Numerical Methods in Mathematical Finance.

Project:
Monte-Carlo Simulation and Calibration of Multi-Dimensional Model
and the Valuation of an Exchange Option

Christian P. Fries  Juan Montes
email@christian-fries.de  montes.jmiguel@gmail.com

June 4, 2014

Version 0.8

Abstract

The following are the exercises of the project home-work of the lecture Numerical Methods in Mathematical Finance (Summer 2014), which is part of the final exam. The project is conducted in the LMU quantLab, but may be prepared at home. Support is provided by

Christian Fries, email@christian-fries.de
Juan Miguel Montes, montes.jmiguel@gmail.com

Resources and further details can be found at
http://www.christian-fries.de/finmath/lecture14.2/project/
Contents

1 Introduction 3
   1.1 Aim of the Project ............................................. 3
   1.2 Evaluation of the Project Results ............................... 3
   1.3 Submission of the Solution ..................................... 3

2 Simulation 4
   2.1 Task: Monte-Carlo Simulation of a Stochastic Process .......... 4

3 Calibration 6
   3.1 Task: Calibration .................................................. 6

4 Hedge Simulation 8
   4.1 Task: Hedge Simulation .......................................... 8

References 9
1 Introduction

The project consists of the implementation of a Monte-Carlo simulation of a risk neutral multi-asset model. The model should then be calibrated to a given product and then used for a hedge simulation.

1.1 Aim of the Project

During the project Monte-Carlo simulations for a multi-asset model will be performed (using Euler scheme). A generic Monte-Carlo calibration will be implemented and compared to an analytic calibration formula. The models will the be tested for valuation of derivatives and (optionally) hedge simulation.

The aim of the project is to perform a collaborative development of a simulation, valuation and calibration algorithm, including testing.

Your solution should be elegant, efficient, easy to read and well documented.

1.2 Evaluation of the Project Results

To successfully pass the review of the project we may ask for a short presentation of a part of the solution (parts will be distributed randomly) and ask you for answering a set of project related questions.

Note: The implementation part of the project can be solved very elegantly using object oriented implementation techniques requiring only few lines of new code. We encourage you to discuss your ideas during the solution in order to improve your solution.

1.3 Submission of the Solution

Each working group will receive its own password protected subversion repository. The solution has to be submitted to the repository until end of July 2nd, 2014.

We may ask some of you to present some parts of their solution and perform a short evaluation. The presentation/evaluation will be on July 3rd/4th, 2014 or July 10th/11th.
2 Simulation

2.1 Task: Monte-Carlo Simulation of a Stochastic Process

Exercise 1 (Euler-Scheme for Stochastic Processes):

Consider the model

\[
\begin{align*}
    dS_1(t) &= rS_1(t)dt + \sigma_1 S_1(t)dW_1 \\
    dS_2(t) &= rS_2(t)dt + \sigma_2 S_2(t)dW_2 \\
    dW_1(t)dW_2(t) &= \rho_{1,2}dt
\end{align*}
\]

given under the equivalent martingale measure \( Q^B \) belonging to the numéraire

\[
B(t) = \exp(rt), \quad B(0) = B_0 = 1
\]

Write one or more Java classes implementing the interface

\[
\text{AssetModelMonteCarloSimulationInterface}
\]

from the package

\[
\text{net.finmath.montecarlo.assetderivativevaluation}
\]

such that objects instantiated from these classes will provide an approximation (simulation) of these models for a given time discretization and number of paths.
Notes:

- You may use

  \texttt{net.finmath.time.TimeDiscretizationInterface}

  for the specification of the time discretization.

- You may use

  \texttt{net.finmath.montecarlo.BrownianMotionInterface}

  for the specification of the Brownian motion.

- You are not obliged to use these classes, but you have to implement the interface

  \texttt{AssetModelMonteCarloSimulationInterface}.

- Since you work together in a small group, we suggest the following approach:
  
  - try to implement the model independently (maybe using different approaches: quick-and-dirty from scratch, using code from the finmath library, duplicating and modifying existing code),
  
  - use the model(s) to value a financial product like an European option (you may use

    \texttt{net.finmath.montecarlo.assetderivativevaluation.products}),

  - use your different implications to value \textit{the same product} (this is possible since you implemented a common interface), then compare your results,
  
  - then, let the others read your code.

