MonteCarloPricer

1.0

Generated by Doxygen 1.8.5

Wed Sep 27 2017 20:52:26

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 BlackScholesModel Class Reference

Modèle de Black Scholes.

```
#include <BlackScholesModel.hpp>
```

Public Member Functions

- BlackScholesModel (int, double, double, PnlVect ∗, PnlVect ∗, PnlVect ∗)

  valeurs initiales du sous-jacent
- ∼BlackScholesModel ()

  Destructeur de classe.
- void asset (PnlMat ∗path, double T, int nbTimeSteps, PnlRng ∗rng)

  Génère une trajectoire du modèle et la stocke dans path.
- void asset (PnlMat ∗path, double t, double T, int nbTimeSteps, PnlRng ∗rng, const PnlMat ∗past)

  Calcule une trajectoire du sous-jacent connaissant le passé jusqu' à la date t.
- void shiftAsset (PnlMat ∗shift_path, const PnlMat ∗path, int d, double h, double t, double timestep)

  Shift d'une trajectoire du sous-jacent.
- PnlMat ∗ simul_market (int H, double T, PnlRng ∗rng)

  Calcule une simulation de marché
- void simul_market (PnlMat ∗path, double T, int H, PnlRng ∗rng)

Data Fields

- int size_
- double r_

  nombre d'actifs du modèle
- double rho_

  taux d'intérêt
- PnlVect ∗ sigma_

  paramètre de corrélation
- PnlVect ∗ spot_

  vecteur de volatilités

Protected Attributes

- PnlVect ∗ trend_
- PnlMat ∗ cholesky

  Tendance du modèle.

Private Attributes

- PnlVect ∗ G_i

  Matrice de Cholesky.
- PnlVect ∗ L_d
- PnlVect ∗ tempRow
- PnlVect ∗ spots_t

4.1.1 Detailed Description

Modèle de Black Scholes.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BlackScholesModel::BlackScholesModel (int *size,* double *r,* double *rho,* PnlVect ∗ *sigma,* PnlVect ∗ *spot,* PnlVect ∗ *trend*)

valeurs initiales du sous-jacent

Création de la matrice de Cholesky

Création des vecteurs temporaires

4.1.2.2 BlackScholesModel::∼BlackScholesModel ( )

Destructeur de classe.

Suppression des vecteurs temporaires

4.1.3 Member Function Documentation

4.1.3.1 void BlackScholesModel::asset (PnlMat ∗ *path,* double *T,* int *nbTimeSteps,* PnlRng ∗ *rng*)

Génère une trajectoire du modèle et la stocke dans path.

Parameters

`out`	*path*	contient une trajectoire du modèle. C'est une matrice de taille (nbTimeSteps+1) x d
`in`	*T*	maturité
`in`	*nbTimeSteps*	nombre de dates de constatation

Simulation de la trajectoire

Referenced by MonteCarlo::delta(), and MonteCarlo::price().

4.1.3.2 void BlackScholesModel::asset (PnlMat ∗ *path,* double *t,* double *T,* int *nbTimeSteps,* PnlRng ∗ *rng,* const PnlMat ∗ *past*)

Calcule une trajectoire du sous-jacent connaissant le passé jusqu' à la date t.

Parameters

out	*path*	contient une trajectoire du sous-jacent donnée jusqu'à l'instant t par la matrice past
in	*t*	date jusqu'à laquelle on connait la trajectoire. t n'est pas forcément une date de discrétisation
in	*nbTimeSteps*	nombre de pas de constatation
in	*T*	date jusqu'à laquelle on simule la trajectoire
in	*past*	trajectoire réalisée jusqu'a la date t

Copie de la trajectoire passée dans la trajectoire totale

Sauvegarde des spots

Simulation de la trajectoire

4.1.3.3 void BlackScholesModel::shiftAsset (PnlMat ∗ *shift_path*, const PnlMat ∗ *path*, int *d*, double *h*, double *t*, double *timestep*)

Shift d'une trajectoire du sous-jacent.

Parameters

in	*path*	contient en input la trajectoire du sous-jacent
out	*shift_path*	contient la trajectoire path dont la composante d a été shiftée par (1+h) à partir de la date t.
in	*t*	date à partir de laquelle on shift
in	*h*	pas de différences finies
in	*d*	indice du sous-jacent à shifter
in	*timestep*	pas de constatation du sous-jacent

Referenced by MonteCarlo::delta().

4.1.3.4 PnlMat ∗ BlackScholesModel::simul_market (int *H*, double *T*, PnlRng ∗ *rng*)

Calcule une simulation de marché

Parameters

in	*H*	le nombre de dates
in	*T*	la maturité
in	*rng*	le générateur aléatoire

Simulation de la trajectoire

Referenced by main().

4.1.3.5 void BlackScholesModel::simul_market (PnlMat ∗ *path*, double *T*, int *H*, PnlRng ∗ *rng*)

4.1.4 Field Documentation

4.1.4.1 PnlMat∗ BlackScholesModel::cholesky [protected]

Tendance du modèle.

4.1.4.2 PnlVect∗ BlackScholesModel::G_i [private]

Matrice de Cholesky.

Déclaration ici afin de limiter les créations et suppressions à chaque appel

4.1.4.3 PnlVect∗ BlackScholesModel::L_d [private]

4.1.4.4 double BlackScholesModel::r_

nombre d'actifs du modèle

Referenced by MonteCarlo::delta(), MonteCarlo::hedgingPAndL(), and MonteCarlo::price().

4.1.4.5 double BlackScholesModel::rho_

taux d'intérêt

4.1.4.6 PnlVect∗ BlackScholesModel::sigma_

paramètre de corrélation

4.1.4.7 int BlackScholesModel::size_

Referenced by MonteCarlo::delta(), MonteCarlo::hedgingPAndL(), and MonteCarlo::MonteCarlo().

