Black and Scholes pricing model

Generated by Doxygen 1.12.0

1	Hierarchical Index	1
	1.1 Class Hierarchy	1
2	Class Index	3
	2.1 Class List	3
2	File Index	5
•	3.1 File List	5
1	Class Documentation	7
•	4.1 ensile::Change Class Reference	7
	4.1.1 Detailed Description	9
	4.1.2 Constructor & Destructor Documentation	9
	4.1.2.1 Change() [1/2]	9
	4.1.2.2 Change() [2/2]	9
	4.1.3 Member Function Documentation	10
	4.1.3.1 get_ds_changed()	10
	4.1.3.2 get_dt_changed()	10
	4.1.3.3 get_f()	10
	4.1.3.4 get_l_changed()	10
	4.1.3.5 get_t_changed()	11
	4.1.3.6 l_transformation()	11
	4.1.3.7 price_transformation()	11
	4.1.3.8 t_transformation()	11
	4.1.4 Member Data Documentation	12
	4.1.4.1 ds_changed	12
	4.1.4.2 dt_changed	12
	4.1.4.3 f	12
	4.2 ensiie::Complete Class Reference	12
	4.2.1 Detailed Description	14
	4.2.2 Constructor & Destructor Documentation	15
	<b>4.2.2.1 Complete()</b> [1/2]	15
	<b>4.2.2.2 Complete()</b> [2/2]	15
	4.2.3 Member Function Documentation	15
	4.2.3.1 coefficients_computation()	15
	4.2.3.2 get_alpha()	16
	4.2.3.3 get_beta()	16
	4.2.3.4 get_gamma()	16
	4.2.3.5 get_low()	16
	4.2.3.6 get_up()	16
	4.2.3.7 lu_factorization()	17
	4.2.3.8 pricing()	17
	4.3 ensiie::CompleteCall Class Reference	17

4.3.1 Detailed Description	19
4.3.2 Constructor & Destructor Documentation	20
4.3.2.1 CompleteCall() [1/2]	20
4.3.2.2 CompleteCall() [2/2]	20
4.3.3 Member Function Documentation	20
4.3.3.1 get_price()	20
4.3.3.2 pricing()	21
4.4 ensiie::CompletePut Class Reference	21
4.4.1 Detailed Description	23
4.4.2 Constructor & Destructor Documentation	23
4.4.2.1 CompletePut() [1/2]	23
4.4.2.2 CompletePut() [2/2]	24
4.4.3 Member Function Documentation	24
4.4.3.1 get_price()	24
4.4.3.2 pricing()	24
4.5 ensiie::Data Class Reference	25
4.5.1 Detailed Description	26
4.5.2 Constructor & Destructor Documentation	26
4.5.2.1 Data()	26
4.5.3 Member Function Documentation	27
4.5.3.1 discretize()	27
4.5.3.2 get_ds()	27
4.5.3.3 get_dt()	27
4.5.3.4 get_K()	27
4.5.3.5 get_L()	27
4.5.3.6 get_l()	28
4.5.3.7 get_M()	28
4.5.3.8 get_N()	28
4.5.3.9 get_r()	28
4.5.3.10 get_sigma()	28
4.5.3.11 get_T()	29
4.5.3.12 get_t()	29
4.5.4 Member Data Documentation	29
4.5.4.1 ds	29
4.5.4.2 dt	29
4.5.4.3 K	29
4.5.4.4 L	29
4.5.4.5 l	30
4.5.4.6 M	30
4.5.4.7 N	30
4.5.4.8 r	30
4.5.4.9 sigma	30

4.5.4.10 T	30
4.5.4.11 t	30
4.6 ensiie::Reduced Class Reference	31
4.6.1 Detailed Description	33
4.6.2 Constructor & Destructor Documentation	34
<b>4.6.2.1 Reduced()</b> [1/2]	34
<b>4.6.2.2 Reduced()</b> [2/2]	34
4.6.3 Member Function Documentation	34
4.6.3.1 get_low()	34
4.6.3.2 get_theta()	35
4.6.3.3 get_up()	35
4.6.3.4 lu_factorization()	35
4.6.3.5 pricing()	35
4.6.4 Member Data Documentation	36
4.6.4.1 theta	36
4.7 ensiie::ReducedCall Class Reference	36
4.7.1 Detailed Description	39
4.7.2 Constructor & Destructor Documentation	39
<b>4.7.2.1</b> ReducedCall() [1/2]	39
<b>4.7.2.2</b> ReducedCall() [2/2]	40
4.7.3 Member Function Documentation	41
4.7.3.1 get_price()	41
4.7.3.2 pricing()	41
4.8 ensiie::ReducedPut Class Reference	42
4.8.1 Detailed Description	44
4.8.2 Constructor & Destructor Documentation	45
4.8.2.1 ReducedPut() [1/2]	45
4.8.2.2 ReducedPut() [2/2]	45
4.8.3 Member Function Documentation	45
4.8.3.1 get_price()	45
4.8.3.2 pricing()	46
4.9 ensiie::SDL Class Reference	46
4.9.1 Detailed Description	47
4.9.2 Constructor & Destructor Documentation	47
4.9.2.1 SDL()	47
4.9.2.2 ∼SDL()	47
4.9.3 Member Function Documentation	47
4.9.3.1 clearScreen()	47
4.9.3.2 drawAxes()	48
4.9.3.3 drawGraph_green()	48
4.9.3.4 drawGraph_red()	48
4.9.3.5 updateScreen()	48

4.9.3.6 waitForClose()	48
5 File Documentation	49
5.1 change.h	49
5.2 complete.h	49
5.3 completecall.h	50
5.4 completeput.h	50
5.5 data.h	50
5.6 reduced.h	51
5.7 reducedcall.h	52
5.8 reducedput.h	52
5.9 sdl.h	52
Index	53

# **Chapter 1**

# **Hierarchical Index**

# 1.1 Class Hierarchy

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

ensiie::Data	25
ensiie::Change	7
ensiie::Reduced	31
ensiie::ReducedCall	
ensiie::ReducedPut	12
ensiie::Complete	2
ensiie::CompleteCall	7
ensiie::CompletePut	21
ensiie::SDL	16

2 Hierarchical Index

# **Chapter 2**

# **Class Index**

# 2.1 Class List

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

ensiie::Change	
Class representing a transformation of variables to determine those to be used to dtermine the	
transformed PDE solution to the otpion pricing	7
ensiie::Complete	
A base abstract class for solving financial models using Crank-Nicolson methods	12
ensiie::CompleteCall	
A class to calculate and store the price of a European call option using Crank-Nicolson method	
to solve the Black-Scholes PDE	17
ensiie::CompletePut	
A class to calculate and store the price of a European put option using Crank-Nicolson method	
to solve the Black-Scholes PDE	21
ensiie::Data	
Contains all the data needed to solve the Black-Scholes pde	25
ensiie::Reduced	
A base abstract class for solving financial models using implicit finite differences methods	31
ensiie::ReducedCall	
A class to calculate and store the price of a European call option using implicit finite differences	
method to solve the modified Black-Scholes Heat Equation	36
ensiie::ReducedPut	
A class to calculate and store the price of a European put option using implicit finite differences	
method to solve the modified Black-Scholes Heat Equation	42
ensiie::SDL	
A class to manage the SDL window and render put and call prices	46

4 Class Index

# **Chapter 3**

# **File Index**

# 3.1 File List

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

change.h		 																		 				49
complete.h		 																		 				49
completecall.h	1																			 				50
completeput.h		 																		 				50
data.h		 																		 				50
reduced.h																								
reducedcall.h																								
reducedput.h																								
sdl h																								52

6 File Index

# **Chapter 4**

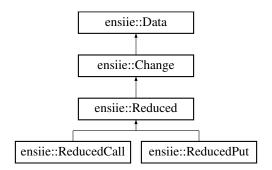
# **Class Documentation**

# 4.1 ensiie::Change Class Reference

Class representing a transformation of variables to determine those to be used to dtermine the transformed PDE solution to the otpion pricing.

```
#include <change.h>
```

Inheritance diagram for ensiie::Change:



#### **Public Member Functions**

• Change (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Change object using financial parameters.

