1996

Finanzia: An Option Valuation Library

Konstantinos N. Pantazopoulos

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

Report Number:
96-067
FINANZIA: AN OPTION VALUATION LIBRARY

Konstantinos N. Pantazopoulos
Elias N. Houstis

Purdue University
Department of Computer Science
West Lafayette, IN 47907

CSD-TR 96-067
November 1996
FINANZIA: An Option Valuation Library

Konstantinos N. Pantazopoulos
Elias N. Houstis

November 4, 1996

Abstract

Option valuation is an important problem in the financial derivative products or options market. A variety of computational techniques have been proposed to approximate its solution based on various models of the financial derivatives process. In this paper we present a software library together with its software architecture for solving the option valuation problem and supporting the financial derivatives engineering process.

1 Introduction

Financial derivative products or options\(^1\) come in many different flavors. The most basic options are calls and puts, often referred to as vanilla options. A call is a financial contract among two parties with the buyer of the contract having the right, but not the obligation, to buy the underlying asset for the strike. A put allows the buyer to sell the underlying asset for the strike price. There are a number of characteristics that differentiate the various derivative products. If the buyer of an option can only exercise the right at the expiration of the option, then the option is European. If the right can be exercised at any time on or before the expiration the option is American. Another important characteristic of an option is the payoff at expiration, i.e. the distribution of the option value with respect to the underlying’s price at expiration. For a call it is given as \(\max(0, S - K)\) and for a put as \(\max(0, K - S)\) where \(S\) is the underlying’s price and \(K\) is the option’s strike.

The primary task is to value an option, i.e. to come up with a “fair”\(^2\) price. This problem is referred as option valuation. The valuation involves solving a mathematical problem, either in discrete or continuous time. Apart from the value, one is interested in the risk that incurs from an option transaction. Knowing the risk allows for protective actions (hedging). Often, one knows the solution to the valuation problem,

---

\(^1\)In general, financial derivatives or state contingent claims include products other than options, such as futures and forwards. In this presentation, we use these terms interchangeably mainly meaning options.

\(^2\)“Fair” in the context of option valuation refers to the price of the option that all agents in an economy would agree independent of their risk preferences; the “fair” option price is defined as the risk-neutral price.
for instance in the case of publicly traded options with quoted prices in the financial markets, and is interested in obtaining values for various parameters that lead to the particular price, for example calculating the implied volatility. In this case an inverse problem is solved based on the mathematical model that provides the option value. In this paper we discuss the design and implementation of an object oriented library, FINANZIA, that can be used as a tool to support the option valuation process.

This library supports three interrelated tasks associated with the option valuation problem: the valuation, the hedging, and the implied parameter calculation. Their inter-relationship is depicted in Figure 1.

The rest of the paper is organized as follows. In Section 2 we discuss the mathematical formulations of the option valuation problem and its derivatives and the algorithmic infrastructure available and proposed to approximate its solution. In Section 3 we describe the design and architecture of the FINANZIA option valuation library and in Section 4 we discuss its implementation. In Section 5 we outline the possible extensions and future directions and we conclude in Section 6.

2 Financial derivatives computations

The mathematical modeling of options was pioneered in the seminal works of Black and Scholes [BS73] and of Merton [Mer73]. Since then, tremendous effort has been devoted in extending and improving the option valuation models and their computational solutions. More recently, partial differential equations (PDEs) based models and their corresponding numerical solutions have been widely applied to option valuation problem with considerable success [WDH93, WDH95], along with the more popular discrete time approximations and probabilistic methods.