4.1.4.8 PnlVect∗ BlackScholesModel::spot_

vecteur de volatilités

4.1.4.9 PnlVect∗ BlackScholesModel::spots_t [private]

4.1.4.10 PnlVect∗ BlackScholesModel::tempRow [private]

4.1.4.11 PnlVect∗ BlackScholesModel::trend_ [protected]

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

- /user/2/.base/margueed/home/pricerMonteCarlo/src/BlackScholesModel.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/BlackScholesModel.cpp

4.2 comp Struct Reference

```
#include <parser.hpp>
```

Public Member Functions

- bool operator() (const std::string &lhs, const std::string &rhs) const

4.2.1 Member Function Documentation

4.2.1.1 bool comp::operator() (const std::string & *lhs*, const std::string & *rhs*) const [inline]

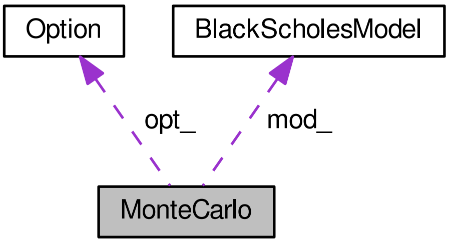
The documentation for this struct was generated from the following file:

- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.hpp

4.3 MonteCarlo Class Reference

```
#include <MonteCarlo.hpp>
```

Collaboration diagram for MonteCarlo:



Public Member Functions

- MonteCarlo (BlackScholesModel ∗mod, Option ∗opt, PnlRng ∗rng, double fdStep, int nbSamples)

    Constructeur de la classe.

- ∼MonteCarlo ()

    Destructeur de classe.

- void price (double &prix, double &ic)

    Calcule le prix de l'option à la date 0.

- void price (const PnlMat ∗past, double t, double &prix, double &ic)

    Calcule le prix de l'option à la date t.

- void delta (const PnlMat ∗past, double t, PnlVect ∗delta)

    Calcule le delta de l'option à la date t.

- double hedgingPAndL (PnlVect ∗result, PnlMat ∗path, int H)

    Calcule le profit&loss du portefeuille de couverture.

Data Fields

- BlackScholesModel ∗ mod_
- Option ∗ opt_
- PnlRng ∗ rng_
- double fdStep_
- int nbSamples_

Private Attributes

- PnlMat ∗ spotsMat
- PnlMat ∗ shift_path
- PnlMat ∗ path

4.3.1 Constructor & Destructor Documentation

4.3.1.1 MonteCarlo::MonteCarlo (BlackScholesModel ∗ *mod,* Option ∗ *opt,* PnlRng ∗ *rng,* double *fdStep,* int *nbSamples*)

Constructeur de la classe.

nombre de tirages Monte Carlo

Parameters

in	*modele*	Black Scholes
in	*l'option*	
in	*le*	générateur
in	*pas*	de différence finie
in	*nombre*	de tirages Monte Carlo

Initialisation des variables temporaires

References fdStep_, mod_, nbSamples_, Option::nbTimeSteps_, opt_, path, rng_, shift_path, BlackScholesModel::size_, and spotsMat.

4.3.1.2 MonteCarlo::∼MonteCarlo ( )

Destructeur de classe.

References path, shift_path, and spotsMat.

4.3.2 Member Function Documentation

4.3.2.1 void MonteCarlo::delta (const PnlMat ∗ *past,* double *t,* PnlVect ∗ *delta*)

Calcule le delta de l'option à la date t.

Parameters

in	*past*	contient la trajectoire du sous-jacent jusqu'à l'instant t
in	*t*	date à laquelle le calcul est fait
out	*delta*	contient le vecteur de delta de confiance sur le calcul du delta

References BlackScholesModel::asset(), fdStep_, mod_, nbSamples_, Option::nbTimeSteps_, opt_, path, Option::payoff(), BlackScholesModel::r_, rng_, shift_path, BlackScholesModel::shiftAsset(), BlackScholesModel::size_, and Option::T_.

Referenced by hedgingPAndL(), and main().

4.3.2.2 double MonteCarlo::hedgingPAndL (PnlVect ∗ *result,* PnlMat ∗ *path,* int *H*)

Calcule le profit&loss du portefeuille de couverture.

Parameters

out	*result*	contient les profit&loss du portefeuille au cours de la trajectoire
in	*path*	contient la trajectoire des sous-jacents

Returns

le profit&loss final du portefeuille de couverture

Initialisation

Calcul du prix initial de l'option

Calcul des deltas initiaux

Complétion de la matrice result

Free the memory

References delta(), mod_, Option::nbTimeSteps_, opt_, Option::payoff(), price(), BlackScholesModel::r_, BlackScholesModel::size_, and Option::T_.

Referenced by main().

4.3.2.3 void MonteCarlo::price (double & *prix*, double & *ic*)

Calcule le prix de l'option à la date 0.

Parameters

out	*prix*	valeur de l'estimateur Monte Carlo
out	*ic*	largeur de l'intervalle de confiance

Calcul du prix

Confidence interval in 95%

References BlackScholesModel::asset(), mod_, nbSamples_, Option::nbTimeSteps_, opt_, Option::payoff(), BlackScholesModel::r_, rng_, spotsMat, and Option::T_.

Referenced by hedgingPAndL(), and main().

4.3.2.4 void MonteCarlo::price (const PnlMat $*$ *past*, double *t*, double & *prix*, double & *ic*)

Calcule le prix de l'option à la date t.

Parameters

in	*past*	contient la trajectoire du sous-jacent jusqu'à l'instant t
in	*t*	date à laquelle le calcul est fait
out	*prix*	contient le prix
out	*ic*	contient la largeur de l'intervalle de confiance sur le calcul du prix

Price computing

Confidence interval in 95%

References BlackScholesModel::asset(), mod_, nbSamples_, Option::nbTimeSteps_, opt_, Option::payoff(), BlackScholesModel::r_, rng_, spotsMat, and Option::T_.

4.3.3 Field Documentation

4.3.3.1 double MonteCarlo::fdStep_

pointeur sur le générateur

Referenced by delta(), and MonteCarlo().

4.3.3.2 BlackScholesModel$*$ MonteCarlo::mod_

Referenced by delta(), hedgingPAndL(), MonteCarlo(), and price().

4.3.3.3 int MonteCarlo::nbSamples_

pas de différence finie

Referenced by delta(), MonteCarlo(), and price().

4.3.3.4 Option∗ MonteCarlo::opt_

pointeur vers le modèle

Referenced by delta(), hedgingPAndL(), MonteCarlo(), and price().

4.3.3.5 PnlMat∗ MonteCarlo::path [private]

Referenced by delta(), MonteCarlo(), and ∼MonteCarlo().

4.3.3.6 PnlRng∗ MonteCarlo::rng_

pointeur sur l'option

Referenced by delta(), MonteCarlo(), and price().

4.3.3.7 PnlMat∗ MonteCarlo::shift_path [private]

Referenced by delta(), MonteCarlo(), and ∼MonteCarlo().

4.3.3.8 PnlMat∗ MonteCarlo::spotsMat [private]

Referenced by MonteCarlo(), price(), and ∼MonteCarlo().

