Projet PAP

Généré par Doxygen 1.9.0

Chapitre 1

README

Le fichier qui utilise l'ensemble des classes que nous avons crée pour construire le vecteur C(0,.) est main.cpp Nous avons donc crée un Makefile pour simplifier la compilation. Voici donc la procédure : 1) il faut taper make à la racine du dossier 2) aller dans le sous dossier bin et taper ./main

Chapitre 2

Index hiérarchique

2.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 3

Index des classes

3.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

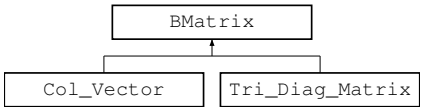
Chapitre 4

Documentation des classes

4.1 Référence de la classe BMatrix

`#include <bmatrix.h>`

Graphe d'héritage de BMatrix:



Fonctions membres publiques

- BMatrix (int nl, int nc)
- ~BMatrix ()
- int **get_nl** () const
- int **get_nc** () const
- double * **get_data** () const
- double operator() (int l, int c) const
- bool same_size (const BMatrix &m) const
- double & operator() (int l, int c)
- BMatrix & operator+= (const BMatrix &m)
- BMatrix & operator-= (const BMatrix &m)
- BMatrix & operator= (const BMatrix &m)

Attributs protégés

- int nl_
- int nc_
- double * data_

Amis

- std::ostream & operator<< (std::ostream &st, const BMatrix &m)
- BMatrix operator+ (const BMatrix &m1, const BMatrix &m2)
- BMatrix operator- (const BMatrix &m1, const BMatrix &m2)
- BMatrix operator* (const BMatrix &m1, const BMatrix &m2)
- BMatrix operator* (double val, const BMatrix &m)

4.1.1 Description détaillée

Classe des matrices de taille quelconque (les elements sont des doubles)

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 BMatrix()

```
BMatrix::BMatrix (
            int nl,
            int nc )
```

Constructeur des BMatrix. Initialise la matrice nulle de taille nl x nc. Throw "invalid size" si la taille est negative ou nulle. Throw "not enough memory" si l'allocation memoire ne s'est pas faite.

Paramètres

nl	(integer) : nombre de lignes
nc	(integer) : nombre de colonnes

4.1.2.2 ~BMatrix()

```
BMatrix::~BMatrix ( )
```

Destructeur des Constructeur des BMatrix. Libere la memoire allouee de la BMatrix.

4.1.3 Documentation des fonctions membres

4.1.3.1 operator()() [1/2]

```
double & BMatrix::operator() (
            int l,
            int c )
```

Operateur d'acces aux elements de la matrice. L'indexation se fait de 0 a nl - 1 pour les lignes et de 0 a nc - 1 pour les colonnes. Throw "wrong index" si l ou c est hors des dimensions.

Paramètres

l	(integer) : ligne d'acces
c	(integer) : colonne d'acces

4.1.3.2 operator()() [2/2]

```
double BMatrix::operator() (
            int l,
            int c ) const
```

Operateur d'acces constant aux elements de la matrice. L'indexation se fait de 0 a nl - 1 pour les lignes et de 0 a nc - 1 pour les colonnes. Throw "wrong index" si l ou c est hors des dimensions.

Paramètres

l	(integer) : ligne d'acces
c	(integer) : colonne d'acces

4.1.3.3 operator+=()

```
BMatrix & BMatrix::operator+= (
            const BMatrix & m )
```

Surcharge de l'operateur d'addition des matrices. Throw "size mismatched" si les tailles ne correspondent pas.

Paramètres

m	(BMatrix) : la matrice a ajoutee

4.1.3.4 operator-=()

```
BMatrix & BMatrix::operator-= (
            const BMatrix & m )
```

Surcharge de l'operateur de soustraction des matrices. Throw "size mismatched" si les tailles ne correspondent pas.

Paramètres

m	(BMatrix) : la matrice a soustraire

4.1.3.5 operator=()

```
BMatrix & BMatrix::operator= (
```

```
            const BMatrix & m )
```

Surcharge de l'operateur d'affectation des matrices.

Paramètres

m	(BMatrix) : la matrice a affectee

4.1.3.6 same_size()

```
bool BMatrix::same_size (
            const BMatrix & m ) const
```

Renvois un boolean determinant si la matrice m est de meme dimension que la matrice courante.