• Change (const Data &d)

Constructs a Reduced object using an existing Data object.

std::vector< double > get t changed () const

Gets the transformed time values.

std::vector< double > get\_l\_changed () const

Gets the transformed price values.

· double get\_dt\_changed () const

Gets the time variation step.

• double get\_ds\_changed () const

Gets the price variation.

· double get\_f () const

Gets the f parameter.

# Public Member Functions inherited from ensiie::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

• double get\_K () const

Gets the strike price of the option.

double get\_L () const

Gets the maximum asset price considered in the model.

• double get\_M () const

Gets the number of time steps in the discretization.

double get\_N () const

Gets the number of asset price steps in the discretization.

• double get dt () const

Gets the time step size.

• double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_I () const

Gets the discretized asset price vector.

#### **Protected Member Functions**

• void t\_transformation ()

Performs the time transformation.

• void I\_transformation ()

Performs the price transformation.

std::vector< double > price\_transformation (const std::vector< double > &v)

Transforms the t=0 price vector.

### Protected Member Functions inherited from ensile::Data

• void discretize ()

Discretizes the time and space domains based on the parameters of the model.

#### **Protected Attributes**

- double f\_
- · double dt\_changed\_
- · double ds\_changed\_
- std::vector< double > t\_changed\_
- std::vector< double > I\_changed\_

#### Protected Attributes inherited from ensiie::Data

```
double T__
double r__
double sigma__
double K__
double L__
double M__
double N__
double dt__
double ds__
std::vector< double > t__
std::vector< double > I__
```

# 4.1.1 Detailed Description

Class representing a transformation of variables to determine those to be used to dtermine the transformed PDE solution to the otpion pricing.

The Change class inherits from the Data class and is responsible for performing transformation of prices and time vectors.

#### 4.1.2 Constructor & Destructor Documentation

# 4.1.2.1 Change() [1/2]

```
ensiie::Change::Change (
double T,
double r,
double sigma,
double K,
double L,
double M,
double N)
```

Constructs a Change object using financial parameters.

Initializes the financial parameters for the option pricing model and calculates transofrmation of time and underlying asset's price.

# Parameters

T	Time to maturity (in years)
r	Market Risk-free interest rate
sigma	Volatility of the underlying asset
K	Strike price of the option
L	Maximum value of the underlying asset
М	Number of time steps
N	Number of price steps

#### 4.1.2.2 Change() [2/2]

Constructs a Reduced object using an existing Data object.

This constructor initializes the model using an existing Data object and calculates change of variables.

#### **Parameters**

d A Data object containing the financial parameters for the model

# 4.1.3 Member Function Documentation

# 4.1.3.1 get\_ds\_changed()

```
double ensiie::Change::get_ds_changed () const
```

Gets the price variation.

#### Returns

The price variation value.

### 4.1.3.2 get\_dt\_changed()

```
double ensiie::Change::get_dt_changed () const
```

Gets the time variation step.

#### Returns

The time variation step.

# 4.1.3.3 get\_f()

```
double ensiie::Change::get_f () const
```

Gets the f parameter.

# Returns

The f parameter value.

# 4.1.3.4 get\_l\_changed()

```
std::vector< double > ensiie::Change::get_l_changed () const
```

Gets the transformed price values.

#### Returns

A vector containing the transformed asset's price values.

#### 4.1.3.5 get\_t\_changed()

```
std::vector< double > ensiie::Change::get_t_changed () const
```

Gets the transformed time values.

Returns

A vector containing the transformed time values.

#### 4.1.3.6 I\_transformation()

```
void ensiie::Change::l_transformation () [protected]
```

Performs the price transformation.

This function calculates the price transformation 1 based on the parameter K and stores the results in  $1\_\leftarrow$  changed\_ and ds\_changed\_. This aproximation is done as it's not possible to transform S=0

#### 4.1.3.7 price\_transformation()

Transforms the t=0 price vector.

**Parameters** 

v The price vector to be transformed.

#### Returns

A vector containing the transformed prices.

This function applies a transformation to the prices obtained by the changed PDE using a formula based on  $f_a$  and  $l_changed_n$ , returning a vector with the corrected prices. It's taken into account just the change of variable for the final time

# 4.1.3.8 t\_transformation()

```
void ensiie::Change::t_transformation () [protected]
```

Performs the time transformation.

This function calculates the time transformation t based on volatility and maturity, and stores the results in  $t\_\leftarrow$  changed\_ and  $dt\_$ changed\_.

# 4.1.4 Member Data Documentation

#### 4.1.4.1 ds\_changed\_

```
double ensiie::Change::ds_changed_ [protected]
```

Changed underlying asset's price step.

# 4.1.4.2 dt\_changed\_

```
double ensiie::Change::dt_changed_ [protected]
```

Changed time step.

#### 4.1.4.3 f

```
double ensiie::Change::f_ [protected]
```

Parameter to execute variable's change.

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

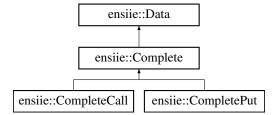
- · change.h
- · change.cpp

# 4.2 ensiie::Complete Class Reference

A base abstract class for solving financial models using Crank-Nicolson methods.

```
#include <complete.h>
```

Inheritance diagram for ensiie::Complete:



#### **Public Member Functions**

• Complete (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Complete object with the given parameters.

Complete (const Data &d)

Constructs a Complete object using an existing Data object.

• virtual void pricing ()=0

Pure virtual function to compute the price of the option.

std::vector< double > get\_alpha () const

Gets the alpha coefficients for the Crank Nicolson scheme.

std::vector< double > get\_beta () const

Gets the beta coefficients for the Crank\_Nicolson scheme.

std::vector< double > get\_gamma () const

Gets the gamma coefficients for the Crank\_Nicolson scheme.

std::vector< double > get\_low () const

Gets the lower matrix values of coefficients' matrix from the LU factorization.

• std::vector< double > get\_up () const

Gets the upper matrix values of coefficients' matrix from the LU factorization.

#### Public Member Functions inherited from ensile::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

• double get K () const

Gets the strike price of the option.

• double get\_L () const

Gets the maximum asset price considered in the model.

double get\_M () const

Gets the number of time steps in the discretization.

• double get\_N () const

Gets the number of asset price steps in the discretization.

• double get\_dt () const

Gets the time step size.

double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_l () const

Gets the discretized asset price vector.

#### **Protected Member Functions**

void coefficients\_computation ()

Computes the coefficients (alpha, beta, gamma) for the Crank-Nicolson scheme.

void lu\_factorization ()

Performs LU factorization of the system's matrix for Crank-Nicholson scheme.

# Protected Member Functions inherited from ensile::Data

· void discretize ()

Discretizes the time and space domains based on the parameters of the model.

#### **Protected Attributes**

```
std::vector< double > alpha_
std::vector< double > beta_
std::vector< double > gamma_
std::vector< double > low_
std::vector< double > up_
```

#### Protected Attributes inherited from ensile::Data

```
double T_
double r_
double sigma_
double K_
double L_
double M_
double N_
double dt_
double ds_
std::vector< double > t_
std::vector< double > I_
```

# 4.2.1 Detailed Description

A base abstract class for solving financial models using Crank-Nicolson methods.

The Complete class extends the Data class and serves as an abstract base for solving partial differential equations (PDEs) related to call and put options pricing. It provides the structure and common functionality for subclasses, involved in deterining prices for call and put optiont, computing coefficients used in Crank-Nicolson schemes, performing LU factorization of the coefficients' matrix, and providing access to the computed values.

This abstract class includes methods for:

- Calculation of the coefficients (alpha, beta, gamma) for Crank-Nicolson methods.
- · LU factorization for efficient solving of tridiagonal systems.

Subclasses derived from Complete implement the specific algorithm of the PDE solver, including boundary conditions, to get the european call and put option pricing.

The class can be initialized via direct parameter input or using an existing Data object.

Note

This class cannot be instantiated directly due to its abstract nature.

#### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Complete() [1/2]

Constructs a Complete object with the given parameters.

This constructor initializes the base <code>Data</code> class and computes the coefficients and LU factorization.

#### **Parameters**

T	Total time to maturity (in years).
r	Risk-free interest rate.
sigma	Volatility of the underlying asset.
K	Strike price.
L	Maximum asset price for discretization.
М	Number of time steps.
N	Number of space steps.