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

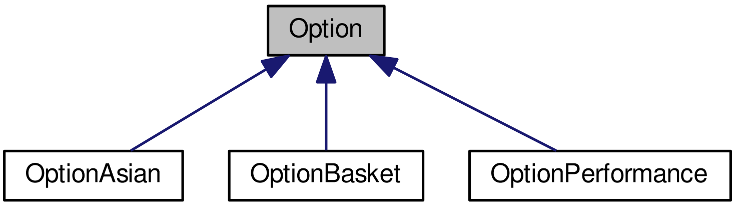
- /user/2/.base/margueed/home/pricerMonteCarlo/src/MonteCarlo.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/MonteCarlo.cpp

4.4 Option Class Reference

Classe Option abstraite.

`#include <Option.hpp>`

Inheritance diagram for Option:



Public Member Functions

- Option (double T_, int nbTimeSteps_, int size_, PnlVect ∗payoffCoeff_)

  coefficient permettant le calcul du payoff
- virtual ∼Option ()

  Destructeur de classe.
- virtual double payoff (const PnlMat ∗path)=0

  Calcule la valeur du payoff sur la trajectoire.
- double maturity ()
- int nbTimeSteps ()
- int size ()
- PnlVect payoffCoeff ()
- double payoffCoeff (int i)

Data Fields

- double T_
- int nbTimeSteps_

  maturité
- int size_

  nombre de pas de temps de discrétisation
- PnlVect ∗ payoffCoeff_

  *dimension du modèle, redondant avec BlackScholesModel::size_*

4.4.1 Detailed Description

Classe Option abstraite.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Option::Option (double *T_*, int *nbTimeSteps_*, int *size_*, PnlVect ∗ *payoffCoeff_*)

coefficient permettant le calcul du payoff

Constructeur de la classe

Parameters

in	*maturité*	
in	*nombre*	de pas de temps de discrétisation
in	*dimension*	du modèle, redondant avec BlackScholesModel::size_
in	*coefficient*	permettant le calcul du payoff

4.4.2.2 Option::∼Option ( ) [virtual]

Destructeur de classe.

References payoffCoeff_.

4.4.3 Member Function Documentation

4.4.3.1 double Option::maturity ( )

References T_.

4.4.3.2 int Option::nbTimeSteps ( )

References nbTimeSteps_.

4.4.3.3 virtual double Option::payoff (const PnlMat ∗ *path*) [pure virtual]

Calcule la valeur du payoff sur la trajectoire.

Parameters

in	*path*	est une matrice de taille (N+1) x d contenant une trajectoire du modèle telle que créée par la fonction asset.

Returns

phi(trajectoire)

Implemented in OptionAsian, OptionBasket, and OptionPerformance.

Referenced by MonteCarlo::delta(), MonteCarlo::hedgingPAndL(), and MonteCarlo::price().

4.4.3.4 PnlVect Option::payoffCoeff ( )

References payoffCoeff_.

4.4.3.5 double Option::payoffCoeff (int *i*)

References payoffCoeff_.

4.4.3.6 int Option::size ( )

References size_.

4.4.4 Field Documentation

4.4.4.1 int Option::nbTimeSteps_

maturité

Referenced by MonteCarlo::delta(), MonteCarlo::hedgingPAndL(), MonteCarlo::MonteCarlo(), nbTimeSteps(), OptionAsian::payoff(), OptionPerformance::payoff(), OptionBasket::payoff(), and MonteCarlo::price().

4.4.4.2 PnlVect∗ Option::payoffCoeff_

dimension du modèle, redondant avec BlackScholesModel::size_

Referenced by OptionAsian::payoff(), OptionPerformance::payoff(), OptionBasket::payoff(), payoffCoeff(), and ∼Option().

4.4.4.3 int Option::size_

nombre de pas de temps de discrétisation

Referenced by OptionAsian::payoff(), OptionPerformance::payoff(), OptionBasket::payoff(), and size().

4.4.4.4 double Option::T_

Referenced by MonteCarlo::delta(), MonteCarlo::hedgingPAndL(), maturity(), and MonteCarlo::price().

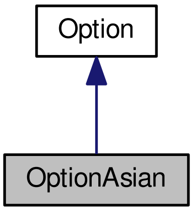
The documentation for this class was generated from the following files:

- /user/2/.base/margueed/home/pricerMonteCarlo/src/Option.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/Option.cpp

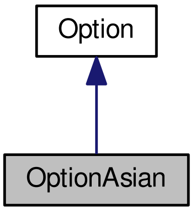
4.5 OptionAsian Class Reference

```
#include <OptionAsian.hpp>
```

Inheritance diagram for OptionAsian:



Collaboration diagram for OptionAsian:



Public Member Functions

- OptionAsian (double T_, int nbTimeSteps_, int size_, PnlVect ∗payoffCoeff_, double strike_)
- virtual double payoff (const PnlMat ∗path)

  Calcule la valeur du payoff sur la trajectoire.
- double strike ()

Protected Attributes

- double strike_

Additional Inherited Members

4.5.1 Constructor & Destructor Documentation

4.5.1.1 OptionAsian::OptionAsian (double *T_*, int *nbTimeSteps_*, int *size_*, PnlVect ∗ *payoffCoeff_*, double *strike_*)

4.5.2 Member Function Documentation

4.5.2.1 double OptionAsian::payoff (const PnlMat ∗ *path*) [virtual]

Calcule la valeur du payoff sur la trajectoire.

Parameters

in	*path*	est une matrice de taille (N+1) x d contenant une trajectoire du modèle telle que créée par la fonction asset.

Returns

phi(trajectoire)

Implements Option.

References Option::nbTimeSteps_, Option::payoffCoeff_, Option::size_, and strike_.

4.5.2.2 double OptionAsian::strike ( )

References strike_.

4.5.3 Field Documentation

4.5.3.1 double OptionAsian::strike_ [protected]

Referenced by payoff(), and strike().