Paramètres

m	(BMatrix) : la matrice a comparee

4.1.4 Documentation des fonctions amies et associées

4.1.4.1 operator* [1/2]

```
BMatrix operator* (
            const BMatrix & m1,
            const BMatrix & m2 )  [friend]
```

Surcharge de l'operateur exterieur de multiplication des matrices. Renvois la matrice resultante de la multiplication. Throw "size mismatched" si les tailles ne correspondent pas.

Paramètres

m1	(BMatrix) : la matrice a gauche de la multiplication
m2	(BMatrix) : la matrice a droite de la multiplication

4.1.4.2 operator* [2/2]

```
BMatrix operator* (
            double val,
            const BMatrix & m )  [friend]
```

Surcharge de l'operateur exterieur de multiplication d'une matrices par un scalaire. Renvois la matrice resultante de la multiplication.

Paramètres

val	(double) : le scalaire de la mutiplication
m	(BMatrix) : la matrice de la multiplication

4.1.4.3 operator+

```
BMatrix operator+ (
            const BMatrix & m1,
            const BMatrix & m2 )  [friend]
```

Surcharge de l'operateur exterieur d'addition des matrices. Renvois la matrice resultante de l'addition. Throw "size mismatched" si les tailles ne correspondent pas.

Paramètres

m1	(BMatrix) : la 1ere matrice a additionner
m2	(BMatrix) : la 2em matrice a additionner

4.1.4.4 operator-

```
BMatrix operator- (
            const BMatrix & m1,
            const BMatrix & m2 )  [friend]
```

Surcharge de l'operateur exterieur de soustraction des matrices. Renvois la matrice resultante de la soustraction. Throw "size mismatched" si les tailles ne correspondent pas.

Paramètres

m1	(BMatrix) : la 1ere matrice a soustraire
m2	(BMatrix) : la 2em matrice a soustraire

4.1.4.5 operator<<

```
std::ostream& operator<< (
            std::ostream & st,
            const BMatrix & m )  [friend]
```

Surcharge de l'operateur de flux. Affichage de la matrice lignes par lignes.

4.1.5 Documentation des données membres

4.1.5.1 data_

```
double* BMatrix::data_  [protected]
```

Tableau de donnees

4.1.5.2 nc_

```
int BMatrix::nc_  [protected]
```

Nombre de colonnes

4.1.5.3 nl_

```
int BMatrix::nl_  [protected]
```

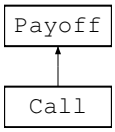
Nombre de lignes

La documentation de cette classe a été générée à partir du fichier suivant :
- include/bmatrix.h
- src/bmatrix.cpp

4.2 Référence de la classe Call

```
#include <call.h>
```

Graphe d'héritage de Call:



Fonctions membres publiques

- Call (int T, int L, int K, double r)
- Call (const Call &p)
- double get_C_T_s (double s) const
- double get_C_t_0 (double t) const
- double get_C_t_L (double t) const

Membres hérités additionnels

4.2.1 Description détaillée

Classe qui gere les Call

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 Call() [1/2]

```
Call::Call (
            int T,
            int L,
            int K,
            double r )
```

Constructeur de la classe Call.

Paramètres

T	(integer): Temps de maturite du call
L	(integer): Prix maximum de l'actif sous jacent
K	(integer): Strike de l'actif
r	(double): taux d'interet en vigueur

4.2.2.2 Call() [2/2]

```
Call::Call (
            const Call & p )
```

Constructeur de copie de la classe Call.

Paramètres

p	(const Call&) : call a copier

4.2.3 Documentation des fonctions membres

4.2.3.1 get_C_t_0()

```
double Call::get_C_t_0 (
            double t ) const  [virtual]
```

Getter de la condition au bord du prix de l'option (lorsque s = 0) Si t $<$ 0 alors le calcul se fait pour t = 0 si t $>$ T le calcul se fait pour t = T.

Paramètres

t	(double) : temps a laquel est evalue le prix de l'option.

Implémente Payoff.

