

Projet PAP

Généré par Doxygen 1.9.0

1 README	1
2 Index hiérarchique	3
2.1 Hiérarchie des classes	3
3 Index des classes	5
3.1 Liste des classes	5
4 Documentation des classes	7
4.1 Référence de la classe BMatrix	7
4.1.1 Description détaillée	8
4.1.2 Documentation des constructeurs et destructeur	8
4.1.2.1 BMatrix()	8
4.1.2.2 ~BMatrix()	8
4.1.3 Documentation des fonctions membres	8
4.1.3.1 operator() [1/2]	8
4.1.3.2 operator() [2/2]	9
4.1.3.3 operator+=()	9
4.1.3.4 operator-=()	9
4.1.3.5 operator=()	9
4.1.3.6 same_size()	10
4.1.4 Documentation des fonctions amies et associées	10
4.1.4.1 operator* [1/2]	10
4.1.4.2 operator* [2/2]	10
4.1.4.3 operator+	11
4.1.4.4 operator-	11
4.1.4.5 operator<<	11
4.1.5 Documentation des données membres	12
4.1.5.1 data_	12
4.1.5.2 nc_	12
4.1.5.3 nl_	12
4.2 Référence de la classe Call	12
4.2.1 Description détaillée	13
4.2.2 Documentation des constructeurs et destructeur	13
4.2.2.1 Call() [1/2]	13
4.2.2.2 Call() [2/2]	13
4.2.3 Documentation des fonctions membres	13
4.2.3.1 get_C_t_0()	14
4.2.3.2 get_C_t_L()	14
4.2.3.3 get_C_T_s()	14
4.3 Référence de la classe Col_Vector	15
4.3.1 Description détaillée	15
4.3.2 Documentation des constructeurs et destructeur	15

4.3.2.1 Col_Vector() [1/2]	15
4.3.2.2 Col_Vector() [2/2]	16
4.3.3 Documentation des fonctions membres	16
4.3.3.1 operator+=()	16
4.3.3.2 operator-=()	16
4.3.3.3 operator=()	17
4.3.3.4 operator[]() [1/2]	17
4.3.3.5 operator[]() [2/2]	17
4.3.4 Documentation des fonctions amies et associées	17
4.3.4.1 operator*	17
4.3.4.2 operator+	18
4.3.4.3 operator-	18
4.4 Référence de la classe Cranck_Nicholson	18
4.4.1 Description détaillée	19
4.4.2 Documentation des constructeurs et destructeur	19
4.4.2.1 Cranck_Nicholson()	19
4.4.3 Documentation des fonctions membres	19
4.4.3.1 get_init_cond()	20
4.4.3.2 step_forward()	20
4.5 Référence de la classe Diff_finies	20
4.5.1 Description détaillée	21
4.5.2 Documentation des constructeurs et destructeur	21
4.5.2.1 Diff_finies()	21
4.5.3 Documentation des fonctions membres	21
4.5.3.1 is_finished()	21
4.5.4 Documentation des données membres	21
4.5.4.1 Ns_	22
4.5.4.2 Nt_	22
4.5.4.3 p_	22
4.5.4.4 sigma_	22
4.5.4.5 step_	22
4.6 Référence de la classe Implicit	22
4.6.1 Description détaillée	23
4.6.2 Documentation des constructeurs et destructeur	23
4.6.2.1 Implicit()	23
4.6.3 Documentation des fonctions membres	23
4.6.3.1 ctilde_to_C()	23
4.6.3.2 get_init_cond()	24
4.6.3.3 step_forward()	24
4.7 Référence de la classe Linear_System	24
4.7.1 Description détaillée	25
4.7.2 Documentation des constructeurs et destructeur	25