### 4.2.2.2 Complete() [2/2]

Constructs a Complete object using an existing Data object.

This constructor takes the parameters from an existing Data object and computes the coefficients and LU factorization

Note: The Data object d is not copied; its values are used to initialize the base class Data members. Therefore, the member of the class 'Data' exists only once in memory, shared between the Data and Complete objects.

#### **Parameters**

```
d A Data object to initialize the base class.
```

#### 4.2.3 Member Function Documentation

#### 4.2.3.1 coefficients\_computation()

```
void ensiie::Complete::coefficients_computation () [protected]
```

Computes the coefficients (alpha, beta, gamma) for the Crank-Nicolson scheme.

These coefficients are used in the matrix representation of the problem. The computation is based on the model parameters and discretization values.

#### 4.2.3.2 get\_alpha()

```
std::vector< double > ensiie::Complete::get_alpha () const
```

Gets the alpha coefficients for the Crank\_Nicolson scheme.

Returns

A vector of alpha coefficients.

# 4.2.3.3 get\_beta()

```
std::vector< double > ensiie::Complete::get_beta () const
```

Gets the beta coefficients for the Crank\_Nicolson scheme.

Returns

A vector of beta coefficients.

#### 4.2.3.4 get\_gamma()

```
std::vector< double > ensiie::Complete::get_gamma () const
```

Gets the gamma coefficients for the Crank\_Nicolson scheme.

Returns

A vector of gamma coefficients.

# 4.2.3.5 get\_low()

```
std::vector< double > ensiie::Complete::get_low () const
```

Gets the lower matrix values of coefficients' matrix from the LU factorization.

Returns

A vector of lower matrix values (low).

#### 4.2.3.6 get\_up()

```
std::vector< double > ensiie::Complete::get_up () const
```

Gets the upper matrix values of coefficients' matrix from the LU factorization.

Returns

A vector of upper matrix values (up).

#### 4.2.3.7 lu\_factorization()

```
void ensiie::Complete::lu_factorization () [protected]
```

Performs LU factorization of the system's matrix for Crank-Nicholson scheme.

This method calculates the lower (low) and upper (up) matrices, which are used to solve the system of equations iteratively.

#### 4.2.3.8 pricing()

```
virtual void ensiie::Complete::pricing () [pure virtual]
```

Pure virtual function to compute the price of the option.

This is a pure virtual function that is implemented by its derived class.

It defines the computation of Call and Put options price. The specific implementation will based on different boundary conditions.

Note

Since this is a pure virtual function, Complete cannot be instantiated directly.

Implemented in ensiie::CompleteCall, and ensiie::CompletePut.

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

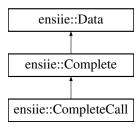
- · complete.h
- · complete.cpp

# 4.3 ensiie::CompleteCall Class Reference

A class to calculate and store the price of a European call option using Crank-Nicolson method to solve the Black-Scholes PDE.

```
#include <completecall.h>
```

Inheritance diagram for ensiie::CompleteCall:



#### **Public Member Functions**

CompleteCall (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a CompleteCall object using financial parameters.

CompleteCall (const Data &d)

Constructs a CompleteCall object using an existing Data object.

· void pricing () override

Computes the price of the European call option using the Crank-Nicolson method.

std::vector< double > get\_price () const

Retrieves the computed prices of the call option C(0, s), which are prices at time 0 for each level of underlying price s.

#### Public Member Functions inherited from ensile::Complete

• Complete (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Complete object with the given parameters.

Complete (const Data &d)

Constructs a Complete object using an existing Data object.

std::vector< double > get\_alpha () const

Gets the alpha coefficients for the Crank\_Nicolson scheme.

std::vector< double > get beta () const

Gets the beta coefficients for the Crank Nicolson scheme.

std::vector< double > get\_gamma () const

Gets the gamma coefficients for the Crank\_Nicolson scheme.

std::vector< double > get\_low () const

Gets the lower matrix values of coefficients' matrix from the LU factorization.

std::vector< double > get\_up () const

Gets the upper matrix values of coefficients' matrix from the LU factorization.

#### Public Member Functions inherited from ensile::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

double get\_sigma () const

Gets the volatility of the underlying asset.

double get\_K () const

Gets the strike price of the option.

double get\_L () const

Gets the maximum asset price considered in the model.

• double get\_M () const

Gets the number of time steps in the discretization.

double get\_N () const

Gets the number of asset price steps in the discretization.

double get\_dt () const

Gets the time step size.

double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_l () const

Gets the discretized asset price vector.

#### **Additional Inherited Members**

# Protected Member Functions inherited from ensile::Complete

• void coefficients\_computation ()

Computes the coefficients (alpha, beta, gamma) for the Crank-Nicolson scheme.

void lu factorization ()

Performs LU factorization of the system's matrix for Crank-Nicholson scheme.

#### Protected Member Functions inherited from ensile::Data

· void discretize ()

Discretizes the time and space domains based on the parameters of the model.

# Protected Attributes inherited from ensiie::Complete

```
    std::vector< double > alpha_
```

- std::vector< double > beta\_
- std::vector< double > gamma\_
- std::vector< double > low\_
- std::vector< double > up\_

#### Protected Attributes inherited from ensile::Data

- double T
- double r\_
- double sigma\_
- double K
- double L
- double M
- double N
- double dt
- double ds
- · double ds\_
- std::vector< double > t\_
- std::vector< double > I

# 4.3.1 Detailed Description

A class to calculate and store the price of a European call option using Crank-Nicolson method to solve the Black-Scholes PDE.

This class is derived from the Complete class and implements the specific pricing algorithm for a European call option, including the computation of boundary conditions and the solution of the PDE using LU factorization.

#### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 CompleteCall() [1/2]

Constructs a CompleteCall object using financial parameters.

Initializes the financial parameters for the option pricing model and calculates the coefficients and LU factorization.

#### **Parameters**

T	Time to maturity (in years)
r	Market Risk-free interest rate
sigma	Volatility of the underlying asset
K	Strike price of the option
L	Maximum value of the underlying asset
М	Number of time steps
N	Number of price steps

#### 4.3.2.2 CompleteCall() [2/2]

Constructs a CompleteCall object using an existing Data object.

This constructor initializes the model using an existing <code>Data</code> object and calculates the coefficients and LU factorization.

#### **Parameters**

```
d A Data object containing the financial parameters for the model
```

#### 4.3.3 Member Function Documentation

#### 4.3.3.1 get\_price()

```
std::vector< double > ensiie::CompleteCall::get_price () const
```

Retrieves the computed prices of the call option C(0, s), which are prices at time 0 for each level of underlying price s.

#### Returns

A vector containing the prices of the call option at time 0.

#### 4.3.3.2 pricing()

```
void ensiie::CompleteCall::pricing () [override], [virtual]
```

Computes the price of the European call option using the Crank-Nicolson method.

This method uses boundary conditions, LU factorization, and a system of equations to solve for the price of the European call option over a grid of time and price steps.

Implements ensiie::Complete.

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

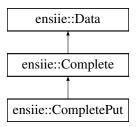
- · completecall.h
- · completecall.cpp

# 4.4 ensile::CompletePut Class Reference

A class to calculate and store the price of a European put option using Crank-Nicolson method to solve the Black-Scholes PDE.

```
#include <completeput.h>
```

Inheritance diagram for ensiie::CompletePut:



# **Public Member Functions**

- · CompletePut (double T, double r, double sigma, double K, double L, double M, double N)
  - Constructs a CompletePut object using financial parameters.
- CompletePut (const Data &d)

Constructs a CompletePut object using an existing Data object.

· void pricing () override

Computes the price of the European put option using the Crank-Nicolson method.

std::vector< double > get\_price () const

Retrieves the computed prices of the put option P(0, s), which are prices at time 0 for each level of underlying price s.

# Public Member Functions inherited from ensile::Complete

· Complete (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Complete object with the given parameters.

Complete (const Data &d)

Constructs a Complete object using an existing Data object.

std::vector< double > get\_alpha () const

Gets the alpha coefficients for the Crank\_Nicolson scheme.

std::vector< double > get\_beta () const

Gets the beta coefficients for the Crank Nicolson scheme.

std::vector< double > get\_gamma () const

Gets the gamma coefficients for the Crank\_Nicolson scheme.

std::vector< double > get\_low () const

Gets the lower matrix values of coefficients' matrix from the LU factorization.

std::vector< double > get\_up () const

Gets the upper matrix values of coefficients' matrix from the LU factorization.