4.2.3.2 get_C_t_L()

```
double Call::get_C_t_L (
            double t ) const  [virtual]
```

Getter de la condition au bord du prix de l'option (lorsque s = L) Si t $<$ 0 alors le calcul se fait pour t = 0 si t $>$ T le calcul se fait pour t = T.

Paramètres

t	(double) : temps a laquel est evalue le prix de l'option

Implémente Payoff.

4.2.3.3 get_C_T_s()

```
double Call::get_C_T_s (
            double s ) const  [virtual]
```

Getter de la condition a maturite du prix de l'option (lorsque t = T). Si s $<$ 0 alors le calcul se fait pour s = 0 si s $>$ L le calcul se fait pour s = L

Paramètres

s	(double): prix de l'actif sous jacent a l'instant T

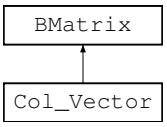
Implémente Payoff.

La documentation de cette classe a été générée à partir du fichier suivant :
- include/call.h
- src/call.cpp

4.3 Référence de la classe Col_Vector

`#include <col_vector.h>`

Graphe d'héritage de Col_Vector:



Fonctions membres publiques

- Col_Vector (int length)
- Col_Vector (const Col_Vector &m)
- int **get_length** () const
- double operator[ ] (int idx) const
- double & operator[ ] (int idx)
- Col_Vector & operator= (const Col_Vector &v)
- Col_Vector & operator+= (const Col_Vector &m)
- Col_Vector & operator-= (const Col_Vector &m)

Amis

- Col_Vector operator∗ (const BMatrix &m1, const Col_Vector &m2)
- Col_Vector operator+ (const Col_Vector &v1, const Col_Vector &v2)
- Col_Vector operator- (const Col_Vector &v1, const Col_Vector &v2)

Membres hérités additionnels

4.3.1 Description détaillée

Classe representant un vecteur colonne

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 Col_Vector() [1/2]

```
Col_Vector::Col_Vector (
            int length )
```

Constructeur des vecteurs colonne Col_Vector de taille length x 1.

Paramètres

length	(interger) : taille du vecteur colonne

4.3.2.2 Col_Vector() [2/2]

```
Col_Vector::Col_Vector (
            const Col_Vector & v )
```

Constructeur de copie des vecteurs colonnes.

Paramètres

v	(Col_Vector) : le vecteur pier

4.3.3 Documentation des fonctions membres

4.3.3.1 operator+=()

```
Col_Vector & Col_Vector::operator+= (
            const Col_Vector & m )
```

Operateur d'addition interne pour les Col_Vector. Throw "size mismatched" si le vecteur a additionner n'est pas de la meme dimension que le vecteur courant.

Paramètres

m	(const Col_Vector&) : vecteur a additionner

4.3.3.2 operator-=()

```
Col_Vector & Col_Vector::operator-= (
            const Col_Vector & m )
```

Operateur de soustraction interne pour les Col_Vector. Throw "size mismatched" si le vecteur a soustraire n'est pas de la meme dimension que le vecteur courant.

Paramètres

m	(const Col_Vector&) : vecteur a soustraire

4.3.3.3 operator=()

```
Col_Vector & Col_Vector::operator= (
            const Col_Vector & v )
```

Surcharge operateur d'affectation. Throw "bad dimensions" si les tailles ne correspondent pas.

Paramètres

v	(Col_Vector): le vecteur a affecter

4.3.3.4 operator[]() [1/2]

```
double & Col_Vector::operator[] (
            int idx )
```

Surcharge operateur d'acces constant aux elements du vecteur colonne. L'acces se fait pour un indice entre 0 et length - 1. Throw "wrong index" si l'indice est hors des dimensions de la matrice

Paramètres

idx	(integer) : index de l'acces

4.3.3.5 operator[]() [2/2]

```
double Col_Vector::operator[] (
            int idx ) const
```

Surcharge operateur d'acces aux elements du vecteur colonne. L'acces se fait pour un indice entre 0 et length - 1. Throw "wrong index" si l'indice est hors des dimensions de la matrice.

Paramètres

idx	(integer) : index de l'acces

4.3.4 Documentation des fonctions amies et associées

4.3.4.1 operator*

```
Col_Vector operator* (
            const BMatrix & m1,
            const Col_Vector & m2 )  [friend]
```

Operateur externe de multiplication de BMatrix a gauche par un Col_Vector a droite. Throw "size mismatched" si les tailles ne sont pas compatible multiplication matricielle.