4.7.2.1 Linear_System()	25
4.7.3 Documentation des fonctions membres	25
4.7.3.1 resolution_system_algo_thomas()	25
4.8 Référence de la classe Payoff	25
4.8.1 Description détaillée	26
4.8.2 Documentation des constructeurs et destructeur	26
4.8.2.1 Payoff() [1/2]	26
4.8.2.2 Payoff() [2/2]	26
4.8.3 Documentation des fonctions membres	27
4.8.3.1 max()	27
4.8.4 Documentation des données membres	27
4.8.4.1 K_	27
4.8.4.2 L_	27
4.8.4.3 r_	27
4.8.4.4 T_	28
4.9 Référence de la classe Put	28
4.9.1 Description détaillée	28
4.9.2 Documentation des constructeurs et destructeur	28
4.9.2.1 Put() [1/2]	28
4.9.2.2 Put() [2/2]	29
4.9.3 Documentation des fonctions membres	29
4.9.3.1 get_C_t_0()	29
4.9.3.2 get_C_t_L()	29
4.9.3.3 get_C_T_s()	30
4.10 Référence de la classe Tri_Diag_Matrix	30
4.10.1 Description détaillée	30
4.10.2 Documentation des constructeurs et destructeur	31
4.10.2.1 Tri_Diag_Matrix() [1/2]	31
4.10.2.2 Tri_Diag_Matrix() [2/2]	31
4.10.3 Documentation des fonctions membres	31
4.10.3.1 get_a()	31
4.10.3.2 get_b()	32
4.10.3.3 get_c()	32
Index	33

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 :

BMatrix	7
Col_Vector	15
Tri_Diag_Matrix	30
Diff_finies	20
Cranck_Nicholson	18
Implicit	22
Linear_System	24
Payoff	25
Call	12
Put	28

Chapitre 3

Index des classes

3.1 Liste des classes

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

BMatrix	7
Call	12
Col_Vector	15
Cranck_Nicholson	18
Diff_finies	20
Implicit	22
Linear_System	24
Payoff	25
Put	28
Tri_Diag_Matrix	30

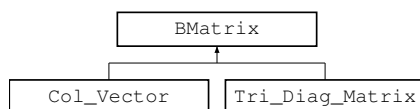
Chapitre 4

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

<i>nl</i>	(integer) : nombre de lignes
<i>nc</i>	(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'accès 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

<i>l</i>	(integer) : ligne d'accès
<i>c</i>	(integer) : colonne d'accès

4.1.3.2 operator() [2/2]

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

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

Paramètres

<i>l</i>	(integer) : ligne d'accès
<i>c</i>	(integer) : colonne d'accès

4.1.3.3 operator+=()

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

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

Paramètres

<i>m</i>	(BMatrix) : la matrice à ajouter
----------	----------------------------------

4.1.3.4 operator-=()

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

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

Paramètres

<i>m</i>	(BMatrix) : la matrice à soustraire
----------	-------------------------------------

4.1.3.5 operator=()

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

```
const BMatrix & m )
```

Surcharge de l'operateur d'affectation des matrices.

Paramètres

<i>m</i>	(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

<i>m</i>	(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

<i>m1</i>	(BMatrix) : la matrice a gauche de la multiplication
<i>m2</i>	(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'opérateur extérieur de multiplication d'une matrices par un scalaire. Renvois la matrice resultante de la multiplication.

Paramètres

<i>val</i>	(double) : le scalaire de la mutiplication
<i>m</i>	(BMatrix) : la matrice de la multiplication

4.1.4.3 operator+

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

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

Paramètres

<i>m1</i>	(BMatrix) : la 1ere matrice a additionner
<i>m2</i>	(BMatrix) : la 2em matrice a additionner

4.1.4.4 operator-

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

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

Paramètres

<i>m1</i>	(BMatrix) : la 1ere matrice a soustraire
<i>m2</i>	(BMatrix) : la 2em matrice a soustraire

4.1.4.5 operator<<

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

Surcharge de l'opérateur 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

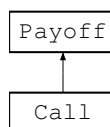
La documentation de cette classe a été générée à partir du fichier suivant :

- 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

<i>T</i>	(integer): Temps de maturite du call
<i>L</i>	(integer): Prix maximum de l'actif sous jacent
<i>K</i>	(integer): Strike de l'actif
<i>r</i>	(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

<i>p</i>	(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

<code>t</code>	(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

<code>t</code>	(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

<code>s</code>	(double): prix de l'actif sous jacent a l'instant T
----------------	---

Implémente [Payoff](#).