Mathematically an option is a function $V$ of the underlying asset $S$, the strike price $K$, the risk-free interest rate $r^3$, the volatility of the underlying asset's returns $\sigma$, the duration of the option contract $T$, the time to expiration $t$, the dividend yield of the asset $\delta$ (for stocks or indices) and the payoff of the option $V_T$. Customarily, only $S$ and $t$ are explicitly included in the value function, while the rest are considered implied parameters and are inputs to the option valuation problem. The option value is denoted as $V(S, t)$. $S$ is modeled as a diffusion process (Ito process), while in simple cases $\sigma$ and $r$ are constants. The option value is the expected value of the payoffs over all possible realizations of $S$ discounted with the risk-free rate $r$ (risk-neutral

---

3The risk-free interest rate is the rate earned by an investment that has no risk such as a government security.
The expectation is taken using the risk-neutral density of $S$ which is the probability density under the risk-neutral measure $Q$. Equation 1 is the Feynmann-Kac solution to the fundamental partial differential equation (PDE)

$$V_t + \frac{1}{2} \sigma^2 S^2 V_{SS} + (r - \delta)SV_S - rV = 0$$

with appropriate boundary conditions for the option that $V$ describes. Equation 2 is valid for options on a single underlying asset under constant dividend yield, risk-free rate and volatility. For multi-asset options, the fundamental PDE has the form

$$V_t + \sum_{i=1}^{n} \sigma_i S_i V_{S_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\sigma_i^2 \rho_{ij} \sigma_j^2 S_i^2 S_j^2 V_{S_i S_j}) - rV = 0$$

where $n$ is the number of underlyings and $\rho_{ij}$ the correlation of asset $i$ with asset $j$.

A number of options such as lookbacks or asians have a payoff which is dependent on the path that the underlying asset follows. This path dependence is easily handled in the European option case by discrete time approximations and Monte Carlo methods mentioned below. In the PDE formulation of the problem an additional independent variable is introduced.

The early exercise feature, e.g. for American or Bermudan options, results in an optimal exercise boundary problem which in discrete or Monte Carlo methods is treated as an optimization problem or in the PDE setting as a free boundary problem. Further extensions include stochastic interest rates, stochastic volatility, varying payoffs and varying underlying assets.

2.1 Review of Option Valuation Algorithms

In broad terms the option valuation problem is attacked using analytic approximations to the fundamental PDE for specific cases, simulating the stochastic process of the underlying asset in discrete time, or simulating the stochastic process in continuous time. The majority of option valuation algorithms are based on the risk-neutral valuation approach introduced in [CR76] and theoretically treated in [HK79, HP81].

Closed form solutions are available in specific cases. The solution for the European option was presented in [BS73, Mer73]. For American call options with dividends results are given in [Rol77, Ges79, Wha81]. For the American put problem early results are presented in [Joh83, Blo86].

Discrete time approximations were first introduced in [CRR79] and include variations of what are known as binomial methods. Several extensions to the original approach have been done in recent years with various degrees of success and applicability. Related work is discussed in [Par77, Boy88, Omb88, RB79, HW88]. Convergence of these types of methods is discussed in [AK94, Lam95, LR95, BD95]. Extensions for multiple assets ( multinomial methods) are described in [Boy88, BEG89,
Extensions of multinomial methods to incorporate real market data are described in [Rub94a]. In general, multinomial methods are time and space consuming but are good tools for prototyping and are used extensively in practice.

Continuous time approximations are attacked mainly with Monte Carlo methods and PDE methods.

Monte Carlo methods for the option valuation problem were introduced in [Boy77] and discussed more recently in [BBG95]. Although Monte Carlo simulation methods are very powerful for handling general path dependent option products in the European case, they are extremely inefficient and difficult to implement for the American option problem. Attempts in this direction have been presented in [Ti193] and [BM95, BG95]. Monte Carlo methods are widely used in practice, mainly because of their powerful simulation capabilities and their usefulness in scenario analysis.