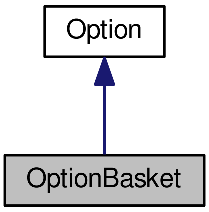
The documentation for this class was generated from the following files:

- /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionAsian.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionAsian.cpp

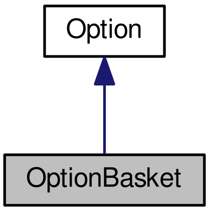
4.6 OptionBasket Class Reference

```
#include <OptionBasket.hpp>
```

Inheritance diagram for OptionBasket:



Collaboration diagram for OptionBasket:



Public Member Functions

- OptionBasket (double T_, int nbTimeSteps_, int size_, PnlVect ∗payoffCoeff_, double strike_)
- virtual double payoff (const PnlMat ∗path)

  Calcule la valeur du payoff sur la trajectoire.
- double strike ()

Protected Attributes

- double strike_

Additional Inherited Members

4.6.1 Constructor & Destructor Documentation

4.6.1.1 OptionBasket::OptionBasket (double *T_*, int *nbTimeSteps_*, int *size_*, PnlVect ∗ *payoffCoeff_*, double *strike_*)

4.6.2 Member Function Documentation

4.6.2.1 double OptionBasket::payoff (const PnlMat $*$ *path*) [virtual]

Calcule la valeur du payoff sur la trajectoire.

Parameters

in	*path*	est une matrice de taille (N+1) x d contenant une trajectoire du modèle telle que créée par la fonction asset.

Returns

phi(trajectoire)

Implements Option.

References Option::nbTimeSteps_, Option::payoffCoeff_, Option::size_, and strike_.

4.6.2.2 double OptionBasket::strike ( )

References strike_.

4.6.3 Field Documentation

4.6.3.1 double OptionBasket::strike_ [protected]

Referenced by payoff(), and strike().

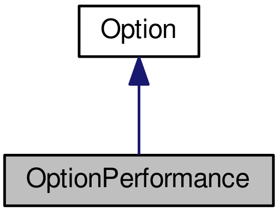
The documentation for this class was generated from the following files:

- /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionBasket.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionBasket.cpp

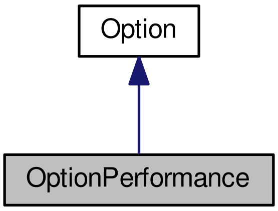
4.7 OptionPerformance Class Reference

```
#include <OptionPerformance.hpp>
```

Inheritance diagram for OptionPerformance:



Collaboration diagram for OptionPerformance:



Public Member Functions

- OptionPerformance (double T_, int nbTimeSteps_, int size_, PnlVect ∗payoffCoeff_)
- virtual double payoff (const PnlMat ∗path)

  Calcule la valeur du payoff sur la trajectoire.

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 OptionPerformance::OptionPerformance (double *T_*, int *nbTimeSteps_*, int *size_*, PnlVect ∗ *payoffCoeff_*)

4.7.2 Member Function Documentation

4.7.2.1 double OptionPerformance::payoff (const PnlMat ∗ *path*) [virtual]

Calcule la valeur du payoff sur la trajectoire.

Parameters

in	*path*	est une matrice de taille (N+1) x d contenant une trajectoire du modèle telle que créée par la fonction asset.

Returns

phi(trajectoire)

Implements Option.

References Option::nbTimeSteps_, Option::payoffCoeff_, and Option::size_.

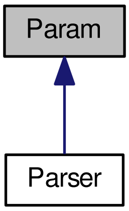
The documentation for this class was generated from the following files:

- /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionPerformance.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionPerformance.cpp

4.8 Param Class Reference

```
#include <parser.hpp>
```

Inheritance diagram for Param:



Public Member Functions

- Param ()
- Param (const Param &)
- ∼Param ()
- Param & operator= (const Param &P)
- template<typename T >
  bool extract (const std::string &key, T &out, bool go_on=true) const
- bool extract (const std::string &key, PnlVect ∗&out, int size, bool go_on=true) const

  Set out according to P.
- template<typename T >
  bool set (const std::string &key, const T &in)
- template<typename T >
  void insert (const std::string &key, const T_type &t, const T &in)

  Insert a new pair in the map or set M[key] to the new value if the key already exists in the map.
- void print () const

Data Fields

- Hash M

Private Member Functions

- bool check_if_key (Hash::const_iterator &it, const std::string &key) const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Friends

- class boost::serialization::access

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Param::Param ( ) [inline]

4.8.1.2 Param::Param (const Param & *P*)

References M.

4.8.1.3 Param::∼Param ( )

References M.

4.8.2 Member Function Documentation

4.8.2.1 bool Param::check_if_key (Hash::const_iterator & *it,* const std::string & *key*) const [private]

References M.

Referenced by extract().

4.8.2.2 template<typename T > bool Param::extract (const std::string & *key,* T & *out,* bool *go_on =* `true`) const [inline]

References check_if_key().

Referenced by main().

4.8.2.3 bool Param::extract (const std::string & *key,* PnlVect ∗& *out,* int *size,* bool *go_on =* `true`) const

Set out according to P.

Parameters

key	the key to be looked for in the map
out	(output) set to the value associated to key in the map
size	size of the vector to be stored
go_on	a boolean, if false and key is not found in the map, the abort function is called

References check_if_key().

4.8.2.4 template<typename T > void Param::insert (const std::string & *key,* const T_type & *t,* const T & *in*) [inline]

Insert a new pair in the map or set M[key] to the new value if the key already exists in the map.

Template Parameters

T	the template type of the element to be inserted

Parameters

key	the key
t	the type of the elements as an integer
in	the element itself

References M, TypeVal::type, and TypeVal::Val.

4.8.2.5 Param & Param::operator= (const Param & *P*)

References M.

4.8.2.6 void Param::print ( ) const [inline]

References M.

4.8.2.7 template<class Archive > void Param::serialize (Archive & *ar,* const unsigned int *version*) [inline], [private]

References M.

4.8.2.8 template<typename T > bool Param::set (const std::string & *key,* const T & *in*) [inline]

References M.

4.8.3 Friends And Related Function Documentation

4.8.3.1 friend class boost::serialization::access [friend]

4.8.4 Field Documentation

4.8.4.1 Hash Param::M

Referenced by Parser::add(), check_if_key(), insert(), operator=(), Param(), print(), serialize(), set(), and ∼Param().

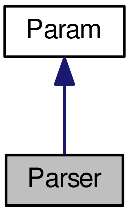
The documentation for this class was generated from the following files:

- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.cpp

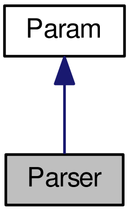
4.9 Parser Class Reference

```
#include <parser.hpp>
```

Inheritance diagram for Parser:



Collaboration diagram for Parser:



Public Member Functions

- Parser ()
- Parser (const char ∗InputFile)
- ∼Parser ()
- void add (char RedLine[ ])

Private Member Functions

- void ReadInputFile (const char ∗InputFile)

Private Attributes

- char type_ldelim
- char type_rdelim

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 Parser::Parser ( )

4.9.1.2 Parser::Parser (const char ∗ *InputFile*)

References ReadInputFile(), type_ldelim, and type_rdelim.

