthEndReference** }
- enum **DayCountConvention** { **ACT\_365**, **ACT\_360**, **Day30\_365**, **Day30\_360** }

### 7.19.1 Typedef Documentation

#### 7.19.1.1 typedef ShortNatural Day

**Author:**

Simon

Definition at line 12 of file date.h.

Referenced by `Date::advance()`, `Date::Date()`, `Date::dayOfMonth()`, `Date::dayOfYear()`, `UsDate::isBusinessDay()`, `Date::lastDayOfMonth()`, `Date::month()`, `CSVParser::operator>>()`, and `Date::setDateToToday()`.



### 7.19.1.2 typedef ShortNatural Year

Definition at line 13 of file date.h.

Referenced by Date::advance(), Date::Date(), Date::endOfMonth(), Date::isLeap(), Date::nthWeekday(), CSVParser::operator>>(), Date::year(), and Date::yearOffset().

## 7.19.2 Enumeration Type Documentation

### 7.19.2.1 enum BusinessDayConvention

Enumeration values:

*Unadjusted*

*Preceding*

*ModifiedPreceding*

*Following*

*ModifiedFollowing*

*MonthEndReference*

Definition at line 54 of file date.h.

Referenced by flowSchedule::getBusDayConv().

### 7.19.2.2 enum DayCountConvention

Enumeration values:

*ACT\_365*

*ACT\_360*

*Day30\_365*

*Day30\_360*

Definition at line 74 of file date.h.

Referenced by yieldPoint::getDayCount(), inputBond(), and mainbond().

### 7.19.2.3 enum Frequency

Enumeration values:

*NoFrequency*

*Once*

*Annual*

*Semiannual*

*EveryFourthMonth*

*Quarterly*

*Bimonthly*

*Monthly*

Definition at line 44 of file date.h.

Referenced by creditCurve::getFrequency(), inputBond(), mainbond(), and SwapLeg::SwapLeg().

#### 7.19.2.4 enum Month

Enumeration values:

*January*

*February*

*March*

*April*

*May*

*June*

*July*

*August*

*September*

*October*

*November*

*December*

Definition at line 24 of file date.h.

Referenced by `Date::advance()`, `Date::endOfMonth()`, `UsDate::isBusinessDay()`, `mainyieldcurve()`, `Date::month()`, `CSVParser::operator>>()`, and `Date::setDateToToday()`.

#### 7.19.2.5 enum TimeUnit

Enumeration values:

*Days*

*Weeks*

*Months*

*Years*

Definition at line 38 of file date.h.

#### 7.19.2.6 enum Weekday

Enumeration values:

*Sunday*

*Monday*

*Tuesday*

*Wednesday*

*Thursday*

*Friday*

*Saturday*

Definition at line 15 of file date.h.

Referenced by `UsDate::isBusinessDay()`, `Date::nextWeekday()`, `Date::nthWeekday()`, and `Date::weekday()`.

## 7.20 Drift.cpp File Reference

```
#include "../../common/types.h"  
#include "../drift.h"  
#include <math.h>
```

## 7.21 Drift.h File Reference

```
#include "../../PartB/yieldCurve.h"  
#include "../../PartE/volsurface.h"  
#include "../../common/date.h"
```

### Classes

- class **Drift**

## 7.22 Exotics.cpp File Reference

```
#include "../Exotics.h"
```

## 7.23 Exotics.h File Reference

```
#include "../PartB/yieldCurve.h"
#include "../PartE/volsurface.h"
#include <valarray>
```

### Classes

- class **Exotics**

### Enumerations

- enum **exoticsType** {  
**AsianCall**, **AsianPut**, **RevLookbackCall**, **RevLookbackPut**,  
**FlooredCliquet**, **CappedCliquet**, **CollaredCliquet**, **BarrierCall**,  
**BarrierPut** }

### Functions

- **Real mainmc** (**Real** Expiry, **Real** Strike, **Real** Spot, **volsurface** \*pvolsurface, **yieldCurve** \*pyieldCurve, **LongNatural** nPaths, **LongNatural** nDates, **Integer** PrdName)

### Variables

- const **Real** defaultShiftVolSurface = 0.01
- const **Real** defaultAdvDays = 1.

### 7.23.1 Enumeration Type Documentation

#### 7.23.1.1 enum **exoticsType**

Enumeration values:

*AsianCall*

*AsianPut*

*RevLookbackCall*

*RevLookbackPut*

*FlooredCliquet*

*CappedCliquet*

*CollaredCliquet*

*BarrierCall*

*BarrierPut*

Definition at line 22 of file Exotics.h.

Referenced by choiceToType(), and inputExoticOptionOnSingleAsset().

## 7.23.2 Function Documentation

### 7.23.2.1 Real mainmc (Real *Expiry*, Real *Strike*, Real *Spot*, volsurface \* *pvolsurface*, yieldCurve \* *pyieldCurve*, LongNatural *nPaths*, LongNatural *nDates*, Integer *PrdName*)

Definition at line 40 of file mainmontecarlo.cpp.

References yieldCurve::discountFactor(), GaussianProcess::GetStepIncrements(), Drift::GetvDates(), Drift::GetvDrift(), Integer, LongNatural, MCEngine::MCResult(), r, Real, MCEngine::RunEngineGeneral(), Date::setDateToToday(), and Random::SetSeed().

Referenced by Exotics::getPrice(), mainbinomialtree(), and mainmontecarlo().

## 7.23.3 Variable Documentation

### 7.23.3.1 const Real defaultAdvDays = 1. [static]

Definition at line 17 of file Exotics.h.

### 7.23.3.2 const Real defaultShiftVolSurface = 0.01 [static]

**Author:**

Simon

Definition at line 16 of file Exotics.h.

## 7.24 filereader.cpp File Reference

```
#include <fstream>
#include "filereader.h"
#include "csvparser.h"
```



## 7.25 flereader.h File Reference

```
#include <string>
#include "..\PartB\yieldCurve.h"
#include "..\PartF\creditCurve.h"
#include "..\PartE\volsurface.h"
```

### Classes

- class **FileReader**

## 7.26 GaussianProcess.cpp File Reference

```
#include "GaussianProcess.h"  
#include <math.h>  
#include <cmath>  
#include <valarray>
```

## 7.27 GaussianProcess.h File Reference

```
#include "../../common/Date.h"  
#include "../../common/types.h"  
#include "../../PartE/volsurface.h"  
#include <valarray>
```

### Classes

- class `GaussianProcess`

### Defines

- `#define GAUSSIANPROCESS_H`

#### 7.27.1 Define Documentation

##### 7.27.1.1 `#define GAUSSIANPROCESS_H`

Definition at line 3 of file `GaussianProcess.h`.

## 7.28 importData.cpp File Reference

```
#include "..\importData.h"
```

## 7.29 importData.h File Reference

```
#include <iostream>
#include <string>
#include "..\PartF\creditCurve.h"
#include "..\common\filereader.h"
#include "..\common\date.h"
#include "..\PartB\yieldCurve.h"
#include "..\PartE\volsurface.h"
```

### Classes

- struct **marketData**
- class **importData**

### Variables

- const string **YCNAME** = "yieldcurve.csv"
- const string **VSNAME** = "volsurface.csv"
- const string **CSNAME** = "creditspreads.csv"

#### 7.29.1 Variable Documentation

**7.29.1.1** `const string CSNAME = "creditspreads.csv" [static]`

Definition at line 20 of file importData.h.

**7.29.1.2** `const string VSNAME = "volsurface.csv" [static]`

Definition at line 19 of file importData.h.

**7.29.1.3** `const string YCNAME = "yieldcurve.csv" [static]`

#### **Author:**

Yann

Definition at line 18 of file importData.h.

## 7.30 interpolator.cpp File Reference

```
#include "../interpolator.h"
```

### Functions

- Integer interpolatormain ()

#### 7.30.1 Function Documentation

##### 7.30.1.1 Integer interpolatormain ()

Definition at line 198 of file interpolator.cpp.

References Integer, interpolator::interpolate(), and Real.

## 7.31 interpolator.h File Reference

```
#include <iostream>
#include <valarray>
#include <fstream>
#include <istream>
#include <ostream>
#include "../types.h"
```

### Classes

- class **interpolator**

## 7.32 main.cpp File Reference

```
#include "main.h"
```

### Functions

- **Natural main** (**Natural** argc, char \*\*argv)

#### 7.32.1 Function Documentation

##### 7.32.1.1 **Natural main** (**Natural** *argc*, char \*\* *argv*)

Definition at line 4 of file main.cpp.

References credits(), maintests(), Natural, and productsCreationMenu().



## 7.33 main.h File Reference

```
#include <string.h>
#include "testObjects.h"
#include "productscreation.h"
#include "importData.h"
```

### Functions

- **Natural maintests** (Natural argc, char \*\*argv)
- **void credits** ()

#### 7.33.1 Function Documentation

##### 7.33.1.1 void credits ()

Definition at line 3 of file credits.cpp.

Referenced by main().

##### 7.33.1.2 Natural maintests (Natural *argc*, char \*\* *argv*)

Definition at line 7 of file testObjects.cpp.

References mainasset(), mainbinomialtree(), mainbond(), mainconvertiblebond(), maincreditcurve(), maindate(), mainfilereader(), maininterpolator(), mainIRVanillaSwap(), mainmatrix(), mainmontecarlo(), mainoption(), mainoptionstrategy(), mainrainbowoptions(), mainvarianceswap(), mainvolsurface(), mainyieldcurve(), Natural, FileReader::setdatadir(), and Short-Natural.

Referenced by main().

## 7.34 mainbinomialtree.cpp File Reference

```
#include "..\PartA\BlackScholes\BlackScholes.h"
#include "../Interface/main.h"
#include "..\PartJ\binomialTree.h"
#include "..\common\utils.h"
```

### Functions

- bool **mainbinomialtree** (void)  
*test of the binomial tree object*

#### 7.34.1 Function Documentation

##### 7.34.1.1 bool mainbinomialtree (void)

test of the binomial tree object

#### Returns:

true if pass, otherwise fail

Definition at line 7 of file mainbinomialtree.cpp.

References Call, BlackScholes::getPrice(), binomialTree::getPrice(), binomialTree::getStockProcess(), LongNatural, mainmc(), Natural, r, Real, realsEqual(), and binomialTree::runEngineCall().

Referenced by maintests().

## 7.35 mainbond.cpp File Reference

```
#include "..\PartB\yieldCurve.h"
#include "..\PartF\creditCurve.h"
#include "..\PartH\bond.h"
#include "..\common\filereader.h"
#include <iostream>
#include <valarray>
```

### Functions

- bool **mainbond** (void)  
*test the bond functionality*

#### 7.35.1 Function Documentation

##### 7.35.1.1 bool mainbond (void)

test the bond functionality

##### Returns:

true if pass, otherwise fail

Definition at line 12 of file mainbond.cpp.

References ACT\_365, bond::convexity(), DayCountConvention, bond::duration(), bond::fairvalue(), Frequency, May, November, Real, Semiannual, and bond::yieldToMaturity().

Referenced by maintests().

## 7.36 mainconvertiblebond.cpp File Reference

```
#include "..\PartJ\convertiblebond.h"  
#include "..\common\utils.h"
```

### Functions

- `bool mainconvertiblebond (void)`  
*test of the convertible bond object*

#### 7.36.1 Function Documentation

##### 7.36.1.1 `bool mainconvertiblebond (void)`

test of the convertible bond object

##### **Returns:**

true if pass, otherwise fail

Definition at line 5 of file mainconvertiblebond.cpp.

References ACT\_365, Days, convertiblebond::fairvalue(), Months, Natural, convertiblebond::parityDelta(), convertiblebond::parityGamma(), Date::plus(), Real, reals-Equal(), convertiblebond::rho(), Date::setDateToToday(), and asset::setPrice().

Referenced by maintests().

## 7.37 maincreditcurves.cpp File Reference

```
#include "..\PartF\creditCurve.h"
#include "..\common\filereader.h"
#include <iostream>
#include <valarray>
#include <time.h>
```

### Functions

- **bool maincreditcurve** (void)  
*test of the credit curve object*

#### 7.37.1 Function Documentation

##### 7.37.1.1 bool maincreditcurve (void)

test of the credit curve object

**Returns:**

true if pass, otherwise fail

Definition at line 9 of file maincreditcurves.cpp.

References `FileReader::buildCreditSpreadPointArray()`, `FileReader::buildYieldPointArray()`, `creditCurve::creditSpread()`, `creditCurve::cumulativeDefaultProbability()`, `creditCurve::defaultProbability()`, `FileReader::getdatadirasString()`, `creditCurve::hazardRate()`, `Natural`, `Real`, `creditCurve::riskyDiscountFactor()`, and `creditCurve::survivalProbability()`.

Referenced by `maintests()`.

## 7.38 maindate.cpp File Reference

```
#include "../Interface/main.h"  
#include "../common/date.h"  
#include <iostream>
```

### Functions

- bool **maindate** (void)  
*test the date class*

### 7.38.1 Function Documentation

#### 7.38.1.1 bool maindate (void)

test the date class

**Returns:**

true if pass, otherwise fail

Definition at line 7 of file maindate.cpp.

References `Date::lastDayOfMonth()`, `Date::setDateToToday()`, and `Date::toString()`.

Referenced by `maintests()`.

## 7.39 mainfilereader.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include "../Interface/main.h"
#include "../common/csvparser.h"
#include "../common/filereader.h"
#include "../PartB/yieldCurve.h"
```

### Functions

- bool **mainfilereader** (void)  
*test the file reader functionality*
- bool **fr\_basic** (void)  
*file reader basic test*

### 7.39.1 Function Documentation

#### 7.39.1.1 bool fr\_basic (void)

file reader basic test

**Returns:**

true if pass, otherwise fail

Definition at line 53 of file mainfilereader.cpp.

References FileReader::fileexists(), and FileReader::getdatadirasString().

Referenced by mainfilereader().

#### 7.39.1.2 bool mainfilereader (void)

test the file reader functionality

**Returns:**

true if pass, otherwise fail

Definition at line 45 of file mainfilereader.cpp.

References fr\_basic().

Referenced by maintests().

## 7.40 maininterpolator.cpp File Reference

```
#include "..\common\date.h"  
#include "..\common\interpolator.h"  
#include <valarray>
```

### Functions

- **bool maininterpolator** (void)  
*test the 2d/3d interpolator functionality*

#### 7.40.1 Function Documentation

##### 7.40.1.1 **bool maininterpolator** (void)

test the 2d/3d interpolator functionality

**Returns:**

true if pass, otherwise fail

Definition at line 5 of file maininterpolator.cpp.

References Integer, interpolator::interpolate(), and Real.

Referenced by maintests().



## 7.41 mainIRVanillaSwap.cpp File Reference

```
#include "../Interface/main.h"
#include "../PartD/VanillaSwap.h"
#include <iostream>
```

### Functions

- `bool mainIRVanillaSwap (void)`  
*Test of the Vanilla Swap - Yann.*

#### 7.41.1 Function Documentation

##### 7.41.1.1 `bool mainIRVanillaSwap (void)`

Test of the Vanilla Swap - Yann.

**Author:**

Yann

Definition at line 9 of file mainIRVanillaSwap.cpp.

References `FileReader::buildYieldPointArray()`, `FileReader::getdatadirasString()`, `CashFlow::getFairValue()`, `Date::plusMonths()`, `Date::plusYears()`, `Real`, `VanillaSwap::returnPrice()`, and `Date::setDateToToday()`.

Referenced by `maintests()`.

## 7.42 mainmatrix.cpp File Reference

```
#include "..\common\matrix.h"
#include "..\common\types.h"
#include "..\common\utils.h"
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <valarray>
```

### Functions

- bool **mainmatrix** (void)  
*test the matrix functions*

### 7.42.1 Function Documentation

#### 7.42.1.1 bool mainmatrix (void)

test the matrix functions

**Author:**

Yann

Definition at line 16 of file mainmatrix.cpp.

References `Matrix::CholeskyDecomposition()`, `Matrix::GetTransposed()`, `Natural`, `Matrix::SetValue()`, `transform1DvalarrayToColumnMatrix()`, `transform2DvalarrayToMatrix()`, `transformColumnMatrixTo1Dvalarray()`, and `transformMatrixTo2Dvalarray()`.

Referenced by `maintests()`.

## 7.43 mainmontecarlo.cpp File Reference

```
#include "../Interface/main.h"
#include "../PartA/MonteCarlo1/ParkMiller.h"
#include "../PartA/MonteCarlo1/MersenneTwister.h"
#include "../PartA/MonteCarlo1/RandC.h"
#include "../PartA/MonteCarlo1/Sobol.h"
#include "../PartA/MonteCarlo1/GaussianProcess.h"
#include "../common/Normals.h"
#include "../PartA/MonteCarlo1/PayOff.h"
#include "../PartA/MonteCarlo1/Random.h"
#include "../PartB/YieldCurve.h"
#include "../PartA/MonteCarlo1/Drift.h"
#include "../PartA/MonteCarlo1/MCEngine.h"
#include "time.h"
#include <iostream>
#include <valarray>
```

### Functions

- **bool mainmontecarlo** (void)  
*test the montecarlo option pricer functionality*
- **Real mainmc** (**Real** Expiry, **Real** Strike, **Real** Spot, **volsurface** \*pvolsurface, **yieldCurve** \*pyieldCurve, **LongNatural** nPaths, **LongNatural** nDates, **Integer** PrdName)

#### 7.43.1 Function Documentation

**7.43.1.1 Real mainmc** (**Real** *Expiry*, **Real** *Strike*, **Real** *Spot*, **volsurface** \**pvolsurface*, **yieldCurve** \* *pyieldCurve*, **LongNatural** *nPaths*, **LongNatural** *nDates*, **Integer** *PrdName*)

Definition at line 40 of file mainmontecarlo.cpp.

References `yieldCurve::discountFactor()`, `GaussianProcess::GetStepIncrements()`, `Drift::GetvDates()`, `Drift::GetvDrift()`, `Integer`, `LongNatural`, `MCEngine::MCResult()`, `r`, `Real`, `MCEngine::RunEngineGeneral()`, `Date::setDateToToday()`, and `Random::SetSeed()`.

Referenced by `Exotics::getPrice()`, `mainbinomialtree()`, and `mainmontecarlo()`.

#### 7.43.1.2 bool mainmontecarlo (void)

test the montecarlo option pricer functionality

#### Returns:

true if pass, otherwise fail

Definition at line 19 of file mainmontecarlo.cpp.

References LongNatural, mainmc(), and Real.

Referenced by maintests().

## 7.44 mainoptionstrategy.cpp File Reference

```
#include "../PartA/BlackScholes/OptionStrategy.h"
#include "../PartA/BlackScholes/BlackScholes.h"
#include <iostream>
```

### Functions

- **bool mainoption** (void)  
*test the BS option pricing functionality*
- **bool mainoptionstrategy** (void)  
*test the option strategy functionality*

#### 7.44.1 Function Documentation

##### 7.44.1.1 bool mainoption (void)

test the BS option pricing functionality

**Returns:**

true if pass, otherwise fail

Definition at line 5 of file mainoptionstrategy.cpp.

References Call, BlackScholes::getDelta(), BlackScholes::getGamma(), BlackScholes::getPrice(), BlackScholes::getRho(), BlackScholes::getTheta(), BlackScholes::getVega(), BlackScholes::getVolatility(), and LongNatural.

Referenced by maintests().

##### 7.44.1.2 bool mainoptionstrategy (void)

test the option strategy functionality

**Returns:**

true if pass, otherwise fail

Definition at line 26 of file mainoptionstrategy.cpp.

References OptionStrategy::addLongButterflySpread(), OptionStrategy::getGlobalDelta(), and OptionStrategy::returnPrice().

Referenced by maintests().

## 7.45 mainrainbowoptions.cpp File Reference

```
#include "../Interface/main.h"
#include "../PartI/rainbowoption.h"
#include "../PartB/YieldCurve.h"
#include <iostream>
#include <valarray>
```

### Functions

- **bool mainrainbowoptions** (void)  
*test of the credit curve object*

#### 7.45.1 Function Documentation

##### 7.45.1.1 bool mainrainbowoptions (void)

test of the credit curve object

**Author:**

Yann

\* Types (MonteCarlo, ClosedForm )

Definition at line 13 of file mainrainbowoptions.cpp.

References AssetsBasketMax, BestOf2AssetsCash, BetterOf2Assets, ClosedForm, RainbowOption::getCorrelRisk(), RainbowOption::getPartialDelta(), RainbowOption::getPartialGamma(), RainbowOption::getPartialVega(), RainbowOption::getPrice(), RainbowOption::getRho(), Max2AssetsCall, Max2AssetsPut, Min2AssetsCall, Min2AssetsPut, MonteCarlo, RealDate::setDateToToday(), RainbowOption::setRainbowType(), SpreadOptionMax, WorseOf2Assets, and WorstOf2AssetsCash.

Referenced by maintests().

## 7.46 maintestasset.cpp File Reference

```
#include "..\PartC\asset.h"
#include "../common/filereader.h"
#include <iostream>
#include <valarray>
#include <time.h>
```

### Functions

- **bool mainasset** (void)  
*test the asset functionality*

#### 7.46.1 Function Documentation

##### 7.46.1.1 bool mainasset (void)

test the asset functionality

**Author:**

Yann

Definition at line 14 of file maintestasset.cpp.

References a, FileReader::buildYieldPointArray(), EUR, asset::forwardPrice(), FileReader::getdatadirasString(), Natural, Date::plusDays(), Date::plusMonths(), Date::plusYears(), Real, Date::setDateToToday(), and Date::toString().

Referenced by maintests().

## 7.47 mainvarianceswap.cpp File Reference

```
#include "../PartA/BlackScholes/OptionStrategy.h"
#include "../PartK/VarianceSwap.h"
#include <iostream>
#include <valarray>
```

### Functions

- **bool mainvarianceswap** (void)  
*test of the variance swaps object*

#### 7.47.1 Function Documentation

##### 7.47.1.1 bool mainvarianceswap (void)

test of the variance swaps object

##### Returns:

true if pass, otherwise fail

Definition at line 9 of file mainvarianceswap.cpp.

References `OptionStrategy::addLongButterflySpread()`, `OptionStrategy::addOneBlackScholes-Object()`, `Call`, `VarianceSwap::getPrice()`, `Natural`, `Put`, and `Real`.

Referenced by `maintests()`.



## 7.48 mainvolsurface.cpp File Reference

```
#include "..\PartB\yieldCurve.h"
#include "..\PartE\volsurface.h"
#include "..\common\filereader.h"
#include <iostream>
#include <fstream>
#include <valarray>
```

### Functions

- bool **mainvolsurface** (void)  
*test the volsurface functionality*

#### 7.48.1 Function Documentation

##### 7.48.1.1 bool mainvolsurface (void)

test the volsurface functionality

##### Returns:

true if pass, otherwise fail

Definition at line 12 of file mainvolsurface.cpp.

References `FileReader::buildVolSurfaceParams()`, `December`, `July`, `March`, `Real`, `September`, `volsurface::setvolsurface()`, `Date::toString()`, and `volsurface::volatility()`.

Referenced by `maintests()`.

## 7.49 mainyieldcurves.cpp File Reference

```
#include "..\PartB\yieldCurve.h"
#include "../common/filereader.h"
#include <iostream>
#include <valarray>
#include <time.h>
```

### Functions

- **bool mainyieldcurve** (void)  
*test the yield curve functionality*

#### 7.49.1 Function Documentation

##### 7.49.1.1 **bool mainyieldcurve** (void)

test the yield curve functionality

**Author:**

Yann

Definition at line 13 of file mainyieldcurves.cpp.

References `FileReader::buildYieldPointArray()`, `Continuous`, `Discrete`, `FileReader::getdatadirasString()`, `yieldCurve::getMaturitiesInTheZCBCurve()`, `Month`, `Natural`, `Date::plusMonths()`, `Real`, `yieldCurve::rotateZCBRateCurve()`, `Date::setDateToToday()`, `yieldCurve::shiftZCBRateCurve()`, and `yieldCurve::spotRate()`.

Referenced by `maintests()`.

## 7.50 matrix.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include "matrix.h"
#include "utils.h"
```

### Functions

- **Matrix & IdentityMatrix** (int Diagonal)
- ostream & **operator<<** (ostream &ostr, const **Matrix** &obj)

#### 7.50.1 Function Documentation

##### 7.50.1.1 Matrix& IdentityMatrix (int *Diagonal*)

Definition at line 1465 of file matrix.cpp.

References Matrix::IdentityMatrix().

Referenced by RainbowOption::RainbowOption().

##### 7.50.1.2 ostream& operator<< (ostream & *ostr*, const Matrix & *obj*)

Definition at line 1472 of file matrix.cpp.

## 7.51 matrix.h File Reference

```
#include <iostream>
#include <fstream>
#include "..\common\types.h"
```

### Classes

- class **Matrix**

### Functions

- **Matrix & IdentityMatrix** (int Diagonal)
- ostream & **operator<<** (ostream &ostr, const **Matrix** &obj)

#### 7.51.1 Function Documentation

##### 7.51.1.1 Matrix& IdentityMatrix (int *Diagonal*)

Definition at line 1465 of file matrix.cpp.

References Matrix::IdentityMatrix().

Referenced by RainbowOption::RainbowOption().

##### 7.51.1.2 ostream& operator<< (ostream & *ostr*, const Matrix & *obj*)

Definition at line 1472 of file matrix.cpp.