PDE methods for the option valuation problem had been originally proposed for European options in [BS73, Mer73]. In the context of the American option problem, finite difference techniques were first presented in [Sch77] and [BE77, BE78]. The quadratic method in [Mac86, BW87] and the method of lines in [CF95] provide exact solutions to approximations of the fundamental PDE. The convergence of finite-difference methods for the American option problem was first discussed in the context of the variational inequality approach to the problem in [JLL90]. Finite difference discretization of the variational inequality formulation of the problem leads to the linear complementarity solutions presented extensively in [WDH93, WDH95]. Front-tracking methods for the solution of the free boundary formulation of the problem are discussed in [PH96] and a comparison between the two approaches is given in [PH96]. PDE methods are very efficient and accurate but are difficult to generalize in the case of multi-asset options.

3 Design and Architecture of the Option Valuation Library

The main objective of an option valuation library for the option management problem is to assist in the financial engineering process and provide the appropriate computational tools. A general model of the process is described in Figure 2. The financial engineer receives a number of requests for price quotes of derivative securities. Using existing data and the appropriate models the request is decomposed into specific steps for hedging the position. Hedging generates a number of transactions which incur costs. Along with the operational costs and the fair price calculated for the derivative the actual quote is generated and returned.

FINANZIA assists in derivative securities computations and the financial engineering process by providing support for the problem specification, the problem solution and the solution management and interpretation. The applicability of FINANZIA is wide, including categories such as

- education, academic research
In order to provide practical support it is necessary to address the dynamics of the financial engineering process. New options are designed on the fly, in order to meet customer needs. Participants’ competitiveness depends on the ability to price, hedge and manage derivatives products as fast as possible. Option calculations are performed millions of times every minute and **turnaround** time is crucial when quotations are pending or new financial products are being analyze.

### 3.1 Analysis of Requirements

The issues addressed by FINANZIA from an operational point of view are outlined below:

- **Portability, maintainability, expandability**
- **Friendliness, configurability, mobility**
- **Robustness, efficiency, accuracy**

Maintainability and expandability cannot be overstated in view of the dynamic nature of the problem domain and the diversity in computing support. *Portable option calculators* are of primary interest mainly due to the vast heterogeneity of real world computing environments. The context of applications is also important in this direction. For example investment banks in their majority use workstation technology for back office and real time support while they incorporate personal computer technology for the human interaction with the applications. Since the algorithmic infrastructure and computational support is substantial in terms of choices and size, option calculators should be delivered on demand based on the particular problem at hand. This will result in reduced frustration by the user and better utilization of resources, e.g. memory, network bandwidth or processing bandwidth. The variation and the
continuously increasing population of the problem domain suggest the definition of a high level abstract problem description language in order to speed up interaction and improve turnaround time. The availability of customizable option calculators is important in order to address many synthetic products whose management can be decomposed in more primitive blocks. Rapid prototyping is crucial for the researcher or financial engineer who invents new products not covered, or covered inefficiently, by existing tools. Polyalgorithmic solution in terms of speed and accuracy is also necessary. For example, algorithms used to generate daily risk reports may have less stringent speed requirements than those used in a real-time trading support system. On the other hand risk analysis might require higher accuracy than real time quoting. Hedging would require not only price calculation but also the computation of various partial derivatives describing the risk characteristics of the option (greeks).

3.2 Class Hierarchy and Abstractions

Figure 3 indicates a sample population of option products according to various basic characteristics such as exercise policy, underlying, dividend model and payoff. Every possible path in this example from top to bottom describes a different problem. An abstraction that assists in the problem specification needs not only to cover existing option problems but also to provide for the description of new problems as they emerge. Object orientation is employed in order to manage the complexity and expose the natural properties of the problem. The challenge is to properly assess the applicability and identify adequate abstractions for the option valuation problem. In FINANZIA abstractions are introduced at three different levels; in option kernels (Figure 3), in valuation kernels (Figure 4) and in numerical kernels (Figure 5). The three basic kernels are naturally interpreted into classes.

The Option Class serves as an abstraction for a derivative security. The various options products are defined through inheritance from the base option class.
derivation of the hierarchy simplifies the problem specification by identifying the characteristics which are important. As suggested in Figure 3 the main points of derivation from the base option class are the payoff, the exercise policy, the underlying and the various specific characteristics such as the dividend model and barrier.