#### Public Member Functions inherited from ensile::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

· double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

double get\_K () const

Gets the strike price of the option.

double get\_L () const

Gets the maximum asset price considered in the model.

double get\_M () const

Gets the number of time steps in the discretization.

• double get\_N () const

Gets the number of asset price steps in the discretization.

• double get\_dt () const

Gets the time step size.

• double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_I () const

Gets the discretized asset price vector.

#### **Additional Inherited Members**

#### Protected Member Functions inherited from ensile::Complete

• void coefficients\_computation ()

Computes the coefficients (alpha, beta, gamma) for the Crank-Nicolson scheme.

void lu\_factorization ()

Performs LU factorization of the system's matrix for Crank-Nicholson scheme.

#### Protected Member Functions inherited from ensile::Data

· void discretize ()

Discretizes the time and space domains based on the parameters of the model.

# Protected Attributes inherited from ensiie::Complete

```
std::vector< double > alpha_
std::vector< double > beta_
std::vector< double > gamma_
std::vector< double > low_
std::vector< double > up_
```

### Protected Attributes inherited from ensiie::Data

```
double T_
double r_
double sigma_
double K_
double L_
double M_
double N_
double dt_
double ds_
std::vector< double > t_
std::vector< double > I_
```

# 4.4.1 Detailed Description

A class to calculate and store the price of a European put option using Crank-Nicolson method to solve the Black-Scholes PDE.

This class is derived from the Complete class and implements the specific pricing algorithm for a European put option, including the computation of boundary conditions and the solution of the PDE using LU factorization.

### 4.4.2 Constructor & Destructor Documentation

### 4.4.2.1 CompletePut() [1/2]

Constructs a CompletePut object using financial parameters.

Initializes the financial parameters for the option pricing model and calculates the coefficients and LU factorization.

#### **Parameters**

T	Time to maturity (in years)
r	Market Risk-free interest rate
sigma	Volatility of the underlying asset
K	Strike price of the option
L	Maximum value of the underlying asset
М	Number of time steps
N	Number of price steps

#### 4.4.2.2 CompletePut() [2/2]

Constructs a CompletePut object using an existing Data object.

This constructor initializes the model using an existing <code>Data</code> object and calculates the coefficients and LU factorization.

#### **Parameters**

d A Data object containing the financial parameters for the model

#### 4.4.3 Member Function Documentation

#### 4.4.3.1 get\_price()

```
std::vector< double > ensiie::CompletePut::get_price () const
```

Retrieves the computed prices of the put option P(0, s), which are prices at time 0 for each level of underlying price s.

#### Returns

A vector containing the prices of the put option at time 0.

#### 4.4.3.2 pricing()

```
void ensiie::CompletePut::pricing () [override], [virtual]
```

Computes the price of the European put option using the Crank-Nicolson method.

This method uses boundary conditions, LU factorization, and a system of equations to solve for the price of the European put option over a grid of time and price steps.

Implements ensiie::Complete.

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

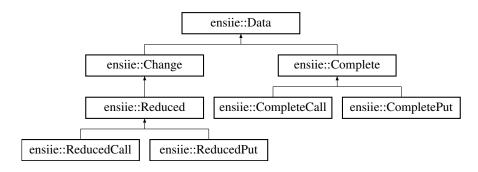
- · completeput.h
- completeput.cpp

# 4.5 ensiie::Data Class Reference

Contains all the data needed to solve the Black-Scholes pde.

#include <data.h>

Inheritance diagram for ensiie::Data:



#### **Public Member Functions**

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

• double get\_K () const

Gets the strike price of the option.

double get\_L () const

Gets the maximum asset price considered in the model.

• double get\_M () const

Gets the number of time steps in the discretization.

• double get\_N () const

 ${\it Gets the number of asset price steps in the discretization}.$ 

• double get\_dt () const

Gets the time step size.

• double get\_ds () const

Gets the asset price step size.

std::vector< double > get t () const

Gets the discretized time vector.

std::vector< double > get\_l () const

Gets the discretized asset price vector.

#### **Protected Member Functions**

· void discretize ()

Discretizes the time and space domains based on the parameters of the model.

# **Protected Attributes**

```
double T_
double r_
double sigma_
double K_
double L_
double M_
double N_
double dt_
double ds_
std::vector< double > t_
std::vector< double > I_
```

# 4.5.1 Detailed Description

Contains all the data needed to solve the Black-Scholes pde.

It also discretize the domain in time and asset value.

#### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 Data()

```
ensiie::Data::Data (
double T,
double r,
double sigma,
double K,
double L,
double M,
double N)
```

Constructs a Data object with the specified parameters.

Initializes the parameters of the model and calls the discretize() method to set up the time and space discretization.

#### **Parameters**

T	Total time to maturity (in years).
r	Market risk-free interest rate.
sigma	Volatility of the underlying asset.
K	Strike price of the option.
L	Maximum asset price considered in the model.
М	Number of time steps in the discretization.
N	Number of asset price steps in the discretization.

# **Exceptions**

std::invalid_argument	If any parameter is negative, or if ${\tt M}$ or ${\tt N}$ is zero.
-----------------------	---

# 4.5.3 Member Function Documentation

#### 4.5.3.1 discretize()

```
void ensiie::Data::discretize () [protected]
```

Discretizes the time and space domains based on the parameters of the model.

This method calculates the time step ( $dt_$ ) and space step ( $ds_$ ) and fills the vectors  $t_$  and  $l_$  with the discretized values.

#### 4.5.3.2 get\_ds()

```
double ensiie::Data::get_ds () const
```

Gets the asset price step size.

#### Returns

Asset price step size (ds).

#### 4.5.3.3 get\_dt()

```
double ensiie::Data::get_dt () const
```

Gets the time step size.

# Returns

Time step size (dt).

# 4.5.3.4 get\_K()

```
double ensiie::Data::get_K () const
```

Gets the strike price of the option.

#### Returns

Strike price (K).

#### 4.5.3.5 get\_L()

```
double ensiie::Data::get_L () const
```

Gets the maximum asset price considered in the model.

# Returns

Maximum asset price (L).

#### 4.5.3.6 get\_I()

```
std::vector< double > ensiie::Data::get_l () const
```

Gets the discretized asset price vector.

# Returns

Discretized asset price vector (I).

# 4.5.3.7 get\_M()

```
double ensiie::Data::get_M () const
```

Gets the number of time steps in the discretization.

#### Returns

Number of time steps (M).

# 4.5.3.8 get\_N()

```
double ensiie::Data::get_N () const
```

Gets the number of asset price steps in the discretization.

#### Returns

Number of asset price steps (N).

# 4.5.3.9 get\_r()

```
double ensiie::Data::get_r () const
```

Gets the market risk-free interest rate.

#### Returns

Market risk-free interest rate (r).

#### 4.5.3.10 get\_sigma()

```
double ensiie::Data::get_sigma () const
```

Gets the volatility of the underlying asset.

### Returns

Volatility (sigma).

#### 4.5.3.11 get\_T()

```
double ensiie::Data::get_T () const
```

Gets the total time to maturity.

#### Returns

Total time to maturity (T).

# 4.5.3.12 get\_t()

```
std::vector< double > ensiie::Data::get_t () const
```

Gets the discretized time vector.

#### Returns

Discretized time vector (t).

# 4.5.4 Member Data Documentation

# 4.5.4.1 ds\_

```
double ensiie::Data::ds_ [protected]
```

Lenght of the step in asset's value.

# 4.5.4.2 dt\_

```
double ensiie::Data::dt_ [protected]
```

Lenght of the step in time.

# 4.5.4.3 K\_

```
double ensiie::Data::K_ [protected]
```

Strike price of the option.

#### 4.5.4.4 L\_

```
double ensiie::Data::L_ [protected]
```

Underlying asset maximum price.

#### 4.5.4.5 I\_

```
std::vector<double> ensiie::Data::l_ [protected]
```

Discretized asset price vector.

# 4.5.4.6 M\_

```
double ensiie::Data::M_ [protected]
```

Number of steps in time discretization.

