Black and Scholes pricing model

Generated by Doxygen 1.12.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

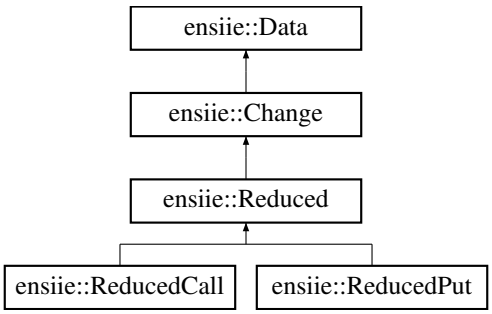
Class Documentation

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_l () const

  Gets the discretized asset price vector.

Protected Member Functions

- 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 ensiie::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 > **l_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 > l_

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
M	Number of time steps
N	Number of price steps

4.1.2.2 Change() [2/2]

```
ensiie::Change::Change (
            const Data & d)
```

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 l_transformation()

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

Performs the price transformation.

This function calculates the price transformation `l` based on the parameter `K` and stores the results in `l_changed_` and `ds_changed_`. This aproximation is done as it's not possible to transform S=0

4.1.3.7 price_transformation()

```
std::vector< double > ensiie::Change::price_transformation (
            const std::vector< double > & v)  [protected]
```

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_` and `l_changed_`, 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_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:

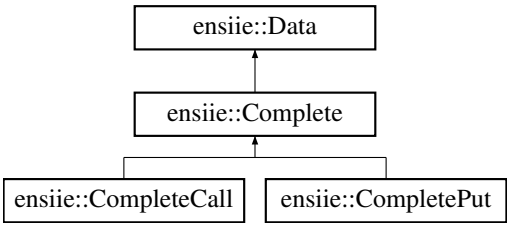
- change.h
- change.cpp

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 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_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 ensiie::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 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 > l_

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]

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

Constructs a Complete object with the given parameters.

This constructor initializes the base Data 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.
M	Number of time steps.
N	Number of space steps.

4.2.2.2 Complete() [2/2]

```
ensiie::Complete::Complete (
            const Data & d)
```

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:

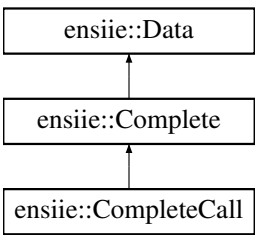
- complete.h
- complete.cpp

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 ensiie::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 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_l () const

  Gets the discretized asset price vector.

Additional Inherited Members

Protected Member Functions inherited from ensiie::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 ensiie::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 > l_

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]

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

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
M	Number of time steps
N	Number of price steps

4.3.2.2 CompleteCall() [2/2]

```
ensiie::CompleteCall::CompleteCall (
            const Data & d)
```

Constructs a CompleteCall object using an existing Data object.

This constructor initializes the model using an existing `Data` 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:

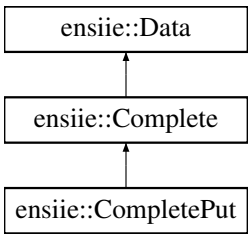
- completecall.h
- completecall.cpp

4.4 ensiie::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 ensiie::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 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_l () const

  Gets the discretized asset price vector.

Additional Inherited Members

Protected Member Functions inherited from ensiie::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 ensiie::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 > l_

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]

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

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
M	Number of time steps
N	Number of price steps

4.4.2.2 CompletePut() [2/2]

```
ensiie::CompletePut::CompletePut (
            const Data & d)
```

Constructs a CompletePut object using an existing Data object.

This constructor initializes the model using an existing `Data` 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:

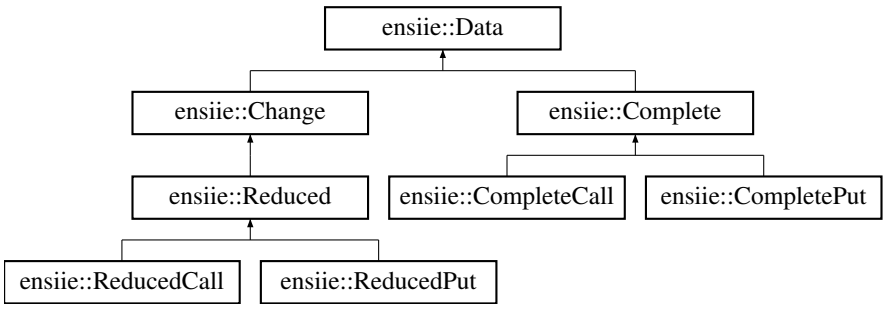
- completeput.h
- completeput.cpp

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

  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 > l_

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.
M	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 `M` or `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_l()

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

Gets the discretized asset price vector.