Paramètres

m1	(const BMatrix &) : La matrice a gauche
v	(const Col_Vector&) : Le vecteur a gauche

4.3.4.2 operator+

```
Col_Vector operator+ (
            const Col_Vector & v1,
            const Col_Vector & v2 )  [friend]
```

Operateur externe d'addition de Col_Vector. Throw "size mismatched" si les tailles des vecteurs ne sont pas identiques.

Paramètres

v1	(const Col_Vector&) : Le vecteur a gauche
v2	(const Col_Vector&) : Le vecteur a droite

4.3.4.3 operator-

```
Col_Vector operator- (
            const Col_Vector & v1,
            const Col_Vector & v2 )  [friend]
```

Operateur externe de soustraction de Col_Vector. Throw "size mismatched" si les tailles des vecteurs ne sont pas identiques.

Paramètres

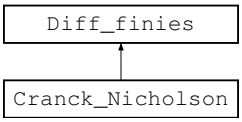
v1	(const Col_Vector&) : Le vecteur a gauche
v2	(const Col_Vector&) : Le vecteur a droite

La documentation de cette classe a été générée à partir du fichier suivant :
- include/col_vector.h
- src/col_vector.cpp

4.4 Référence de la classe Cranck_Nicholson

```
#include <cranck_nicholson.h>
```

Graphe d'héritage de Cranck_Nicholson:



Fonctions membres publiques

- Cranck_Nicholson (int Ns, int Nt, double sigma, const Payoff &p)
- Col_Vector get_init_cond () const
- Col_Vector step_forward (const Col_Vector &C)

Membres hérités additionnels

4.4.1 Description détaillée

Classe represantant un schema de Cranck-Nicholson pour l'equation complete de black-scholes

4.4.2 Documentation des constructeurs et destructeur

4.4.2.1 Cranck_Nicholson()

```
Cranck_Nicholson::Cranck_Nicholson (
            int Ns,
            int Nt,
            double sigma,
            const Payoff & p )
```

Constructeur de la classe Cranck_Nicholson

Paramètres

Ns	(int): Nombre d'intervalles de discretisation de [0,S]
Nt	(int): Nombre d'intervalles de discretisation de [0,T]
sigma	(double): volatilit l'actif sous jacent
p	(const Payoff& p): Option en vigueur

4.4.3 Documentation des fonctions membres

4.4.3.1 get_init_cond()

```
Col_Vector Cranck_Nicholson::get_init_cond ( ) const  [virtual]
```

Getter du vecteur des conditions initiales relatives au Payoff p_ et de la discretisation en vigueur

Implémente Diff_finies.

4.4.3.2 step_forward()

```
Col_Vector Cranck_Nicholson::step_forward (
            const Col_Vector & C )  [virtual]
```

Fonction renvoyant l'etape suivante dans le calcul du schema de cranck-nicholson et mettant a jour l'indicateur de l'etape de la methode.

Paramètres

(const	Col_Vector &) : Vecteur des valeurs calcular l'etape precedent du schema

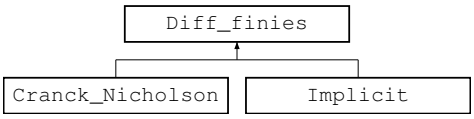
Implémente Diff_finies.

La documentation de cette classe a été générée à partir du fichier suivant :
- include/cranck_nicholson.h
- src/cranck_nicholson.cpp

4.5 Référence de la classe Diff_finies

```
#include <diff_finies.h>
```

Graphe d'héritage de Diff_finies:



Fonctions membres publiques

- Diff_finies (int Ns, int Nt, double sigma, const Payoff &p)
- int **get_Ns** () const
- int **get_Nt** () const
- double **get_sigma** () const
- int **get_step** () const
- const Payoff & **get_payoff** () const
- virtual Col_Vector **get_init_cond** () const =0
- virtual Col_Vector **step_forward** (const Col_Vector &C)=0
- bool is_finished () const