4.9.1.3 Parser::∼Parser ( )

4.9.2 Member Function Documentation

4.9.2.1 void Parser::add (char *RedLine[ ]*)

References charPtrTovector(), Param::M, T_DOUBLE, T_INT, T_LONG, T_STRING, T_VECTOR, TypeVal::type, type_ldelim, type_rdelim, and TypeVal::Val.

Referenced by ReadInputFile().

4.9.2.2 void Parser::ReadInputFile (const char ∗ *InputFile*) [private]

References add(), and MAX_CHAR_LINE.

Referenced by Parser().

4.9.3 Field Documentation

4.9.3.1 char Parser::type_ldelim [private]

Referenced by add(), and Parser().

4.9.3.2 char Parser::type_rdelim [private]

Referenced by add(), and Parser().

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

- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.cpp

4.10 TypeVal Class Reference

```
#include <parser.hpp>
```

Public Member Functions

- TypeVal ()
- TypeVal (const TypeVal &)
- ∼TypeVal ()
- TypeVal & operator= (const TypeVal &v)
- void print (const std::string &s) const

Data Fields

- T_type type
- boost::variant< int, size_t,
  double, std::vector< double >
  , std::string > Val

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Friends

- class boost::serialization::access

4.10.1 Constructor & Destructor Documentation

4.10.1.1 TypeVal::TypeVal ( )

4.10.1.2 TypeVal::TypeVal (const **TypeVal** & *v*)

References T_DOUBLE, T_INT, T_LONG, T_STRING, T_VECTOR, type, and Val.

4.10.1.3 TypeVal::∼TypeVal ( )

4.10.2 Member Function Documentation

4.10.2.1 **TypeVal** & TypeVal::operator= (const **TypeVal** & *v*)

References T_DOUBLE, T_INT, T_LONG, T_STRING, T_VECTOR, type, and Val.

4.10.2.2 void TypeVal::print (const std::string & *s*) const

References T_DOUBLE, T_INT, T_LONG, T_STRING, and T_VECTOR.

4.10.2.3 template<class Archive > void TypeVal::serialize (Archive & *ar,* const unsigned int *version*) `[inline]`, `[private]`

References type, and Val.

4.10.3 Friends And Related Function Documentation

4.10.3.1 friend class boost::serialization::access `[friend]`

4.10.4 Field Documentation

4.10.4.1 **T_type** TypeVal::type

Referenced by Parser::add(), Param::insert(), operator=(), serialize(), and TypeVal().

4.10.4.2 boost::variant<int, size_t, double, std::vector<double>, std::string> TypeVal::Val

Referenced by Parser::add(), Param::insert(), operator=(), serialize(), and TypeVal().

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

- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.hpp
- /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.cpp

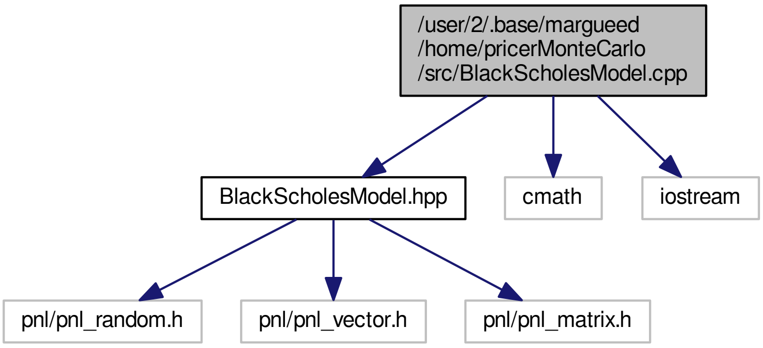
Chapter 5

File Documentation

5.1 /user/2/.base/margueed/home/pricerMonteCarlo/src/BlackScholesModel.cpp File Reference

```
#include "BlackScholesModel.hpp"
#include <cmath>
#include <iostream>
```

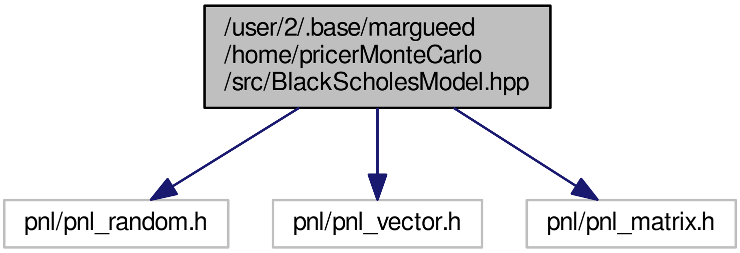
Include dependency graph for BlackScholesModel.cpp:



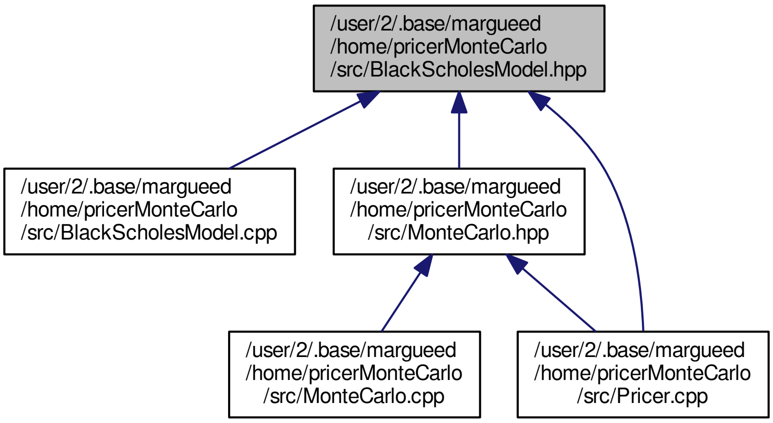
5.2 /user/2/.base/margueed/home/pricerMonteCarlo/src/BlackScholesModel.hpp File Reference

```
#include "pnl/pnl_random.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
```

Include dependency graph for BlackScholesModel.hpp:



This graph shows which files directly or indirectly include this file:



Data Structures

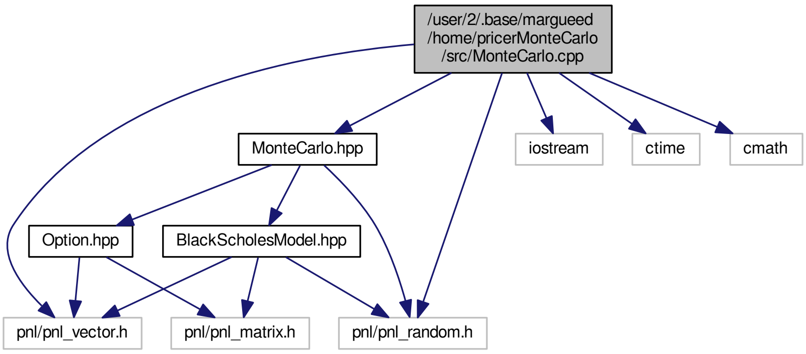
- class BlackScholesModel

  Modèle de Black Scholes.