References Matrix::GetColumns(), and Matrix::GetRows().

## 7.52 MCEngine.cpp File Reference

```
#include "..\mceengine.h"
```

## 7.53 MCEngine.h File Reference

```
#include "Random.h"  
#include "GaussianProcess.h"  
#include "PayOff.h"  
#include "../../common/Normals.h"  
#include "../../common/matrix.h"
```

### Classes

- class **MCEngine**

## 7.54 MersenneTwister.cpp File Reference

```
#include "MersenneTwister.h"
#include "time.h"
```

### Variables

- const **LongNatural** **N** = 624
- const **LongNatural** **M** = 397
- const **LongNatural** **MATRIX\_A** = 0x9908b0dfUL
- const **LongNatural** **UPPER\_MASK** = 0x80000000UL
- const **LongNatural** **LOWER\_MASK** = 0x7ffffffUL

### 7.54.1 Variable Documentation

#### 7.54.1.1 const LongNatural LOWER\_MASK = 0x7ffffffUL

Definition at line 12 of file MersenneTwister.cpp.

Referenced by MersenneTwister::GetOneRandomInteger().

#### 7.54.1.2 const LongNatural M = 397

Definition at line 6 of file MersenneTwister.cpp.

Referenced by volsurface::forwardvolsurface(), MersenneTwister::GetOneRandomInteger(), interpolator::interpolate(), yieldCurve::SequentDiscountFactorsByInvertSwapMatrix(), volsurface::shiftedvolsurface(), transformColumnMatrixTo1Dvalarray(), and transformMatrixTo2Dvalarray().

#### 7.54.1.3 const LongNatural MATRIX\_A = 0x9908b0dfUL

Definition at line 8 of file MersenneTwister.cpp.

Referenced by MersenneTwister::GetOneRandomInteger().

#### 7.54.1.4 const LongNatural N = 624

Definition at line 5 of file MersenneTwister.cpp.

Referenced by volsurface::forwardvolsurface(), MersenneTwister::GetOneRandomInteger(), interpolator::interpolate(), MersenneTwister::MersenneTwister(), MersenneTwister::SetSeed(), volsurface::setvolsurface(), and volsurface::shiftedvolsurface().

#### 7.54.1.5 const LongNatural UPPER\_MASK = 0x80000000UL

Definition at line 10 of file MersenneTwister.cpp.

Referenced by MersenneTwister::GetOneRandomInteger().

## 7.55 MersenneTwister.h File Reference

```
#include "../RandomGenerator.h"  
#include "../../Common/types.h"  
#include <valarray>
```

### Classes

- class `MersenneTwister`



## 7.56 Normals.cpp File Reference

```
#include ".\normals.h"
#include <cmath>
```

### Functions

- **Real NormalDensity** (**Real** *x*)
- **Real InverseCumulativeNormal** (**Real** *u*)
- **Real CumulativeNormal** (**Real** *x*)
- **Real Average** (*valarray*< **Real** > *Ptr*, **LongNatural** *dim*)
- **Real Maximize** (*valarray*< **Real** > *Ptr*, **LongNatural** *dim*)
- **Real CumulativeBivariateNormal** (**Real** *a*, **Real** *b*, **Real** *rho*)
  - Bivariate normal distribution - from Hull's book.*
- **Real SubFunctionForBivariateNormal** (**Real** *X*, **Real** *y*, **Real** *ap*, **Real** *bp*, **Real** *rho*)

### Variables

- **const Real OneOverRootTwoPi** = 0.398942280401433

### 7.56.1 Function Documentation

#### 7.56.1.1 Real Average (*valarray*< **Real** > *Ptr*, **LongNatural** *dim*)

Definition at line 116 of file Normals.cpp.

References **LongNatural**, and **Real**.

Referenced by **PayOff::AsianCall()**, and **PayOff::AsianPut()**.

#### 7.56.1.2 Real CumulativeBivariateNormal (**Real** *a*, **Real** *b*, **Real** *rho*)

Bivariate normal distribution - from Hull's book.

#### Author:

Yann

Definition at line 144 of file Normals.cpp.

References *a*, **CumulativeNormal()**, **Natural**, **Real**, **sign()**, and **SubFunctionForBivariateNormal()**.

Referenced by **RainbowOption::compute\_A()**, **RainbowOption::compute\_B()**, and **RainbowOption::compute\_C()**.

#### 7.56.1.3 Real CumulativeNormal (**Real** *x*)

Definition at line 82 of file Normals.cpp.

References *a*, **NormalDensity()**, and **Real**.

Referenced by `RainbowOption::compute_A()`, `RainbowOption::compute_B()`, `CumulativeBivariateNormal()`, `BlackScholes::getDelta()`, `BlackScholes::getPrice()`, `BlackScholes::getRho()`, and `BlackScholes::getTheta()`.

#### 7.56.1.4 Real InverseCumulativeNormal (Real $u$ )

Definition at line 23 of file `Normals.cpp`.

References `a`, `r`, and `Real`.

Referenced by `Random::GetGaussian()`, and `Random::GetGaussians()`.

#### 7.56.1.5 Real Maximize (valarray< Real > $Ptr$ , LongNatural $dim$ )

Definition at line 130 of file `Normals.cpp`.

References `LongNatural`, and `Real`.

Referenced by `PayOff::RevLookbackCall()`, and `PayOff::RevLookbackPut()`.

#### 7.56.1.6 Real NormalDensity (Real $x$ )

Definition at line 17 of file `Normals.cpp`.

References `OneOverRootTwoPi`, and `Real`.

Referenced by `CumulativeNormal()`, `BlackScholes::getGamma()`, `BlackScholes::getTheta()`, and `BlackScholes::getVega()`.

#### 7.56.1.7 Real SubFunctionForBivariateNormal (Real $X$ , Real $y$ , Real $ap$ , Real $bp$ , Real $rho$ )

Definition at line 189 of file `Normals.cpp`.

References `r`, and `Real`.

Referenced by `CumulativeBivariateNormal()`.

### 7.56.2 Variable Documentation

#### 7.56.2.1 `const Real OneOverRootTwoPi = 0.398942280401433`

Definition at line 14 of file `Normals.cpp`.

Referenced by `NormalDensity()`.

## 7.57 Normals.h File Reference

```
#include <valarray>
#include "../types.h"
#include "../utils.h"
```

### Functions

- **Real NormalDensity** (**Real** *x*)
- **Real CumulativeNormal** (**Real** *x*)
- **Real InverseCumulativeNormal** (**Real** *x*)
- **Real Average** (*valarray*< **Real** > *Ptr*, **LongNatural** *dim*)
- **Real Maximize** (*valarray*< **Real** > *Ptr*, **LongNatural** *dim*)
- **Real CumulativeBivariateNormal** (**Real** *a*, **Real** *b*, **Real** *rho*)
 

*Bivariate normal distribution - from Hull's book.*
- **Real SubFunctionForBivariateNormal** (**Real** *X*, **Real** *y*, **Real** *ap*, **Real** *bp*, **Real** *rho*)

### 7.57.1 Function Documentation

#### 7.57.1.1 Real Average (*valarray*< **Real** > *Ptr*, **LongNatural** *dim*)

Definition at line 116 of file Normals.cpp.

References **LongNatural**, and **Real**.

Referenced by `PayOff::AsianCall()`, and `PayOff::AsianPut()`.

#### 7.57.1.2 Real CumulativeBivariateNormal (**Real** *a*, **Real** *b*, **Real** *rho*)

Bivariate normal distribution - from Hull's book.

#### Author:

Yann

Definition at line 144 of file Normals.cpp.

References *a*, `CumulativeNormal()`, **Natural**, **Real**, `sign()`, and `SubFunctionForBivariateNormal()`.

Referenced by `RainbowOption::compute_A()`, `RainbowOption::compute_B()`, and `RainbowOption::compute_C()`.

#### 7.57.1.3 Real CumulativeNormal (**Real** *x*)

Definition at line 82 of file Normals.cpp.

References *a*, `NormalDensity()`, and **Real**.

Referenced by `RainbowOption::compute_A()`, `RainbowOption::compute_B()`, `CumulativeBivariateNormal()`, `BlackScholes::getDelta()`, `BlackScholes::getPrice()`, `BlackScholes::getRho()`, and `BlackScholes::getTheta()`.

**7.57.1.4 Real InverseCumulativeNormal (Real  $x$ )**

Definition at line 23 of file Normals.cpp.

References `a`, `r`, and `Real`.

Referenced by `Random::GetGaussian()`, and `Random::GetGaussians()`.

**7.57.1.5 Real Maximize (valarray< Real > Ptr, LongNatural dim)**

Definition at line 130 of file Normals.cpp.

References `LongNatural`, and `Real`.

Referenced by `PayOff::RevLookbackCall()`, and `PayOff::RevLookbackPut()`.

**7.57.1.6 Real NormalDensity (Real  $x$ )**

Definition at line 17 of file Normals.cpp.

References `OneOverRootTwoPi`, and `Real`.

Referenced by `CumulativeNormal()`, `BlackScholes::getGamma()`, `BlackScholes::getTheta()`, and `BlackScholes::getVega()`.

**7.57.1.7 Real SubFunctionForBivariateNormal (Real  $X$ , Real  $y$ , Real  $ap$ , Real  $bp$ , Real  $\rho$ )**

Definition at line 189 of file Normals.cpp.

References `r`, and `Real`.

Referenced by `CumulativeBivariateNormal()`.

## 7.58 OptionStrategy.cpp File Reference

```
#include "..\optionstrategy.h"
```

### Functions

- ostream & **operator<<** (ostream &os, const **OptionStrategy** &optionStrategy)

#### 7.58.1 Function Documentation

##### 7.58.1.1 ostream& operator<< (ostream & os, const OptionStrategy & optionStrategy)

###### Parameters:

*os*: the output stream to direct output to

*optionStrategy*: the option strategy to display

###### Returns:

output stream as is standard for operator<<

Definition at line 196 of file OptionStrategy.cpp.

References BlackScholes::getMaturity(), BlackScholes::getRate(), BlackScholes::getSpot(), BlackScholes::getStrike(), BlackScholes::getVolatility(), BlackScholes::isCall(), Natural, OptionStrategy::returnNbOptions(), OptionStrategy::returnOption(), and OptionStrategy::returnOptionQuantity().

## 7.59 OptionStrategy.h File Reference

```
#include "BlackScholes.h"  
#include "../common/types.h"  
#include <valarray>
```

### Classes

- class **OptionStrategy**

### Variables

- const **Natural maxNbOptions** = 500
- const **Real defaultshiftRate** = 0.0001
- const **Real defaultshiftVol** = 0.0001
- const **Real defaultshiftMat** = 0.01
- const **Real defaultshiftSpot** = 0.0001
- const **Real defaultshiftStrike** = 0.0001

### 7.59.1 Variable Documentation

**7.59.1.1** `const Real defaultshiftMat = 0.01` [static]

Definition at line 16 of file OptionStrategy.h.

**7.59.1.2** `const Real defaultshiftRate = 0.0001` [static]

Definition at line 14 of file OptionStrategy.h.

**7.59.1.3** `const Real defaultshiftSpot = 0.0001` [static]

Definition at line 17 of file OptionStrategy.h.

**7.59.1.4** `const Real defaultshiftStrike = 0.0001` [static]

Definition at line 18 of file OptionStrategy.h.

**7.59.1.5** `const Real defaultshiftVol = 0.0001` [static]

Definition at line 15 of file OptionStrategy.h.

**7.59.1.6** `const Natural maxNbOptions = 500` [static]

Definition at line 13 of file OptionStrategy.h.

## 7.60 ParkMiller.cpp File Reference

```
#include ".\parkmiller.h"  
#include "time.h"
```

## 7.61 ParkMiller.h File Reference

```
#include "../common/types.h"
#include "RandomGenerator.h"
```

### Classes

- class **ParkMiller**

### Variables

- const **LongInteger a** = 16807
- const **LongInteger m** = 2147483647
- const **LongInteger q** = 127773
- const **LongInteger r** = 2836

### 7.61.1 Variable Documentation

#### 7.61.1.1 const LongInteger a = 16807

Definition at line 8 of file ParkMiller.h.

Referenced by `RainbowOption::compute_C()`, `CumulativeBivariateNormal()`, `CumulativeNormal()`, `ParkMiller::GetOneRandomInteger()`, `InverseCumulativeNormal()`, and `mainasset()`.

#### 7.61.1.2 const LongInteger m = 2147483647

Definition at line 9 of file ParkMiller.h.

Referenced by `Date::advance()`, `Date::Date()`, `Date::endOfMonth()`, `ParkMiller::GetOneRandomInteger()`, `ParkMiller::getUniform()`, `UsDate::isBusinessDay()`, `ParkMiller::Max()`, `Date::month()`, `Date::monthLength()`, `Date::monthOffset()`, `Date::nthWeekday()`, `yieldCurve::operator==()`, `CSVParser::operator>>()`, `yieldPoint::setMaturity()`, and `CreditSpreadPoint::setMaturity()`.

#### 7.61.1.3 const LongInteger q = 127773

Definition at line 10 of file ParkMiller.h.

Referenced by `Matrix::Determinant()`, `ParkMiller::GetOneRandomInteger()`, `Matrix::IdentityMatrix()`, `Matrix::operator *`, `Matrix::RightAppendIdentity()`, and `binomialTree::setClaimVariables()`.

#### 7.61.1.4 const LongInteger r = 2836

Definition at line 11 of file ParkMiller.h.

Referenced by `OptionStrategy::addLongButterflySpread()`, `OptionStrategy::addLongCallSpread()`, `OptionStrategy::addLongPutSpread()`, `OptionStrategy::addLongRatioCallSpread()`, `OptionStrategy::addLongStraddle()`, `OptionStrategy::addLongStrangle()`, `OptionStrategy::addOneOptionToStrategy()`, `yieldCurve::assignFlatRate()`, and `creditCurve::assignFlatSpread()`.



binomialTree::binomialTree(), BlackScholes::BlackScholes(), CreditSpreadPoint::CreditSpreadPoint(), ParkMiller::GetOneRandomInteger(), InverseCumulativeNormal(), volsurface::invertBSformula(), mainbinomialtree(), mainmc(), yieldPoint::setRate(), CreditSpreadPoint::setRate(), volsurface::setvolsurface(), SubFunctionForBivariateNormal(), and yieldPoint::yieldPoint().

## 7.62 PayOff.cpp File Reference

```
#include "./payoff.h"  
#include "../../common/Normals.h"  
#include <minmax.h>
```

## 7.63 PayOff.h File Reference

```
#include "../../common/types.h"  
#include <valarray>
```

### Classes

- class **PayOff**

## 7.64 PortFolio.cpp File Reference

```
#include "../PortFolio.h"
```

## 7.65 PortFolio.h File Reference

```
#include "../PartA/BlackScholes/OptionStrategy.h"
#include "../PartK/VarianceSwap.h"
#include "../PartD/VanillaSwap.h"
#include "../PartI/rainbowoption.h"
#include "../PartH/bond.h"
#include "../PartC/asset.h"
#include "../PartL/Exotics.h"
#include <valarray>
```

### Classes

- class **Portfolio**

### Variables

- const **Natural** `MAX_SIZE` = 100
- const **Natural** `MAX_SIZE_NAME` = 50

#### 7.65.1 Variable Documentation

##### 7.65.1.1 const Natural `MAX_SIZE` = 100

Definition at line 17 of file PortFolio.h.

Referenced by Portfolio::Portfolio().

##### 7.65.1.2 const Natural `MAX_SIZE_NAME` = 50

Definition at line 18 of file PortFolio.h.

Referenced by Portfolio::getCurrencyAsString(), and Portfolio::Portfolio().

## 7.66 productsCreation.cpp File Reference

```
#include "..\productscreation.h"
```

### Functions

- **bool productsCreationMenu** (**marketData** data)  
*Menu for this category.*
- **BlackScholes \* inputBSOption** (**marketData** data)  
*User interface to input and store a BS option.*
- **string outputCallPut** (char c)  
*Transform the input c/C/p/P into "Call" or "Put".*
- **OptionStrategy inputOptionStrategy** (**marketData** data)  
*User interface to input and store an option strategy and options in it.*
- **void inputSpecificOptionStrategy** (**marketData** data, **OptionStrategy** &strategy)  
*add something else to the strategy than a Call/Put*
- **void inputButterflySpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a butterfly spread to the strategy*
- **void inputCallSpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a call spread to the strategy*
- **void inputPutSpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a put spread to the strategy*
- **void inputRatioCallSpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a ratio call spread to the strategy*
- **void inputStraddle** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a Straddle to the strategy*
- **void inputStrangle** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a strangle to the strategy*
- **Exotics \* inputExoticOptionOnSingleAsset** (**marketData** &data)  
*User interface to input and store an Exotic Option with a single underlying (MC Price).*
- **convertiblebond \* inputConvertibleBond** (**marketData** &data)  
*User interface to input a convertible bond.*

- **bond \* inputBond (marketData &data)**  
*User interface to input a bond.*
- **exoticsType choiceToType (Natural choice)**  
*converts the 1 to 9 choice in a Exotics(p. 86) Type*
- **VanillaSwap \* inputVanillaSwap (marketData data)**  
*User interface to input an interest rate vanilla swap.*
- **rainbowType chooseRainbowType ()**
- **priceType choosePricingType ()**
- **RainbowOption \* inputRainbowOption (marketData data)**  
*To input a Rainbow Option.*

## 7.66.1 Function Documentation

### 7.66.1.1 exoticsType choiceToType (Natural choice)

converts the 1 to 9 choice in a **Exotics**(p. 86) Type

Definition at line 832 of file productsCreation.cpp.

References AsianCall, AsianPut, BarrierCall, BarrierPut, CappedCliquet, CollaredCliquet, exoticsType, FlooredCliquet, Natural, RevLookbackCall, and RevLookbackPut.

Referenced by inputExoticOptionOnSingleAsset().

### 7.66.1.2 priceType choosePricingType ()

Definition at line 1014 of file productsCreation.cpp.

References ClosedForm, MonteCarlo, Natural, and priceType.

Referenced by inputRainbowOption().

### 7.66.1.3 rainbowType chooseRainbowType ()

Definition at line 957 of file productsCreation.cpp.

References AssetsBasketMax, BestOf2AssetsCash, BetterOf2Assets, Max2AssetsCall, Max2AssetsPut, Min2AssetsCall, Min2AssetsPut, Natural, rainbowType, SpreadOptionMax, WorseOf2Assets, and WorstOf2AssetsCash.

Referenced by inputRainbowOption().

### 7.66.1.4 bond\* inputBond (marketData & data)

User interface to input a bond.

Definition at line 639 of file productsCreation.cpp.

References ACT\_360, ACT\_365, Annual, Bimonthly, bond::convexity(), marketData::creditcurve, Day30\_360, Day30\_365, DayCountConvention, bond::duration(), EveryFourthMonth, bond::fairvalue(), Frequency, Integer, Monthly, Natural, NoFrequency, Once, Date::plusDays(), Quarterly, Real, Semiannual, Date::setDateToToday(), marketData::yieldcurve, and bond::yieldToMaturity().

Referenced by productsCreationMenu().

#### 7.66.1.5 BlackScholes\* inputBSOption (marketData data)

User interface to input and store a BS option.

Definition at line 52 of file productsCreation.cpp.

References Call, BlackScholes::getDelta(), BlackScholes::getGamma(), BlackScholes::getPrice(), BlackScholes::getRho(), BlackScholes::getTheta(), BlackScholes::getVega(), Integer, Natural, outputCallPut(), Date::plusDays(), Put, Real, Date::setDateToToday(), yieldCurve::spotRate(), TypeOptionBS, volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputOptionStrategy(), and productsCreationMenu().

#### 7.66.1.6 void inputButterflySpread (marketData data, OptionStrategy & strategy, bool useMarketData)

add a butterfly spread to the strategy

Definition at line 214 of file productsCreation.cpp.

References OptionStrategy::addLongButterflySpread(), Integer, Natural, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

#### 7.66.1.7 void inputCallSpread (marketData data, OptionStrategy & strategy, bool useMarketData)

add a call spread to the strategy

Definition at line 263 of file productsCreation.cpp.

References OptionStrategy::addLongCallSpread(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

#### 7.66.1.8 convertiblebond\* inputConvertibleBond (marketData & data)

User interface to input a convertible bond.

Definition at line 533 of file productsCreation.cpp.

References ACT\_365, marketData::creditcurve, Integer, Natural, Date::plusDays(), Real, Date::setDateToToday(), and marketData::yieldcurve.

Referenced by productsCreationMenu().



### 7.66.1.9 Exotics\* inputExoticOptionOnSingleAsset (marketData & data)

User interface to input and store an Exotic Option with a single underlying (MC Price).

Definition at line 424 of file productsCreation.cpp.

References choiceToType(), CollaredCliquet, exoticsType, importData::getData(), Exotics::getDelta(), Exotics::getPrice(), Exotics::getRho(), Exotics::getTheta(), Exotics::getVega(), LongNatural, Natural, Real, importData::runUserDefinedInterface(), marketData::vols, and marketData::yieldcurve.

Referenced by productsCreationMenu().

### 7.66.1.10 OptionStrategy inputOptionStrategy (marketData data)

User interface to input and store an option strategy and options in it.

Definition at line 119 of file productsCreation.cpp.

References OptionStrategy::addOneBlackScholesObject(), OptionStrategy::getGlobalDelta(), OptionStrategy::getGlobalGamma(), OptionStrategy::getGlobalRho(), OptionStrategy::getGlobalTheta(), OptionStrategy::getGlobalVega(), inputBSOption(), inputSpecificOptionStrategy(), Natural, Real, and OptionStrategy::returnPrice().

Referenced by productsCreationMenu().

### 7.66.1.11 void inputPutSpread (marketData data, OptionStrategy & strategy, bool useMarketData)

add a put spread to the strategy

Definition at line 296 of file productsCreation.cpp.

References OptionStrategy::addLongPutSpread(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

### 7.66.1.12 RainbowOption\* inputRainbowOption (marketData data)

To input a Rainbow Option.

Definition at line 1039 of file productsCreation.cpp.

References choosePricingType(), chooseRainbowType(), RainbowOption::getCorrelRisk(), RainbowOption::getPartialDelta(), RainbowOption::getPartialGamma(), RainbowOption::getPartialVega(), RainbowOption::getPrice(), RainbowOption::getRho(), Natural, priceType, rainbowType, Real, Date::setDateToToday(), marketData::vols, and marketData::yieldcurve.

Referenced by productsCreationMenu().

### 7.66.1.13 void inputRatioCallSpread (marketData data, OptionStrategy & strategy, bool useMarketData)

add a ratio call spread to the strategy

Definition at line 329 of file productsCreation.cpp.

References `OptionStrategy::addLongRatioCallSpread()`, `Integer`, `Date::plusDays()`, `Real`, `Date::setDateToToday()`, `yieldCurve::spotRate()`, `volsurface::volatility()`, `marketData::vols`, and `marketData::yieldcurve`.

Referenced by `inputSpecificOptionStrategy()`.

#### **7.66.1.14 `void inputSpecificOptionStrategy (marketData data, OptionStrategy & strategy)`**

add something else to the strategy than a Call/Put

Definition at line 169 of file `productsCreation.cpp`.

References `inputButterflySpread()`, `inputCallSpread()`, `inputPutSpread()`, `inputRatioCallSpread()`, `inputStraddle()`, `inputStrangle()`, and `Natural`.

Referenced by `inputOptionStrategy()`.

#### **7.66.1.15 `void inputStraddle (marketData data, OptionStrategy & strategy, bool useMarketData)`**

add a Straddle to the strategy

Definition at line 362 of file `productsCreation.cpp`.

References `OptionStrategy::addLongStraddle()`, `Integer`, `Date::plusDays()`, `Real`, `Date::setDateToToday()`, `yieldCurve::spotRate()`, `volsurface::volatility()`, `marketData::vols`, and `marketData::yieldcurve`.

Referenced by `inputSpecificOptionStrategy()`.

#### **7.66.1.16 `void inputStrangle (marketData data, OptionStrategy & strategy, bool useMarketData)`**

add a strangle to the strategy

Definition at line 390 of file `productsCreation.cpp`.

References `OptionStrategy::addLongStrangle()`, `Integer`, `Date::plusDays()`, `Real`, `Date::setDateToToday()`, `yieldCurve::spotRate()`, `volsurface::volatility()`, `marketData::vols`, and `marketData::yieldcurve`.

Referenced by `inputSpecificOptionStrategy()`.

#### **7.66.1.17 `VanillaSwap* inputVanillaSwap (marketData data)`**

User interface to input an interest rate vanilla swap.

Definition at line 866 of file `productsCreation.cpp`.

References `Following`, `VanillaSwap::getFairValue1()`, `VanillaSwap::getFairValue2()`, `VanillaSwap::getRho()`, `VanillaSwap::getTheta()`, `Integer`, `Natural`, `Date::plusDays()`, `Real`, `VanillaSwap::returnPrice()`, `Date::setDateToToday()`, and `marketData::yieldcurve`.

Referenced by `productsCreationMenu()`.

**7.66.1.18 string outputCallPut (char *c*)**

Transform the input *c*/C/p/P into "Call" or "Put".

Definition at line 108 of file productsCreation.cpp.

Referenced by inputBSOption().

**7.66.1.19 bool productsCreationMenu (marketData *data*)**

Menu for this category.

Definition at line 4 of file productsCreation.cpp.