#### 4.5.4.7 N\_

```
double ensiie::Data::N_ [protected]
```

Number of steps in asset price discretization.

#### 4.5.4.8 r\_

```
double ensiie::Data::r_ [protected]
```

Market risk-free interest rate.

# 4.5.4.9 sigma\_

```
double ensiie::Data::sigma_ [protected]
```

Underlying asset volatility.

#### 4.5.4.10 T\_

```
double ensiie::Data::T_ [protected]
```

Time to maturity.

# 4.5.4.11 t\_

```
std::vector<double> ensiie::Data::t_ [protected]
```

Discretized time vector.

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

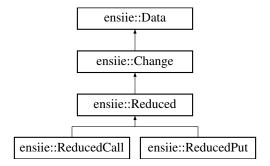
- · data.h
- data.cpp

## 4.6 ensiie::Reduced Class Reference

A base abstract class for solving financial models using implicit finite differences methods.

#include <reduced.h>

Inheritance diagram for ensiie::Reduced:



#### **Public Member Functions**

- Reduced (double T, double r, double sigma, double K, double L, double M, double N)
   Constructs a Reduced object using financial parameters.
- Reduced (const Data &d)

Constructs a Reduced object using an existing Data object.

• virtual void pricing ()=0

Pure virtual function to compute the price of the option.

• double get\_theta () const

Gets the theta coefficient for the implicit finite differences scheme.

std::vector< double > get\_low () const

 ${\it Gets the lower matrix values of coefficients' matrix from the LU factorization.}$ 

std::vector< double > get\_up () const

Gets the upper matrix values of coefficients' matrix from the LU factorization.

## Public Member Functions inherited from ensile::Change

• Change (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Change object using financial parameters.

Change (const Data &d)

Constructs a Reduced object using an existing Data object.

std::vector< double > get\_t\_changed () const

Gets the transformed time values.

std::vector< double > get\_l\_changed () const

Gets the transformed price values.

· double get\_dt\_changed () const

Gets the time variation step.

double get\_ds\_changed () const

Gets the price variation.

double get\_f () const

Gets the f parameter.

## Public Member Functions inherited from ensile::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

• double get\_K () const

Gets the strike price of the option.

• double get\_L () const

Gets the maximum asset price considered in the model.

• double get\_M () const

Gets the number of time steps in the discretization.

double get\_N () const

Gets the number of asset price steps in the discretization.

• double get dt () const

Gets the time step size.

• double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_I () const

Gets the discretized asset price vector.

#### **Protected Member Functions**

• void lu\_factorization ()

Performs LU factorization of the system's matrix for implicit finite differences scheme.

## Protected Member Functions inherited from ensile::Change

• void t\_transformation ()

Performs the time transformation.

• void I\_transformation ()

Performs the price transformation.

std::vector< double > price\_transformation (const std::vector< double > &v)

*Transforms the t=0 price vector.* 

## Protected Member Functions inherited from ensile::Data

• void discretize ()

Discretizes the time and space domains based on the parameters of the model.

#### **Protected Attributes**

```
double theta_
std::vector< double > low_
std::vector< double > up_
```

## Protected Attributes inherited from ensiie::Change

```
double f_
double dt_changed_
double ds_changed_
std::vector< double > t_changed_
std::vector< double > I changed
```

#### Protected Attributes inherited from ensile::Data

```
double T_
double r_
double sigma_
double K_
double L_
double M_
double N_
double dt_
double ds_
std::vector< double > t_
std::vector< double > I
```

## 4.6.1 Detailed Description

A base abstract class for solving financial models using implicit finite differences methods.

The Reduced class extends the Data and 'Change' classes and serves as an abstract base for solving the modified partial differential equations (PDEs) related to call and put options pricing. It provides the structure and common functionality for subclasses, involved in deterining prices for call and put options, computing coefficient used in implicit finite differences schemes, performing LU factorization of the coefficients' matrix, and providing access to the computed values.

This abstract class includes the method for LU factorization for efficient solving of tridiagonal systems.

Subclasses derived from Reduced implement the specific algorithm of the PDE solver, including terminal condition, to get the modified european call and put option pricing.

The class can be initialized via direct parameter input or using an existing Data object.

Note

This class cannot be instantiated directly due to its abstract nature.

## 4.6.2 Constructor & Destructor Documentation

## 4.6.2.1 Reduced() [1/2]

```
ensiie::Reduced::Reduced (
double T,
double r,
double sigma,
double K,
double L,
double M,
double N)
```

Constructs a Reduced object using financial parameters.

Initializes the financial parameters for the option pricing model and calculates change of variables, parameters and LU factorization.

#### **Parameters**

T	Time to maturity (in years)
r	Market Risk-free interest rate
sigma	Volatility of the underlying asset
K	Strike price of the option
L	Maximum value of the underlying asset
М	Number of time steps
N	Number of price steps

#### 4.6.2.2 Reduced() [2/2]

Constructs a Reduced object using an existing Data object.

This constructor initializes the model using an existing <code>Data</code> object and calculates change of variables, parameters and LU factorization.

#### **Parameters**

```
d A Data object containing the financial parameters for the model
```

## 4.6.3 Member Function Documentation

## 4.6.3.1 get\_low()

```
std::vector< double > ensiie::Reduced::get_low () const
```

Gets the lower matrix values of coefficients' matrix from the LU factorization.

#### Returns

A vector of lower matrix values (low).

#### 4.6.3.2 get\_theta()

```
double ensiie::Reduced::get_theta () const
```

Gets the theta coefficient for the implicit finite differences scheme.

#### Returns

A double with the theta coefficient.

## 4.6.3.3 get\_up()

```
std::vector< double > ensiie::Reduced::get_up () const
```

Gets the upper matrix values of coefficients' matrix from the LU factorization.

#### Returns

A vector of upper matrix values (up).

## 4.6.3.4 lu\_factorization()

```
void ensiie::Reduced::lu_factorization () [protected]
```

Performs LU factorization of the system's matrix for implicit finite differences scheme.

This method calculates the lower (low) and upper (up) matrices, which are used to solve the system of equations iteratively.

## 4.6.3.5 pricing()

```
virtual void ensiie::Reduced::pricing () [pure virtual]
```

Pure virtual function to compute the price of the option.

This is a pure virtual function that is implemented by its derived class.

It defines the computation of Modified Call and Put options prices. The specific implementation will based on different terminal conditions.

Note

Since this is a pure virtual function, Complete cannot be instantiated directly.

Implemented in ensiie::ReducedCall, and ensiie::ReducedPut.

## 4.6.4 Member Data Documentation

#### 4.6.4.1 theta

```
double ensiie::Reduced::theta_ [protected]
```

Parameter to solve the linear system.

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

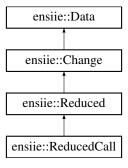
- · reduced.h
- reduced.cpp

## 4.7 ensiie::ReducedCall Class Reference

A class to calculate and store the price of a European call option using implicit finite differences method to solve the modified Black-Scholes Heat Equation.

```
#include <reducedcall.h>
```

Inheritance diagram for ensiie::ReducedCall:



#### **Public Member Functions**

- ReducedCall (double T, double r, double sigma, double K, double L, double M, double N)
  - Constructs a ReducedCall object using financial parameters.
- · ReducedCall (const Data &d)
  - Constructs a ReducedCall object using an existing Data object.
- void pricing () override
  - Computes the price of the European call option using the finite difference method on transformed PDE.
- std::vector< double > get\_price () const
  - Retrieves the computed prices of the put option  $P_{tilda}(0, s)$ , which are prices at time 0 for each level of underlying price s.

#### Public Member Functions inherited from ensile::Reduced

• Reduced (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Reduced object using financial parameters.

• Reduced (const Data &d)

Constructs a Reduced object using an existing Data object.

double get\_theta () const

Gets the theta coefficient for the implicit finite differences scheme.

std::vector< double > get\_low () const

Gets the lower matrix values of coefficients' matrix from the LU factorization.

std::vector< double > get\_up () const

Gets the upper matrix values of coefficients' matrix from the LU factorization.

## Public Member Functions inherited from ensile::Change

• Change (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Change object using financial parameters.