5.3 /user/2/.base/margueed/home/pricerMonteCarlo/src/MonteCarlo.cpp File Reference

```
#include "MonteCarlo.hpp"
#include <iostream>
#include <ctime>
#include <cmath>
#include "pnl/pnl_random.h"
#include "pnl/pnl_vector.h"
```

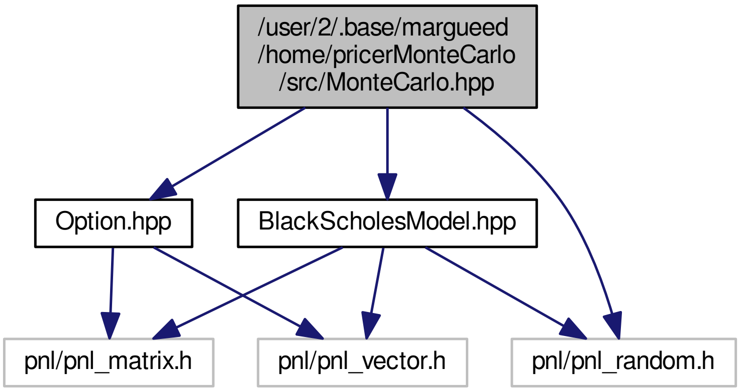
Include dependency graph for MonteCarlo.cpp:



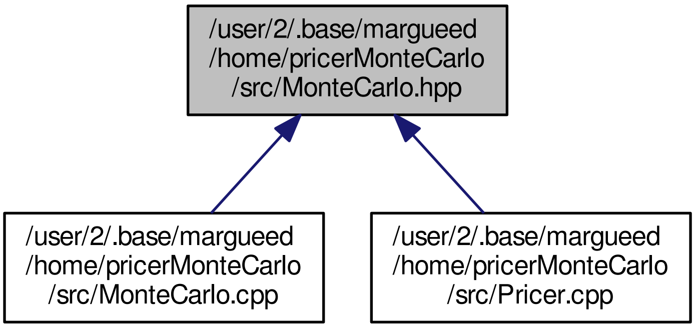
5.4 /user/2/.base/margueed/home/pricerMonteCarlo/src/MonteCarlo.hpp File Reference

```
#include "Option.hpp"
#include "BlackScholesModel.hpp"
#include "pnl/pnl_random.h"
```

Include dependency graph for MonteCarlo.hpp:



This graph shows which files directly or indirectly include this file:


/user/2/.base/margueed /home/pricerMonteCarlo /src/MonteCarlo.hpp
/user/2/.base/margueed /home/pricerMonteCarlo /src/MonteCarlo.cpp
/user/2/.base/margueed /home/pricerMonteCarlo /src/Pricer.cpp

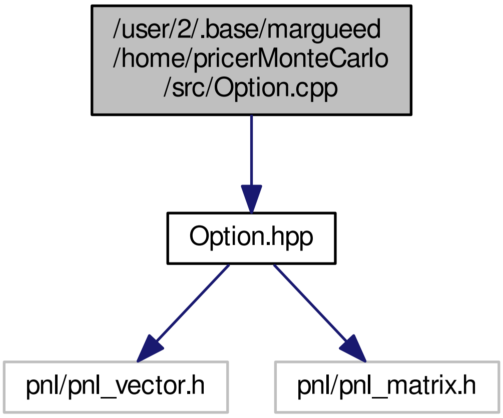
Data Structures

- class MonteCarlo

5.5 /user/2/.base/margueed/home/pricerMonteCarlo/src/Option.cpp File Reference

```
#include "Option.hpp"
```

Include dependency graph for Option.cpp:

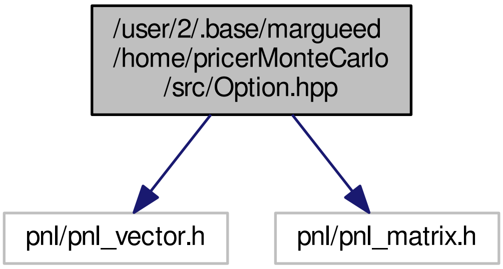

/user/2/.base/margueed /home/pricerMonteCarlo /src/Option.cpp
Option.hpp
pnl/pnl_vector.h
pnl/pnl_matrix.h

5.6 /user/2/.base/margueed/home/pricerMonteCarlo/src/Option.hpp File Reference

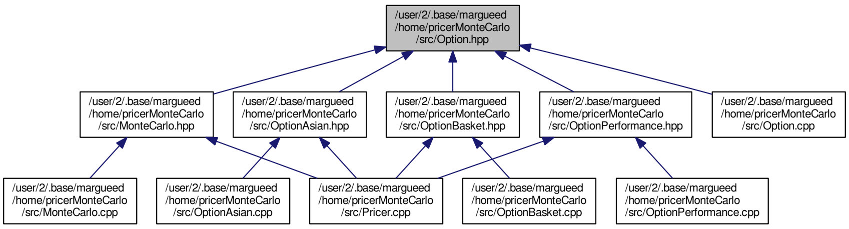
```
#include "pnl/pnl_vector.h"
```

```
#include "pnl/pnl_matrix.h"
```

Include dependency graph for Option.hpp:



This graph shows which files directly or indirectly include this file:



Data Structures

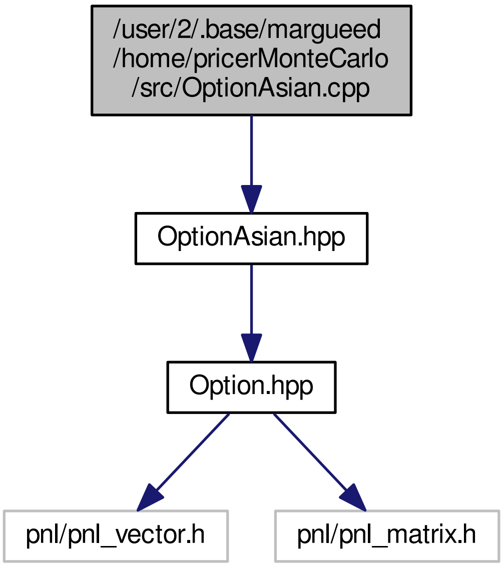
- class Option

  Classe Option abstraite.