References inputBond(), inputBSOption(), inputConvertibleBond(), inputExoticOptionOnSingleAsset(), inputOptionStrategy(), inputRainbowOption(), inputVanillaSwap(), and Natural.

Referenced by main().

## 7.67 productsCreation.h File Reference

```
#include "../PartB/yieldCurve.h"
#include "../PartE/volsurface.h"
#include "../PartA/BlackScholes/BlackScholes.h"
#include "../PartA/BlackScholes/OptionStrategy.h"
#include "../PartL/Exotics.h"
#include "../PartH/bond.h"
#include "../PartD/VanillaSwap.h"
#include "../PartI/rainbowoption.h"
#include "../PartJ/convertiblebond.h"
#include "../importData.h"
#include <minmax.h>
```

### Functions

- bool **productsCreationMenu** (**marketData** data)
 

*Menu for this category.*
- **BlackScholes \* inputBSOption** (**marketData** data)
 

*User interface to input and store a BS option.*
- string **outputCallPut** (char c)
 

*Transform the input c/C/p/P into "Call" or "Put".*
- **OptionStrategy inputOptionStrategy** (**marketData** data)
 

*User interface to input and store an option strategy and options in it.*
- void **inputSpecificOptionStrategy** (**marketData** data, **OptionStrategy** &strategy)
 

*add something else to the strategy than a Call/Put*
- void **inputButterflySpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)
 

*add a butterfly spread to the strategy*
- void **inputCallSpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)
 

*add a call spread to the strategy*
- void **inputPutSpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)
 

*add a put spread to the strategy*
- void **inputRatioCallSpread** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)
 

*add a ratio call spread to the strategy*

- void **inputStraddle** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a Straddle to the strategy*
- void **inputStrangle** (**marketData** data, **OptionStrategy** &strategy, bool useMarketData)  
*add a strangle to the strategy*
- **Exotics** \* **inputExoticOptionOnSingleAsset** (**marketData** &data)  
*User interface to input and store an Exotic Option with a single underlying (MC Price).*
- **exoticsType** **choiceToType** (**Natural** choice)  
*converts the 1 to 9 choice in a **Exotics**(p. 86) Type*
- **bond** \* **inputBond** (**marketData** &data)  
*User interface to input a bond.*
- **convertiblebond** \* **inputConvertibleBond** (**marketData** &data)  
*User interface to input a convertible bond.*
- **VanillaSwap** \* **inputVanillaSwap** (**marketData** data)  
*User interface to input an interest rate vanilla swap.*
- **priceType** **choosePricingType** (**Natural** choice)  
*For Rainbow options, converts the input into a priceType.*
- **rainbowType** **chooseRainbowType** (**Natural** choice)  
*For Rainbow options, converts the input into a rainbowType.*
- **RainbowOption** \* **inputRainbowOption** (**marketData** data)  
*To input a Rainbow Option.*

## 7.67.1 Function Documentation

### 7.67.1.1 **exoticsType** **choiceToType** (**Natural** *choice*)

converts the 1 to 9 choice in a **Exotics**(p. 86) Type

Definition at line 832 of file productsCreation.cpp.

References **AsianCall**, **AsianPut**, **BarrierCall**, **BarrierPut**, **CappedCliquet**, **CollaredCliquet**, **exoticsType**, **FlooredCliquet**, **Natural**, **RevLookbackCall**, and **RevLookbackPut**.

Referenced by **inputExoticOptionOnSingleAsset**().

### 7.67.1.2 **priceType** **choosePricingType** (**Natural** *choice*)

For Rainbow options, converts the input into a priceType.

### 7.67.1.3 rainbowType chooseRainbowType (Natural *choice*)

For Rainbow options, converts the input into a rainbowType.

### 7.67.1.4 bond\* inputBond (marketData & data)

User interface to input a bond.

Definition at line 639 of file productsCreation.cpp.

References ACT\_360, ACT\_365, Annual, Bimonthly, bond::convexity(), marketData::creditcurve, Day30\_360, Day30\_365, DayCountConvention, bond::duration(), EveryFourthMonth, bond::fairvalue(), Frequency, Integer, Monthly, Natural, NoFrequency, Once, Date::plusDays(), Quarterly, Real, Semiannual, Date::setDateToToday(), marketData::yieldcurve, and bond::yieldToMaturity().

Referenced by productsCreationMenu().

### 7.67.1.5 BlackScholes\* inputBSOption (marketData data)

User interface to input and store a BS option.

Definition at line 52 of file productsCreation.cpp.

References Call, BlackScholes::getDelta(), BlackScholes::getGamma(), BlackScholes::getPrice(), BlackScholes::getRho(), BlackScholes::getTheta(), BlackScholes::getVega(), Integer, Natural, outputCallPut(), Date::plusDays(), Put, Real, Date::setDateToToday(), yieldCurve::spotRate(), TypeOptionBS, volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputOptionStrategy(), and productsCreationMenu().

### 7.67.1.6 void inputButterflySpread (marketData data, OptionStrategy & strategy, bool useMarketData)

add a butterfly spread to the strategy

Definition at line 214 of file productsCreation.cpp.

References OptionStrategy::addLongButterflySpread(), Integer, Natural, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

### 7.67.1.7 void inputCallSpread (marketData data, OptionStrategy & strategy, bool useMarketData)

add a call spread to the strategy

Definition at line 263 of file productsCreation.cpp.

References OptionStrategy::addLongCallSpread(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

**7.67.1.8 convertiblebond\* inputConvertibleBond (marketData & data)**

User interface to input a convertible bond.

Definition at line 533 of file productsCreation.cpp.

References ACT\_365, marketData::creditcurve, Integer, Natural, Date::plusDays(), Real, Date::setDateToToday(), and marketData::yieldcurve.

Referenced by productsCreationMenu().

**7.67.1.9 Exotics\* inputExoticOptionOnSingleAsset (marketData & data)**

User interface to input and store an Exotic Option with a single underlying (MC Price).

Definition at line 424 of file productsCreation.cpp.

References choiceToType(), CollaredCliquet, exoticsType, importData::getData(), Exotics::getDelta(), Exotics::getPrice(), Exotics::getRho(), Exotics::getTheta(), Exotics::getVega(), LongNatural, Natural, Real, importData::runUserDefinedInterface(), marketData::vols, and marketData::yieldcurve.

Referenced by productsCreationMenu().

**7.67.1.10 OptionStrategy inputOptionStrategy (marketData data)**

User interface to input and store an option strategy and options in it.

Definition at line 119 of file productsCreation.cpp.

References OptionStrategy::addOneBlackScholesObject(), OptionStrategy::getGlobalDelta(), OptionStrategy::getGlobalGamma(), OptionStrategy::getGlobalRho(), OptionStrategy::getGlobalTheta(), OptionStrategy::getGlobalVega(), inputBSOption(), inputSpecificOptionStrategy(), Natural, Real, and OptionStrategy::returnPrice().

Referenced by productsCreationMenu().

**7.67.1.11 void inputPutSpread (marketData data, OptionStrategy & strategy, bool useMarketData)**

add a put spread to the strategy

Definition at line 296 of file productsCreation.cpp.

References OptionStrategy::addLongPutSpread(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

**7.67.1.12 RainbowOption\* inputRainbowOption (marketData data)**

To input a Rainbow Option.

Definition at line 1039 of file productsCreation.cpp.

References choosePricingType(), chooseRainbowType(), RainbowOption::getCorrelRisk(), RainbowOption::getPartialDelta(), RainbowOption::getPartialGamma(), RainbowOption::get-

PartialVega(), RainbowOption::getPrice(), RainbowOption::getRho(), Natural, priceType, rainbowType, Real, Date::setDateToToday(), marketData::vols, and marketData::yieldcurve.

Referenced by productsCreationMenu().

#### **7.67.1.13 void inputRatioCallSpread (marketData data, OptionStrategy & strategy, bool useMarketData)**

add a ratio call spread to the strategy

Definition at line 329 of file productsCreation.cpp.

References OptionStrategy::addLongRatioCallSpread(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

#### **7.67.1.14 void inputSpecificOptionStrategy (marketData data, OptionStrategy & strategy)**

add something else to the strategy than a Call/Put

Definition at line 169 of file productsCreation.cpp.

References inputButterflySpread(), inputCallSpread(), inputPutSpread(), inputRatioCallSpread(), inputStraddle(), inputStrangle(), and Natural.

Referenced by inputOptionStrategy().

#### **7.67.1.15 void inputStraddle (marketData data, OptionStrategy & strategy, bool useMarketData)**

add a Straddle to the strategy

Definition at line 362 of file productsCreation.cpp.

References OptionStrategy::addLongStraddle(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().

#### **7.67.1.16 void inputStrangle (marketData data, OptionStrategy & strategy, bool useMarketData)**

add a strangle to the strategy

Definition at line 390 of file productsCreation.cpp.

References OptionStrategy::addLongStrangle(), Integer, Date::plusDays(), Real, Date::setDateToToday(), yieldCurve::spotRate(), volsurface::volatility(), marketData::vols, and marketData::yieldcurve.

Referenced by inputSpecificOptionStrategy().



**7.67.1.17 VanillaSwap\* inputVanillaSwap (marketData *data*)**

User interface to input an interest rate vanilla swap.

Definition at line 866 of file productsCreation.cpp.

References Following, VanillaSwap::getFairValue1(), VanillaSwap::getFairValue2(), VanillaSwap::getRho(), VanillaSwap::getTheta(), Integer, Natural, Date::plusDays(), Real, VanillaSwap::returnPrice(), Date::setDateToToday(), and marketData::yieldcurve.

Referenced by productsCreationMenu().

**7.67.1.18 string outputCallPut (char *c*)**

Transform the input *c*/C/p/P into "Call" or "Put".

Definition at line 108 of file productsCreation.cpp.

Referenced by inputBSOption().

**7.67.1.19 bool productsCreationMenu (marketData *data*)**

Menu for this category.

Definition at line 4 of file productsCreation.cpp.

References inputBond(), inputBSOption(), inputConvertibleBond(), inputExoticOptionOnSingleAsset(), inputOptionStrategy(), inputRainbowOption(), inputVanillaSwap(), and Natural.

Referenced by main().

## 7.68 rainbowoption.cpp File Reference

```
#include "..\rainbowoption.h"
```

## 7.69 rainbowoption.h File Reference

```
#include "../common/types.h"
#include "../common/matrix.h"
#include "../common/date.h"
#include "../PartB\yieldCurve.h"
#include "../PartE\volsurface.h"
#include <valarray>
#include "../PartA/MonteCarlo1/MersenneTwister.h"
#include "../PartA/MonteCarlo1/GaussianProcess.h"
#include "../PartA/MonteCarlo1/PayOff.h"
#include "../PartA/MonteCarlo1/Random.h"
#include "../PartA/MonteCarlo1/Drift.h"
#include "../PartA/MonteCarlo1/MCEngine.h"
#include "../PartA/BlackScholes/BlackScholes.h"
```

### Classes

- class **RainbowOption**

### Defines

- #define **RO\_DEFAULT\_STRIKE** 100
- #define **RO\_DEFAULT\_VOL** 0.15
- #define **RO\_DEFAULT\_RATE** 0.02
- #define **RO\_DEFAULT\_MATURITY** 1
- #define **RO\_DEFAULT\_NB\_ASSETS** 2
- #define **RO\_DEFAULT\_MULTIPLIER** 1
- #define **RO\_NPATHS** 100000
- #define **RO\_SEED** 100000000
- #define **EPSILON** 0.00000001
- #define **GREEKAPPROX** 0.01

### Enumerations

- enum **priceType** { **MonteCarlo**, **ClosedForm** }
- enum **rainbowType** {  
    **SpreadOptionMax**, **AssetsBasketMax**, **BestOf2AssetsCash**, **WorstOf2AssetsCash**,  
    **BetterOf2Assets**, **WorseOf2Assets**, **Max2AssetsCall**, **Min2AssetsCall**,  
    **Max2AssetsPut**, **Min2AssetsPut** }

## 7.69.1 Define Documentation

### 7.69.1.1 `#define EPSILON 0.00000001`

Definition at line 32 of file rainbowoption.h.

Referenced by `RainbowOption::PriceByClosedForm_BetterOf2()`, `RainbowOption::PriceByClosedForm_WorseOf2()`, `RainbowOption::PriceByMc_BetterOf2Assets()`, and `RainbowOption::PriceByMc_WorseOf2Assets()`.

### 7.69.1.2 `#define GREEKAPPROX 0.01`

Definition at line 33 of file rainbowoption.h.

Referenced by `RainbowOption::getCorrelRisk()`, `RainbowOption::getPartialDelta()`, `RainbowOption::getPartialGamma()`, `RainbowOption::getPartialVega()`, and `RainbowOption::getRho()`.

### 7.69.1.3 `#define RO_DEFAULT_MATURITY 1`

Definition at line 25 of file rainbowoption.h.

Referenced by `RainbowOption::RainbowOption()`.

### 7.69.1.4 `#define RO_DEFAULT_MULTIPLIER 1`

Definition at line 27 of file rainbowoption.h.

### 7.69.1.5 `#define RO_DEFAULT_NB_ASSETS 2`

Definition at line 26 of file rainbowoption.h.

Referenced by `RainbowOption::RainbowOption()`.

### 7.69.1.6 `#define RO_DEFAULT_RATE 0.02`

Definition at line 24 of file rainbowoption.h.

Referenced by `RainbowOption::RainbowOption()`.

### 7.69.1.7 `#define RO_DEFAULT_STRIKE 100`

Definition at line 22 of file rainbowoption.h.

Referenced by `RainbowOption::RainbowOption()`.

### 7.69.1.8 `#define RO_DEFAULT_VOL 0.15`

Definition at line 23 of file rainbowoption.h.

Referenced by `RainbowOption::RainbowOption()`.

### 7.69.1.9 #define RO\_NPATHS 100000

Definition at line 29 of file rainbowoption.h.

### 7.69.1.10 #define RO\_SEED 100000000

Definition at line 30 of file rainbowoption.h.

Referenced by RainbowOption::instanciateMCVariables(), and RainbowOption::RainbowOption().

## 7.69.2 Enumeration Type Documentation

### 7.69.2.1 enum priceType

**Author:**

Yann to simplify, only non path dependant Rainbows, and no quanto

**Enumeration values:**

*MonteCarlo*

*ClosedForm*

Definition at line 41 of file rainbowoption.h.

Referenced by choosePricingType(), and inputRainbowOption().

### 7.69.2.2 enum rainbowType

**Enumeration values:**

*SpreadOptionMax*

*AssetsBasketMax*

*BestOf2AssetsCash*

*WorstOf2AssetsCash*

*BetterOf2Assets*

*WorseOf2Assets*

*Max2AssetsCall*

*Min2AssetsCall*

*Max2AssetsPut*

*Min2AssetsPut*

Definition at line 46 of file rainbowoption.h.

Referenced by chooseRainbowType(), RainbowOption::getRainbowType(), and inputRainbowOption().

## 7.70 RandC.cpp File Reference

```
#include "./randc.h"  
#include "../../common/types.h"  
#include <stdlib.h>  
#include <time.h>
```

### Variables

- `const Natural Maxim = 32767`

#### 7.70.1 Variable Documentation

##### 7.70.1.1 `const Natural Maxim = 32767`

Definition at line 6 of file RandC.cpp.

Referenced by `RandC::GetOneRandomInteger()`, and `RandC::Max()`.

## 7.71 RandC.h File Reference

```
#include "../RandomGenerator.h"
```

### Classes

- class `RandC`

## 7.72 Random.cpp File Reference

```
#include "..\random.h"
```



## 7.73 Random.h File Reference

```
#include "../../common/normals.h"  
#include "../../common/types.h"  
#include "RandomGenerator.h"  
#include <valarray>
```

### Classes

- class **Random**

## 7.74 RandomGenerator.cpp File Reference

```
#include "..\randomgenerator.h"
```

## 7.75 RandomGenerator.h File Reference

```
#include "../../Common/types.h"
```

### Classes

- class **RandomGenerator**

## 7.76 Sobol.cpp File Reference

```
#include ".\sobol.h"  
#include <minmax.h>
```

## 7.77 Sobol.h File Reference

```
#include "../RandomGenerator.h"  
#include "../../common/types.h"  
#include <valarray>
```

### Classes

- class `Sobol`

### Variables

- const `Natural MAXBIT = 30MAXDIM=6`

#### 7.77.1 Variable Documentation

##### 7.77.1.1 `const Natural MAXBIT = 30MAXDIM=6`

Definition at line 10 of file `Sobol.h`.

Referenced by `Sobol::sobseq()`.

## 7.78 StringTokenizer.cpp File Reference

```
#include "StringTokenizer.h"
```

## 7.79 StringTokenizer.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string>
```

### Classes

- class `StringTokenizer`

## 7.80 SwapLeg.cpp File Reference

```
#include "..\swapleg.h"
```



## 7.81 SwapLeg.h File Reference

```
#include "../Common/Date.h"  
#include "../Common/types.h"  
#include <valarray>
```

### Classes

- class `SwapLeg`

## 7.82 testObjects.cpp File Reference

```
#include <iostream>
#include <string>
#include "testObjects.h"
#include "../common/filereader.h"
```

### Functions

- **Natural maintests** (**Natural** argc, char \*\*argv)

#### 7.82.1 Function Documentation

##### 7.82.1.1 **Natural maintests** (**Natural** *argc*, char \*\* *argv*)

Definition at line 7 of file testObjects.cpp.

References `mainasset()`, `mainbinomialtree()`, `mainbond()`, `mainconvertiblebond()`, `maincreditcurve()`, `maindate()`, `mainfilereader()`, `maininterpolator()`, `mainIRVanillaSwap()`, `mainmatrix()`, `mainmontecarlo()`, `mainoption()`, `mainoptionstrategy()`, `mainrainbowoptions()`, `mainvarianceswap()`, `mainvolsurface()`, `mainyieldcurve()`, `Natural`, `FileReader::setdatadir()`, and `ShortNatural`.

Referenced by `main()`.

## 7.83 testObjects.h File Reference