Attributs protégés

- int Ns_
- int Nt_
- double sigma_
- const Payoff & p_
- int step_

4.5.1 Description détaillée

Classe representant les methodes aux differences finies pour l'equation de black scholes

4.5.2 Documentation des constructeurs et destructeur

4.5.2.1 Diff_finies()

```
Diff_finies::Diff_finies (
            int Ns,
            int Nt,
            double sigma,
            const Payoff & p )
```

Constructeur de la classe Diff_finies

Paramètres

Ns	(int): Nombre d'intervalles de discretisation de [0,S]
Nt	(int): Nombre d'intervalles de discretisation de [0,T]
sigma	(double): volatilit l'actif sous jacent
p	(const Payoff& p): Option en vigueur

4.5.3 Documentation des fonctions membres

4.5.3.1 is_finished()

```
bool Diff_finies::is_finished ( ) const
```

Indique si la methode a finis les calculs

4.5.4 Documentation des données membres

4.5.4.1 Ns_

```
int Diff_finies::Ns_  [protected]
```

Nombre d'intervalles de discretisation de [0,S]

4.5.4.2 Nt_

```
int Diff_finies::Nt_  [protected]
```

Nombre d'intervalles de discretisation de [0,T]

4.5.4.3 p_

```
const Payoff& Diff_finies::p_  [protected]
```

Option en vigueur

4.5.4.4 sigma_

```
double Diff_finies::sigma_  [protected]
```

Volatilite de l'actif sous jacent

4.5.4.5 step_

```
int Diff_finies::step_  [protected]
```

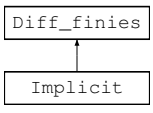
indicateur interne du nombre d'etape calculee

La documentation de cette classe a été générée à partir du fichier suivant :
- include/diff_finies.h
- src/diff_finies.cpp

4.6 Référence de la classe Implicit

```
#include <implicit.h>
```

Graphe d'héritage de Implicit:



Fonctions membres publiques

- Implicit (int Ns, int Nt, double sigma, const Payoff &p)
- Col_Vector get_init_cond () const
- Col_Vector step_forward (const Col_Vector &C)
- Col_Vector ctilde_to_C (const Col_Vector &C, double tTilde) const

Membres hérités additionnels

4.6.1 Description détaillée

Classe represantant un schema aux differences finies implicite pour l'equation reduite de black-scholes

4.6.2 Documentation des constructeurs et destructeur

4.6.2.1 Implicit()

```
Implicit::Implicit (
            int Ns,
            int Nt,
            double sigma,
            const Payoff & p )
```

Constructeur de la classe Implicit

Paramètres

Ns	(int): Nombre d'intervalles de discretisation de [0,S]
Nt	(int): Nombre d'intervalles de discretisation de [0,T]
sigma	(double): volatilite de l'actif sous jacent
p	(const Payoff& p): Option en vigueur

4.6.3 Documentation des fonctions membres

4.6.3.1 ctilde_to_C()

```
Col_Vector Implicit::ctilde_to_C (
            const Col_Vector & c_tilde,
            double tTilde ) const
```

Fonction renvoyant le vecteur correspondant aux valeur de C etant donner le vecteur Ctilde et tTilde

Paramètres

c_tilde	(const Col_Vector &) : Vecteur des valeur Ctilde
tTilde	(double) : temps du calcul du vecteur c_tilde

4.6.3.2 get_init_cond()

```
Col_Vector Implicit::get_init_cond ( ) const  [virtual]
```

Getter du vecteur des conditions initiales relatives au Payoff p_ et de la discretisation en vigueur

Implémente Diff_finies.

4.6.3.3 step_forward()

```
Col_Vector Implicit::step_forward (
            const Col_Vector & C )  [virtual]
```

Fonction renvoyant l'etape suivante dans le calcul du schema de cranck-nicholson et mettant a jour l'indicateur de l'etape de la methode.

Paramètres

(const	Col_Vector &) : Vecteur des valeurs calcul$\blacklozenge$e par l'etape precedent du schema

Implémente Diff_finies.