• Change (const Data &d)

Constructs a Reduced object using an existing Data object.

std::vector< double > get\_t\_changed () const

Gets the transformed time values.

std::vector< double > get\_l\_changed () const

Gets the transformed price values.

double get\_dt\_changed () const

Gets the time variation step.

· double get ds changed () const

Gets the price variation.

• double get\_f () const

Gets the f parameter.

## Public Member Functions inherited from ensile::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

double get\_K () const

Gets the strike price of the option.

• double get L () const

Gets the maximum asset price considered in the model.

double get\_M () const

Gets the number of time steps in the discretization.

• double get N () const

Gets the number of asset price steps in the discretization.

double get\_dt () const

Gets the time step size.

• double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_l () const

Gets the discretized asset price vector.

#### **Additional Inherited Members**

## Protected Member Functions inherited from ensile::Reduced

• void lu\_factorization ()

Performs LU factorization of the system's matrix for implicit finite differences scheme.

## Protected Member Functions inherited from ensile::Change

void t\_transformation ()

Performs the time transformation.

void l\_transformation ()

Performs the price transformation.

std::vector< double > price\_transformation (const std::vector< double > &v)

Transforms the t=0 price vector.

## Protected Member Functions inherited from ensile::Data

· void discretize ()

Discretizes the time and space domains based on the parameters of the model.

## Protected Attributes inherited from ensile::Reduced

```
· double theta_
```

- $std::vector < double > low_$
- std::vector< double > up\_

## Protected Attributes inherited from ensiie::Change

- double f\_
- · double dt\_changed\_
- · double ds\_changed\_
- std::vector< double > t\_changed\_
- std::vector< double > I\_changed\_

## Protected Attributes inherited from ensiie::Data

```
double T_
double r_
double sigma_
double K_
double L_
double M_
double N_
double dt_
double ds_
std::vector< double > t_
```

## 4.7.1 Detailed Description

std::vector< double > l\_

A class to calculate and store the price of a European call option using implicit finite differences method to solve the modified Black-Scholes Heat Equation.

This class is derived from the Reduced class and implements the specific pricing algorithm for a European call option, including the computation of terminal condition and the solution of the modified PDE using LU factorization.

#### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 ReducedCall() [1/2]

Constructs a ReducedCall object using financial parameters.

Initializes the financial parameters for the option pricing model and calculates change of variables, parameters and LU factorization.

#### **Parameters**

T	Time to maturity (in years)
r	Market Risk-free interest rate
sigma	Volatility of the underlying asset
K	Strike price of the option
L	Maximum value of the underlying asset
М	Number of time steps
N	Number of price steps

## 4.7.2.2 ReducedCall() [2/2]

```
\begin{tabular}{ll} \tt ensiie::ReducedCall::ReducedCall::Call ( & d) \end{tabular}
```

Constructs a  $\ensuremath{\mathsf{ReducedCall}}$  object using an existing  $\ensuremath{\mathsf{Data}}$  object.

This constructor initializes the model using an existing  $\mathtt{Data}$  object and calculates change of variables, parameters and LU factorization.

#### **Parameters**

d A Data object containing the financial parameters for the model

#### 4.7.3 Member Function Documentation

## 4.7.3.1 get\_price()

```
std::vector< double > ensiie::ReducedCall::get_price () const
```

Retrieves the computed prices of the put option P\_tilda(0, s), which are prices at time 0 for each level of underlying price s.

#### Returns

A vector containing the prices of the put option at time 0.

#### 4.7.3.2 pricing()

```
void ensiie::ReducedCall::pricing () [override], [virtual]
```

Computes the price of the European call option using the finite difference method on transformed PDE.

This method uses terminal condition, LU factorization, and a system of equations to solve for the price of the European call option over a grid of time and price steps. In the methods the matrix prices has the form prices[raws j],[columns i]

Terminal condition for t=T

Iterative solution of the matrix prices by column

Solution to the problem Ly=b, computing b and then y

Solution to the problem Ux=y

Extraction of the column of the matrix corresponding to C\_tilda(0, s)

Changing of the price vector with real prices

Implements ensiie::Reduced.

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

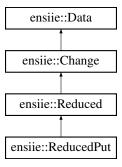
- · reducedcall.h
- · reducedcall.cpp

## 4.8 ensije::ReducedPut Class Reference

A class to calculate and store the price of a European put option using implicit finite differences method to solve the modified Black-Scholes Heat Equation.

#include <reducedput.h>

Inheritance diagram for ensiie::ReducedPut:



#### **Public Member Functions**

• ReducedPut (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a ReducedPut object using financial parameters.

ReducedPut (const Data &d)

Constructs a ReducedPut object using an existing Data object.

· void pricing () override

Computes the price of the European put option using the finite difference method on transformed PDE.

• std::vector< double > get\_price () const

Retrieves the computed prices of the put option P\_tilda(0, s), which are prices at time 0 for each level of underlying price s.

## Public Member Functions inherited from ensiie::Reduced

• Reduced (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Reduced object using financial parameters.

• Reduced (const Data &d)

Constructs a Reduced object using an existing Data object.

• double get\_theta () const

Gets the theta coefficient for the implicit finite differences scheme.

std::vector< double > get\_low () const

Gets the lower matrix values of coefficients' matrix from the LU factorization.

std::vector< double > get\_up () const

Gets the upper matrix values of coefficients' matrix from the LU factorization.

## Public Member Functions inherited from ensile::Change

• Change (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Change object using financial parameters.

Change (const Data &d)

Constructs a Reduced object using an existing Data object.

std::vector< double > get\_t\_changed () const

Gets the transformed time values.

• std::vector< double > get\_l\_changed () const

Gets the transformed price values.

· double get\_dt\_changed () const

Gets the time variation step.

• double get ds changed () const

Gets the price variation.

double get\_f () const

Gets the f parameter.

#### Public Member Functions inherited from ensile::Data

• Data (double T, double r, double sigma, double K, double L, double M, double N)

Constructs a Data object with the specified parameters.

double get\_T () const

Gets the total time to maturity.

• double get\_r () const

Gets the market risk-free interest rate.

• double get\_sigma () const

Gets the volatility of the underlying asset.

• double get\_K () const

Gets the strike price of the option.

• double get\_L () const

Gets the maximum asset price considered in the model.

• double get\_M () const

Gets the number of time steps in the discretization.

double get\_N () const

Gets the number of asset price steps in the discretization.

• double get\_dt () const

Gets the time step size.

double get\_ds () const

Gets the asset price step size.

std::vector< double > get\_t () const

Gets the discretized time vector.

std::vector< double > get\_l () const

Gets the discretized asset price vector.

#### **Additional Inherited Members**

## Protected Member Functions inherited from ensite::Reduced

void lu\_factorization ()

Performs LU factorization of the system's matrix for implicit finite differences scheme.

## Protected Member Functions inherited from ensile::Change

```
    void t_transformation ()
```

Performs the time transformation.

void I transformation ()

Performs the price transformation.

std::vector< double > price\_transformation (const std::vector< double > &v)

Transforms the t=0 price vector.

#### Protected Member Functions inherited from ensile::Data

· void discretize ()

Discretizes the time and space domains based on the parameters of the model.

#### Protected Attributes inherited from ensile::Reduced

```
    double theta
```

- std::vector< double > low\_
- std::vector< double > up\_

## Protected Attributes inherited from ensiie::Change

```
    double f
```

- double dt changed
- · double ds\_changed\_
- std::vector< double > t\_changed\_
- std::vector< double > I\_changed\_

#### Protected Attributes inherited from ensile::Data

```
    double T
```

- double r
- double sigma\_
- double K
- double L\_
- double M
- double N\_
- double dt\_
- double ds
- std::vector< double > t\_
- std::vector< double > I

## 4.8.1 Detailed Description

A class to calculate and store the price of a European put option using implicit finite differences method to solve the modified Black-Scholes Heat Equation.

This class is derived from the Reduced class and implements the specific pricing algorithm for a European put option, including the computation of terminal condition and the solution of the modified PDE using LU factorization.

## 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 ReducedPut() [1/2]

Constructs a ReducedPut object using financial parameters.

Initializes the financial parameters for the option pricing model and calculates change of variables, parameters and LU factorization.