```
#include <string>
#include "../PartB/yieldCurve.h"
#include "../PartE/volsurface.h"
```

### Functions

- bool **maindate** (void)  
*test the date class*
- bool **mainmontecarlo** (void)  
*test the montecarlo option pricer functionality*
- double **mainmc** (double Expiry, double Strike, double Spot, **volsurface** \*Vol, **yieldCurve** \*r, unsigned long nPaths, unsigned long nDates, int PrdName)  
*main routine which runs the montecarlo pricer*
- bool **mainoption** (void)  
*test the BS option pricing functionality*
- bool **mainoptionstrategy** (void)  
*test the option strategy functionality*
- bool **mainmatrix** (void)  
*test the matrix functions*
- bool **maininterpolator** (void)  
*test the 2d/3d interpolator functionality*
- bool **mainyieldcurve** (void)  
*test the yield curve functionality*
- bool **mainasset** (void)  
*test the asset functionality*
- bool **mainfilereader** (void)  
*test the file reader functionality*
- bool **fr\_basic** (void)  
*file reader basic test*
- bool **mainvolsurface** (void)  
*test the volsurface functionality*
- bool **maincreditcurve** (void)  
*test of the credit curve object*
- bool **mainrainbowoptions** (void)

*test of the credit curve object*

- bool **mainbond** (void)  
*test the bond functionality*
- bool **mainvarianceswap** (void)  
*test of the variance swaps object*
- bool **mainbinomialtree** (void)  
*test of the binomial tree object*
- bool **mainconvertiblebond** (void)  
*test of the convertible bond object*
- bool **mainIRVanillaSwap** (void)  
*Test of the Vanilla Swap - Yann.*

## 7.83.1 Function Documentation

### 7.83.1.1 bool fr\_basic (void)

file reader basic test

**Returns:**

true if pass, otherwise fail

Definition at line 53 of file mainfilereader.cpp.

References FileReader::fileexists(), and FileReader::getdatadirasString().

Referenced by mainfilereader().

### 7.83.1.2 bool mainasset (void)

test the asset functionality

**Author:**

Yann

Definition at line 14 of file maintestasset.cpp.

References a, FileReader::buildYieldPointArray(), EUR, asset::forwardPrice(), FileReader::getdatadirasString(), Natural, Date::plusDays(), Date::plusMonths(), Date::plusYears(), Real, Date::setDateToToday(), and Date::toString().

Referenced by maintests().

### 7.83.1.3 bool mainbinomialtree (void)

test of the binomial tree object

**Returns:**

true if pass, otherwise fail

Definition at line 7 of file mainbinomialtree.cpp.

References Call, binomialTree::getPrice(), BlackScholes::getPrice(), binomialTree::getStockProcess(), LongNatural, mainmc(), Natural, r, Real, realsEqual(), and binomialTree::runEngineCall().

Referenced by maintests().

**7.83.1.4 bool mainbond (void)**

test the bond functionality

**Returns:**

true if pass, otherwise fail

Definition at line 12 of file mainbond.cpp.

References ACT\_365, bond::convexity(), DayCountConvention, bond::duration(), bond::fairvalue(), Frequency, May, November, Real, Semiannual, and bond::yieldToMaturity().

Referenced by maintests().

**7.83.1.5 bool mainconvertiblebond (void)**

test of the convertible bond object

**Returns:**

true if pass, otherwise fail

Definition at line 5 of file mainconvertiblebond.cpp.

References ACT\_365, Days, convertiblebond::fairvalue(), Months, Natural, convertiblebond::parityDelta(), convertiblebond::parityGamma(), Date::plus(), Real, realsEqual(), convertiblebond::rho(), Date::setDateToToday(), and asset::setPrice().

Referenced by maintests().

**7.83.1.6 bool maincreditcurve (void)**

test of the credit curve object

**Returns:**

true if pass, otherwise fail

Definition at line 9 of file maincreditcurves.cpp.

References FileReader::buildCreditSpreadPointArray(), FileReader::buildYieldPointArray(), creditCurve::creditSpread(), creditCurve::cumulativeDefaultProbability(), creditCurve::defaultProbability(), FileReader::getdatadirasString(), creditCurve::hazardRate(), Natural, Real, creditCurve::riskyDiscountFactor(), and creditCurve::survivalProbability().

Referenced by maintests().

**7.83.1.7 bool maindate (void)**

test the date class

**Returns:**

true if pass, otherwise fail

Definition at line 7 of file maindate.cpp.

References Date::lastDayOfMonth(), Date::setDateToToday(), and Date::toString().

Referenced by maintests().

**7.83.1.8 bool mainfilereader (void)**

test the file reader functionality

**Returns:**

true if pass, otherwise fail

Definition at line 45 of file mainfilereader.cpp.

References fr\_basic().

Referenced by maintests().

**7.83.1.9 bool maininterpolator (void)**

test the 2d/3d interpolator functionality

**Returns:**

true if pass, otherwise fail

Definition at line 5 of file maininterpolator.cpp.

References Integer, interpolator::interpolate(), and Real.

Referenced by maintests().

**7.83.1.10 bool mainIRVanillaSwap (void)**

Test of the Vanilla Swap - Yann.

**Author:**

Yann

Definition at line 9 of file mainIRVanillaSwap.cpp.

References FileReader::buildYieldPointArray(), FileReader::getdatadirasString(), CashFlow::getFairValue(), Date::plusMonths(), Date::plusYears(), Real, VanillaSwap::returnPrice(), and Date::setDateToToday().

Referenced by maintests().

**7.83.1.11 bool mainmatrix (void)**

test the matrix functions

**Author:**

Yann

Definition at line 16 of file mainmatrix.cpp.

References Matrix::CholeskyDecomposition(), Matrix::GetTransposed(), Natural, Matrix::SetValue(), transform1DvalarrayToColumnMatrix(), transform2DvalarrayToMatrix(), transformColumnMatrixTo1Dvalarray(), and transformMatrixTo2Dvalarray().

Referenced by maintests().

**7.83.1.12 double mainmc (double *Expiry*, double *Strike*, double *Spot*, volsurface \* *Vol*, yieldCurve \* *r*, unsigned long *nPaths*, unsigned long *nDates*, int *PrdName*)**

main routine which runs the montecarlo pricer

**7.83.1.13 bool mainmontecarlo (void)**

test the montecarlo option pricer functionality

**Returns:**

true if pass, otherwise fail

Definition at line 19 of file mainmontecarlo.cpp.

References LongNatural, mainmc(), and Real.

Referenced by maintests().

**7.83.1.14 bool mainoption (void)**

test the BS option pricing functionality

**Returns:**

true if pass, otherwise fail

Definition at line 5 of file mainoptionstrategy.cpp.

References Call, BlackScholes::getDelta(), BlackScholes::getGamma(), BlackScholes::getPrice(), BlackScholes::getRho(), BlackScholes::getTheta(), BlackScholes::getVega(), BlackScholes::getVolatility(), and LongNatural.

Referenced by maintests().

**7.83.1.15 bool mainoptionstrategy (void)**

test the option strategy functionality

**Returns:**

true if pass, otherwise fail

Definition at line 26 of file mainoptionstrategy.cpp.

References `OptionStrategy::addLongButterflySpread()`, `OptionStrategy::getGlobalDelta()`, and `OptionStrategy::returnPrice()`.

Referenced by `maintests()`.

**7.83.1.16 bool mainrainbowoptions (void)**

test of the credit curve object

**Author:**

Yann

\* Types (MonteCarlo, ClosedForm )

Definition at line 13 of file mainrainbowoptions.cpp.

References `AssetsBasketMax`, `BestOf2AssetsCash`, `BetterOf2Assets`, `ClosedForm`, `RainbowOption::getCorrelRisk()`, `RainbowOption::getPartialDelta()`, `RainbowOption::getPartialGamma()`, `RainbowOption::getPartialVega()`, `RainbowOption::getPrice()`, `RainbowOption::getRho()`, `Max2AssetsCall`, `Max2AssetsPut`, `Min2AssetsCall`, `Min2AssetsPut`, `MonteCarlo`, `RealDate::setDateToToday()`, `RainbowOption::setRainbowType()`, `SpreadOptionMax`, `WorseOf2Assets`, and `WorstOf2AssetsCash`.

Referenced by `maintests()`.

**7.83.1.17 bool mainvarianceswap (void)**

test of the variance swaps object

**Returns:**

true if pass, otherwise fail

Definition at line 9 of file mainvarianceswap.cpp.

References `OptionStrategy::addLongButterflySpread()`, `OptionStrategy::addOneBlackScholesObject()`, `Call`, `VarianceSwap::getPrice()`, `Natural`, `Put`, and `Real`.

Referenced by `maintests()`.

**7.83.1.18 bool mainvolsurface (void)**

test the volsurface functionality

**Returns:**

true if pass, otherwise fail

Definition at line 12 of file mainvolsurface.cpp.

References `FileReader::buildVolSurfaceParams()`, `December`, `July`, `March`, `Real`, `September`, `volsurface::setvolsurface()`, `Date::toString()`, and `volsurface::volatility()`.

Referenced by `maintests()`.



**7.83.1.19 bool mainyieldcurve (void)**

test the yield curve functionality

**Author:**

Yann

Definition at line 13 of file mainyieldcurves.cpp.

References `FileReader::buildYieldPointArray()`, `Continuous`, `Discrete`, `FileReader::getdatadirasString()`, `yieldCurve::getMaturitiesInTheZCBCurve()`, `Month`, `Natural`, `Date::plusMonths()`, `Real`, `yieldCurve::rotateZCBRateCurve()`, `Date::setDateToToday()`, `yieldCurve::shiftZCBRateCurve()`, and `yieldCurve::spotRate()`.

Referenced by `maintests()`.

## 7.84 types.h File Reference

### Classes

- struct `cachedval`

### Defines

- `#define TN_INTEGER int`
- `#define TN_LONG_INTEGER long`
- `#define TN_REAL double`
- `#define TN_INFINITY 9999`

### Typedefs

- `typedef short TN_INTEGER ShortInteger`
- `typedef unsigned short TN_INTEGER ShortNatural`
- `typedef TN_INTEGER Integer`
- `typedef unsigned TN_INTEGER Natural`
- `typedef TN_LONG_INTEGER LongInteger`
- `typedef unsigned TN_LONG_INTEGER LongNatural`
- `typedef TN_REAL Real`
- `typedef long long VeryLongInteger`
- `typedef unsigned long long VeryLongNatural`

### Enumerations

- enum `Currency` { `USD = 1`, `EUR = 2`, `CAD = 3` }

#### 7.84.1 Define Documentation

##### 7.84.1.1 `#define TN_INFINITY 9999`

Definition at line 9 of file `types.h`.

Referenced by `mergeunique()`.

##### 7.84.1.2 `#define TN_INTEGER int`

Definition at line 6 of file `types.h`.

##### 7.84.1.3 `#define TN_LONG_INTEGER long`

Definition at line 7 of file `types.h`.

##### 7.84.1.4 `#define TN_REAL double`

Definition at line 8 of file `types.h`.

Referenced by `Date::dayCount()`, and `bond::fairvalue()`.

## 7.84.2 Typedef Documentation

### 7.84.2.1 typedef TN\_INTEGER Integer

Definition at line 14 of file types.h.

Referenced by `Date::advance()`, `volsurface::forwardvolsurface()`, `bond::getCashflow()`, `interpolator::getInterpolation()`, `interpolator::getPlace()`, `interpolator::getPlaceOnXi()`, `VarianceSwap::getPrice()`, `Portfolio::getPrice()`, `Exotics::getTheta()`, `inputBond()`, `inputBSOption()`, `inputButterflySpread()`, `inputCallSpread()`, `inputConvertibleBond()`, `inputPutSpread()`, `inputRatioCallSpread()`, `inputStraddle()`, `inputStrangle()`, `inputVanillaSwap()`, `interpolator::interpolate()`, `interpolatormain()`, `maininterpolator()`, `mainmc()`, `Date::month()`, `Date::monthLength()`, `Date::monthOffset()`, `Date::plus()`, `Date::plusDays()`, `Date::plusMonths()`, `Date::plusWeeks()`, `Date::plusYears()`, `Portfolio::returnSensibilityToRate()`, `Portfolio::returnSensibilityToTime()`, `Portfolio::returnSensibilityToVol()`, `binomialTree::runEngineCall()`, `binomialTree::runEngineConvertibleBond()`, `volsurface::shiftedvolsurface()`, `Sobol::sobseq()`, `SwapLeg::SwapLeg()`, and `Date::weekday()`.

### 7.84.2.2 typedef TN\_LONG\_INTEGER LongInteger

Definition at line 16 of file types.h.

Referenced by `Date::applyConvention()`, `Date::Date()`, `ParkMiller::GetOneRandomInteger()`, `Drift::GetTimeBtwDates()`, `Date::maximumSerialNumber()`, `Date::minimumSerialNumber()`, `Date::operator++()`, `Date::operator++()`, `Date::operator+=()`, `Date::operator-()`, `Date::operator-()`, `Date::operator-=()`, `SwapLeg::returnSize()`, `Date::serialNumber()`, `Sobol::sobseq()`, `SwapLeg::SwapLeg()`, and `Date::yearOffset()`.

### 7.84.2.3 typedef unsigned TN\_LONG\_INTEGER LongNatural

Definition at line 17 of file types.h.

Referenced by `PayOff::AsianCall()`, `PayOff::AsianPut()`, `Average()`, `PayOff::BarrierCall()`, `PayOff::BarrierPut()`, `GaussianProcess::BuildPath()`, `Drift::Drift()`, `Exotics::Exotics()`, `GaussianProcess::GaussianProcess()`, `Random::GetDimensionality()`, `Drift::GetDriftattimei()`, `Random::GetGaussians()`, `Sobol::GetOneRandomInteger()`, `RandC::GetOneRandomInteger()`, `ParkMiller::GetOneRandomInteger()`, `MersenneTwister::GetOneRandomInteger()`, `RainbowOption::getPrice()`, `GaussianProcess::GetStepIncrements()`, `Drift::GetTimeBtwDates()`, `Random::GetUniforms()`, `inputExoticOptionOnSingleAsset()`, `mainbinomialtree()`, `mainmc()`, `mainmontecarlo()`, `mainoption()`, `Maximize()`, `MCEngine::MCEngine()`, `MersenneTwister::MersenneTwister()`, `Sobol::Min()`, `RandC::Min()`, `ParkMiller::Min()`, `MersenneTwister::Min()`, `ParkMiller::ParkMiller()`, `RainbowOption::PriceByMc_2AssetsBasketMax()`, `RainbowOption::PriceByMc_2SpreadOptionMax()`, `RainbowOption::PriceByMc_BestOf2AssetsCash()`, `RainbowOption::PriceByMc_BetterOf2Assets()`, `RainbowOption::PriceByMc_Max2AssetsCall()`, `RainbowOption::PriceByMc_Max2AssetsPut()`, `RainbowOption::PriceByMc_Min2AssetsCall()`, `RainbowOption::PriceByMc_Min2AssetsPut()`, `RainbowOption::PriceByMc_WorseOf2Assets()`, `RainbowOption::PriceByMc_WorstOf2AssetsCash()`, `RainbowOption::RainbowOption()`, `RandC::RandC()`, `Random::Random()`, `RandomGenerator::RandomGenerator()`, `Random::ResetDimensionality()`, `PayOff::RevLookbackCall()`, `PayOff::RevLookbackPut()`, `MCEngine::RunEngineAsianCall()`, `MCEngine::RunEngineAsianPut()`, `MCEngine::RunEngineBarrierCall()`, `MCEngine::RunEngineBarrierPut()`, `MCEngine::RunEngineCall()`, `MCEngine::RunEngineCappedCliquet()`, `MCEngine::RunEngineFlooredCliquet()`, `MCEngine::RunEnginePut()`, `MCEngine::RunEngineRainbow2AssetsBasketMax()`, `MCEngine::RunEngineRainbow2SpreadOptionMax()`, `MCEngine::RunEngineRainbow`

BestOf2AssetsCash(), MCEngine::RunEngineRainbowMax2AssetsCall(), MCEngine::RunEngineRainbowMax2AssetsPut(), MCEngine::RunEngineRainbowMin2AssetsCall(), MCEngine::RunEngineRainbowMin2AssetsPut(), MCEngine::RunEngineRainbowWorstOf2AssetsCash(), MCEngine::RunEngineRevLookbackCall(), MCEngine::RunEngineRevLookbackPut(), Sobol::SetSeed(), RandomGenerator::SetSeed(), Random::SetSeed(), RandC::SetSeed(), ParkMiller::SetSeed(), MersenneTwister::SetSeed(), Random::Skip(), Sobol::Sobol(), and Sobol::sobseq().

#### 7.84.2.4 typedef unsigned TN\_INTEGER Natural

Definition at line 15 of file types.h.

Referenced by yieldCurve::assignFlatRate(), creditCurve::assignFlatSpread(), yieldCurve::assignZCBRateAtIndex(), binomialTree::binomialTree(), FileReader::buildCreditSpreadPointArray(), FileReader::buildVolSurfaceParams(), FileReader::buildYieldPointArray(), CashFlow::CashFlow(), choiceToType(), choosePricingType(), chooseRainbowType(), creditCurve::combineUnderlyingAndSpreads(), yieldCurve::computeZCBRatesBootstrap(), binomialTree::constructStockProcess(), convertiblebond::convertiblebond(), bond::convexity(), creditCurve::createSpreadCurve(), CumulativeBivariateNormal(), creditCurve::defaultProbability(), importData::displayFileFormatsMenu(), Drift::Drift(), bond::duration(), bond::fairvalue(), asset::forwardPrice(), yieldCurve::forwardZCBCurve(), GaussianProcess::GaussianProcess(), bond::getCashflow(), binomialTree::getClaimProcess(), RainbowOption::getCorrelRisk(), RainbowOption::getDelta(), CashFlow::getFairValue(), RainbowOption::getGamma(), yieldCurve::getMaturitiesInTheMarketCurve(), yieldCurve::getMaturitiesInTheZCBCurve(), RainbowOption::getPartialDelta(), RainbowOption::getPartialGamma(), RainbowOption::getPartialVega(), yieldCurve::getPointAtMaturity(), VarianceSwap::getPrice(), binomialTree::getRate(), yieldCurve::getSequentSwapRates(), convertiblebond::getSteps(), binomialTree::getSteps(), binomialTree::getStockProcess(), yieldCurve::getSwapRates(), Drift::GetvDates(), Drift::GetvDrift(), RainbowOption::getVega(), creditCurve::indexOfCurrentSpread(), creditCurve::indexOfPreviousSpread(), inputBond(), inputBSOption(), inputButterflySpread(), inputConvertibleBond(), inputExoticOptionOnSingleAsset(), inputOptionStrategy(), inputRainbowOption(), inputSpecificOptionStrategy(), inputVanillaSwap(), RainbowOption::instanciateMCVariables(), main(), mainasset(), mainbinomialtree(), mainconvertiblebond(), maincreditcurve(), mainmatrix(), maintests(), mainvarianceswap(), mainyieldcurve(), mergeunique(), operator<<(), yieldCurve::operator==(), CSVParser::operator>>(), productsCreationMenu(), riskybond::quotedPrice(), bond::quotedPrice(), RainbowOption::reassignVolsAtThemoney(), RainbowOption::reassignVolsAtThestrike(), creditCurve::resampleSpread(), OptionStrategy::returnNbOptions(), OptionStrategy::returnOption(), OptionStrategy::returnOptionQuantity(), yieldCurve::rotateZCBRateCurve(), binomialTree::runEngineCall(), binomialTree::runEngineConvertibleBond(), MCEngine::RunEngineGeneral(), MCEngine::RunEngineRainbow2AssetsBasketMax(), MCEngine::RunEngineRainbow2SpreadOptionMax(), MCEngine::RunEngineRainbowBestOf2AssetsCash(), MCEngine::RunEngineRainbowMax2AssetsCall(), MCEngine::RunEngineRainbowMax2AssetsPut(), MCEngine::RunEngineRainbowMin2AssetsCall(), MCEngine::RunEngineRainbowMin2AssetsPut(), MCEngine::RunEngineRainbowWorstOf2AssetsCash(), importData::runInterface(), yieldCurve::SequentDiscountFactorsByInvertSwapMatrix(), binomialTree::setClaimVariables(), yieldCurve::shiftZCBRateCurve(), Sobol::sobseq(), yieldCurve::sortCashSwap(), yieldCurve::sortMarketRatesByMaturity(), yieldCurve::spotRate(), creditCurve::survivalProbability(), creditCurve::swapFees(), SwapLeg::SwapLeg(), transform1DvalarrayToColumnMatrix(), transform2DvalarrayToMatrix(), transformColumnMatrixTo1Dvalarray(), transformMatrixTo2Dvalarray(), valarrayRealToString(), and bond::yieldToMaturity().

## 7.84.2.5 typedef TN\_REAL Real

Definition at line 18 of file types.h.

Referenced by `absolute()`, `Portfolio::addAsset()`, `Portfolio::addBond()`, `Portfolio::addExoticOption()`, `OptionStrategy::addLongButterflySpread()`, `OptionStrategy::addLongCallSpread()`, `OptionStrategy::addLongPutSpread()`, `OptionStrategy::addLongRatioCallSpread()`, `OptionStrategy::addLongStraddle()`, `OptionStrategy::addLongStrangle()`, `OptionStrategy::addOneBlackScholesObject()`, `OptionStrategy::addOneOptionToStrategy()`, `Portfolio::addRainbowOption()`, `Portfolio::addVanillaSwap()`, `Portfolio::addVarianceSwap()`, `convertiblebond::adjustedConversionRatio()`, `PayOff::AsianCall()`, `PayOff::AsianPut()`, `asset::asset()`, `yieldCurve::assignFlatRate()`, `creditCurve::assignFlatSpread()`, `yieldCurve::assignZCBrateAtIndex()`, `Average()`, `PayOff::BarrierCall()`, `PayOff::BarrierPut()`, `binomialTree::binomialTree()`, `BlackScholes::BlackScholes()`, `bond::bond()`, `GaussianProcess::BuildPath()`, `GaussianProcess::BuildTerminalPoint()`, `PayOff::Call()`, `PayOff::CappedCliquet()`, `CashFlow::CashFlow()`, `OptionStrategy::changeMaturity()`, `BlackScholes::changeMaturity()`, `OptionStrategy::changeRate()`, `BlackScholes::changeRate()`, `OptionStrategy::changeSpot()`, `BlackScholes::changeSpot()`, `OptionStrategy::changeStrike()`, `BlackScholes::changeStrike()`, `OptionStrategy::changeVol()`, `BlackScholes::changeVol()`, `Matrix::CholeskyDecomposition()`, `RainbowOption::compute_A()`, `RainbowOption::compute_B()`, `RainbowOption::compute_C()`, `RainbowOption::compute_d1()`, `RainbowOption::compute_d2()`, `RainbowOption::compute_d3()`, `RainbowOption::compute_d4()`, `RainbowOption::compute_rho1()`, `RainbowOption::compute_rho2()`, `RainbowOption::compute_sigmaA()`, `yieldCurve::computeZCBRatesBootstrap()`, `PayOff::Convertible()`, `convertiblebond::convertiblebond()`, `bond::convexity()`, `creditCurve::creditCurve()`, `creditCurve::creditSpread()`, `CreditSpreadPoint::CreditSpreadPoint()`, `CumulativeBivariateNormal()`, `creditCurve::cumulativeDefaultProbability()`, `CumulativeNormal()`, `Date::dayCount()`, `creditCurve::defaultProbability()`, `convertiblebond::delta()`, `yieldCurve::discountFactor()`, `creditCurve::discountFactor()`, `Drift::Drift()`, `bond::duration()`, `Exotics::Exotics()`, `convertiblebond::fairvalue()`, `bond::fairvalue()`, `PayOff::FlooredCliquet()`, `flowSchedule::flowSchedule()`, `yieldCurve::forwardDiscountFactor()`, `asset::forwardPrice()`, `yieldCurve::forwardRate()`, `creditCurve::forwardRate()`, `volsurface::forwardVolatility()`, `yieldCurve::forwardZCBCurve()`, `convertiblebond::gamma()`, `GaussianProcess::GaussianProcess()`, `flowSchedule::getAmount()`, `bond::getCashflow()`, `RainbowOption::getCorrelRisk()`, `RainbowOption::getDelta()`, `Exotics::getDelta()`, `BlackScholes::getDelta()`, `asset::getDelta()`, `Drift::GetDriftatimei()`, `bond::getFaceAmount()`, `CashFlow::getFairValue()`, `VanillaSwap::getFairValue1()`, `VanillaSwap::getFairValue2()`, `RainbowOption::getGamma()`, `BlackScholes::getGamma()`, `Random::GetGaussian()`, `Random::GetGaussians()`, `OptionStrategy::getGlobalDelta()`, `OptionStrategy::getGlobalGamma()`, `OptionStrategy::getGlobalRho()`, `OptionStrategy::getGlobalTheta()`, `OptionStrategy::getGlobalVega()`, `interpolator::getInterpolation()`, `yieldPoint::getMaturity()`, `CreditSpreadPoint::getMaturity()`, `BlackScholes::getMaturity()`, `binomialTree::getMaturity()`, `bond::getMaturityInYears()`, `RainbowOption::getPartialDelta()`, `RainbowOption::getPartialGamma()`, `RainbowOption::getPartialVega()`, `interpolator::getPlace()`, `interpolator::getPlaceOnXi()`, `yieldCurve::getPointAtMaturity()`, `VarianceSwap::getPrice()`, `RainbowOption::getPrice()`, `Portfolio::getPrice()`, `Exotics::getPrice()`, `BlackScholes::getPrice()`, `asset::getPrice()`, `yieldPoint::getRate()`, `CreditSpreadPoint::getRate()`, `BlackScholes::getRate()`, `binomialTree::getRate()`, `asset::getRate()`, `creditCurve::getRecoveryRate()`, `VarianceSwap::getRho()`, `VanillaSwap::getRho()`, `RainbowOption::getRho()`, `Exotics::getRho()`, `BlackScholes::getRho()`, `asset::getRho()`, `yieldCurve::getSequentSwapRates()`, `binomialTree::getSigma()`, `binomialTree::getSo()`, `BlackScholes::getSpot()`, `BlackScholes::getStrike()`, `VarianceSwap::getTheta()`, `VanillaSwap::getTheta()`, `RainbowOption::getTheta()`, `Exotics::getTheta()`, `BlackScholes::getTheta()`, `Sobol::getUniform()`, `RandomGenerator::getUniform()`, `Random::GetUniform()`, `RandC::getUniform()`, `ParkMiller::getUniform()`, `MersenneTwister::getUniform()`, `VarianceSwap::getVega()`, `RainbowOption::getVega()`, `Exotics::getVega()`, `BlackScholes::getVega()`, `BlackScholes::getVolatility()`, `asset::GetVolatility()`, `creditCurve::hazardRate()`, `importData::importVolSurface()`, `creditCurve::indexOfCurrentSpread()`,

creditCurve::indexOfPreviousSpread(), inputBond(), inputBSOption(), inputButterflySpread(), inputCallSpread(), inputConvertibleBond(), inputExoticOptionOnSingleAsset(), inputOptionStrategy(), inputPutSpread(), inputRainbowOption(), inputRatioCallSpread(), inputStraddle(), inputStrangle(), inputVanillaSwap(), convertiblebond::interestRateDelta(), interpolator::interpolate(), interpolator::main(), InverseCumulativeNormal(), volsurface::invertBSformula(), mainasset(), mainbinomialtree(), mainbond(), mainconvertiblebond(), maincreditcurve(), maininterpolator(), mainIRVanillaSwap(), mainmc(), mainmontecarlo(), mainrainbowoptions(), mainvarianceswap(), mainvolsurface(), mainyieldcurve(), Maximize(), MCEngine::MCEngine(), MCEngine::MCResult(), mergeunique(), NormalDensity(), PayOff::operator()(), yieldCurve::operator==(), convertiblebond::parity(), convertiblebond::parityDelta(), convertiblebond::parityGamma(), PayOff::PayOff(), asset::Price(), RainbowOption::PriceByClosedForm\_BestOf2\_plusCash(), RainbowOption::PriceByClosedForm\_BetterOf2(), RainbowOption::PriceByClosedForm\_MaxOf2\_call(), RainbowOption::PriceByClosedForm\_MaxOf2\_put(), RainbowOption::PriceByClosedForm\_MinOf2\_call(), RainbowOption::PriceByClosedForm\_MinOf2\_put(), RainbowOption::PriceByClosedForm\_WorseOf2(), RainbowOption::PriceByMc\_2AssetsBasketMax(), RainbowOption::PriceByMc\_2SpreadOptionMax(), RainbowOption::PriceByMc\_BestOf2AssetsCash(), RainbowOption::PriceByMc\_BetterOf2Assets(), RainbowOption::PriceByMc\_Max2AssetsCall(), RainbowOption::PriceByMc\_Max2AssetsPut(), RainbowOption::PriceByMc\_Min2AssetsCall(), RainbowOption::PriceByMc\_Min2AssetsPut(), RainbowOption::PriceByMc\_WorseOf2Assets(), RainbowOption::PriceByMc\_WorstOf2AssetsCash(), PayOff::Put(), riskybond::quotedPrice(), bond::quotedPrice(), PayOff::Rainbow2AssetsBasketMax(), PayOff::Rainbow2SpreadOptionMax(), PayOff::RainbowBestOf2AssetsCash(), PayOff::RainbowMax2AssetsCall(), PayOff::RainbowMax2AssetsPut(), PayOff::RainbowMin2AssetsCall(), PayOff::RainbowMin2AssetsPut(), RainbowOption::RainbowOption(), PayOff::RainbowWorstOf2AssetsCash(), realsEqual(), OptionStrategy::recalcPrice(), OptionStrategy::returnOptionQuantity(), VanillaSwap::returnPrice(), OptionStrategy::returnPrice(), Portfolio::returnSensibilityToRate(), Portfolio::returnSensibilityToTime(), Portfolio::returnSensibilityToVol(), PayOff::RevLookbackCall(), PayOff::RevLookbackPut(), convertiblebond::rho(), riskybond::rho(), treasurybond::rho(), riskybond::riskybond(), creditCurve::riskyDiscountFactor(), yieldCurve::rotateZCBRateCurve(), binomialTree::runEngineConvertibleBond(), MCEngine::RunEngineRainbow2AssetsBasketMax(), MCEngine::RunEngineRainbow2SpreadOptionMax(), MCEngine::RunEngineRainbowBestOf2AssetsCash(), MCEngine::RunEngineRainbowMax2AssetsCall(), MCEngine::RunEngineRainbowMax2AssetsPut(), MCEngine::RunEngineRainbowMin2AssetsCall(), MCEngine::RunEngineRainbowMin2AssetsPut(), MCEngine::RunEngineRainbowWorstOf2AssetsCash(), importData::runUserDefinedInterface(), flowSchedule::setAmount(), binomialTree::setClaimVariables(), asset::setDivAsRate(), yieldPoint::setMaturity(), CreditSpreadPoint::setMaturity(), asset::setPrice(), yieldPoint::setRate(), CreditSpreadPoint::setRate(), PayOff::SetStrike(), asset::setVolatility(), volsurface::setvolsurface(), riskybond::shiftedbond(), treasurybond::shiftedbond(), convertiblebond::shiftedcbond(), volsurface::shiftedvolsurface(), volsurface::shiftedYCVolsurface(), yieldCurve::shiftZCBRateCurve(), sign(), Sobol::sobseq(), yieldCurve::sortMarketRatesByMaturity(), yieldCurve::spotRate(), creditCurve::spotRate(), SubFunctionForBivariateNormal(), creditCurve::survivalProbability(), creditCurve::swapFees(), SwapLeg::SwapLeg(), creditCurve::timeOfCurrentSpread(), creditCurve::timeOfPreviousSpread(), treasurybond::treasurybond(), volsurface::variance(), VarianceSwap::VarianceSwap(), volsurface::volatility(), volsurface::volsurface(), yieldCurve::yieldCurve(), yieldPoint::yieldPoint(), and bond::yieldToMaturity().

### 7.84.2.6 typedef short TN\_INTEGER ShortInteger

Definition at line 12 of file types.h.

Referenced by Date::nthWeekday().

### 7.84.2.7 typedef unsigned short TN\_INTEGER ShortNatural

Definition at line 13 of file types.h.

Referenced by Date::Date(), maintests(), and VanillaSwap::VanillaSwap().

### 7.84.2.8 typedef long long VeryLongInteger

Definition at line 19 of file types.h.

### 7.84.2.9 typedef unsigned long long VeryLongNatural

Definition at line 20 of file types.h.

Referenced by Sobol::Max(), RandC::Max(), ParkMiller::Max(), and MersenneTwister::Max().

## 7.84.3 Enumeration Type Documentation

### 7.84.3.1 enum Currency

Enumeration values:

*USD*

*EUR*

*CAD*

Definition at line 22 of file types.h.

Referenced by Portfolio::getCurrency(), creditCurve::getCurrency(), and asset::GetCurrencyFormat().

## 7.85 UsDate.cpp File Reference