La documentation de cette classe a été générée à partir du fichier suivant :
- include/implicit.h
- src/implicit.cpp

4.7 Référence de la classe Linear_System

```
#include <linear_system.h>
```

Fonctions membres publiques

- Linear_System (int nb_inconnue, Tri_Diag_Matrix &A, Col_Vector &B)
- **Linear_System** (const Linear_System &m)
- int **get_nb_inconnue** () const
- Tri_Diag_Matrix **get_A** () const
- Col_Vector **get_B** () const
- Col_Vector resolution_system_algo_thomas ()

4.7.1 Description détaillée

Classe represantant un systeme lineaire tridiagonale AX = B

4.7.2 Documentation des constructeurs et destructeur

4.7.2.1 Linear_System()

```
Linear_System::Linear_System (
            int nb_inconnue,
            Tri_Diag_Matrix & A,
            Col_Vector & B )
```

Constructeur des systeme lineaire AX = B.

Paramètres

nb_inconnue	(integer): nombre d'inconnues du systeme.
A	(Tri_Diag_Matrix) : la matrice des coefficients.
B	(Col_Vector) : le vecteur colonne

4.7.3 Documentation des fonctions membres

4.7.3.1 resolution_system_algo_thomas()

```
Col_Vector Linear_System::resolution_system_algo_thomas ( )
```

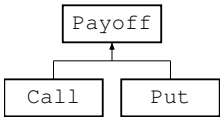
Resolution d'un syst tridiagonale rapide par l'algorithme de Thomas

La documentation de cette classe a été générée à partir du fichier suivant :
- include/linear_system.h
- src/linear_system.cpp

4.8 Référence de la classe Payoff

```
#include <payoff.h>
```

Graphe d'héritage de Payoff:



Fonctions membres publiques

- Payoff (int T, int L, int K, double r)
- Payoff (const Payoff &p)
- int **get_T** () const
- int **get_L** () const
- int **get_K** () const
- double **get_r** () const
- virtual double **get_C_T_s** (double s) const =0
- virtual double **get_C_t_0** (double t) const =0
- virtual double **get_C_t_L** (double t) const =0
- int max (int K, int s) const

Attributs protégés

- int T_
- int L_
- int K_
- double r_

4.8.1 Description détaillée

Classe abstraite represantant un Payoff de maniere generale

4.8.2 Documentation des constructeurs et destructeur

4.8.2.1 Payoff() [1/2]

```
Payoff::Payoff (
            int T,
            int L,
            int K,
            double r )
```

Constructeur de la classe Payoff.

Paramètres

T	(integer): Temps de maturite de l'option
L	(integer): Prix maximum de l'actif sous jacent
K	(integer): Strike de l'actif
r	(double): taux d'interet en vigueur

4.8.2.2 Payoff() [2/2]

```
Payoff::Payoff (
            const Payoff & p )
```

Constructeur de copie de la classe Payoff.

Paramètres

p	(const Call&) : call a copier

4.8.3 Documentation des fonctions membres

4.8.3.1 max()

```
int Payoff::max (
            int K,
            int s ) const
```

Fonction max entre 2 entiers

Paramètres

K	(integer): 1er entier a comparer
s	(integer): 2em entier a comparer

4.8.4 Documentation des données membres

4.8.4.1 K_

```
int Payoff::K_  [protected]
```

Strike K de l'option

4.8.4.2 L_

```
int Payoff::L_  [protected]
```

Maximum de la valeur de l'actif sous jacent

4.8.4.3 r_

```
double Payoff::r_  [protected]
```

Taux d'interet en vigueur

4.8.4.4 T_

```
int Payoff::T_  [protected]
```

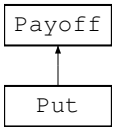
Temps de matrite T

La documentation de cette classe a été générée à partir du fichier suivant :
- include/payoff.h
- src/payoff.cpp

4.9 Référence de la classe Put

```
#include <put.h>
```

Graphe d'héritage de Put:



Fonctions membres publiques

- Put (int T, int L, int K, double r)
- Put (const Put &p)
- double get_C_T_s (double s) const
- double get_C_t_0 (double t) const
- double get_C_t_L (double t) const

Membres hérités additionnels

4.9.1 Description détaillée

Classe represantant un put.

4.9.2 Documentation des constructeurs et destructeur

4.9.2.1 Put() [1/2]

```
Put::Put (
            int T,
            int L,
            int K,
            double r )
```

Constructeur de la classe ut.