5.7 /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionAsian.cpp File Reference

```
#include "OptionAsian.hpp"
```

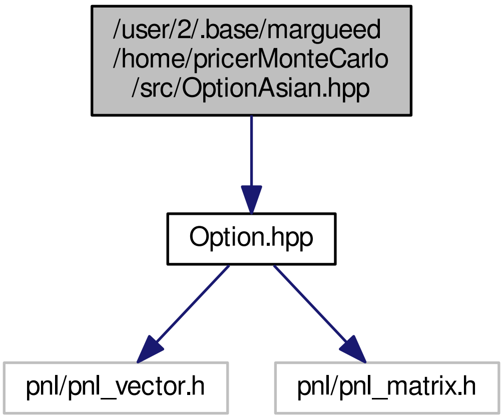
Include dependency graph for OptionAsian.cpp:



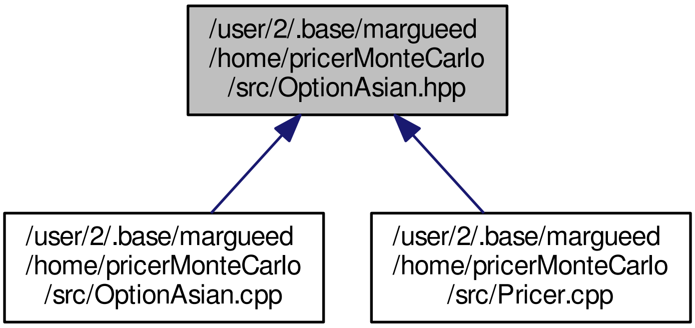
5.8 /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionAsian.hpp File Reference

```
#include "Option.hpp"
```

Include dependency graph for OptionAsian.hpp:



This graph shows which files directly or indirectly include this file:



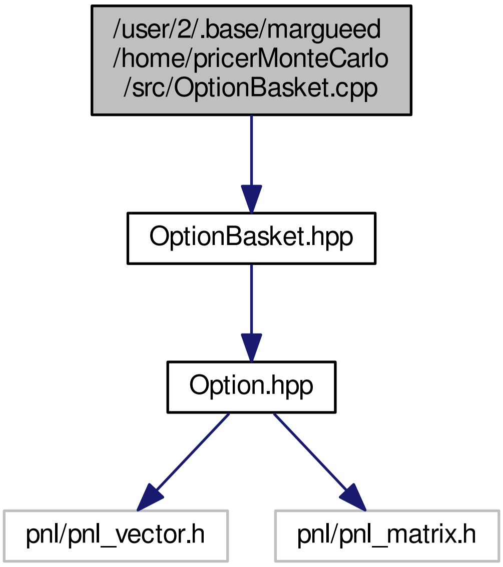
Data Structures

- class OptionAsian

5.9 /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionBasket.cpp File Reference

```
#include "OptionBasket.hpp"
```

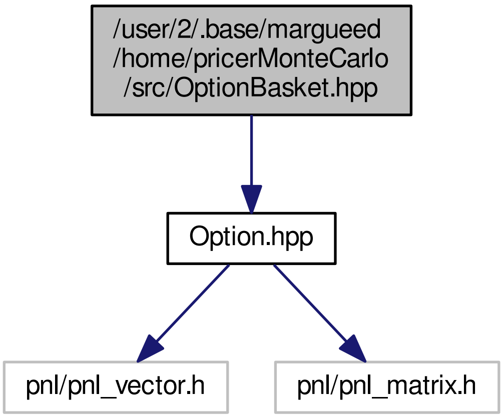
Include dependency graph for OptionBasket.cpp:



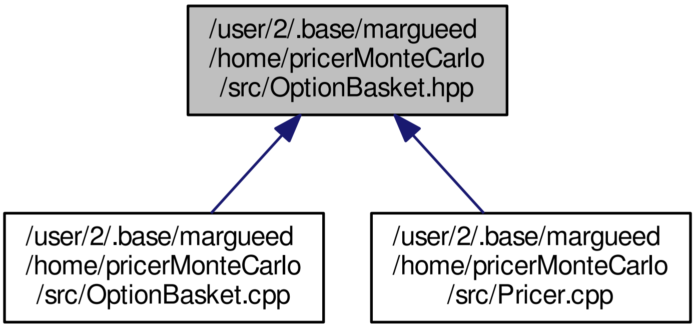
5.10 /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionBasket.hpp File Reference

```
#include "Option.hpp"
```

Include dependency graph for OptionBasket.hpp:


/user/2/.base/margueed /home/pricerMonteCarlo /src/OptionBasket.hpp
Option.hpp
pnl/pnl_vector.h
pnl/pnl_matrix.h

This graph shows which files directly or indirectly include this file:


/user/2/.base/margueed /home/pricerMonteCarlo /src/OptionBasket.hpp
/user/2/.base/margueed /home/pricerMonteCarlo /src/OptionBasket.cpp
/user/2/.base/margueed /home/pricerMonteCarlo /src/Pricer.cpp

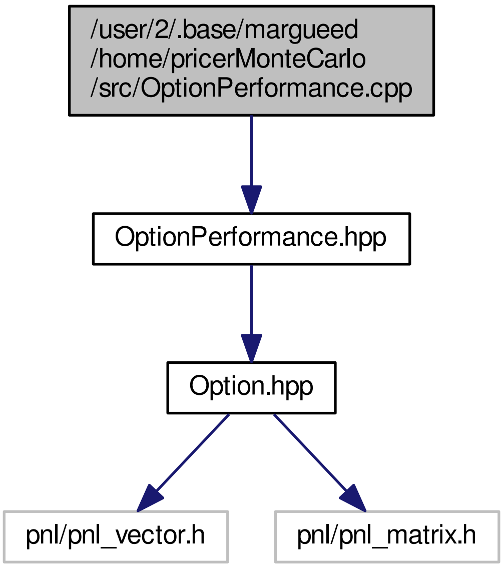
Data Structures

- class OptionBasket

5.11 /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionPerformance.cpp File Reference

```
#include "OptionPerformance.hpp"
```