```
#include "..\usdate.h"
```



## 7.86 UsDate.h File Reference

```
#include "date.h"
```

### Classes

- class UsDate

## 7.87 utils.cpp File Reference

```
#include "utils.h"
```

### Functions

- `valarray< Real > mergeunique (valarray< Real > a1, valarray< Real > a2)`  
*helper function to merge two valarrays of reals sorted in ascending order and remove duplicates.*
- short int `sign (Real x)`  
*sign of a Real*
- void `ErrorMsg (string str, bool mustexit)`
- `Matrix transform1DvalarrayToColumnMatrix (valarray< Real > array)`  
*Transforms a 1D array into a column vector - easier to handle our structures and do operations on them.*
- `Matrix transform2DvalarrayToMatrix (valarray< valarray< Real > > array)`  
*Transforms a 2D array into a matrix.*
- `valarray< Real > transformColumnMatrixTo1Dvalarray (Matrix M)`  
*Transforms a column matrix to a 1D array.*
- `valarray< valarray< Real > > transformMatrixTo2Dvalarray (Matrix M)`  
*Transforms a normal matrix to a 2D array.*
- int `getint (istream &istr)`
- bool `realsEqual (Real realOne, Real realTwo, Real precision)`
- string `valarrayRealToString (const valarray< Real > &theArray)`

### 7.87.1 Function Documentation

#### 7.87.1.1 void ErrorMsg (string *str*, bool *mustexit*)

Definition at line 74 of file `utils.cpp`.

Referenced by `Matrix::CMAC()`, `Matrix::CMAR()`, `Matrix::Determinant()`, `Matrix::DivideColumn()`, `Matrix::DivideRow()`, `Matrix::GetInverse()`, `Matrix::operator*()`, `Matrix::operator()()`, `Matrix::operator+()`, `Matrix::operator-()`, `Matrix::operator/()`, and `Matrix::operator[]()`.

#### 7.87.1.2 int getint (istream & *istr*)

Definition at line 143 of file `utils.cpp`.

Referenced by `Matrix::Input()`.

#### 7.87.1.3 valarray<Real> mergeunique (valarray< Real > *a1*, valarray< Real > *a2*)

helper function to merge two valarrays of reals sorted in ascending order and remove duplicates.

*a1* - first array to merge *a2* - second array to merge

**Returns:**

the merged array

Definition at line 12 of file utils.cpp.

References Natural, Real, and TN\_INFINITY.

Referenced by creditCurve::combineUnderlyingAndSpreads(), and yieldCurve::operator==().

**7.87.1.4 bool realsEqual (Real *realOne*, Real *realTwo*, Real *precision*)**

Definition at line 157 of file utils.cpp.

References Real.

Referenced by mainbinomialtree(), and mainconvertiblebond().

**7.87.1.5 short int sign (Real *x*)**

sign of a Real

**Author:**

Yann

Definition at line 62 of file utils.cpp.

References Real.

Referenced by CumulativeBivariateNormal().

**7.87.1.6 Matrix transform1DvalarrayToColumnMatrix (valarray< Real > *array*)**

Transforms a 1D array into a column vector - easier to handle our structures and do operations on them.

Definition at line 96 of file utils.cpp.

References Natural, and Matrix::SetValue().

Referenced by mainmatrix().

**7.87.1.7 Matrix transform2DvalarrayToMatrix (valarray< valarray< Real > > *array*)**

Transforms a 2D array into a matrix.

Definition at line 107 of file utils.cpp.

References Natural, and Matrix::SetValue().

Referenced by mainmatrix().

**7.87.1.8 valarray<Real> transformColumnMatrixTo1Dvalarray (Matrix *M*)**

Transforms a column matrix to a 1D array.

Definition at line 119 of file utils.cpp.

References M, and Natural.

Referenced by `mainmatrix()`.

#### **7.87.1.9** `valarray<valarray<Real> > transformMatrixTo2Dvalarray (Matrix M)`

Transforms a normal matrix to a 2D array.

Definition at line 128 of file `utils.cpp`.

References M, and Natural.

Referenced by `mainmatrix()`.

#### **7.87.1.10** `string valarrayRealToString (const valarray< Real > & theArray)`

Definition at line 162 of file `utils.cpp`.

References Natural.

Referenced by `operator<<()`.

## 7.88 utils.h File Reference

```
#include "types.h"
#include "matrix.h"
#include <valarray>
#include <iostream>
#include <string>
```

### Defines

- `#define DEFAULT_PRECISION 0.00001`

### Functions

- short int **sign** (**Real** x)  
*sign of a Real*
- **Matrix transform1DvalarrayToColumnMatrix** (valarray< **Real** > array)  
*Transforms a 1D array into a column vector - easier to handle our structures and do operations on them.*
- **Matrix transform2DvalarrayToMatrix** (valarray< valarray< **Real** > > array)  
*Transforms a 2D array into a matrix.*
- valarray< **Real** > **transformColumnMatrixTo1Dvalarray** (**Matrix** M)  
*Transforms a column matrix to a 1D array.*
- valarray< valarray< **Real** > > **transformMatrixTo2Dvalarray** (**Matrix** M)  
*Transforms a normal matrix to a 2D array.*
- valarray< **Real** > **mergeunique** (valarray< **Real** > a1, valarray< **Real** > a2)  
*helper function to merge two valarrays of reals sorted in ascending order and remove duplicates.*
- void **ErrorMsg** (string str, bool mustexit=false)
- int **getint** (istream &istr)
- bool **realsEqual** (**Real** realOne, **Real** realTwo, **Real** precision=0.00001)
- string **valarrayRealToString** (const valarray< **Real** > &theArray)

### 7.88.1 Define Documentation

#### 7.88.1.1 `#define DEFAULT_PRECISION 0.00001`

Definition at line 11 of file utils.h.

## 7.88.2 Function Documentation

### 7.88.2.1 void `ErrorMsg` (string *str*, bool *mustexit* = false)

Definition at line 74 of file `utils.cpp`.

Referenced by `Matrix::CMAC()`, `Matrix::CMAR()`, `Matrix::Determinant()`, `Matrix::DivideColumn()`, `Matrix::DivideRow()`, `Matrix::GetInverse()`, `Matrix::operator*()`, `Matrix::operator()()`, `Matrix::operator+()`, `Matrix::operator-()`, `Matrix::operator/()`, and `Matrix::operator[]()`.

### 7.88.2.2 int `getint` (istream & *istr*)

Definition at line 143 of file `utils.cpp`.

Referenced by `Matrix::Input()`.

### 7.88.2.3 valarray<Real> `mergeunique` (valarray< Real > *a1*, valarray< Real > *a2*)

helper function to merge two valarrays of reals sorted in ascending order and remove duplicates.

*a1* - first array to merge *a2* - second array to merge

#### Returns:

the merged array

Definition at line 12 of file `utils.cpp`.

References `Natural`, `Real`, and `TN_INFINITY`.

Referenced by `creditCurve::combineUnderlyingAndSpreads()`, and `yieldCurve::operator==(())`.

### 7.88.2.4 bool `realsEqual` (Real *realOne*, Real *realTwo*, Real *precision* = 0.00001)

Definition at line 157 of file `utils.cpp`.

References `Real`.

Referenced by `mainbinomialtree()`, and `mainconvertiblebond()`.

### 7.88.2.5 short int `sign` (Real *x*)

sign of a Real

#### Author:

Yann

Definition at line 62 of file `utils.cpp`.

References `Real`.

Referenced by `CumulativeBivariateNormal()`.

**7.88.2.6 Matrix transform1DvalarrayToColumnMatrix (valarray< Real > array)**

Transforms a 1D array into a column vector - easier to handle our structures and do operations on them.

Definition at line 96 of file utils.cpp.

References Natural, and Matrix::SetValue().

Referenced by mainmatrix().

**7.88.2.7 Matrix transform2DvalarrayToMatrix (valarray< valarray< Real > > array)**

Transforms a 2D array into a matrix.

Definition at line 107 of file utils.cpp.

References Natural, and Matrix::SetValue().

Referenced by mainmatrix().

**7.88.2.8 valarray<Real> transformColumnMatrixTo1Dvalarray (Matrix M)**

Transforms a column matrix to a 1D array.

Definition at line 119 of file utils.cpp.

References M, and Natural.

Referenced by mainmatrix().

**7.88.2.9 valarray<valarray<Real> > transformMatrixTo2Dvalarray (Matrix M)**

Transforms a normal matrix to a 2D array.

Definition at line 128 of file utils.cpp.

References M, and Natural.

Referenced by mainmatrix().

**7.88.2.10 string valarrayRealToString (const valarray< Real > & theArray)**

Definition at line 162 of file utils.cpp.

References Natural.

Referenced by operator<<().

## 7.89 VanillaSwap.cpp File Reference

```
#include "..\VanillaSwap.h"
```



## 7.90 VanillaSwap.h File Reference

```
#include "../CashFlow.h"
```

### Classes

- class **VanillaSwap**

### Variables

- const **ShortNatural** **MAX\_LETTERS** = 30
- const **Real** **defaultprecisionInPrice** = 1.

#### 7.90.1 Variable Documentation

**7.90.1.1** const **Real** **defaultprecisionInPrice** = 1. [static]

Definition at line 6 of file VanillaSwap.h.

**7.90.1.2** const **ShortNatural** **MAX\_LETTERS** = 30 [static]

Definition at line 5 of file VanillaSwap.h.

## 7.91 VarianceSwap.cpp File Reference

```
#include "../VarianceSwap.h"
```

## 7.92 VarianceSwap.h File Reference

```
#include "../PartA/BlackScholes/OptionStrategy.h"
```

### Classes

- class **VarianceSwap**

## 7.93 volsurface.cpp File Reference

```
#include "..\volsurface.h"
```

## 7.94 volsurface.h File Reference

```
#include "../common/types.h"
#include "../common/date.h"
#include "../common/interpolator.h"
#include <valarray>
#include <cmath>
#include "../PartB/yieldCurve.h"
#include "../PartA/BlackScholes/BlackScholes.h"
```

### Classes

- class **volsurfaceparams**
- class **volsurface**

### Defines

- #define **PI** 3.141592653589793238462643

#### 7.94.1 Define Documentation

##### 7.94.1.1 #define PI 3.141592653589793238462643

Definition at line 14 of file volsurface.h.

## 7.95 yieldCurve.cpp File Reference

```
#include ".\yieldCurve.h"
```

### Functions

- ostream & **operator**<< (ostream &os, const **yieldCurve** &c)

#### 7.95.1 Function Documentation

##### 7.95.1.1 ostream& operator<< (ostream & os, const yieldCurve & c)

**Parameters:**

- os* - the output stream to direct output to
- c* - the curve to display

**Returns:**

output stream as is standard for operator<<

Definition at line 457 of file yieldCurve.cpp.

References `yieldCurve::getMaturitiesInTheZCBCurve()`, `Natural`, and `yieldCurve::spotRate()`.

## 7.96 yieldCurve.h File Reference