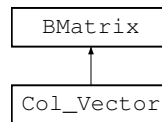
La documentation de cette classe a été générée à partir du fichier suivant :

- `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 représentant 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

<i>length</i>	(integer) : 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

<i>v</i>	(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 )
```

Opérateur 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

<i>m</i>	(const Col_Vector&) : vecteur a additionner
----------	---

4.3.3.2 operator-=()

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

Opérateur 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

<i>m</i>	(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

<i>v</i>	(Col_Vector): le vecteur a affecter
----------	-------------------------------------

4.3.3.4 operator[]() [1/2]

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

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

Paramètres

<i>idx</i>	(integer) : index de l'accès
------------	------------------------------

4.3.3.5 operator[]() [2/2]

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

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

Paramètres

<i>idx</i>	(integer) : index de l'accès
------------	------------------------------

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]
```

Opérateur 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

<i>m1</i>	(const BMatrix &) : La matrice a gauche
<i>v</i>	(const Col_Vector &) : Le vecteur a gauche

4.3.4.2 operator+

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

Opérateur externe d'addition de [Col_Vector](#). Throw "size mismatched" si les tailles des vecteurs ne sont pas identiques.

Paramètres

<i>v1</i>	(const Col_Vector &) : Le vecteur a gauche
<i>v2</i>	(const Col_Vector &) : Le vecteur a droite

4.3.4.3 operator-

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

Opérateur externe de soustraction de [Col_Vector](#). Throw "size mismatched" si les tailles des vecteurs ne sont pas identiques.

Paramètres

<i>v1</i>	(const Col_Vector &) : Le vecteur a gauche
<i>v2</i>	(const Col_Vector &) : Le vecteur a droite

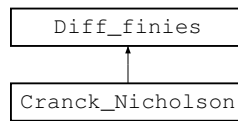
La documentation de cette classe a été générée à partir du fichier suivant :

- 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 representant 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

<i>Ns</i>	(int): Nombre d'intervalles de discretisation de [0,S]
<i>Nt</i>	(int): Nombre d'intervalles de discretisation de [0,T]
<i>sigma</i>	(double): volatilité l'actif sous jacent
<i>p</i>	(const <code>Payoff</code> & <i>p</i>): 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](#).

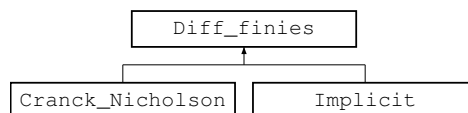
La documentation de cette classe a été générée à partir du fichier suivant :

- 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

<i>Ns</i>	(int): Nombre d'intervalles de discretisation de [0,S]
<i>Nt</i>	(int): Nombre d'intervalles de discretisation de [0,T]
<i>sigma</i>	(double): volatilité l'actif sous jacent
<i>p</i>	(const <code>Payoff</code> & 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

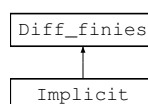
La documentation de cette classe a été générée à partir du fichier suivant :

- 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 representant 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

<i>Ns</i>	(int): Nombre d'intervalles de discretisation de [0,S]
<i>Nt</i>	(int): Nombre d'intervalles de discretisation de [0,T]
<i>sigma</i>	(double): volatilité de l'actif sous jacent
<i>p</i>	(const <code>Payoff</code> & <i>p</i>): 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

<i>c_tilde</i>	(const Col_Vector &) : Vecteur des valeur Ctilde
<i>tTilde</i>	(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 calculée 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 representant 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

<i>nb_inconnue</i>	(integer): nombre d'inconnues du systeme.
<i>A</i>	(Tri_Diag_Matrix) : la matrice des coefficients.
<i>B</i>	(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

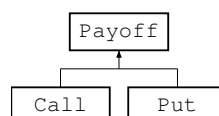
La documentation de cette classe a été générée à partir du fichier suivant :

- `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 representant 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

<i>T</i>	(integer): Temps de maturite de l'option
<i>L</i>	(integer): Prix maximum de l'actif sous jacent
<i>K</i>	(integer): Strike de l'actif
<i>r</i>	(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

<i>p</i>	(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

<i>K</i>	(integer): 1er entier a comparer
<i>s</i>	(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

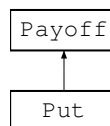
La documentation de cette classe a été générée à partir du fichier suivant :

- 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](#).