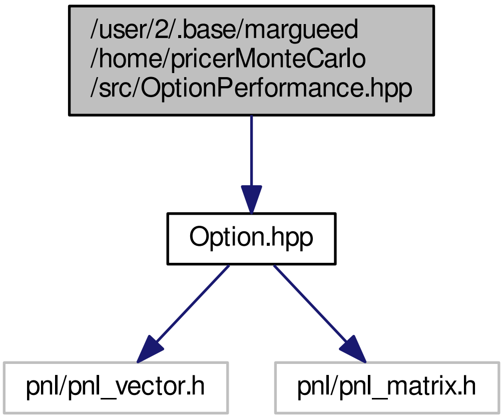
Include dependency graph for OptionPerformance.cpp:



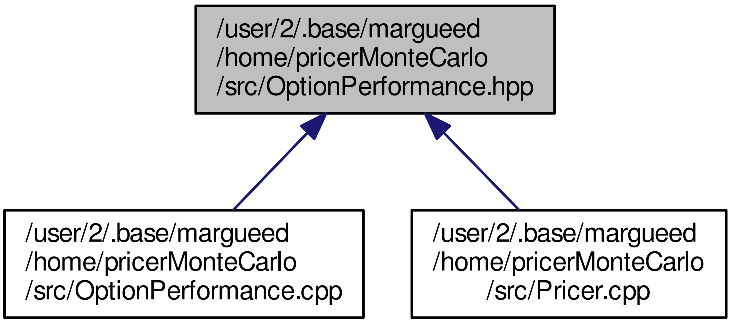
5.12 /user/2/.base/margueed/home/pricerMonteCarlo/src/OptionPerformance.hpp File Reference

```
#include "Option.hpp"
```

Include dependency graph for OptionPerformance.hpp:



This graph shows which files directly or indirectly include this file:



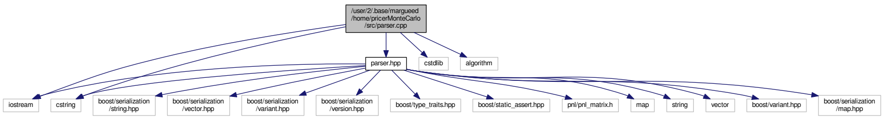
Data Structures

- class OptionPerformance

5.13 /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include "parser.hpp"
```

Include dependency graph for parser.cpp:



Functions

- template<typename T >
  std::ostream & operator<< (std::ostream &out, const std::vector< T > &v)
- static vector< double > charPtrTovector (const char ∗s)

5.13.1 Function Documentation

5.13.1.1 static vector<double> charPtrTovector (const char ∗ *s*) [static]

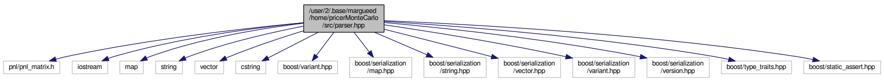
Referenced by Parser::add().

5.13.1.2 template<typename T > std::ostream& operator<< (std::ostream & *out,* const std::vector< T > & *v*)

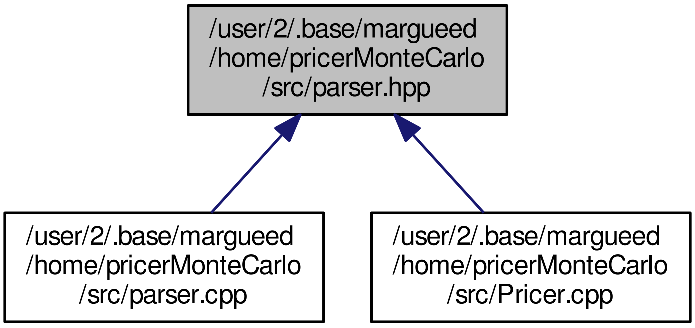
5.14 /user/2/.base/margueed/home/pricerMonteCarlo/src/parser.hpp File Reference

```
#include "pnl/pnl_matrix.h"
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include <cstring>
#include <boost/variant.hpp>
#include <boost/serialization/map.hpp>
#include <boost/serialization/string.hpp>
#include <boost/serialization/vector.hpp>
#include <boost/serialization/variant.hpp>
#include <boost/serialization/version.hpp>
#include <boost/type_traits.hpp>
#include <boost/static_assert.hpp>
```

Include dependency graph for parser.hpp:



This graph shows which files directly or indirectly include this file:


/user/2/.base/margueed /home/pricerMonteCarlo /src/parser.hpp
/user/2/.base/margueed /home/pricerMonteCarlo /src/parser.cpp
/user/2/.base/margueed /home/pricerMonteCarlo /src/Pricer.cpp

Data Structures

- class TypeVal
- struct comp
- class Param
- class Parser

Macros

- #define MAX_CHAR_LINE 1024

Typedefs

- typedef std::map< std::string, TypeVal, comp > Hash

Enumerations

- enum T_type {
  T_NULL, T_INT, T_LONG, T_DOUBLE,
  T_VECTOR, T_STRING }

5.14.1 Macro Definition Documentation

5.14.1.1 #define MAX_CHAR_LINE 1024

Referenced by Parser::ReadInputFile().

5.14.2 Typedef Documentation

5.14.2.1 typedef std::map<std::string, TypeVal, comp> Hash

5.14.3 Enumeration Type Documentation

5.14.3.1 enum T_type

Enumerator

T_NULL

T_INT

T_LONG

T_DOUBLE

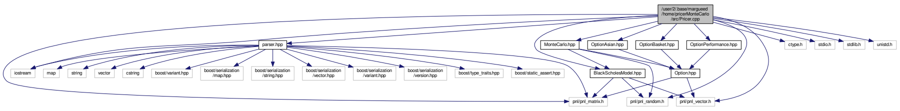
T_VECTOR

T_STRING

5.15 /user/2/.base/margueed/home/pricerMonteCarlo/src/Pricer.cpp File Reference

```
#include <iostream>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_random.h"
#include "pnl/pnl_vector.h"
#include "parser.hpp"
#include "BlackScholesModel.hpp"
#include "MonteCarlo.hpp"
#include "OptionAsian.hpp"
#include "OptionBasket.hpp"
#include "OptionPerformance.hpp"
```

Include dependency graph for Pricer.cpp:



Functions

- int main (int argc, char ∗∗argv)

5.15.1 Function Documentation

5.15.1.1 int main (int *argc,* char ∗∗ *argv*)

References MonteCarlo::delta(), Param::extract(), MonteCarlo::hedgingPAndL(), MonteCarlo::price(), and BlackScholesModel::simul_market().

Index