Returns

Discretized asset price vector (l).

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 l_

```
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:

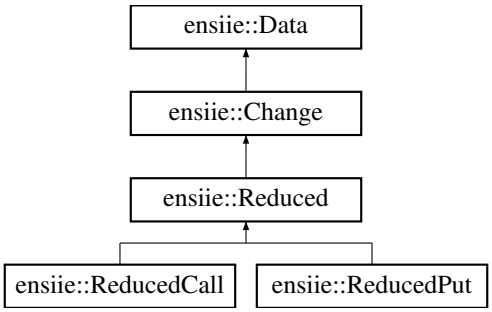
- data.h
- data.cpp

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

  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 ensiie::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 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_l () 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 ensiie::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 ensiie::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 > **l_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 > l_

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
M	Number of time steps
N	Number of price steps

4.6.2.2 Reduced() [2/2]

```
ensiie::Reduced::Reduced (
            const Data & d)
```

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 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:

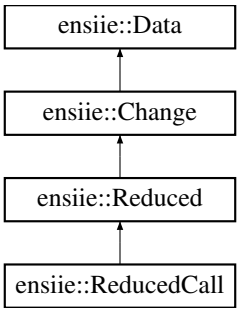
- reduced.h
- reduced.cpp

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 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 ensiie::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 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_l () const

  Gets the discretized asset price vector.

Additional Inherited Members

Protected Member Functions inherited from ensiie::Reduced

- void lu_factorization ()

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

Protected Member Functions inherited from ensiie::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 ensiie::Data

- void discretize ()

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

Protected Attributes inherited from ensiie::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 > **l_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 > l_

4.7.1 Detailed Description

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]

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

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
M	Number of time steps
N	Number of price steps

4.7.2.2 ReducedCall() [2/2]

```
ensiie::ReducedCall::ReducedCall (
            const Data & d)
```

Constructs a ReducedCall object using an existing Data object.

This constructor initializes the model using an existing `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:

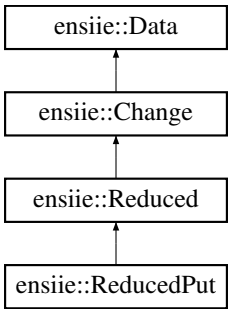
- reducedcall.h
- reducedcall.cpp

4.8 ensiie::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 ensiie::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 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_l () const

  Gets the discretized asset price vector.

Additional Inherited Members

Protected Member Functions inherited from ensiie::Reduced

- void lu_factorization ()

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

Protected Member Functions inherited from ensiie::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 ensiie::Data

- void discretize ()

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

Protected Attributes inherited from ensiie::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 > **l_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 > l_

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]

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

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
M	Number of time steps
N	Number of price steps

4.8.2.2 ReducedPut() [2/2]

```
ensiie::ReducedPut::ReducedPut (
            const Data & d)
```

Constructs a ReducedPut object using an existing Data object.

This constructor initializes the model using an existing `Data` 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()

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

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()

```
void ensiie::SDL::drawGraph_green (
            const std::vector< double > & values)
```

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()

```
void ensiie::SDL::drawGraph_red (
            const std::vector< double > & values)
```

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:
00019         double f_;
00020         double dt_changed_;
00021         double ds_changed_;
00022         std::vector<double> t_changed_;
00023         std::vector<double> l_changed_;
00024
00031         void t_transformation();
00032
00039         void l_transformation();
00040
00050         std::vector<double> price_transformation(const std::vector<double>& v);
00051
00052     public:
00053
00068         Change(double T, double r, double sigma, double K, double L, double M, double N);
00069
00078         Change(const Data& d);
00079
00085         std::vector<double> get_t_changed() const;
00086
00092         std::vector<double> get_l_changed() const;
00093
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_;
00035         std::vector<double> gamma_;
```

```
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"
00003
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
```

```
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_;
00026         std::vector<double> l_;
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_l() 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:
00030         double theta_;
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
00080         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 }
```

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
00021     public:
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     public:
00026
00038         SDL(const std::string& title, int width, int height);
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