Option Pricer

We look at the various classes involved in the design of the project and highlight the design choices taken while writing the classes. OOP paradigms such as composition, inheritance and polymorphism (both static and runtime) have been used appropriately to encapsulate functionality and increase code readability and maintainability.

Equity Class

This class represents the stock securities from which the options are derived for the purview of this project.

The class has the following as its member variables:

- Stock Price
- Cost of Carry
- Volatility of the Equity

Option Class

This class represents all options in the purview of this project. It is considered as an **abstract** base class which is inherited by other “specialized” options.

The class has the following as its member variables:

- OptionType
- Equity
- Strike Price
- Risk Free Rate

An option can be of type *put* or *call* which is kept as an enum represented by OptionType. Enum is better than string as it is more efficient to pass around in function calls and gives the benefit of avoiding string matching and subsequent error handling for wrong strings. OptionType is not considered to change once an option is created, therefore there is no toggle and no explicit setter for this member variable.

The Option-Equity relationship is “has a” which is referred to as composition. An option has equity as the underlying security.

Important methods like Price() of an option are made pure virtual to enforce child classes to implement this method. Other methods are kept virtual and a “NoImplementationException” is thrown when a child class does not override these methods.

EuropeanOption class

This is a derived class from the Option class to represent European Options with no option to change time to expiry.
It implements the Price(), Delta(), Gamma() methods based on Black Scholes model.
The class has the following as its member variables:

- Time to expiry
- (Inherited members from Option class)

Since a European option “is a” Option, this relationship is called inheritance which is what we do.

PerpetualAmericanOption class

This is a derived class from the Option class to represent American options with time to expiry always infinity.

Mesh class

This class represents the mesh creation functionality and encapsulates this functionality pretty well. It is a generic class. Static polymorphism is used here.

The class has the following as its member variables:

- Start element value of mesh
- Step size for the mesh
- Size of the mesh

It is made as a generic class as this design is more flexible and robust. It can be used by other non-primitive types as well to create a mesh of that type.
Note: for the use case of this project, only the use of double is employed.

It has the GenerateMatrix() member function which encapsulates the mesh creation functionality according to the guidelines of the Class’s member variables.

Since, this is made as a generic class, to enhance the flexibility of this class and the consumers of this class, the GenerateMesh() function only expects the + operator to be overloaded by the consumer. Therefore, the use of * operator was avoided while making a design decision here.

Matrix Class

Represents a 2D matrix class.

The class has the following as its member variables:

- vector of vectors which carries the matrix data

- Number of rows
- Number of Columns

It uses the standard library's vector for the data store part and uses vector's functionalities which are readily available. Functionality like push_back, element access, memory management, assignment operator all come in handy due to the use of vector and thus we avoid reinventing the wheel by using a self made Array class.

The class also has a member function Transpose() which transposes the matrix. This functionality comes handy as part of MeshMatrixGenerator (discussed ahead).

MatrixGeneratorInterface

This is an abstract class which expects a GenerateMatrix method to be implemented by the child class.

There can be numerous ways of generating a matrix. Every unique way can be implemented by the child class while implementing the GenerateMatrix method.
We show two ways of generating the matrix and implement one fully.

ConsoleMatrixGenerator

This inherits the MatrixGeneratorInterface as it "is a" type of Matrix Generator.
Its implementation can take user input for the creation of matrix.
Note: The implementation is incomplete, it is only shown for the design approach.

MeshMatrixGenerator

This class inherits the MatrixGeneratorInterface as it "is a" type of matrix generator.

The class has the following as its member variables:

- Vector of type Mesh

The vector of type Mesh can have any number of parameters whose individual mesh ranges can be defined in a mesh object and passed to MeshMatrixGenerator.
This satisfies the requirement where five option parameters can define their individual mesh system and the MeshMatrixGenerator as part of its GenerateMatrix() method *delegates* the creation of mesh to the Mesh class.

All the created Meshes are then arranged to form a matrix and transposed using the Matrix class functionality.
This transposed matrix becomes a matrix of numerous options with five parameters.

OptionMatrixPricer class

This class represents an option matrix pricer. This class takes a 2D Matrix in its member methods as an argument and prices those options in each row. It also has methods for Delta and Gamma of the option matrix. This class calculates the option prices and return a vector fo the same.

The class has the following as its member variables:

- Pointer to an option

Runtime polymorphism is used here to calculate Price(), Delta(), Gamma() for various kinds of options (EuropeanOption, PerpetualAmericanOption). The member variable which is a pointer to an option type encapsulates all “specialized” option classes and therefore the OptionMatrixPricer does not take into account these “specialized” options. It uses the Option base class’s pointer and virtual overridden member methods like Price() and Delta() are called at runtime according to the type of the variable at the memory location pointed by the Option pointer.