```
#include "../common/types.h"
#include "../common/date.h"
#include "../common/interpolator.h"
#include "../common/matrix.h"
#include "../common/utills.h"
#include <string>
#include <math.h>
#include <algorithm>
#include <time.h>
```

### Classes

- class `yieldPoint`
- class `yieldCurve`

### Defines

- `#define YC_NAME_STRLEN 128`
- `#define YC_DEFAULT_NUMER_POINTS 15`
- `#define YC_MAX_NUMBER_POINTS 50`

### Enumerations

- enum `interestComposition` { `Discrete`, `Continuous` }
- enum `TypeOfRate` { `Cash`, `Swap` }

### Variables

- `const Real defaultshiftfactorForShortRate = 0.0001`

#### 7.96.1 Define Documentation

##### 7.96.1.1 `#define YC_DEFAULT_NUMER_POINTS 15`

Definition at line 18 of file `yieldCurve.h`.

##### 7.96.1.2 `#define YC_MAX_NUMBER_POINTS 50`

Definition at line 19 of file `yieldCurve.h`.

Referenced by `FileReader::buildYieldPointArray()`, `yieldCurve::getSwapRates()`, `yieldCurve::sortMarketRatesByMaturity()`, and `yieldCurve::yieldCurve()`.

### 7.96.1.3 `#define YC_NAME_STRLEN 128`

Definition at line 17 of file yieldCurve.h.

## 7.96.2 Enumeration Type Documentation

### 7.96.2.1 `enum interestComposition`

**Author:**

Yann

**Enumeration values:**

*Discrete*

*Continuous*

Definition at line 24 of file yieldCurve.h.

### 7.96.2.2 `enum TypeOfRate`

**Enumeration values:**

*Cash*

*Swap*

Definition at line 29 of file yieldCurve.h.

Referenced by `FileReader::buildYieldPointArray()`, and `yieldPoint::getType()`.

## 7.96.3 Variable Documentation

### 7.96.3.1 `const Real defaultshiftfactorForShortRate = 0.0001 [static]`

Definition at line 90 of file yieldCurve.h.



# Index

- \_CorrelationMatrix
  - RainbowOption, 176
- \_DFTomaturity
  - RainbowOption, 176
- \_Drifts
  - RainbowOption, 176
- \_FlowAmountInPercent
  - flowSchedule, 95
- \_K
  - BlackScholes, 31
- \_MCEngine
  - RainbowOption, 177
- \_Multiplier
  - RainbowOption, 177
- \_NumberOfAssets
  - RainbowOption, 177
- \_So
  - binomialTree, 25
- \_Strike
  - RainbowOption, 178
- \_T
  - BlackScholes, 31
- \_TerminalPoints
  - RainbowOption, 178
- \_announcedDividendFlows
  - asset, 18
- \_areDividendsAsGrowingRate
  - asset, 18
- \_assets
  - Portfolio, 162
- \_bond
  - convertiblebond, 48
- \_bonds
  - Portfolio, 162
- \_bt
  - convertiblebond, 48
- \_btCached
  - convertiblebond, 49
- \_businessDayConventionOnPaymentDate
  - flowSchedule, 95
- \_callPrice
  - convertiblebond, 49
- \_callputprices
  - volsurface, 217
- \_cashflows
  - cashflow, 40
- \_cc
  - importData, 103
  - riskybond, 194
- \_claimProcess
  - binomialTree, 24
- \_combined
  - creditCurve, 63
- \_constantvol
  - volsurface, 217
- \_conversionRatio
  - convertiblebond, 49
- \_coupon
  - bond, 37
- \_currency
  - creditCurve, 63
  - Portfolio, 162
- \_currentPrice
  - asset, 18
- \_curve
  - VanillaSwap, 208
- \_d
  - binomialTree, 24
- \_datadir
  - FileReader, 92
  - importData, 103
- \_dateOfFlowPayment
  - flowSchedule, 95
- \_dateSchedule
  - SwapLeg, 202
- \_datedeltaCached
  - convertiblebond, 49
- \_dategammaCached
  - convertiblebond, 49
- \_dateinterestRateDeltaCached
  - convertiblebond, 49
- \_datepriceCached
  - convertiblebond, 49
- \_dates
  - cashflow, 41
- \_dayCount
  - yieldPoint, 233
- \_daycount
  - bond, 37
- \_defaultProbability

- creditCurve, 63
- \_delta
  - convertiblebond, 49
- \_deltaCached
  - convertiblebond, 49
- \_denomCur
  - asset, 18
- \_discountFactor
  - binomialTree, 24
- \_dividendGrowingRate
  - asset, 19
- \_dt
  - binomialTree, 24
- \_exoticsOptions
  - Portfolio, 162
- \_expiry
  - Exotics, 88
- \_expiryInYears
  - RainbowOption, 176
- \_faceamount
  - bond, 37
- \_firstcoupondate
  - bond, 37
- \_flowSchedule
  - SwapLeg, 202
- \_forward
  - VarianceSwap, 211
- \_freq
  - bond, 37
- \_frequency
  - creditCurve, 63
- \_gamma
  - convertiblebond, 50
- \_gammaCached
  - convertiblebond, 50
- \_gaussianSample
  - RainbowOption, 176
- \_impliedvolsurface
  - volsurface, 217
- \_insideOptions
  - OptionStrategy, 147
- \_insideQuantities
  - OptionStrategy, 147
- \_interestRateDelta
  - convertiblebond, 50
- \_interestRateDeltaCached
  - convertiblebond, 50
- \_interpvol surf
  - volsurface, 217
- \_isallputprices
  - volsurface, 217
- \_issue
  - bond, 37
- \_leg1
  - VanillaSwap, 208
- \_leg2
  - VanillaSwap, 209
- \_marketData
  - importData, 103
- \_marketRates
  - yieldCurve, 229
- \_maturities
  - volsurface, 217
- \_maturity
  - binomialTree, 24
  - bond, 37
  - CreditSpreadPoint, 67
  - VarianceSwap, 211
  - yieldPoint, 233
- \_n
  - binomialTree, 25
  - convertiblebond, 50
- \_nDates
  - Exotics, 88
- \_nPaths
  - Exotics, 88
- \_name
  - Portfolio, 162
  - yieldCurve, 229
- \_name1
  - VanillaSwap, 209
- \_name2
  - VanillaSwap, 209
- \_nbAssets
  - Portfolio, 162
- \_nbBonds
  - Portfolio, 162
- \_nbExoticsOptions
  - Portfolio, 163
- \_nbOptions
  - OptionStrategy, 147
- \_nbRainbowOptions
  - Portfolio, 163
- \_nbVanSwaps
  - Portfolio, 163
- \_nbVarSwaps
  - Portfolio, 163
- \_optionStrategy
  - Portfolio, 163
- \_options
  - VarianceSwap, 211
- \_outputMsgs
  - RainbowOption, 177
- \_pHazardRateProcesses
  - RainbowOption, 177
- \_pRandom
  - RainbowOption, 177
- \_price

- BlackScholes, 31
- convertiblebond, 50
- OptionStrategy, 147
- \_priceCached
  - convertiblebond, 50
- \_putPrice
  - convertiblebond, 50
- \_q
  - binomialTree, 25
- \_quantityAssets
  - Portfolio, 163
- \_quantityBonds
  - Portfolio, 163
- \_quantityExoticsOptions
  - Portfolio, 163
- \_quantityRainbowOptions
  - Portfolio, 164
- \_quantityVanSwaps
  - Portfolio, 164
- \_quantityVarSwaps
  - Portfolio, 164
- \_r
  - BlackScholes, 31
- \_rainbowOptions
  - Portfolio, 164
- \_rate
  - CreditSpreadPoint, 67
  - yieldPoint, 233
- \_recoveryRate
  - creditCurve, 63
- \_seed
  - RainbowOption, 178
- \_serialNumber
  - Date, 81
- \_sigma
  - binomialTree, 25
- \_spot
  - BlackScholes, 31
  - Exotics, 88
- \_spots
  - RainbowOption, 178
- \_spreads
  - creditCurve, 63
- \_spreadtype
  - CreditSpreadPoint, 67
- \_startDate
  - RainbowOption, 178
- \_stock
  - convertiblebond, 50
- \_stockPrice
  - volsurface, 217
- \_stockProcess
  - binomialTree, 25
- \_strike
  - Exotics, 88
  - GaussianProcess, 98
- \_strike2
  - Exotics, 88
- \_strikes
  - volsurface, 217
- \_survivalProbability
  - creditCurve, 63
- \_swapFees
  - creditCurve, 63
- \_thePayOff
  - RainbowOption, 178
- \_today
  - volsurface, 217
- \_type
  - BlackScholes, 31
  - Exotics, 88
  - RainbowOption, 179
  - yieldPoint, 233
- \_u
  - binomialTree, 25
- \_underlying
  - creditCurve, 64
- \_vanSwaps
  - Portfolio, 164
- \_varSwaps
  - Portfolio, 164
- \_vol
  - BlackScholes, 31
  - GaussianProcess, 98
- \_volSurface
  - Exotics, 89
- \_volatilities
  - RainbowOption, 179
- \_volatilitiesSurfaces
  - RainbowOption, 179
- \_volatility
  - asset, 19
- \_volsurfconst
  - volsurface, 218
- \_vs
  - importData, 103
- \_weights
  - RainbowOption, 179
- \_x
  - interpolator, 106
- \_x1
  - interpolator, 106
- \_x2
  - interpolator, 106
- \_y
  - interpolator, 106
- \_yc
  - asset, 19

- bond, 37
  - importData, 103
  - RainbowOption, 179
- \_yieldCurve
  - Exotics, 89
  - volsurface, 218
- \_ymat
  - interpolator, 107
- \_zcbRates
  - yieldCurve, 229
- ~BlackScholes
  - BlackScholes, 28
- ~CashFlow
  - CashFlow, 42
- ~CreditSpreadPoint
  - CreditSpreadPoint, 66
- ~Date
  - Date, 74
- ~Drift
  - Drift, 83
- ~Exotics
  - Exotics, 87
- ~GaussianProcess
  - GaussianProcess, 97
- ~MCEngine
  - MCEngine, 129
- ~Matrix
  - Matrix, 112
- ~MersenneTwister
  - MersenneTwister, 138
- ~OptionStrategy
  - OptionStrategy, 142
- ~ParkMiller
  - ParkMiller, 148
- ~PayOff
  - PayOff, 152
- ~Portfolio
  - Portfolio, 159
- ~RainbowOption
  - RainbowOption, 168
- ~RandC
  - RandC, 182
- ~RandomGenerator
  - RandomGenerator, 189
- ~Sobol
  - Sobol, 195
- ~StringTokenizer
  - StringTokenizer, 198
- ~SwapLeg
  - SwapLeg, 201
- ~VanillaSwap
  - VanillaSwap, 207
- ~VarianceSwap
  - VarianceSwap, 210
- ~asset
  - asset, 15
- ~binomialTree
  - binomialTree, 21
- ~bond
  - bond, 34
- ~cashflow
  - cashflow, 40
- ~convertiblebond
  - convertiblebond, 45
- ~creditCurve
  - creditCurve, 55
- ~flowSchedule
  - flowSchedule, 93
- ~riskybond
  - riskybond, 193
- ~treasurybond
  - treasurybond, 204
- ~volsurface
  - volsurface, 215
- ~yieldCurve
  - yieldCurve, 223
- ~yieldPoint
  - yieldPoint, 231
- A
  - RainbowOption, 179
- a
  - ParkMiller.h, 304
- Absolute
  - creditCurve.h, 251
- absolute
  - BlackScholes.cpp, 240
- ACT\_360
  - date.h, 257
- ACT\_365
  - date.h, 257
- addAsset
  - Portfolio, 160
- addBond
  - Portfolio, 160
- AddColumns
  - Matrix, 112
- addExoticOption
  - Portfolio, 160
- addLongButterflySpread
  - OptionStrategy, 142
- addLongCallSpread
  - OptionStrategy, 142
- addLongPutSpread
  - OptionStrategy, 142
- addLongRatioCallSpread
  - OptionStrategy, 143
- addLongStraddle

- OptionStrategy, 143
  - addLongStrangle
    - OptionStrategy, 143
  - addOneBlackScholesObject
    - OptionStrategy, 143
  - addOneOptionToStrategy
    - OptionStrategy, 143
  - addOptionStrategy
    - Portfolio, 160
  - addRainbowOption
    - Portfolio, 160
  - AddRows
    - Matrix, 112
  - addVanillaSwap
    - Portfolio, 160
  - addVarianceSwap
    - Portfolio, 160
  - adjustedConversionRatio
    - convertiblebond, 46
  - advance
    - Date, 74
  - Annual
    - date.h, 257
  - applyConvention
    - Date, 74
  - April
    - date.h, 258
  - areDivAsRate
    - asset, 15
  - AsianCall
    - Exotics.h, 262
    - PayOff, 152
  - AsianPut
    - Exotics.h, 262
    - PayOff, 153
  - asset, 13
    - \_announcedDividendFlows, 18
    - \_areDividendsAsGrowingRate, 18
    - \_currentPrice, 18
    - \_denomCur, 18
    - \_dividendGrowingRate, 19
    - \_volatility, 19
    - \_yc, 19
    - ~asset, 15
    - areDivAsRate, 15
    - asset, 15
    - forwardPrice, 15, 16
    - GetCurrencyFormat, 16
    - getDelta, 16
    - getFlowSchedule, 16
    - getPrice, 16
    - getRate, 16
    - getRho, 17
    - GetVolatility, 17
    - getYieldCurve, 17
    - Price, 17
    - setCcy, 17
    - setDivAsRate, 17
    - setDivFlows, 17
    - setPrice, 18
    - setVolatility, 18
    - setYieldCurve, 18
  - asset.cpp, 235
  - asset.h, 236
    - ASSET\_DEFAULT\_VOL, 236
  - ASSET\_DEFAULT\_VOL
    - asset.h, 236
  - AssetsBasketMax
    - rainbowoption.h, 325
  - assignFlatRate
    - yieldCurve, 223
  - assignFlatSpread
    - creditCurve, 56
  - assignZCBrateAtIndex
    - yieldCurve, 223
  - August
    - date.h, 258
  - Average
    - Normals.cpp, 297
    - Normals.h, 299
- B**
- RainbowOption, 179
  - BarrierCall
    - Exotics.h, 262
    - PayOff, 153
  - BarrierPut
    - Exotics.h, 262
    - PayOff, 153
  - BestOf2AssetsCash
    - rainbowoption.h, 325
  - BetterOf2Assets
    - rainbowoption.h, 325
  - Bimonthly
    - date.h, 257
  - binomialTree, 20
    - binomialTree, 21
  - binomialTree
    - \_So, 25
    - \_claimProcess, 24
    - \_d, 24
    - \_discountFactor, 24
    - \_dt, 24
    - \_maturity, 24
    - \_n, 25
    - \_q, 25
    - \_sigma, 25
    - \_stockProcess, 25

- `_u`, 25
- `~binomialTree`, 21
- `binomialTree`, 21
- `constructStockProcess`, 22
- `copyObj`, 22
- `getClaimProcess`, 22
- `getMaturity`, 22
- `getPrice`, 22
- `getRate`, 22
- `getSigma`, 22
- `getSo`, 23
- `getSteps`, 23
- `getStockProcess`, 23
- `operator<<`, 24
- `operator=`, 23
- `runEngineCall`, 23
- `runEngineConvertibleBond`, 23
- `setClaimVariables`, 23, 24
- `binomialTree.cpp`, 237
- `binomialTree.cpp`
  - `operator<<`, 237
- `binomialTree.h`, 238
- `binomialTree.h`
  - `BT_DEFAULT_MATURITY`, 238
  - `BT_DEFAULT_RATE`, 238
  - `BT_DEFAULT_SIGMA`, 238
  - `BT_DEFAULT_SO`, 238
  - `BT_DEFAULT_STEPS`, 238
- `BlackScholes`, 26
  - `BlackScholes`, 27
- `BlackScholes`
  - `_K`, 31
  - `_T`, 31
  - `_price`, 31
  - `_r`, 31
  - `_spot`, 31
  - `_type`, 31
  - `_vol`, 31
  - `~BlackScholes`, 28
  - `BlackScholes`, 27
  - `changeMaturity`, 28
  - `changeRate`, 28
  - `changeSpot`, 28
  - `changeStrike`, 28
  - `changeVol`, 28
  - `d1`, 32
  - `d2`, 32
  - `getDelta`, 28
  - `getGamma`, 29
  - `getMaturity`, 29
  - `getPrice`, 29
  - `getRate`, 29
  - `getRho`, 29
  - `getSpot`, 29
  - `getStrike`, 30
  - `getTheta`, 30
  - `getVega`, 30
  - `getVolatility`, 30
  - `isCall`, 30
    - `OptionStrategy`, 32
    - `recalcInformation`, 31
- `BlackScholes.cpp`, 240
- `BlackScholes.cpp`
  - `absolute`, 240
- `BlackScholes.h`, 241
  - `Call`, 241
  - `Put`, 241
- `BlackScholes.h`
  - `TypeOptionBS`, 241
- `bond`, 33
  - `_coupon`, 37
  - `_daycount`, 37
  - `_faceamount`, 37
  - `_firstcoupondate`, 37
  - `_freq`, 37
  - `_issue`, 37
  - `_maturity`, 37
  - `_yc`, 37
  - `~bond`, 34
  - `bond`, 34
  - `convexity`, 34
  - `duration`, 34
  - `fairvalue`, 35
  - `getCashflow`, 35
  - `getFaceAmount`, 35
  - `getIssue`, 35
  - `getMaturity`, 35
  - `getMaturityInYears`, 36
  - `quotedPrice`, 36
  - `yieldToMaturity`, 36
- `bond.cpp`, 242
- `bond.h`, 243
- `BT_DEFAULT_MATURITY`
  - `binomialTree.h`, 238
- `BT_DEFAULT_RATE`
  - `binomialTree.h`, 238
- `BT_DEFAULT_SIGMA`
  - `binomialTree.h`, 238
- `BT_DEFAULT_SO`
  - `binomialTree.h`, 238
- `BT_DEFAULT_STEPS`
  - `binomialTree.h`, 238
- `buildCreditSpreadPointArray`
  - `FileReader`, 90
- `BuildPath`
  - `GaussianProcess`, 97
- `BuildTerminalPoint`
  - `GaussianProcess`, 97

- buildVolSurfaceParams
  - FileReader, 90
- buildYieldPointArray
  - FileReader, 91
- BusinessDayConvention
  - date.h, 257
- C
  - RainbowOption, 179
- cachedval, 39
  - isCached, 39
  - value, 39
- CAD
  - types.h, 351
- Call
  - BlackScholes.h, 241
  - PayOff, 153
- callputprices
  - volsurfaceparams, 219
- CappedCliquet
  - Exotics.h, 262
  - PayOff, 153
- Cash
  - yieldCurve.h, 368
- CashFlow, 42
  - CashFlow, 42
- CashFlow
  - ~CashFlow, 42
  - CashFlow, 42
  - flowAmount, 43
  - flowDates, 43
  - getFairValue, 43
- cashflow, 40
  - \_cashflows, 40
  - \_dates, 41
  - ~cashflow, 40
  - cashflow, 40
  - getCashflows, 40
  - getDates, 40
- CashFlow.cpp, 244
- CashFlow.h, 245
- CB\_DEFAULT\_CALLPRICE
  - convertiblebond.h, 247
- CB\_DEFAULT\_DAYCOUNT
  - convertiblebond.h, 247
- CB\_DEFAULT\_FACEAMOUNT
  - convertiblebond.h, 247
- CB\_DEFAULT\_MATURITY
  - convertiblebond.h, 247
- CB\_DEFAULT\_PUTPRICE
  - convertiblebond.h, 247
- CB\_DEFAULT\_RATE
  - convertiblebond.h, 248
- CB\_DEFAULT\_RATIO
  - convertiblebond.h, 248
- CB\_DEFAULT\_SIGMA
  - convertiblebond.h, 248
- CB\_DEFAULT\_SO
  - convertiblebond.h, 248
- CB\_DEFAULT\_SPREAD
  - convertiblebond.h, 248
- CB\_DEFAULT\_STEPS
  - convertiblebond.h, 248
- CC\_DEFAULT\_CURRENCY
  - creditCurve.h, 250
- CC\_DEFAULT\_FREQUENCY
  - creditCurve.h, 250
- CC\_DEFAULT\_NAME
  - creditCurve.h, 250
- CC\_DEFAULT\_RECOVERY\_RATE
  - creditCurve.h, 250
- CC\_MAX\_NUM\_SPREADS
  - creditCurve.h, 250
- changeMaturity
  - BlackScholes, 28
  - OptionStrategy, 144
- changeRate
  - BlackScholes, 28
  - OptionStrategy, 144
- changeSpot
  - BlackScholes, 28
  - OptionStrategy, 144
- changeStrike
  - BlackScholes, 28
  - OptionStrategy, 144
- changeVol
  - BlackScholes, 28
  - OptionStrategy, 144
- choiceToType
  - productsCreation.cpp, 311
  - productsCreation.h, 317
- CholeskyDecomposition
  - Matrix, 112
- choosePricingType
  - productsCreation.cpp, 311
  - productsCreation.h, 317
- chooseRainbowType
  - productsCreation.cpp, 311
  - productsCreation.h, 317
- Clear
  - Matrix, 113
- ClearColumn
  - Matrix, 113
- ClearRow
  - Matrix, 113
- clone
  - Random, 186
- ClosedForm

- rainbowoption.h, 325
- CMAC
  - Matrix, 113
- CMAR
  - Matrix, 113
- CollaredCliquet
  - Exotics.h, 262
- combineUnderlyingAndSpreads
  - creditCurve, 56
- compute\_A
  - RainbowOption, 168
- compute\_B
  - RainbowOption, 168
- compute\_C
  - RainbowOption, 168
- compute\_ClosedFormsParameters
  - RainbowOption, 168
- compute\_d1
  - RainbowOption, 169
- compute\_d2
  - RainbowOption, 169
- compute\_d3
  - RainbowOption, 169
- compute\_d4
  - RainbowOption, 169
- compute\_rho1
  - RainbowOption, 169
- compute\_rho2
  - RainbowOption, 169
- compute\_sigmaA
  - RainbowOption, 169
- computeZCBRatesBootstrap
  - yieldCurve, 223
- ConcatenateColumn
  - Matrix, 113
- ConcatenateRow
  - Matrix, 113
- constructStockProcess
  - binomialTree, 22
- Continuous
  - yieldCurve.h, 368
- Convertible
  - PayOff, 153
- convertiblebond, 44
  - \_bond, 48
  - \_bt, 48
  - \_btCached, 49
  - \_callPrice, 49
  - \_conversionRatio, 49
  - \_datedeltaCached, 49
  - \_dategammaCached, 49
  - \_dateinterestRateDeltaCached, 49
  - \_datepriceCached, 49
  - \_delta, 49
  - \_deltaCached, 49
  - \_gamma, 50
  - \_gammaCached, 50
  - \_interestRateDelta, 50
  - \_interestRateDeltaCached, 50
  - \_n, 50
  - \_price, 50
  - \_priceCached, 50
  - \_putPrice, 50
  - \_stock, 50
  - ~convertiblebond, 45
  - adjustedConversionRatio, 46
  - convertiblebond, 45
  - copyObj, 46
  - delta, 46
  - fairvalue, 46
  - gamma, 46
  - getSteps, 47
  - interestRateDelta, 47
  - operator<<, 48
  - parity, 47
  - parityDelta, 47
  - parityGamma, 47
  - rho, 48
  - shiftedcbond, 48
- convertiblebond.cpp, 246
  - operator<<, 246
- convertiblebond.h, 247
  - CB\_DEFAULT\_CALLPRICE, 247
  - CB\_DEFAULT\_DAYCOUNT, 247
  - CB\_DEFAULT\_FACEAMOUNT, 247
  - CB\_DEFAULT\_MATURITY, 247
  - CB\_DEFAULT\_PUTPRICE, 247
  - CB\_DEFAULT\_RATE, 248
  - CB\_DEFAULT\_RATIO, 248
  - CB\_DEFAULT\_SIGMA, 248
  - CB\_DEFAULT\_SO, 248
  - CB\_DEFAULT\_SPREAD, 248
  - CB\_DEFAULT\_STEPS, 248
- convexity
  - bond, 34
- copyObj
  - binomialTree, 22
  - convertiblebond, 46
  - creditCurve, 56
- countTokens
  - StringTokenizer, 198
- createSpreadCurve
  - creditCurve, 56
- creditCurve, 52
  - creditCurve, 54, 55
- creditCurve
  - \_combined, 63
  - \_currency, 63



- \_defaultProbability, 63
  - \_frequency, 63
  - \_recoveryRate, 63
  - \_spreads, 63
  - \_survivalProbability, 63
  - \_swapFees, 63
  - \_underlying, 64
  - ~creditCurve, 55
  - assignFlatSpread, 56
  - combineUnderlyingAndSpreads, 56
  - copyObj, 56
  - createSpreadCurve, 56
  - creditCurve, 54, 55
  - creditSpread, 56
  - cumulativeDefaultProbability, 56
  - defaultProbability, 57
  - discountFactor, 57
  - forwardRate, 57, 58
  - getCombined, 58
  - getCurrency, 58
  - getDefaultProbability, 58
  - getFrequency, 58
  - getMaturitiesInTheZCBCurve, 59
  - getName, 59
  - getRecoveryRate, 59
  - getSpreads, 59
  - getSurvivalProbability, 59
  - getSwapFees, 59
  - getUnderlying, 60
  - hazardRate, 60
  - indexOfCurrentSpread, 60
  - indexOfPreviousSpread, 60
  - operator<<, 62
  - operator=, 60
  - resampleSpread, 61
  - riskyDiscountFactor, 61
  - spotRate, 61
  - survivalProbability, 62
  - swapFees, 62
  - timeOfCurrentSpread, 62
  - timeOfPreviousSpread, 62
- creditcurve
  - marketData, 108
- creditCurve.cpp, 249
- creditCurve.cpp
  - operator<<, 249
- creditCurve.h, 250
  - Absolute, 251
  - Relative, 251
- creditCurve.h
  - CC\_DEFAULT\_CURRENCY, 250
  - CC\_DEFAULT\_FREQUENCY, 250
  - CC\_DEFAULT\_NAME, 250
  - CC\_DEFAULT\_RECOVERY\_RATE, 250
  - CC\_MAX\_NUM\_SPREADS, 250
- CreditSpreadType, 251
- credits
  - credits.cpp, 252
  - main.h, 273
- credits.cpp, 252
  - credits, 252
- creditSpread
  - creditCurve, 56
- CreditSpreadPoint, 65
  - CreditSpreadPoint, 65
- CreditSpreadPoint
  - \_maturity, 67
  - \_rate, 67
  - \_spreadtype, 67
  - ~CreditSpreadPoint, 66
- CreditSpreadPoint, 65
  - getMaturity, 66
  - getRate, 66
  - getSpreadType, 66
  - setMaturity, 66
  - setRate, 66
  - setType, 66
  - TypeAsString, 67
- CreditSpreadType
  - creditCurve.h, 251
- CSNAME
  - importData.h, 269
- CSVParser, 68
  - CSVParser, 68
  - m\_nPos, 70
  - m\_sData, 70
  - operator<<, 68
  - operator>>, 68, 69
  - SkipSpaces, 69
- csvparser.cpp, 253
- csvparser.h, 254
- CumulativeBivariateNormal
  - Normals.cpp, 297
  - Normals.h, 299
- cumulativeDefaultProbability
  - creditCurve, 56
- CumulativeNormal
  - Normals.cpp, 297
  - Normals.h, 299
- Currency
  - types.h, 351
- d1
  - BlackScholes, 32
  - RainbowOption, 180
- d2

- BlackScholes, 32
- RainbowOption, 180
- d3
  - RainbowOption, 180
- d4
  - RainbowOption, 180
- Date, 71
  - \_serialNumber, 81
  - ~Date, 74
  - advance, 74
  - applyConvention, 74
  - Date, 73
  - dayCount, 74
  - dayOfMonth, 74
  - dayOfYear, 75
  - endOfMonth, 75
  - isBusinessDay, 75
  - isEndOfMonth, 75
  - isEOM, 75
  - isLeap, 75
  - lastDayOfMonth, 75
  - maxDate, 76
  - maximumSerialNumber, 76
  - minDate, 76
  - minimumSerialNumber, 76
  - month, 76
  - monthLength, 76
  - monthOffset, 76
  - nextWeekday, 77
  - nthWeekday, 77
  - operator!=, 77
  - operator+, 77
  - operator++, 77
  - operator+=", 77
  - operator-, 78
  - operator-, 78
  - operator=, 78
  - operator<, 78
  - operator<=", 78
  - operator==, 78
  - operator>, 79
  - operator>=", 79
  - plus, 79
  - plusDays, 79
  - plusMonths, 79
  - plusWeeks, 79
  - plusYears, 79
  - returnDateConvention, 79
  - serialNumber, 80
  - setDateToToday, 80
  - toString, 80
  - weekday, 80
  - year, 80
  - yearOffset, 81
- date.cpp, 255
- date.h, 256
  - ACT\_360, 257
  - ACT\_365, 257
  - Annual, 257
  - April, 258
  - August, 258
  - Bimonthly, 257
  - BusinessDayConvention, 257
  - Day, 256
  - Day30\_360, 257
  - Day30\_365, 257
  - DayCountConvention, 257
  - Days, 258
  - December, 258
  - EveryFourthMonth, 257
  - February, 258
  - Following, 257
  - Frequency, 257
  - Friday, 258
  - January, 258
  - July, 258
  - June, 258
  - March, 258
  - May, 258
  - ModifiedFollowing, 257
  - ModifiedPreceding, 257
  - Monday, 258
  - Month, 257
  - MonthEndReference, 257
  - Monthly, 257
  - Months, 258
  - NoFrequency, 257
  - November, 258
  - October, 258
  - Once, 257
  - Preceding, 257
  - Quarterly, 257
  - Saturday, 258
  - Semiannual, 257
  - September, 258
  - Sunday, 258
  - Thursday, 258
  - TimeUnit, 258
  - Tuesday, 258
  - Unadjusted, 257