- You may like to practice \textit{pair programming} (a technique from extrem programming), see \url{http://en.wikipedia.org/wiki/Pair_programming}.\footnote{©2013 Christian Fries}

\textit{http://www.christianfries.com/finmath/}
3 Calibration

3.1 Task: Calibration

Exercise 2 (Generic Calibration): Write a generic calibration for the parameter $\rho$ using a Monte-Carlo calibration and a given product and target value. That is: Given a financial product, i.e., a Java class implementing a method

\[ \text{getValue(AssetModelMonteCarloSimulationInterface)} \]

which returns the value of that product under a given simulation. Now, given a target value, determine $\rho$ such that the valuation of that product matches the given target value. Proceed as follows (however, you may create your own solution):

- To your class of Exercise 1 add a method

\[ \text{getCloneWithModifiedData(Map<String, Object> newData)} \]

which returns a new object of type

\[ \text{AssetModelMonteCarloSimulationInterface} \]

having exactly the same parameters and specification (as this) except that the values of the parameters specified in the map \text{newData} are replaced.

- Modify the library by adding this method to the interface

\[ \text{AssetModelMonteCarloSimulationInterface} \]

Note: If required, add default implementations to classes implementing the interface. The default operation should be simply be

\[ \text{throw new UnsupportedOperationException();} \]

- Create a class \text{CalibratedModel} which provides a method

\[ \text{getModel(AssetModelMonteCarloSimulationInterface model, AbstractAssetMonteCarloProduct product, double targetValue)} \]

Implement this method such that it returns a model for which $\rho = \rho_{1,2}$ is calibrated such that the given product matched the given target value.
The implementation of the methods is comparable small. Use a one dimensional optimizer (like GoldenSectionSearch) and minimize the objective function
\[ \rho \mapsto (V(\rho) - \text{targetValue})^2 \] (4)
where
\[ V(\rho) = \text{product.getValue(newModel)}; \]
with
\[ \text{newModel} = \text{model.getCloneWithModifiedData(newData)}; \]
and \text{newData} is \text{HashMap} with \text{newData.put("rho", new Double(rho))}.

Notes:

- For the calibration you have to minimize the objective function above. Use the class \text{GoldenSectionSearch} from the package \text{net.finmath.optimizer} to perform this. There is a small example in this class, minimizing a function \( y = f(x) \). In our case, \( x (= \text{getNextPoint in the example}) \) will be the \( \rho \) and \( y (= \text{setValue in the example}) \) will be the evaluation of \((V(\rho) - \text{targetValue})^2\) - which involves the Monte-Carlo valuation of the product \( V \) using a \text{clone} of the model with the new parameter \( \rho \). \textbf{Hint:} It is important that you choose the search interval correctly. Correlation is a parameter between \(-1\) and \(1\) and the numerical algorithm will get a problem for some values.

- Performing the calibration it is important that all clones created in each iteration share the same seed for the random numbers. (Why?). The method \text{getCloneWithModifiedSigma} can to take care of this.

- Reusing the pre-calculated \text{BrownianMotion} will greatly improve performance. This is possible since the object \text{BrownianMotion} is immutable\(^1\). (Why is this important?) The method \text{getCloneWithModifiedData} can to take care of this.

\textbf{Exercise 3 (Test of Calibration Algorithm):} Test your calibration, by checking that the calibration product matches the \text{targetValue}, when evaluated by the calibrated model.

Perform this, i.e.,

- calibrate \text{rho} and
- check the calibration

\(^1\) We say that an object is of immutable state, if its state cannot be altered after construction, i.e., the result of a method only depends on its arguments.
for an exchange option, i.e., an product with payoff

\[ \max(S_1(T) - S_2(T), 0) \]

and target values

<table>
<thead>
<tr>
<th>Group</th>
<th>Product Payoff in $T$</th>
<th>Target Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>...</td>
<td>values will be provided</td>
</tr>
<tr>
<td>Group 2</td>
<td>...</td>
<td>values will be provided</td>
</tr>
</tbody>
</table>

while the other model parameters are $S_1(0) = 100$, $S_2(0) = 100$, $N(0) = 1$, $r = 5\%$, $\alpha = 1/2$ and the product parameters are $T = 2$. 

©2013 Christian Fries
4 Hedge Simulation

4.1 Task: Hedge Simulation

Exercise 4 (Hedge Simulation): Since your model implements the interface

\[
\text{AssetModelMonteCarloSimulationInterface}
\]

we may use the model to evaluate all single assets product, and also the

\[
\text{BlackScholesDeltaHedgedPortfolio}
\]

of the package

\[
\text{net.finmath.montecarlo.assetderivativevaluation.products}
\]

This will perform a delta hedge for a European option using only the asset $S_1$.
Adapt the code of that class to a new class

\[
\text{MultiAssetBlackScholesDeltaHedgedPortfolio}
\]

and perform a delta hedge of an exchange option. To do so you need to derivat the delta
of an exchange option (see http://en.wikipedia.org/wiki/Margrabe%27s_formula).

Create a plot of the hedge error, similar to the following

![Replication Portfolio Values](http://www.christianfries.com/finmath/)
References

   http://finmath.net/java.

FRIES, CHRISTIAN P.: finmath lib - api documentation.
http://finmath.net/java/finmath-lib/apidocs.