The Valuator Class serves as an abstraction for the type of problem as described in Figure 1. The other points of derivation depend on the interest rate model (i.e. term structure), the volatility model and the actual valuation algorithm to be used. As discussed in Section 2 there are four main categories, the multinomial, numerical, Monte Carlo and customized. The customized family includes closed formulas that have been suggested and other specific algorithms for a given problem.

The Calculator Class serves as an abstraction for the actual valuation algorithm that is going to be used. In Figure 5 an example of a derivation for a PDE type of algorithm is described. Other types of methods are derived accordingly from a base calculator class.
A number of secondary abstractions are also encapsulated into class definitions. Such classes include specific numerical algorithms such as root finding routines, arrays, vectors, and interpolation methods. Assistant classes are defined as wrappers to main classes in order to assist in the faster instantiation of a particular object. For example, the option assistant simplifies the instantiation of an object from the option class. Analogous assistant classes are available for calculators and valuators. The assistant classes are an attempt to address in a specific way the abstract problem description.

4 Implementation and Examples

The current implementation of FINANZIA uses C++ and is available in UNIX and Windows platforms. The interface is that of the classical library interface, i.e. the user links an application envelope with the FINANZIA library and instantiates the necessary methods to solve a particular problem.

Option Classes are available for a number of option products. A list of the derivation tree is shown in Figure 6. Square brackets indicate optional paths while curly brackets indicate mandatory paths.

All classes provide methods for setting up the various characteristics of the option, such as strike, spot price, etc. The general format of these methods is defined as

\[
\text{< class > Set } \left\{ \begin{array}{c}
S \\
K \\
T \\
D \\
R \\
< \text{vol} > \\
< \text{other} > \\
\end{array} \right\} (\text{< argument >})
\]

The payoff is virtually defined at the base option class and overloaded for the derivation of the various types of options. All classes provide a \text{< option > Payoff(< spot >)} method which, however, is different for a call, put or other exotic option. The payoff has to be defined when the particular option type is installed and cannot be changed dynamically. There is a large number of methods that deal for example
with specific issues such as boundary conditions necessary for each of the PDE solvers. These methods are protected and are overloaded by the user as needed. An option for example might not need to provide the methods needed to use a front-tracking type of solver. Of course this means that such a solver cannot be requested by the valuator of this option.

Calculator Classes are available for several types of algorithms. A list of the derivation tree is shown in Figure 7.

Vanilla European or other options with closed formulas can be priced using a Black–Scholes calculator of type $\text{BSCalc}(<\text{option}>)$ or using the binomial method ($\text{BinCalc}(<\text{option}>)$). Exotic options can be priced using either binomial ($\text{Bin}$), Monte Carlo ($\text{MoC}$) or PDE ($\text{PDE}$) type calculators. American options are necessarily handled by either binomial or PDE calculators. The available PDE calculators include explicit time stepping algorithms ($\text{Exx}$) or implicit ($\text{Ix}$) with both finite differences ($\text{FD}$) and finite elements ($\text{FM}$). The free boundary problem for the American option is solved using either linear complementarity ($\text{LC}$) or front-tracking ($\text{FT}$) techniques. There are numerous private and protected methods to tune up the various solvers which must be handled by the expert user. Default values are provided for users that are not interested in fine tuning the algorithms.

Valuator Classes are available for the three main tasks. The derivation tree is given in Figure 8. The implied parameter calculation is currently supported for volatility ($\text{Vol}$) rates ($\text{Rt}$) and dividends ($\text{Dv}$).