#### **Parameters**

T	Time to maturity (in years)
r	Market Risk-free interest rate
sigma	Volatility of the underlying asset
K	Strike price of the option
L	Maximum value of the underlying asset
М	Number of time steps
N	Number of price steps

## 4.8.2.2 ReducedPut() [2/2]

Constructs a ReducedPut object using an existing Data object.

This constructor initializes the model using an existing <code>Data</code> object and calculates change of variables, parameters and LU factorization.

#### **Parameters**

d A Data object containing the financial parameters for the model

#### 4.8.3 Member Function Documentation

#### 4.8.3.1 get\_price()

```
std::vector< double > ensiie::ReducedPut::get_price () const
```

Retrieves the computed prices of the put option  $P_{tilda}(0, s)$ , which are prices at time 0 for each level of underlying price s.

#### Returns

A vector containing the prices of the put option at time 0.

## 4.8.3.2 pricing()

```
void ensiie::ReducedPut::pricing () [override], [virtual]
```

Computes the price of the European put option using the finite difference method on transformed PDE.

This method uses terminal condition, LU factorization, and a system of equations to solve for the price of the European put option over a grid of time and price steps. In the methods the matrix prices has the form prices[raws j],[columns i]

Terminal condition for t=T

Iterative solution of the matrix prices by column

Solution to the problem Ly=b, comuting b and then y

Solution to the problem Ux=y

Extraction of the column of the matrix corresponding to P\_tilda(0, s)

Changing of the price vector with real prices

Implements ensiie::Reduced.

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

- · reducedput.h
- · reducedput.cpp

## 4.9 ensiie::SDL Class Reference

A class to manage the SDL window and render put and call prices.

```
#include <sdl.h>
```

#### **Public Member Functions**

SDL (const std::string &title, int width, int height)

Constructs the SDL object and initializes the SDL video subsystem.

• ∼SDL ()

Destructor to clean up SDL resources.

• void clearScreen ()

Clears the screen with a white background.

• void drawAxes ()

Draws the axes on the screen.

void drawGraph\_red (const std::vector< double > &values)

Draws a graph on the screen based on the provided values.

void drawGraph\_green (const std::vector< double > &values)

Draws a graph on the screen based on the provided values.

void updateScreen ()

Updates the screen with the rendered content.

void waitForClose ()

Waits for the user to close the window.

## 4.9.1 Detailed Description

A class to manage the SDL window and render put and call prices.

This class provides functionalities to create an SDL window, render graphical elements (axes and graphs), and manage events for the window. It includes methods for clearing the screen, drawing axes, drawing graphs, updating the screen, and handling window close events.

## 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 SDL()

Constructs the SDL object and initializes the SDL video subsystem.

This constructor creates an SDL window with the specified title and dimensions and sets up the renderer. If any SDL function call fails, an exception is thrown.

#### **Parameters**

title	The title of the SDL window.
width	The width of the SDL window.
height	The height of the SDL window.

#### **Exceptions**

```
std::runtime_error | If SDL initialization, window creation, or renderer creation fails.
```

## 4.9.2.2 ∼SDL()

```
ensiie::SDL::~SDL ()
```

Destructor to clean up SDL resources.

This destructor destroys the SDL renderer and window and calls SDL Quit to clean up the SDL library.

#### 4.9.3 Member Function Documentation

#### 4.9.3.1 clearScreen()

```
void ensiie::SDL::clearScreen ()
```

Clears the screen with a white background.

This method sets the drawing color to white and clears the renderers target, essentially clearing the screen.

#### 4.9.3.2 drawAxes()

```
void ensiie::SDL::drawAxes ()
```

Draws the axes on the screen.

This method draws the X and Y axes on the screen. The X axis is drawn along the bottom of the window, and the Y axis is drawn along the left side.

#### 4.9.3.3 drawGraph\_green()

Draws a graph on the screen based on the provided values.

This method plots a graph by drawing lines between data points. The graph is scaled according to the screen size and the range of the data. The graph is drawn in green color.

#### **Parameters**

values	A vector containing the data points to be plotted on the graph.
--------	---

#### 4.9.3.4 drawGraph\_red()

Draws a graph on the screen based on the provided values.

This method plots a graph by drawing lines between data points. The graph is scaled according to the screen size and the range of the data. The graph is drawn in red color.

## **Parameters**

values	A vector containing the data points to be plotted on the graph.
--------	---

#### 4.9.3.5 updateScreen()

```
void ensiie::SDL::updateScreen ()
```

Updates the screen with the rendered content.

This method presents the rendered content to the screen. It is called to update the window after drawing operations.

#### 4.9.3.6 waitForClose()

```
void ensiie::SDL::waitForClose ()
```

Waits for the user to close the window.

This method enters a loop that listens for SDL events, and the window will stay open until the user closes it. Once the user closes the window, the loop terminates.

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

- sdl.h
- · sdl.cpp

## **Chapter 5**

## **File Documentation**

## 5.1 change.h

```
00001 #pragma once
00002 #include "data.h"
00003 #include <cmath>
00004
00005 namespace ensiie
00006 {
00016
          class Change : public Data
00017
00018
          protected:
              double f_;
double dt_changed_;
double ds_changed_;
00019
00020
00021
              std::vector<double> t_changed_;
00023
              std::vector<double> l_changed_;
00024
00031
              void t_transformation();
00032
00039
              void 1 transformation();
00040
00050
               std::vector<double> price_transformation(const std::vector<double>& v);
00051
00052
          public:
00053
              Change (double T, double r, double sigma, double K, double L, double M, double N);
00068
00069
00078
               Change(const Data& d);
00079
00085
               std::vector<double> get_t_changed() const;
00086
00092
               std::vector<double> get 1 changed() const;
00099
               double get_dt_changed() const;
00100
00106
               double get_ds_changed() const;
00107
00113
               double get_f() const;
00114
          };
00115 }
```

## 5.2 complete.h

```
00001 #pragma once
00002 #include "data.h"
00003 #include <algorithm>
00004 #include <cmath>
00005
00006 namespace ensiie
00007 {
00030
          class Complete : public Data
00031
00032
         protected:
00033
             std::vector<double> alpha ;
00034
             std::vector<double> beta_;
             std::vector<double> gamma_;
```

50 File Documentation

```
00036
              std::vector<double> low_;
00037
              std::vector<double> up_;
00038
00045
              void coefficients_computation();
00046
00053
              void lu_factorization();
00054
00055
          public:
00056
00071
              Complete (double T, double r, double sigma, double K, double L, double M, double N);
00072
00085
              Complete (const Data& d);
00086
00097
              virtual void pricing() = 0;
00098
00103
              std::vector<double> get_alpha() const;
00104
00109
              std::vector<double> get_beta() const;
00110
00115
              std::vector<double> get_gamma() const;
00116
00123
              std::vector<double> get_low() const;
00124
00131
              std::vector<double> get_up() const;
00132
          };
00133 }
```

## 5.3 completecall.h

```
00001 #pragma once
00002 #include "complete.h"
00004 namespace ensiie
00005 {
00016
          class CompleteCall : public Complete
00017
00019
              std::vector<double> price_;
00020
00021
          public:
00022
00037
              CompleteCall(double T, double r, double sigma, double K, double L, double M, double N);
00038
00047
              CompleteCall(const Data& d);
00048
00055
              void pricing() override;
00056
00063
              std::vector<double> get_price() const;
00064
          } ;
00065 }
```

## 5.4 completeput.h

```
00001 #pragma once
00002 #include "complete.h"
00003
00004 namespace ensiie
00005 {
00016
          class CompletePut : public Complete
00017
00019
              std::vector<double> price_;
00020
00021
          public:
00022
00037
              CompletePut (double T, double r, double sigma, double K, double L, double M, double N);
00038
00047
              CompletePut(const Data& d);
00048
00055
              void pricing() override;
00056
00063
              std::vector<double> get_price() const;
00064
          };
00065 }
```