  - Wednesday, 258
  - Weekday, 258
  - Weeks, 258
  - Year, 256
  - Years, 258
- Day
  - date.h, 256
  - Day30\_360

- date.h, 257
- Day30\_365
  - date.h, 257
- dayCount
  - Date, 74
- DayCountConvention
  - date.h, 257
- dayOfMonth
  - Date, 74
- dayOfYear
  - Date, 75
- Days
  - date.h, 258
- December
  - date.h, 258
- DEFAULT\_PRECISION
  - utils.h, 357
- defaultAdvDays
  - Exotics.h, 263
- defaultprecisionInPrice
  - VanillaSwap.h, 361
- defaultProbability
  - creditCurve, 57
- defaultshiftfactorForShortRate
  - yieldCurve.h, 368
- defaultshiftMat
  - OptionStrategy.h, 302
- defaultshiftRate
  - OptionStrategy.h, 302
- defaultshiftSpot
  - OptionStrategy.h, 302
- defaultshiftStrike
  - OptionStrategy.h, 302
- defaultshiftVol
  - OptionStrategy.h, 302
- defaultShiftVolSurface
  - Exotics.h, 263
- delim
  - StringTokenizer, 199
- delta
  - convertiblebond, 46
- Determinant
  - Matrix, 113
- Dimensionality
  - Random, 187
- discountFactor
  - creditCurve, 57
  - yieldCurve, 224
- Discrete
  - yieldCurve.h, 368
- Display
  - Matrix, 114
- displayFileFormatsMenu
  - importData, 101
- DivideColumn
  - Matrix, 114
- DivideRow
  - Matrix, 114
- Drift, 82
  - ~Drift, 83
  - Drift, 82, 83
  - GetDriftatimei, 84
  - GetTimeBtwDates, 84
  - GetvDates, 84
  - GetvDrift, 84
  - m\_nDates, 84
  - vDates, 84
  - vDrift, 84
- Drift.cpp, 259
- Drift.h, 260
- duration
  - bond, 34
- endOfMonth
  - Date, 75
- EPSILON
  - rainbowoption.h, 324
- ErrorMsg
  - utils.cpp, 354
  - utils.h, 358
- EUR
  - types.h, 351
- EveryFourthMonth
  - date.h, 257
- Exotics, 86
  - \_expiry, 88
  - \_nDates, 88
  - \_nPaths, 88
  - \_spot, 88
  - \_strike, 88
  - \_strike2, 88
  - \_type, 88
  - \_volSurface, 89
  - \_yieldCurve, 89
  - ~Exotics, 87
  - Exotics, 86
  - getDelta, 87
  - getPrice, 87
  - getRho, 87
  - getTheta, 87
  - getVega, 88
- Exotics.cpp, 261
- Exotics.h, 262
  - AsianCall, 262
  - AsianPut, 262
  - BarrierCall, 262
  - BarrierPut, 262
  - CappedCliquet, 262

- CollaredCliquet, 262
- defaultAdvDays, 263
- defaultShiftVolSurface, 263
- exoticsType, 262
- FlooredCliquet, 262
- mainmc, 263
- RevLookbackCall, 262
- RevLookbackPut, 262
- exoticsType
  - Exotics.h, 262
- fairvalue
  - bond, 35
  - convertiblebond, 46
- February
  - date.h, 258
- fileexists
  - FileReader, 91
- FileReader, 90
- FileReader
  - \_datadir, 92
  - buildCreditSpreadPointArray, 90
  - buildVolSurfaceParams, 90
  - buildYieldPointArray, 91
  - fileexists, 91
  - getdatadir, 92
  - getdatadirasString, 92
  - setdatadir, 92
- filereader.cpp, 264
- filereader.h, 265
- Fill
  - Matrix, 114
- FillColumn
  - Matrix, 114
- FillRow
  - Matrix, 114
- filterNextToken
  - StringTokenizer, 198
- FlooredCliquet
  - Exotics.h, 262
  - PayOff, 154
- flowAmount
  - CashFlow, 43
- flowDates
  - CashFlow, 43
- flowSchedule, 93
  - flowSchedule, 93, 94
- flowSchedule
  - \_FlowAmountInPercent, 95
  - \_businessDayConventionOnPaymentDate, 95
  - \_dateOfFlowPayment, 95
  - ~flowSchedule, 93
  - flowSchedule, 93, 94
  - getAmount, 94
  - getBusDayConv, 94
  - getDate, 94
  - setAmount, 94
  - setBusDayConv, 94
  - setDate, 95
- Following
  - date.h, 257
- forwardDiscountFactor
  - yieldCurve, 224
- forwardPrice
  - asset, 15, 16
- forwardRate
  - creditCurve, 57, 58
  - yieldCurve, 224, 225
- forwardVolatility
  - volsurface, 215
- forwardvolsurface
  - volsurface, 215
- forwardZCBCurve
  - yieldCurve, 225
- fr\_basic
  - mainfilereader.cpp, 279
  - testObjects.h, 340
- Frequency
  - date.h, 257
- Friday
  - date.h, 258
- gamma
  - convertiblebond, 46
- GaussianProcess, 96
  - GaussianProcess, 96, 97
- GaussianProcess
  - \_strike, 98
  - \_vol, 98
  - ~GaussianProcess, 97
  - BuildPath, 97
  - BuildTerminalPoint, 97
  - GaussianProcess, 96, 97
  - GetStepIncrements, 98
  - m\_dbInitialRate, 98
  - m\_dbMeanReversionSpeed, 98
  - m\_dbVol, 98
  - m\_nDates, 98
  - m\_vDates, 98
  - m\_vDrift, 99
  - m\_vStepSize, 99
- GaussianProcess.cpp, 266
- GaussianProcess.h, 267
- GaussianProcess.h
  - GAUSSIANPROCESS\_H, 267
- GAUSSIANPROCESS\_H
  - GaussianProcess.h, 267

- getAmount
  - flowSchedule, 94
- getBusDayConv
  - flowSchedule, 94
- getCashflow
  - bond, 35
- getCashflows
  - cashflow, 40
- getClaimProcess
  - binomialTree, 22
- GetCMAC
  - Matrix, 114
- GetCMAR
  - Matrix, 114
- GetColumnMax
  - Matrix, 115
- GetColumnMin
  - Matrix, 115
- GetColumnRange
  - Matrix, 115
- GetColumns
  - Matrix, 115
- getCombined
  - creditCurve, 58
- getCorrelRisk
  - RainbowOption, 170
- GetCovariant
  - Matrix, 115
- getCreditCurve
  - importData, 101
- getCurrency
  - creditCurve, 58
  - Portfolio, 161
- getCurrencyAsString
  - Portfolio, 161
- GetCurrencyFormat
  - asset, 16
- getData
  - importData, 101
- getdatadir
  - FileReader, 92
- getdatadirasString
  - FileReader, 92
- GetDataOneDimen
  - Matrix, 115
- GetDataTwoDimen
  - Matrix, 115
- getDate
  - flowSchedule, 94
- getDates
  - cashflow, 40
- getDayCount
  - yieldPoint, 231
- getDefaultProbability
  - creditCurve, 58
- getDelta
  - asset, 16
  - BlackScholes, 28
  - Exotics, 87
  - RainbowOption, 170
- GetDimensionality
  - Random, 186
- GetDriftattimei
  - Drift, 84
- getFaceAmount
  - bond, 35
- getFairValue
  - CashFlow, 43
- getFairValue1
  - VanillaSwap, 208
- getFairValue2
  - VanillaSwap, 208
- getFlowSchedule
  - asset, 16
- getFrequency
  - creditCurve, 58
- getGamma
  - BlackScholes, 29
  - RainbowOption, 170
- GetGaussian
  - Random, 186
- GetGaussians
  - Random, 186
- getGlobalDelta
  - OptionStrategy, 144
- getGlobalGamma
  - OptionStrategy, 145
- getGlobalRho
  - OptionStrategy, 145
- getGlobalTheta
  - OptionStrategy, 145
- getGlobalVega
  - OptionStrategy, 145
- getInt
  - utils.cpp, 354
  - utils.h, 358
- getInterpolation
  - interpolator, 105
- GetInverse
  - Matrix, 115
- getIssue
  - bond, 35
- getMaturitiesInTheMarketCurve
  - yieldCurve, 225
- getMaturitiesInTheZCBCurve
  - creditCurve, 59
  - yieldCurve, 225
- getMaturity

- binomialTree, 22
- BlackScholes, 29
- bond, 35
- CreditSpreadPoint, 66
- yieldPoint, 231
- getMaturityInYears
  - bond, 36
- GetMax
  - Matrix, 116
- GetMin
  - Matrix, 116
- GetMinor
  - Matrix, 116
- GetMinorNew
  - Matrix, 116
- getName
  - creditCurve, 59
  - Portfolio, 161
  - yieldCurve, 226
- GetNormalized
  - Matrix, 116
- GetNumericRange
  - Matrix, 116
- GetNumericRangeOfColumn
  - Matrix, 116
- GetNumericRangeOfRow
  - Matrix, 117
- GetOneRandomInteger
  - MersenneTwister, 138
  - ParkMiller, 149
  - RandC, 183
  - RandomGenerator, 190
  - Sobol, 196
- getPartialDelta
  - RainbowOption, 170
- getPartialGamma
  - RainbowOption, 170
- getPartialVega
  - RainbowOption, 171
- getPlace
  - interpolator, 105
- getPlaceOnXi
  - interpolator, 105
- getPointAtMaturity
  - yieldCurve, 226
- getPrice
  - asset, 16
  - binomialTree, 22
  - BlackScholes, 29
  - Exotics, 87
  - Portfolio, 161
  - RainbowOption, 171
  - VarianceSwap, 211
- getRainbowType
  - RainbowOption, 171
- GetRange
  - Matrix, 117
- getRate
  - asset, 16
  - binomialTree, 22
  - BlackScholes, 29
  - CreditSpreadPoint, 66
  - yieldPoint, 231
- getRecoveryRate
  - creditCurve, 59
- GetREF
  - Matrix, 117
- getRho
  - asset, 17
  - BlackScholes, 29
  - Exotics, 87
  - RainbowOption, 171
  - VanillaSwap, 208
  - VarianceSwap, 211
- GetRowMax
  - Matrix, 117
- GetRowMin
  - Matrix, 117
- GetRowRange
  - Matrix, 117
- GetRows
  - Matrix, 117
- GetRREF
  - Matrix, 117
- getSequentSwapRates
  - yieldCurve, 226
- getSigma
  - binomialTree, 22
- getSo
  - binomialTree, 23
- getSpot
  - BlackScholes, 29
- getSpreads
  - creditCurve, 59
- getSpreadType
  - CreditSpreadPoint, 66
- GetStepIncrements
  - GaussianProcess, 98
- getSteps
  - binomialTree, 23
  - convertiblebond, 47
- getStockProcess
  - binomialTree, 23
- getStrike
  - BlackScholes, 30
- GetSubMatrix
  - Matrix, 118
- getSurvivalProbability

- creditCurve, 59
- getSwapFees
  - creditCurve, 59
- getSwapRates
  - yieldCurve, 226
- getTheta
  - BlackScholes, 30
  - Exotics, 87
  - RainbowOption, 171
  - VanillaSwap, 208
  - VarianceSwap, 211
- GetTimeBtwDates
  - Drift, 84
- GetTransposed
  - Matrix, 118
- getType
  - yieldPoint, 232
- getUnderlying
  - creditCurve, 60
- GetUniform
  - Random, 186
- getUniform
  - MersenneTwister, 138
  - ParkMiller, 149
  - RandC, 183
  - RandomGenerator, 190
  - Sobol, 196
- GetUniforms
  - Random, 187
- GetvDates
  - Drift, 84
- GetvDrift
  - Drift, 84
- getVega
  - BlackScholes, 30
  - Exotics, 88
  - RainbowOption, 172
  - VarianceSwap, 211
- GetVolatility
  - asset, 17
- getVolatility
  - BlackScholes, 30
- getVolatilitySurface
  - importData, 101
- getvolsurface
  - volsurface, 215
- getYieldCurve
  - asset, 17
  - importData, 101
- GREEKAPPROX
  - rainbowoption.h, 324
- hasMoreTokens
  - StringTokenizer, 198
- haveClosedFormVariablesBeenComputed
  - RainbowOption, 180
- hazardRate
  - creditCurve, 60
- IdentityMatrix
  - Matrix, 118
  - matrix.cpp, 291
  - matrix.h, 292
- importCreditCurve
  - importData, 101
- importData, 100
  - importData, 101
- importData
  - \_cc, 103
  - \_datadir, 103
  - \_marketData, 103
  - \_vs, 103
  - \_yc, 103
  - displayFileFormatsMenu, 101
  - getCreditCurve, 101
  - getData, 101
  - getVolatilitySurface, 101
  - getYieldCurve, 101
  - importCreditCurve, 101
  - importData, 101
  - importVolSurface, 102
  - importYieldCurve, 102
  - runInterface, 102
  - runUserDefinedInterface, 102
  - setMarketData, 102
- importData.cpp, 268
- importData.h, 269
- importData.h
  - CSNAME, 269
  - VSNAME, 269
  - YCNAME, 269
- importVolSurface
  - importData, 102
- importYieldCurve
  - importData, 102
- indexOfCurrentSpread
  - creditCurve, 60
- indexOfPreviousSpread
  - creditCurve, 60
- InitialSeed
  - Random, 187
- InnerGenerator
  - Random, 188
- Input
  - Matrix, 118
- inputBond
  - productsCreation.cpp, 311
  - productsCreation.h, 318

- inputBSOption
  - productsCreation.cpp, 312
  - productsCreation.h, 318
- inputButterflySpread
  - productsCreation.cpp, 312
  - productsCreation.h, 318
- inputCallSpread
  - productsCreation.cpp, 312
  - productsCreation.h, 318
- inputConvertibleBond
  - productsCreation.cpp, 312
  - productsCreation.h, 318
- inputExoticOptionOnSingleAsset
  - productsCreation.cpp, 312
  - productsCreation.h, 319
- inputOptionStrategy
  - productsCreation.cpp, 313
  - productsCreation.h, 319
- inputPutSpread
  - productsCreation.cpp, 313
  - productsCreation.h, 319
- inputRainbowOption
  - productsCreation.cpp, 313
  - productsCreation.h, 319
- inputRatioCallSpread
  - productsCreation.cpp, 313
  - productsCreation.h, 320
- inputSpecificOptionStrategy
  - productsCreation.cpp, 314
  - productsCreation.h, 320
- inputStraddle
  - productsCreation.cpp, 314
  - productsCreation.h, 320
- inputStrangle
  - productsCreation.cpp, 314
  - productsCreation.h, 320
- inputVanillaSwap
  - productsCreation.cpp, 314
  - productsCreation.h, 320
- instantiateMCVariables
  - RainbowOption, 172
- Integer
  - types.h, 347
- interestComposition
  - yieldCurve.h, 368
- interestRateDelta
  - convertiblebond, 47
- interpolate
  - interpolator, 105, 106
- interpolator, 104
  - \_x, 106
  - \_x1, 106
  - \_x2, 106
  - \_y, 106
  - \_ymat, 107
- getInterpolation, 105
- getPlace, 105
- getPlaceOnXi, 105
- interpolate, 105, 106
- interpolator, 104, 105
- interpolator.cpp, 270
  - interpolatormain, 270
- interpolator.h, 271
- interpolatormain
  - interpolator.cpp, 270
- InverseCumulativeNormal
  - Normals.cpp, 298
  - Normals.h, 299
- Invert
  - Matrix, 118
- invertBSformula
  - volsurface, 215
- isBusinessDay
  - Date, 75
  - UsDate, 206
- isCached
  - cachedval, 39
- isCall
  - BlackScholes, 30
- iscallputprices
  - volsurfaceparams, 219
- IsEmpty
  - Matrix, 118
- isEndOfMonth
  - Date, 75
- isEOM
  - Date, 75
- IsIdentity
  - Matrix, 118
- isLeap
  - Date, 75
- January
  - date.h, 258
- July
  - date.h, 258
- June
  - date.h, 258
- lastDayOfMonth
  - Date, 75
- LeftRemoveIdentity
  - Matrix, 118
- LongInteger
  - types.h, 347
- LongNatural
  - types.h, 347
- LOWER\_MASK



- MersenneTwister.cpp, 295
- M
- MersenneTwister.cpp, 295
- m
- ParkMiller.h, 304
- m\_dbInitialRate
  - GaussianProcess, 98
- m\_dbMeanReversionSpeed
  - GaussianProcess, 98
- m\_dbVol
  - GaussianProcess, 98
- m\_DiscFactor
  - MCEngine, 135
- m\_nCols
  - Matrix, 125
- m\_nDates
  - Drift, 84
  - GaussianProcess, 98
  - MCEngine, 135
- m\_nPaths
  - MCEngine, 135
- m\_nPos
  - CSVParser, 70
- m\_nRows
  - Matrix, 125
- m\_pData
  - Matrix, 125
- m\_price
  - MCEngine, 135
- m\_sData
  - CSVParser, 70
- m\_vDates
  - GaussianProcess, 98
- m\_vDrift
  - GaussianProcess, 99
- m\_vStepSize
  - GaussianProcess, 99
- main
  - main.cpp, 272
- main.cpp, 272
  - main, 272
- main.h, 273
  - credits, 273
  - maintests, 273
- mainasset
  - maintestasset.cpp, 287
  - testObjects.h, 340
- mainbinomialtree
  - mainbinomialtree.cpp, 274
  - testObjects.h, 340
- mainbinomialtree.cpp, 274
  - mainbinomialtree, 274
- mainbond
  - mainbond.cpp, 275
  - testObjects.h, 341
- mainbond.cpp, 275
  - mainbond, 275
- mainconvertiblebond
  - mainconvertiblebond.cpp, 276
  - testObjects.h, 341
- mainconvertiblebond.cpp, 276
  - mainconvertiblebond, 276
- maincreditcurve
  - maincreditcurves.cpp, 277
  - testObjects.h, 341
- maincreditcurves.cpp, 277
  - maincreditcurve, 277
- maindate
  - maindate.cpp, 278
  - testObjects.h, 341
- maindate.cpp, 278
  - maindate, 278
- mainfilereader
  - mainfilereader.cpp, 279
  - testObjects.h, 342
- mainfilereader.cpp, 279
  - fr\_basic, 279
  - mainfilereader, 279
- maininterpolator
  - maininterpolator.cpp, 280
  - testObjects.h, 342
- maininterpolator.cpp, 280
  - maininterpolator, 280
- mainIRVanillaSwap
  - mainIRVanillaSwap.cpp, 281
  - testObjects.h, 342
- mainIRVanillaSwap.cpp, 281
  - mainIRVanillaSwap, 281
- mainmatrix
  - mainmatrix.cpp, 282
  - testObjects.h, 342
- mainmatrix.cpp, 282
  - mainmatrix, 282
- mainmc
  - Exotics.h, 263
  - mainmontecarlo.cpp, 283
  - testObjects.h, 343
- mainmontecarlo
  - mainmontecarlo.cpp, 283
  - testObjects.h, 343
- mainmontecarlo.cpp, 283
  - mainmc, 283
  - mainmontecarlo, 283
- mainoption
  - mainoptionstrategy.cpp, 285
  - testObjects.h, 343

- mainoptionstrategy
  - mainoptionstrategy.cpp, 285
  - testObjects.h, 343
- mainoptionstrategy.cpp, 285
  - mainoption, 285
  - mainoptionstrategy, 285
- mainrainbowoptions
  - mainrainbowoptions.cpp, 286
  - testObjects.h, 344
- mainrainbowoptions.cpp, 286
  - mainrainbowoptions, 286
- maintestasset.cpp, 287
  - mainasset, 287
- maintests
  - main.h, 273
  - testObjects.cpp, 338
- mainvarianceswap
  - mainvarianceswap.cpp, 288
  - testObjects.h, 344
- mainvarianceswap.cpp, 288
  - mainvarianceswap, 288
- mainvolsurface
  - mainvolsurface.cpp, 289
  - testObjects.h, 344
- mainvolsurface.cpp, 289
  - mainvolsurface, 289
- mainyieldcurve
  - mainyieldcurves.cpp, 290
  - testObjects.h, 344
- mainyieldcurves.cpp, 290
  - mainyieldcurve, 290
- MakeCovariant
  - Matrix, 119
- March
  - date.h, 258
- marketData, 108
- marketData
  - creditcurve, 108
  - vols, 108
  - yieldcurve, 108
- Matrix, 109
  - ~Matrix, 112
  - AddColumns, 112
  - AddRows, 112
  - CholeskyDecomposition, 112
  - Clear, 113
  - ClearColumn, 113
  - ClearRow, 113
  - CMAC, 113
  - CMAR, 113
  - ConcatenateColumn, 113
  - ConcatenateRow, 113
  - Determinant, 113
  - Display, 114
  - DivideColumn, 114
  - DivideRow, 114
  - Fill, 114
  - FillColumn, 114
  - FillRow, 114
  - GetCMAC, 114
  - GetCMAR, 114
  - GetColumnMax, 115
  - GetColumnMin, 115
  - GetColumnRange, 115
  - GetColumns, 115
  - GetCovariant, 115
  - GetDataOneDimen, 115
  - GetDataTwoDimen, 115
  - GetInverse, 115
  - GetMax, 116
  - GetMin, 116
  - GetMinor, 116
  - GetMinorNew, 116
  - GetNormalized, 116
  - GetNumericRange, 116
  - GetNumericRangeOfColumn, 116
  - GetNumericRangeOfRow, 117
  - GetRange, 117
  - GetREF, 117
  - GetRowMax, 117
  - GetRowMin, 117
  - GetRowRange, 117
  - GetRows, 117
  - GetRREF, 117
  - GetSubMatrix, 118
  - GetTransposed, 118
  - IdentityMatrix, 118
  - Input, 118
  - Invert, 118
  - IsEmpty, 118
  - IsIdentity, 118
  - LeftRemoveIdentity, 118
  - m\_nCols, 125
  - m\_nRows, 125
  - m\_pData, 125
  - MakeCovariant, 119
  - Matrix, 111, 112
  - MultiplyColumn, 119
  - MultiplyRow, 119
  - Normalize, 119
  - operator \*, 119
  - operator \*&, 119, 120
  - operator !=, 120
  - operator(), 120
  - operator+, 120
  - operator+&, 120
  - operator-, 120
  - operator-&, 120

- operator/, 120, 121
- operator/=: 121
- operator=: 121
- operator==, 121
- operator[], 121
- operator~, 121
- Output, 121
- Read, 122
- REF, 122
- RemoveColumn, 122
- RemoveRow, 122
- RightAppendIdentity, 122
- RREF, 122
- SetSubMatrix, 122
- SetValue, 122
- SortAscend, 123
- SortDescend, 123
- SpliceInColumn, 123
- SpliceInRow, 123
- SumAll, 123
- SumAllSquared, 123
- SumColumn, 124
- SumColumnSquared, 124
- SumRow, 124
- SumRowSquared, 124
- SwapCols, 124
- SwapRows, 124
- Transpose, 124
- Write, 124
- matrix.cpp, 291
  - IdentityMatrix, 291
  - operator<<, 291
- matrix.h, 292
  - IdentityMatrix, 292
  - operator<<, 292
- MATRIX\_A
  - MersenneTwister.cpp, 295
- maturities
  - volsurfaceparams, 219
- Max
  - MersenneTwister, 138
  - ParkMiller, 149
  - RandC, 183
  - RandomGenerator, 190
  - Sobol, 196
- Max2AssetsCall
  - rainbowoption.h, 325
- Max2AssetsPut
  - rainbowoption.h, 325
- MAX\_LETTERS
  - VanillaSwap.h, 361
- MAX\_SIZE
  - PortFolio.h, 309
- MAX\_SIZE\_NAME
  - PortFolio.h, 309
- MAXBIT
  - Sobol.h, 333
- maxDate
  - Date, 76
- Maxim
  - RandC.cpp, 326
- Maximize
  - Normals.cpp, 298
  - Normals.h, 300
- maximumSerialNumber
  - Date, 76
- maxNbOptions
  - OptionStrategy.h, 302
- May
  - date.h, 258
- MCEngine, 127
  - ~MCEngine, 129
  - m\_DiscFactor, 135
  - m\_nDates, 135
  - m\_nPaths, 135
  - m\_price, 135
  - MCEngine, 129, 130
  - MCRResult, 130
  - RunEngineAsianCall, 130
  - RunEngineAsianPut, 130
  - RunEngineBarrierCall, 131
  - RunEngineBarrierPut, 131
  - RunEngineCall, 131
  - RunEngineCappedCliquet, 131
  - RunEngineFlooredCliquet, 132
  - RunEngineGeneral, 132
  - RunEnginePut, 132
  - RunEngineRainbow2AssetsBasketMax, 132
  - RunEngineRainbow2SpreadOptionMax, 133
  - RunEngineRainbowBestOf2AssetsCash, 133
  - RunEngineRainbowMax2AssetsCall, 133
  - RunEngineRainbowMax2AssetsPut, 133
  - RunEngineRainbowMin2AssetsCall, 134
  - RunEngineRainbowMin2AssetsPut, 134
  - RunEngineRainbow-  
WorstOf2AssetsCash, 134
  - RunEngineRevLookbackCall, 134
  - RunEngineRevLookbackPut, 135
- MCEngine.cpp, 293
- MCEngine.h, 294
- MCRResult
  - MCEngine, 130

- mergeunique
  - utils.cpp, 354
  - utils.h, 358
- MersenneTwister, 137
  - MersenneTwister, 137
- MersenneTwister
  - ~MersenneTwister, 138
  - GetOneRandomInteger, 138
  - getUniform, 138
  - Max, 138
  - MersenneTwister, 137
  - Min, 138
  - mt, 139
  - mti, 139
  - seed, 139
  - SetSeed, 138
- MersenneTwister.cpp, 295
- MersenneTwister.cpp
  - LOWER\_MASK, 295
  - M, 295
  - MATRIX\_A, 295
  - N, 295
  - UPPER\_MASK, 295
- MersenneTwister.h, 296
- Min
  - MersenneTwister, 138
  - ParkMiller, 149
  - RandC, 183
  - RandomGenerator, 190
  - Sobol, 196
- Min2AssetsCall
  - rainbowoption.h, 325
- Min2AssetsPut
  - rainbowoption.h, 325
- minDate
  - Date, 76
- minimumSerialNumber
  - Date, 76
- ModifiedFollowing
  - date.h, 257
- ModifiedPreceding
  - date.h, 257
- Monday
  - date.h, 258
- MonteCarlo
  - rainbowoption.h, 325
- Month
  - date.h, 257
- month
  - Date, 76
- MonthEndReference
  - date.h, 257
- monthLength
  - Date, 76
- Monthly
  - date.h, 257
- monthOffset
  - Date, 76
- Months
  - date.h, 258
- mt
  - MersenneTwister, 139
- mti
  - MersenneTwister, 139
- MultiplyColumn
  - Matrix, 119
- MultiplyRow
  - Matrix, 119
- N
  - MersenneTwister.cpp, 295
- n\_
  - Sobol, 197
- Natural
  - types.h, 348
- nextFloatToken
  - StringTokenizer, 199
- nextIntToken
  - StringTokenizer, 199
- nextToken
  - StringTokenizer, 199
- nextWeekday
  - Date, 77
- NoFrequency
  - date.h, 257
- NormalDensity
  - Normals.cpp, 298
  - Normals.h, 300
- Normalize
  - Matrix, 119
- Normals.cpp, 297
  - Average, 297
  - CumulativeBivariateNormal, 297
  - CumulativeNormal, 297
  - InverseCumulativeNormal, 298
  - Maximize, 298
  - NormalDensity, 298
  - OneOverRootTwoPi, 298
  - SubFunctionForBivariateNormal, 298
- Normals.h, 299
  - Average, 299
  - CumulativeBivariateNormal, 299
  - CumulativeNormal, 299
  - InverseCumulativeNormal, 299
  - Maximize, 300
  - NormalDensity, 300
  - SubFunctionForBivariateNormal, 300
- November

- date.h, 258
- nthWeekday
  - Date, 77
- October
  - date.h, 258
- Once
  - date.h, 257
- OneOverRootTwoPi
  - Normals.cpp, 298
- operator \*
  - Matrix, 119
- operator \*=
  - Matrix, 119, 120
- operator !=
  - Date, 77
  - Matrix, 120
  - yieldCurve, 226
- operator()
  - Matrix, 120
  - PayOff, 154
- operator +
  - Date, 77
  - Matrix, 120
- operator ++
  - Date, 77
- operator +=
  - Date, 77
  - Matrix, 120
- operator -
  - Date, 78
  - Matrix, 120
- operator -=
  - Date, 78
  - Matrix, 120
- operator /
  - Matrix, 120, 121
- operator /=
  - Matrix, 121
- operator <
  - Date, 78
- operator <<
  - binomialTree, 24
  - binomialTree.cpp, 237
  - convertiblebond, 48
  - convertiblebond.cpp, 246
  - creditCurve, 62
  - creditCurve.cpp, 249
  - CSVParser, 68
  - matrix.cpp, 291
  - matrix.h, 292
  - OptionStrategy, 146
- OptionStrategy.cpp, 301
- yieldCurve, 229
- yieldCurve.cpp, 366
- operator <=
  - Date, 78
- operator =
  - binomialTree, 23
  - creditCurve, 60
  - Matrix, 121
- operator ==
  - Date, 78
  - Matrix, 121
  - yieldCurve, 227
- operator >
  - Date, 79
- operator >=
  - Date, 79
- operator >>
  - CSVParser, 68, 69
- operator []
  - Matrix, 121
- operator ~
  - Matrix, 121
- OptionStrategy, 140
  - BlackScholes, 32
  - OptionStrategy, 142
- OptionStrategy
  - \_insideOptions, 147