Paramètres

T	(integer): Temps de maturite de l'option
L	(integer): Prix maximum de l'actif sous jacent
K	(integer): Strike de l'actif
r	(double): taux d'interet en vigueur

4.9.2.2 Put() [2/2]

```
Put::Put (
            const Put & p )
```

Constructeur de copie de la classe Put.

Paramètres

p	(const Call&) : put a copier

4.9.3 Documentation des fonctions membres

4.9.3.1 get_C_t_0()

```
double Put::get_C_t_0 (
            double t ) const  [virtual]
```

Getter de la condition au bord du prix de l'option (lorsque s = 0). Si t $<$ 0 alors le calcul se fait pour t = 0 et si t $>$ T le calcul se fait pour t = T.

Paramètres

t	(double) : temps a laquel est evalue le prix de l'option

Implémente Payoff.

4.9.3.2 get_C_t_L()

```
double Put::get_C_t_L (
            double t ) const  [virtual]
```

Getter de la condition au bord du prix de l'option (lorsque s = L) Si t $<$ 0 alors le calcul se fait pour t = 0 et si t $>$ T le calcul se fait pour t = T.

Paramètres

t	(double) : temps a laquel est evalue le prix de l'option

Implémente Payoff.

4.9.3.3 get_C_T_s()

```
double Put::get_C_T_s (
            double s ) const  [virtual]
```

Getter de la condition a maturite du prix de l'option (lorsque t = T). Si s $<$ 0 alors le calcul se fait pour s = 0 et si s $>$ L le calcul se fait pour s = L.

Paramètres

s	(double): prix de l'actif sous jacent a l'instant T

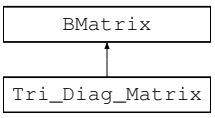
Implémente Payoff.

La documentation de cette classe a été générée à partir du fichier suivant :
- include/put.h
- src/put.cpp

4.10 Référence de la classe Tri_Diag_Matrix

```
#include <tri_diag_matrix.h>
```

Graphe d'héritage de Tri_Diag_Matrix:



Fonctions membres publiques

- Tri_Diag_Matrix (int nl, const Col_Vector &a, const Col_Vector &b, const Col_Vector &c)
- Tri_Diag_Matrix (const Tri_Diag_Matrix &m)
- Col_Vector get_a () const
- Col_Vector get_b () const
- Col_Vector get_c () const

Membres hérités additionnels

4.10.1 Description détaillée

Classe represantant une matrice tridiagonale

4.10.2 Documentation des constructeurs et destructeur

4.10.2.1 Tri_Diag_Matrix() [1/2]

```
Tri_Diag_Matrix::Tri_Diag_Matrix (
            int nl,
            const Col_Vector & a,
            const Col_Vector & b,
            const Col_Vector & c )
```

Constructeur d'une matrice tridiagonale carree de taille nl x nl. Throw "bad vector shapes" si les vecteurs en arguments ne sont pas de bonne dimensions.

Paramètres

nl	(integer) : nombre de lignes.
a	(Col_Vector) : La sous diagonale a
b	(Col_vector) : La diagonale b
c	(Col_Vector) : La sur diagonale c

4.10.2.2 Tri_Diag_Matrix() [2/2]

```
Tri_Diag_Matrix::Tri_Diag_Matrix (
            const Tri_Diag_Matrix & m )
```

Constructeur de copie des matrices tridiagonales. Throw "Bad shaped" si la matrice a copier na pas les memes dimensions.

Paramètres

m	(Tri_Diag_Matrix) : La matrice a copier

4.10.3 Documentation des fonctions membres

4.10.3.1 get_a()

```
Col_Vector Tri_Diag_Matrix::get_a ( ) const
```

Renvois la sous diagonale.

4.10.3.2 get_b()

```
Col_Vector Tri_Diag_Matrix::get_b ( ) const
```

Renvois la diagonale.

4.10.3.3 get_c()

```
Col_Vector Tri_Diag_Matrix::get_c ( ) const
```

Renvois la sur-diagonale.

La documentation de cette classe a été générée à partir du fichier suivant :
- include/tri_diag_matrix.h
- src/tri_diag_matrix.cpp

Index