## 5.5 data.h

00001 #pragma once

5.6 reduced.h

```
00002 #include <vector>
00003 #include <stdexcept>
00004
00005 namespace ensiie
00006 {
00013
          class Data
00014
00015
          protected:
00016
              double T_;
00017
              double r_{-};
00018
              double sigma_;
00019
              double K_;
00020
              double L_;
00021
              double M_;
00022
              double N_{-};
00023
              double dt_;
00024
              double ds ;
00025
              std::vector<double> t_;
              std::vector<double> 1_;
00026
00034
              void discretize();
00035
00036
          public:
00053
              Data(double T, double r, double sigma, double K, double L, double M, double N);
00054
00059
              double get_T() const;
00060
00065
              double get_r() const;
00066
00071
              double get_sigma() const;
00072
00077
              double get K() const;
00078
00083
              double get_L() const;
00084
00089
              double get_M() const;
00090
00095
              double get N() const;
00096
00101
              double get_dt() const;
00102
00107
              double get_ds() const;
00108
00113
              std::vector<double> get t() const;
00114
00119
              std::vector<double> get_1() const;
00120
00121
          };
00122 }
```

## 5.6 reduced.h

```
00001 #pragma once
00002 #include "change.h"
00003 #include <algorithm>
00004
00005 namespace ensiie
00006 {
00027
          class Reduced : public Change
00028
00029
          protected:
              double theta_;
00030
00031
              std::vector<double> low ;
00032
              std::vector<double> up_;
00033
00040
              void lu_factorization();
00041
00042
          public:
00043
00058
              Reduced (double T, double r, double sigma, double K, double L, double M, double N);
00059
00068
              Reduced(const Data& d);
00069
08000
              virtual void pricing() = 0;
00081
00086
              double get_theta() const;
00087
00094
              std::vector<double> get_low() const;
00095
00102
              std::vector<double> get_up() const;
00103
          };
00104 }
```

52 File Documentation

## 5.7 reducedcall.h

```
00001 #pragma once
00002 #include "reduced.h"
00003
00004 namespace ensiie
00005 {
00016
          class ReducedCall : public Reduced
00017
00019
              std::vector<double> price_;
00020
00021
          public:
00022
00037
              ReducedCall(double T, double r, double sigma, double K, double L, double M, double N);
00038
00047
              ReducedCall(const Data& d);
00048
00056
              void pricing() override;
00057
00064
              std::vector<double> get_price() const;
00065
00066 }
```

## 5.8 reducedput.h

```
00001 #pragma once
00002 #include "reduced.h"
00003
00004 namespace ensiie
00005 {
00016
          class ReducedPut : public Reduced
00017
00019
              std::vector<double> price_;
00020
          public:
00021
00022
00037
              ReducedPut (double T, double r, double sigma, double K, double L, double M, double N);
00038
00047
              ReducedPut(const Data& d);
00048
00056
              void pricing() override;
00057
00064
              std::vector<double> get_price() const;
00065
          };
00066 }
```

## 5.9 sdl.h

```
00001 #pragma once
00002 #include <SDL2/SDL.h>
00003 #include <vector>
00004 #include <string>
00005
00006 namespace ensiie {
00007
00017
          class SDL
00018
          {
00019
00020
              SDL Window* window;
00021
              SDL_Renderer* renderer;
00022
              int screenWidth;
00023
              int screenHeight;
00025
00026
              SDL(const std::string& title, int width, int height);
00038
00039
00045
              ~SDL();
00046
00052
              void clearScreen();
00053
00060
              void drawAxes();
00061
00070
              void drawGraph red(const std::vector<double>& values);
00071
00080
              void drawGraph_green(const std::vector<double>& values);
00081
00087
              void updateScreen();
00088
00095
              void waitForClose();
00096
          };
00097 }
```

# Index

~SDL	Complete, 15
ensiie::SDL, 47	get_alpha, 15
	get_beta, 16
Change	get_gamma, 16
ensiie::Change, 9	get_low, 16
clearScreen	get_up, 16
ensiie::SDL, 47	lu_factorization, 16
coefficients_computation	pricing, 17
ensiie::Complete, 15	ensiie::CompleteCall, 17
Complete	CompleteCall, 20
ensiie::Complete, 15	get price, 20
CompleteCall	pricing, 20
ensiie::CompleteCall, 20	ensiie::CompletePut, 21
CompletePut	CompletePut, 23, 24
ensiie::CompletePut, 23, 24	get_price, 24
	pricing, 24
Data	ensiie::Data, 25
ensiie::Data, 26	•
discretize	Data, 26 discretize, 27
ensiie::Data, 27	,
drawAxes	ds_, 29
ensiie::SDL, 47	dt_, 29
drawGraph green	get_ds, 27
ensiie::SDL, 48	get_dt, 27
drawGraph_red	get_K, 27
ensiie::SDL, 48	get_L, 27
ds_	get_l, <mark>27</mark>
ensiie::Data, 29	get_M, 28
	get_N, 28
ds_changed_	get_r, 28
ensiie::Change, 12	get_sigma, 28
dt_	get_T, 28
ensiie::Data, 29	get_t, 29
dt_changed_	K_, 29
ensiie::Change, 12	L_, 29
ensiie::Change, 7	l_, 29
Change, 9	M_, 30
	N_, 30
ds_changed_, 12	r_, 30
dt_changed_, 12	sigma_, <mark>30</mark>
f_, 12	T_, 30
get_ds_changed, 10	t_, 30
get_dt_changed, 10	ensiie::Reduced, 31
get_f, 10	get_low, 34
get_l_changed, 10	get_theta, 34
get_t_changed, 10	get_up, 35
I_transformation, 11	lu_factorization, 35
price_transformation, 11	pricing, 35
t_transformation, 11	Reduced, 34
ensiie::Complete, 12	theta_, 36
coefficients_computation, 15	

54 INDEX

ensiie::ReducedCall, 36	get_sigma
get_price, 41	ensiie::Data, <mark>28</mark>
pricing, 41	get_T
ReducedCall, 39	ensiie::Data, <mark>28</mark>
ensiie::ReducedPut, 42	get_t
get_price, 45	ensiie::Data, 29
pricing, 45	get_t_changed
ReducedPut, 45	ensiie::Change, 10
ensiie::SDL, 46	get theta
~SDL, 47	ensiie::Reduced, 34
clearScreen, 47	get_up
drawAxes, 47	ensiie::Complete, 16
drawGraph_green, 48	ensiie::Reduced, 35
drawGraph_red, 48	К
SDL, 47	<del>-</del>
updateScreen, 48	ensiie::Data, 29
waitForClose, 48	1
	L_
f_	ensiie::Data, 29
ensiie::Change, 12	<u></u>
	ensiie::Data, 29
get_alpha	I_transformation
ensiie::Complete, 15	ensiie::Change, 11
get_beta	lu_factorization
ensiie::Complete, 16	ensiie::Complete, 16
get_ds	ensiie::Reduced, 35
ensiie::Data, 27	
get_ds_changed	M_
ensiie::Change, 10	ensiie::Data, 30
get_dt	
ensiie::Data, 27	N_
	ensiie::Data, 30
get_dt_changed	
ensiie::Change, 10	price_transformation
get_f	ensiie::Change, 11
ensiie::Change, 10	pricing
get_gamma	ensiie::Complete, 17
ensiie::Complete, 16	ensiie::CompleteCall, 20
get_K	ensiie::CompletePut, 24
ensiie::Data, <mark>27</mark>	ensiie::Reduced, 35
get_L	ensiie::ReducedCall, 41
ensiie::Data, 27	
get I	ensiie::ReducedPut, 45
ensiie::Data, 27	
get_I_changed	r_
ensiie::Change, 10	ensiie::Data, 30
get_low	Reduced
ensiie::Complete, 16	ensiie::Reduced, 34
•	ReducedCall
ensiie::Reduced, 34	ensiie::ReducedCall, 39
get_M	ReducedPut
ensiie::Data, 28	ensiie::ReducedPut, 45
get_N	
ensiie::Data, <mark>28</mark>	SDL
get_price	ensiie::SDL, 47
ensiie::CompleteCall, 20	sigma_
ensiie::CompletePut, 24	ensiie::Data, 30
ensiie::ReducedCall, 41	555 ata, 50
ensiie::ReducedPut, 45	Т
get_r	ensiie::Data, 30
ensiie::Data, 28	t
, <del></del>	<del>-</del> _

INDEX 55

```
ensiie::Data, 30
t_transformation
ensiie::Change, 11
theta_
ensiie::Reduced, 36
updateScreen
ensiie::SDL, 48
waitForClose
ensiie::SDL, 48
```