Utility Classes are included to provide support and secondary operations. A list of the most important ones is given in Table 1.
Class Services
Normal Distribution Calculation of normally distributed random variables
Matrix, Vector Abstractions and management of arrays and vectors
Root Finding Bisection, Secant and Newton methods used in implied calculations
Interpolation Linear and spline interpolation algorithms for one and two dimensions
Linear Solvers Direct and Iterative linear solvers (tridiagonal LU, LU, SOR, Projected SOR)

Table 1: FINANZIA Utility Classes

4.1 Examples
In the following we give some examples to illustrate the use of FINANZIA. In order to price a particular option, the option must be defined, the valuator must be instantiated for pricing and the calculator must be selected for the specific algorithm to use.

Pricing an American Option. Assume that we need to price an American call option on an underlying stock whose spot price is 100, the strike price is 120 with constant interest rate 5%, constant dividend yield 2% and time to expiration six months (0.5 years). A possible implementation is shown in Figure 9.

Using the Assistant Classes. In case a more complicated option with a more general algorithm needs to be priced an assistant can be used in order to simplify the necessary instantiation sequences. A generic example using the assistant classes is shown in Figure 10.

In general the assistant will return error information in case the request is invalid. In a robust use the appropriate error handling must also be considered. Instantiating other types of problems such as hedging and implied parameter calculation is straightforward.

Adding a new Component. The extension of FINANZIA to include new options, valuation problems or calculation algorithms can be done in a natural way. In order to add a new option problem the appropriate class must be derived from the hierarchy and the new type must be registered with the system. The same procedure is followed for new valulators and calculators.

5 Future Extensions
In order to achieve some of the stated objectives, a number of extensions are necessary. In this direction we are considering three main approaches. In order to facilitate the “calculator on demand” feature an Internet based implementation of the library is needed; Java is a the most intriguing candidate for such an extension because the class
#include <Finanzia.h>
...
// other stuff
...

// Instantiate an option of the particular type
CallAmerOption myOption;

// Initialize the option
myOption.SetS(100).SetK(120).SetR(0.05).SetD(0.02).SetT(0.5);

// Instantiate a calculator of binomial type
BinCalc myCalculator;

// Instantiate a valuator
PriceVal myValuator;

// Solve the problem
double myOptionValue = myValuator(myOption, myCalculator);

...
// other stuff
...

Figure 9: Pricing an American Option
#include <Finanzia.h>

...  
// other stuff  
...

// Instantiate the assistants
OptionAssistant myOptionAssistant;
ValuatorAssistant myValuatorAssistant;
CalculatorAssistant myCalculatorAssistant;
Option myOption;
Valuator myValuator;
Calculator myCalculator;

// Set up the problem
myOptionAssistant.Asset(TYPE).Exercise(TYPE).DModel(TYPE).Payoff(TYPE);
myValuatorAssistant.ProblemType(TYPE).RateModel(TYPE);
myCalculatorAssistant.Method(TYPE).Algorithm(TYPE).Accuracy(#number);

// Create the appropriate instances
myOption = myOptionAssistant.newOption();
myValuator = myValuatorAssistant.newValuator();
myCalculator = myCalculatorAssistant.newCalculator();

// Initialize myOption, myValuator, myCalculator

// Solve the problem

...
// other stuff  
...

Figure 10: Using the Assistant Classes
hierarchy and methods can be interpreted almost directly. A Java implementation will allow networking computing to be used in conjunction with FINAZIA.

In order to provide an integrated interface for FINANZIA, a link with a problem solving environment of some sort is necessary. We are considering the integration of the library into MATLAB and EXCEL. For the multi-dimensional PDE calculators we are considering the integration with J/ELLPACK which provides a number of tools including mesh generators, visualization tools and parallel numerical libraries.

6 Conclusions

We have discussed and presented the option valuation problem from a computational perspective. We have presented a design and implementation overview for an option valuation library, FINANZIA, which addresses the problem in an integrated, incremental way. The design of FINANZIA follows a hierarchical abstraction of the option management problem. Using object oriented techniques we have encoded the option valuation problem through a derivation tree. The problem specification has been simplified using inheritance. The multiplicity of solution approaches has been managed using dynamic binding and the complexity has been encapsulated in the implementation of the various classes.

References