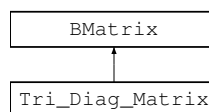
La documentation de cette classe a été générée à partir du fichier suivant :

- `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 representant 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

<i>nl</i>	(integer) : nombre de lignes.
<i>a</i>	(Col_Vector) : La sous diagonale a
<i>b</i>	(Col_vector) : La diagonale b
<i>c</i>	(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

<i>m</i>	(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

- ~BMatrix
 - BMatrix, [8](#)
- BMatrix, [7](#)
 - ~BMatrix, [8](#)
 - BMatrix, [8](#)
 - data_, [12](#)
 - nc_, [12](#)
 - nl_, [12](#)
 - operator<<, [11](#)
 - operator*, [10](#)
 - operator(), [8](#), [9](#)
 - operator+, [11](#)
 - operator+=, [9](#)
 - operator-, [11](#)
 - operator=, [9](#)
 - operator=, [9](#)
 - same_size, [10](#)
- Call, [12](#)
 - Call, [13](#)
 - get_C_t_0, [13](#)
 - get_C_t_L, [14](#)
 - get_C_T_s, [14](#)
- Col_Vector, [15](#)
 - Col_Vector, [15](#), [16](#)
 - operator*, [17](#)
 - operator+, [18](#)
 - operator+=, [16](#)
 - operator-, [18](#)
 - operator=, [16](#)
 - operator=, [16](#)
 - operator[], [17](#)
- Cranck_Nicholson, [18](#)
 - Cranck_Nicholson, [19](#)
 - get_init_cond, [19](#)
 - step_forward, [20](#)
- ctilde_to_C
 - Implicit, [23](#)
- data_
 - BMatrix, [12](#)
- Diff_finies, [20](#)
 - Diff_finies, [21](#)
 - is_finished, [21](#)
 - Ns_, [21](#)
 - Nt_, [22](#)
 - p_, [22](#)
 - sigma_, [22](#)
 - step_, [22](#)
- get_a
 - Tri_Diag_Matrix, [31](#)
- get_b
 - Tri_Diag_Matrix, [31](#)
- get_c
 - Tri_Diag_Matrix, [32](#)
- get_C_t_0
 - Call, [13](#)
 - Put, [29](#)
- get_C_t_L
 - Call, [14](#)
 - Put, [29](#)
- get_C_T_s
 - Call, [14](#)
 - Put, [30](#)
- get_init_cond
 - Cranck_Nicholson, [19](#)
 - Implicit, [24](#)
- Implicit, [22](#)
 - ctilde_to_C, [23](#)
 - get_init_cond, [24](#)
 - Implicit, [23](#)
 - step_forward, [24](#)
- is_finished
 - Diff_finies, [21](#)
- K_
 - Payoff, [27](#)
- L_
 - Payoff, [27](#)
- Linear_System, [24](#)
 - Linear_System, [25](#)
 - resolution_system_algo_thomas, [25](#)
- max
 - Payoff, [27](#)
- nc_
 - BMatrix, [12](#)
- nl_
 - BMatrix, [12](#)
- Ns_
 - Diff_finies, [21](#)
- Nt_
 - Diff_finies, [22](#)
- operator<<
 - BMatrix, [11](#)
- operator*

- BMatrix, 10
- Col_Vector, 17
- operator()
 - BMatrix, 8, 9
- operator+
 - BMatrix, 11
 - Col_Vector, 18
- operator+=
 - BMatrix, 9
 - Col_Vector, 16
- operator-
 - BMatrix, 11
 - Col_Vector, 18
- operator-=
 - BMatrix, 9
 - Col_Vector, 16
- operator=
 - BMatrix, 9
 - Col_Vector, 16
- operator[]
 - Col_Vector, 17
- p_
 - Diff_finies, 22
- Payoff, 25
 - K_, 27
 - L_, 27
 - max, 27
 - Payoff, 26
 - r_, 27
 - T_, 27
- Put, 28
 - get_C_t_0, 29
 - get_C_t_L, 29
 - get_C_T_s, 30
 - Put, 28, 29
- r_
 - Payoff, 27
- resolution_system_algo_thomas
 - Linear_System, 25
- same_size
 - BMatrix, 10
- sigma_
 - Diff_finies, 22
- step_
 - Diff_finies, 22
- step_forward
 - Cranck_Nicholson, 20
 - Implicit, 24
- T_
 - Payoff, 27
- Tri_Diag_Matrix, 30
 - get_a, 31
 - get_b, 31
 - get_c, 32
 - Tri_Diag_Matrix, 31