  - \_insideQuantities, 147
  - \_nbOptions, 147
  - \_price, 147
  - ~OptionStrategy, 142
  - addLongButterflySpread, 142
  - addLongCallSpread, 142
  - addLongPutSpread, 142
  - addLongRatioCallSpread, 143
  - addLongStraddle, 143
  - addLongStrangle, 143
  - addOneBlackScholesObject, 143
  - addOneOptionToStrategy, 143
  - changeMaturity, 144
  - changeRate, 144
  - changeSpot, 144
  - changeStrike, 144
  - changeVol, 144
  - getGlobalDelta, 144
  - getGlobalGamma, 145
  - getGlobalRho, 145
  - getGlobalTheta, 145
  - getGlobalVega, 145
  - operator <<, 146
  - OptionStrategy, 142
  - recalcPrice, 145
  - returnNbOptions, 145

- returnOption, 146
- returnOptionQuantity, 146
- returnPrice, 146
- OptionStrategy.cpp, 301
- OptionStrategy.h, 302
  - operator<<, 301
- OptionStrategy.h, 302
  - defaultshiftMat, 302
  - defaultshiftRate, 302
  - defaultshiftSpot, 302
  - defaultshiftStrike, 302
  - defaultshiftVol, 302
  - maxNbOptions, 302
- Output
  - Matrix, 121
- outputCallPut
  - productsCreation.cpp, 314
  - productsCreation.h, 321
- parity
  - convertiblebond, 47
- parityDelta
  - convertiblebond, 47
- parityGamma
  - convertiblebond, 47
- ParkMiller, 148
  - ParkMiller, 148
- ParkMiller
  - ~ParkMiller, 148
  - GetOneRandomInteger, 149
  - getUniform, 149
  - Max, 149
  - Min, 149
  - ParkMiller, 148
  - Seed, 149
  - SetSeed, 149
- ParkMiller.cpp, 303
- ParkMiller.h, 304
- ParkMiller.h
  - a, 304
  - m, 304
  - q, 304
  - r, 304
- PayOff, 151
  - PayOff, 152
- PayOff
  - ~PayOff, 152
  - AsianCall, 152
  - AsianPut, 153
  - BarrierCall, 153
  - BarrierPut, 153
  - Call, 153
  - CappedCliquet, 153
  - Convertible, 153
  - FlooredCliquet, 154
  - operator(), 154
  - PayOff, 152
  - Put, 154
  - Rainbow2AssetsBasketMax, 154
  - Rainbow2SpreadOptionMax, 154
  - RainbowBestOf2AssetsCash, 155
  - RainbowMax2AssetsCall, 155
  - RainbowMax2AssetsPut, 155
  - RainbowMin2AssetsCall, 155
  - RainbowMin2AssetsPut, 156
  - RainbowWorstOf2AssetsCash, 156
  - RevLookbackCall, 156
  - RevLookbackPut, 156
  - SetStrike, 156
  - Strike, 157
- PayOff.cpp, 306
- PayOff.h, 307
- PI
  - volsurface.h, 365
- plus
  - Date, 79
- plusDays
  - Date, 79
- plusMonths
  - Date, 79
- plusWeeks
  - Date, 79
- plusYears
  - Date, 79
- Portfolio, 158
  - \_assets, 162
  - \_bonds, 162
  - \_currency, 162
  - \_exoticsOptions, 162
  - \_name, 162
  - \_nbAssets, 162
  - \_nbBonds, 162
  - \_nbExoticsOptions, 163
  - \_nbRainbowOptions, 163
  - \_nbVanSwaps, 163
  - \_nbVarSwaps, 163
  - \_optionStrategy, 163
  - \_quantityAssets, 163
  - \_quantityBonds, 163
  - \_quantityExoticsOptions, 163
  - \_quantityRainbowOptions, 164
  - \_quantityVanSwaps, 164
  - \_quantityVarSwaps, 164
  - \_rainbowOptions, 164
  - \_vanSwaps, 164
  - \_varSwaps, 164
  - ~Portfolio, 159

- addAsset, 160
- addBond, 160
- addExoticOption, 160
- addOptionStrategy, 160
- addRainbowOption, 160
- addVanillaSwap, 160
- addVarianceSwap, 160
- getCurrency, 161
- getCurrencyAsString, 161
- getName, 161
- getPrice, 161
- Portfolio, 159
- returnSensibilityToRate, 161
- returnSensibilityToTime, 161
- returnSensibilityToVol, 162
- PortFolio.cpp, 308
- PortFolio.h, 309
- PortFolio.h
  - MAX\_SIZE, 309
  - MAX\_SIZE\_NAME, 309
- Preceding
  - date.h, 257
- Price
  - asset, 17
- PriceByClosedForm\_BestOf2\_plusCash
  - RainbowOption, 172
- PriceByClosedForm\_BetterOf2
  - RainbowOption, 172
- PriceByClosedForm\_MaxOf2\_call
  - RainbowOption, 172
- PriceByClosedForm\_MaxOf2\_put
  - RainbowOption, 173
- PriceByClosedForm\_MinOf2\_call
  - RainbowOption, 173
- PriceByClosedForm\_MinOf2\_put
  - RainbowOption, 173
- PriceByClosedForm\_WorseOf2
  - RainbowOption, 173
- PriceByMc\_2AssetsBasketMax
  - RainbowOption, 173
- PriceByMc\_2SpreadOptionMax
  - RainbowOption, 173
- PriceByMc\_BestOf2AssetsCash
  - RainbowOption, 174
- PriceByMc\_BetterOf2Assets
  - RainbowOption, 174
- PriceByMc\_Max2AssetsCall
  - RainbowOption, 174
- PriceByMc\_Max2AssetsPut
  - RainbowOption, 174
- PriceByMc\_Min2AssetsCall
  - RainbowOption, 175
- PriceByMc\_Min2AssetsPut
  - RainbowOption, 175
- PriceByMc\_WorseOf2Assets
  - RainbowOption, 175
- PriceByMc\_WorstOf2AssetsCash
  - RainbowOption, 175
- priceType
  - rainbowoption.h, 325
- productsCreation.cpp, 310
- productsCreation.cpp
  - choiceToType, 311
  - choosePricingType, 311
  - chooseRainbowType, 311
  - inputBond, 311
  - inputBSOption, 312
  - inputButterflySpread, 312
  - inputCallSpread, 312
  - inputConvertibleBond, 312
  - inputExoticOptionOnSingleAsset, 312
  - inputOptionStrategy, 313
  - inputPutSpread, 313
  - inputRainbowOption, 313
  - inputRatioCallSpread, 313
  - inputSpecificOptionStrategy, 314
  - inputStraddle, 314
  - inputStrangle, 314
  - inputVanillaSwap, 314
  - outputCallPut, 314
  - productsCreationMenu, 315
- productsCreation.h, 316
- productsCreation.h
  - choiceToType, 317
  - choosePricingType, 317
  - chooseRainbowType, 317
  - inputBond, 318
  - inputBSOption, 318
  - inputButterflySpread, 318
  - inputCallSpread, 318
  - inputConvertibleBond, 318
  - inputExoticOptionOnSingleAsset, 319
  - inputOptionStrategy, 319
  - inputPutSpread, 319
  - inputRainbowOption, 319
  - inputRatioCallSpread, 320
  - inputSpecificOptionStrategy, 320
  - inputStraddle, 320
  - inputStrangle, 320
  - inputVanillaSwap, 320
  - outputCallPut, 321
  - productsCreationMenu, 321
- productsCreationMenu
  - productsCreation.cpp, 315
  - productsCreation.h, 321
- Put
  - BlackScholes.h, 241
  - PayOff, 154

- q  
 ParkMiller.h, 304  
 Quarterly  
 date.h, 257  
 quotedPrice  
 bond, 36  
 riskybond, 193
- r  
 ParkMiller.h, 304  
 Rainbow2AssetsBasketMax  
 PayOff, 154  
 Rainbow2SpreadOptionMax  
 PayOff, 154  
 RainbowBestOf2AssetsCash  
 PayOff, 155  
 RainbowMax2AssetsCall  
 PayOff, 155  
 RainbowMax2AssetsPut  
 PayOff, 155  
 RainbowMin2AssetsCall  
 PayOff, 155  
 RainbowMin2AssetsPut  
 PayOff, 156  
 RainbowOption, 165  
 RainbowOption, 167  
 RainbowOption  
 \_CorrelationMatrix, 176  
 \_DFTomaturity, 176  
 \_Drifts, 176  
 \_MCEngine, 177  
 \_Multiplier, 177  
 \_NumberOfAssets, 177  
 \_Strike, 178  
 \_TerminalPoints, 178  
 \_expiryInYears, 176  
 \_gaussianSample, 176  
 \_outputMsgs, 177  
 \_pHazardRateProcesses, 177  
 \_pRandom, 177  
 \_seed, 178  
 \_spots, 178  
 \_startDate, 178  
 \_thePayOff, 178  
 \_type, 179  
 \_volatilities, 179  
 \_volatilitiesSurfaces, 179  
 \_weights, 179  
 \_yc, 179  
 ~RainbowOption, 168  
 A, 179  
 B, 179  
 C, 179  
 compute\_A, 168  
 compute\_B, 168  
 compute\_C, 168  
 compute\_ClosedFormsParameters, 168  
 compute\_d1, 169  
 compute\_d2, 169  
 compute\_d3, 169  
 compute\_d4, 169  
 compute\_rho1, 169  
 compute\_rho2, 169  
 compute\_sigmaA, 169  
 d1, 180  
 d2, 180  
 d3, 180  
 d4, 180  
 getCorrelRisk, 170  
 getDelta, 170  
 getGamma, 170  
 getPartialDelta, 170  
 getPartialGamma, 170  
 getPartialVega, 171  
 getPrice, 171  
 getRainbowType, 171  
 getRho, 171  
 getTheta, 171  
 getVega, 172  
 haveClosedFormVariablesBeenCom-  
 puted, 180  
 instantiateMCVariables, 172  
 PriceByClosedForm\_BestOf2\_-  
 plusCash, 172  
 PriceByClosedForm\_BetterOf2, 172  
 PriceByClosedForm\_MaxOf2\_call, 172  
 PriceByClosedForm\_MaxOf2\_put, 173  
 PriceByClosedForm\_MinOf2\_call, 173  
 PriceByClosedForm\_MinOf2\_put, 173  
 PriceByClosedForm\_WorseOf2, 173  
 PriceByMc\_2AssetsBasketMax, 173  
 PriceByMc\_2SpreadOptionMax, 173  
 PriceByMc\_BestOf2AssetsCash, 174  
 PriceByMc\_BetterOf2Assets, 174  
 PriceByMc\_Max2AssetsCall, 174  
 PriceByMc\_Max2AssetsPut, 174  
 PriceByMc\_Min2AssetsCall, 175  
 PriceByMc\_Min2AssetsPut, 175  
 PriceByMc\_WorseOf2Assets, 175  
 PriceByMc\_WorstOf2AssetsCash, 175  
 RainbowOption, 167  
 reassignVolsAtThemoney, 175  
 reassignVolsAtThestrike, 175  
 rho1, 180  
 rho2, 180  
 setRainbowType, 176  
 sigmaA, 180  
 rainbowoption.cpp, 322



- rainbowoption.h, 323
  - AssetsBasketMax, 325
  - BestOf2AssetsCash, 325
  - BetterOf2Assets, 325
  - ClosedForm, 325
  - EPSILON, 324
  - GREEKAPPROX, 324
  - Max2AssetsCall, 325
  - Max2AssetsPut, 325
  - Min2AssetsCall, 325
  - Min2AssetsPut, 325
  - MonteCarlo, 325
  - priceType, 325
  - rainbowType, 325
  - RO\_DEFAULT\_MATURITY, 324
  - RO\_DEFAULT\_MULTIPLIER, 324
  - RO\_DEFAULT\_NB\_ASSETS, 324
  - RO\_DEFAULT\_RATE, 324
  - RO\_DEFAULT\_STRIKE, 324
  - RO\_DEFAULT\_VOL, 324
  - RO\_NPATHS, 324
  - RO\_SEED, 325
  - SpreadOptionMax, 325
  - WorseOf2Assets, 325
  - WorstOf2AssetsCash, 325
- rainbowType
  - rainbowoption.h, 325
- RainbowWorstOf2AssetsCash
  - PayOff, 156
- RandC, 182
  - RandC, 182
- RandC
  - ~RandC, 182
  - GetOneRandomInteger, 183
  - getUniform, 183
  - Max, 183
  - Min, 183
  - RandC, 182
  - Seed, 184
  - SetSeed, 183
- RandC.cpp, 326
- RandC.cpp
  - Maxim, 326
- RandC.h, 327
- Random, 185
  - clone, 186
  - Dimensionality, 187
  - GetDimensionality, 186
  - GetGaussian, 186
  - GetGaussians, 186
  - GetUniform, 186
  - GetUniforms, 187
  - InitialSeed, 187
  - InnerGenerator, 188
  - Random, 185
  - Reciprocal, 188
  - Reset, 187
  - ResetDimensionality, 187
  - SetSeed, 187
  - Skip, 187
- Random.cpp, 328
- Random.h, 329
- RandomGenerator, 189
  - RandomGenerator, 189
- RandomGenerator
  - ~RandomGenerator, 189
  - GetOneRandomInteger, 190
  - getUniform, 190
  - Max, 190
  - Min, 190
  - RandomGenerator, 189
  - Seed, 191
  - SetSeed, 190
- RandomGenerator.cpp, 330
- RandomGenerator.h, 331
- Read
  - Matrix, 122
- Real
  - types.h, 348
- realsEqual
  - utils.cpp, 355
  - utils.h, 358
- reassignVolsAtThemoney
  - RainbowOption, 175
- reassignVolsAtThestrike
  - RainbowOption, 175
- recalcInformation
  - BlackScholes, 31
- recalcPrice
  - OptionStrategy, 145
- Reciprocal
  - Random, 188
- REF
  - Matrix, 122
- Relative
  - creditCurve.h, 251
- remainingString
  - StringTokenizer, 199
- RemoveColumn
  - Matrix, 122
- RemoveRow
  - Matrix, 122
- resampleSpread
  - creditCurve, 61
- Reset
  - Random, 187
- ResetDimensionality
  - Random, 187

- returnAmounts
  - SwapLeg, 202
- returnDateConvention
  - Date, 79
- returnDates
  - SwapLeg, 202
- returnNbOptions
  - OptionStrategy, 145
- returnOption
  - OptionStrategy, 146
- returnOptionQuantity
  - OptionStrategy, 146
- returnPrice
  - OptionStrategy, 146
  - VanillaSwap, 208
- returnSensibilityToRate
  - Portfolio, 161
- returnSensibilityToTime
  - Portfolio, 161
- returnSensibilityToVol
  - Portfolio, 162
- returnSize
  - SwapLeg, 202
- RevLookbackCall
  - Exotics.h, 262
  - PayOff, 156
- RevLookbackPut
  - Exotics.h, 262
  - PayOff, 156
- rho
  - convertiblebond, 48
  - riskybond, 193
  - treasurybond, 204
- rho1
  - RainbowOption, 180
- rho2
  - RainbowOption, 180
- RightAppendIdentity
  - Matrix, 122
- riskybond, 192
  - \_cc, 194
  - ~riskybond, 193
  - quotedPrice, 193
  - rho, 193
  - riskybond, 192, 193
  - shiftedbond, 194
- riskyDiscountFactor
  - creditCurve, 61
- RO\_DEFAULT\_MATURITY
  - rainbowoption.h, 324
- RO\_DEFAULT\_MULTIPLIER
  - rainbowoption.h, 324
- RO\_DEFAULT\_NB\_ASSETS
  - rainbowoption.h, 324
- RO\_DEFAULT\_RATE
  - rainbowoption.h, 324
- RO\_DEFAULT\_STRIKE
  - rainbowoption.h, 324
- RO\_DEFAULT\_VOL
  - rainbowoption.h, 324
- RO\_NPATHS
  - rainbowoption.h, 324
- RO\_SEED
  - rainbowoption.h, 325
- rotateZCBRateCurve
  - yieldCurve, 227
- RREF
  - Matrix, 122
- RunEngineAsianCall
  - MCEngine, 130
- RunEngineAsianPut
  - MCEngine, 130
- RunEngineBarrierCall
  - MCEngine, 131
- RunEngineBarrierPut
  - MCEngine, 131
- RunEngineCall
  - MCEngine, 131
- runEngineCall
  - binomialTree, 23
- RunEngineCappedCliquet
  - MCEngine, 131
- runEngineConvertibleBond
  - binomialTree, 23
- RunEngineFlooredCliquet
  - MCEngine, 132
- RunEngineGeneral
  - MCEngine, 132
- RunEnginePut
  - MCEngine, 132
- RunEngineRainbow2AssetsBasketMax
  - MCEngine, 132
- RunEngineRainbow2SpreadOptionMax
  - MCEngine, 133
- RunEngineRainbowBestOf2AssetsCash
  - MCEngine, 133
- RunEngineRainbowMax2AssetsCall
  - MCEngine, 133
- RunEngineRainbowMax2AssetsPut
  - MCEngine, 133
- RunEngineRainbowMin2AssetsCall
  - MCEngine, 134
- RunEngineRainbowMin2AssetsPut
  - MCEngine, 134
- RunEngineRainbowWorstOf2AssetsCash
  - MCEngine, 134
- RunEngineRevLookbackCall
  - MCEngine, 134

- RunEngineRevLookbackPut
  - MCEngine, 135
- runInterface
  - importData, 102
- runUserDefinedInterface
  - importData, 102
- Saturday
  - date.h, 258
- Seed
  - ParkMiller, 149
  - RandC, 184
  - RandomGenerator, 191
  - Sobol, 197
- seed
  - MersenneTwister, 139
- Semiannual
  - date.h, 257
- September
  - date.h, 258
- SequentDiscountFactorsByInvertSwapMatrix
  - yieldCurve, 227
- serialNumber
  - Date, 80
- setAmount
  - flowSchedule, 94
- setBusDayConv
  - flowSchedule, 94
- setCcy
  - asset, 17
- setClaimVariables
  - binomialTree, 23, 24
- setdatadir
  - FileReader, 92
- setDate
  - flowSchedule, 95
- setDateToToday
  - Date, 80
- setDayCount
  - yieldPoint, 232
- setDivAsRate
  - asset, 17
- setDivFlows
  - asset, 17
- setMarketData
  - importData, 102
- setMaturity
  - CreditSpreadPoint, 66
  - yieldPoint, 232
- setPrice
  - asset, 18
- setRainbowType
  - RainbowOption, 176
- setRate
  - CreditSpreadPoint, 66
  - yieldPoint, 232
- SetSeed
  - MersenneTwister, 138
  - ParkMiller, 149
  - RandC, 183
  - Random, 187
  - RandomGenerator, 190
  - Sobol, 196
- SetStrike
  - PayOff, 156
- SetSubMatrix
  - Matrix, 122
- setType
  - CreditSpreadPoint, 66
  - yieldPoint, 232
- SetValue
  - Matrix, 122
- setVolatility
  - asset, 18
- setvolsurface
  - volsurface, 216
- setYieldCurve
  - asset, 18
- shiftedbond
  - riskybond, 194
  - treasurybond, 204
- shiftedcbond
  - convertiblebond, 48
- shiftedvolsurface
  - volsurface, 216
- shiftedYCvolsurface
  - volsurface, 216
- shiftZCBRateCurve
  - yieldCurve, 227
- ShortInteger
  - types.h, 350
- ShortNatural
  - types.h, 350
- sigmaA
  - RainbowOption, 180
- sign
  - utils.cpp, 355
  - utils.h, 358
- Skip
  - Random, 187
- SkipSpaces
  - CSVParser, 69
- Sobol, 195
  - ~Sobol, 195
  - GetOneRandomInteger, 196
  - getUniform, 196
  - Max, 196
  - Min, 196

- n\_, 197
- Seed, 197
- SetSeed, 196
- Sobol, 195
- sobseq, 196
- x\_, 197
- Sobol.cpp, 332
- Sobol.h, 333
  - MAXBIT, 333
- sobseq
  - Sobol, 196
- SortAscend
  - Matrix, 123
- sortCashSwap
  - yieldCurve, 228
- SortDescend
  - Matrix, 123
- sortMarketRatesByMaturity
  - yieldCurve, 228
- SpliceInColumn
  - Matrix, 123
- SpliceInRow
  - Matrix, 123
- spotRate
  - creditCurve, 61
  - yieldCurve, 228
- SpreadOptionMax
  - rainbowoption.h, 325
- std, 11
- Strike
  - PayOff, 157
- strikes
  - volsurfaceparams, 219
- StringTokenizer, 198
  - StringTokenizer, 198
- StringTokenizer
  - ~StringTokenizer, 198
  - countTokens, 198
  - delim, 199
  - filterNextToken, 198
  - hasMoreTokens, 198
  - nextFloatToken, 199
  - nextIntToken, 199
  - nextToken, 199
  - remainingString, 199
  - StringTokenizer, 198
  - token\_str, 199
- StringTokenizer.cpp, 334
- StringTokenizer.h, 335
- SubFunctionForBivariateNormal
  - Normals.cpp, 298
  - Normals.h, 300
- SumAll
  - Matrix, 123
- SumAllSquared
  - Matrix, 123
- SumColumn
  - Matrix, 124
- SumColumnSquared
  - Matrix, 124
- SumRow
  - Matrix, 124
- SumRowSquared
  - Matrix, 124
- Sunday
  - date.h, 258
- survivalProbability
  - creditCurve, 62
- Swap
  - yieldCurve.h, 368
- SwapCols
  - Matrix, 124
- swapFees
  - creditCurve, 62
- SwapLeg, 201
  - SwapLeg, 201
- SwapLeg
  - \_dateSchedule, 202
  - \_flowSchedule, 202
  - ~SwapLeg, 201
  - returnAmounts, 202
  - returnDates, 202
  - returnSize, 202
  - SwapLeg, 201
- SwapLeg.cpp, 336
- SwapLeg.h, 337
- SwapRows
  - Matrix, 124
- testObjects.cpp, 338
- testObjects.cpp
  - maintests, 338
- testObjects.h, 339
- testObjects.h
  - fr\_basic, 340
  - mainasset, 340
  - mainbinomialtree, 340
  - mainbond, 341
  - mainconvertiblebond, 341
  - maincreditcurve, 341
  - maindate, 341
  - mainfilereader, 342
  - maininterpolator, 342
  - mainIRVanillaSwap, 342
  - mainmatrix, 342
  - mainmc, 343
  - mainmontecarlo, 343
  - mainoption, 343

- mainoptionstrategy, 343
- mainrainbowoptions, 344
- mainvarianceswap, 344
- mainvolsurface, 344
- mainyieldcurve, 344
- Thursday
  - date.h, 258
- timeOfCurrentSpread
  - creditCurve, 62
- timeOfPreviousSpread
  - creditCurve, 62
- TimeUnit
  - date.h, 258
- TN\_INFINITY
  - types.h, 346
- TN\_INTEGER
  - types.h, 346
- TN\_LONG\_INTEGER
  - types.h, 346
- TN\_REAL
  - types.h, 346
- token\_str
  - StringTokenizer, 199
- toString
  - Date, 80
- transform1DvalarrayToColumnMatrix
  - utils.cpp, 355
  - utils.h, 358
- transform2DvalarrayToMatrix
  - utils.cpp, 355
  - utils.h, 359
- transformColumnMatrixTo1Dvalarray
  - utils.cpp, 355
  - utils.h, 359
- transformMatrixTo2Dvalarray
  - utils.cpp, 356
  - utils.h, 359
- Transpose
  - Matrix, 124
- treasurybond, 203
  - ~treasurybond, 204
  - rho, 204
  - shiftedbond, 204
  - treasurybond, 203, 204
- Tuesday
  - date.h, 258
- TypeAsString
  - CreditSpreadPoint, 67
  - yieldPoint, 232
- TypeOfRate
  - yieldCurve.h, 368
- TypeOptionBS
  - BlackScholes.h, 241
- types.h, 346
- CAD, 351
- Currency, 351
- EUR, 351
- Integer, 347
- LongInteger, 347
- LongNatural, 347
- Natural, 348
- Real, 348
- ShortInteger, 350
- ShortNatural, 350
- TN\_INFINITY, 346
- TN\_INTEGER, 346
- TN\_LONG\_INTEGER, 346
- TN\_REAL, 346
- USD, 351
- VeryLongInteger, 351
- VeryLongNatural, 351
- Unadjusted
  - date.h, 257
- UPPER\_MASK
  - MersenneTwister.cpp, 295
- USD
  - types.h, 351
- UsDate, 206
- UsDate
  - isBusinessDay, 206
- UsDate.cpp, 352
- UsDate.h, 353
- utils.cpp, 354
  - ErrorMsg, 354
  - getint, 354
  - mergeunique, 354
  - realsEqual, 355
  - sign, 355
  - transform1DvalarrayToColumnMatrix, 355
  - transform2DvalarrayToMatrix, 355
  - transformColumnMatrixTo1Dvalarray, 355
  - transformMatrixTo2Dvalarray, 356
  - valarrayRealToString, 356
- utils.h, 357
  - DEFAULT\_PRECISION, 357
  - ErrorMsg, 358
  - getint, 358
  - mergeunique, 358
  - realsEqual, 358
  - sign, 358
  - transform1DvalarrayToColumnMatrix, 358
  - transform2DvalarrayToMatrix, 359
  - transformColumnMatrixTo1Dvalarray, 359

- transformMatrixTo2Dvalarray, 359
- valarrayRealToString, 359
- valarrayRealToString
  - utils.cpp, 356
  - utils.h, 359
- value
  - cachedval, 39
- VanillaSwap, 207
  - VanillaSwap, 207
- VanillaSwap
  - \_curve, 208
  - \_leg1, 208
  - \_leg2, 209
  - \_name1, 209
  - \_name2, 209
  - ~VanillaSwap, 207
  - getFairValue1, 208
  - getFairValue2, 208
  - getRho, 208
  - getTheta, 208
  - returnPrice, 208
  - VanillaSwap, 207
- VanillaSwap.cpp, 360
- VanillaSwap.h, 361
- VanillaSwap.h
  - defaultprecisionInPrice, 361
  - MAX\_LETTERS, 361
- variance
  - volsurface, 216
- VarianceSwap, 210
  - VarianceSwap, 210
- VarianceSwap
  - \_forward, 211
  - \_maturity, 211
  - \_options, 211
  - ~VarianceSwap, 210
  - getPrice, 211
  - getRho, 211
  - getTheta, 211
  - getVega, 211
  - VarianceSwap, 210
- VarianceSwap.cpp, 362
- VarianceSwap.h, 363
- vDates
  - Drift, 84
- vDrift
  - Drift, 84
- VeryLongInteger
  - types.h, 351
- VeryLongNatural
  - types.h, 351
- volatility
  - volsurface, 216
- vols
  - marketData, 108
- volsurface, 213
  - \_callputprices, 217
  - \_constantvol, 217
  - \_impliedvolsurface, 217
  - \_interpolvolsurf, 217
  - \_iscallputprices, 217
  - \_maturities, 217
  - \_stockPrice, 217
  - \_strikes, 217
  - \_today, 217
  - \_volsurfconst, 218
  - \_yieldCurve, 218
  - ~volsurface, 215
  - forwardVolatility, 215
  - forwardvolsurface, 215
  - getvolsurface, 215
  - invertBSformula, 215
  - setvolsurface, 216
  - shiftedvolsurface, 216
  - shiftedYCVolsurface, 216
  - variance, 216
  - volatility, 216
  - volsurface, 214, 215
- volsurface.cpp, 364
- volsurface.h, 365
  - PI, 365
- volsurfaceparams, 219
  - callputprices, 219
  - iscallputprices, 219
  - maturities, 219
  - strikes, 219
- VSNAME
  - importData.h, 269
- Wednesday
  - date.h, 258
- Weekday
  - date.h, 258
- weekday
  - Date, 80
- Weeks
  - date.h, 258
- WorseOf2Assets
  - rainbowoption.h, 325
- WorstOf2AssetsCash
  - rainbowoption.h, 325
- Write
  - Matrix, 124
- x\_
  - Sobol, 197

- YC\_DEFAULT\_NUMER\_POINTS
  - yieldCurve.h, 367
- YC\_MAX\_NUMBER\_POINTS
  - yieldCurve.h, 367
- YC\_NAME\_STRLEN
  - yieldCurve.h, 367
- YCNNAME
  - importData.h, 269
- Year
  - date.h, 256
- year
  - Date, 80
- yearOffset
  - Date, 81
- Years
  - date.h, 258
- yieldCurve, 220
  - yieldCurve, 222
- yieldCurve
  - \_marketRates, 229
  - \_name, 229
  - \_zcbRates, 229
  - ~yieldCurve, 223
  - assignFlatRate, 223
  - assignZCBrateAtIndex, 223
  - computeZCBratesBootstrap, 223
  - discountFactor, 224
  - forwardDiscountFactor, 224
  - forwardRate, 224, 225
  - forwardZCBCurve, 225
  - getMaturitiesInTheMarketCurve, 225
  - getMaturitiesInTheZCBCurve, 225
  - getName, 226
  - getPointAtMaturity, 226
  - getSequentSwapRates, 226
  - getSwapRates, 226
  - operator!=, 226
  - operator<<, 229
  - operator==, 227
  - rotateZCBrateCurve, 227
  - SequentDiscountFactorsByInvertSwap-  
Matrix, 227
  - shiftZCBrateCurve, 227
  - sortCashSwap, 228
  - sortMarketRatesByMaturity, 228
  - spotRate, 228
  - yieldCurve, 222
- yieldcurve
  - marketData, 108
- yieldCurve.cpp, 366
- yieldCurve.cpp
  - operator<<, 366
- yieldCurve.h, 367
  - Cash, 368
  - Continuous, 368
  - Discrete, 368
  - Swap, 368
- yieldCurve.h
  - defaultshiftfactorForShortRate, 368
  - interestComposition, 368
  - TypeOfRate, 368
  - YC\_DEFAULT\_NUMER\_POINTS,  
367
  - YC\_MAX\_NUMBER\_POINTS, 367
  - YC\_NAME\_STRLEN, 367
- yieldPoint, 230
  - yieldPoint, 231
- yieldPoint
  - \_dayCount, 233
  - \_maturity, 233
  - \_rate, 233
  - \_type, 233
  - ~yieldPoint, 231
  - getDayCount, 231
  - getMaturity, 231
  - getRate, 231
  - getType, 232
  - setDayCount, 232
  - setMaturity, 232
  - setRate, 232
  - setType, 232
  - TypeAsString, 232
  - yieldPoint, 231
- yieldToMaturity
  - bond, 36