Fitob R Package: Manual & Tutorial

Janos Benk, benkjanos@gmail.com

2012.11.14

Contents

1 Introduction

What is Fitob? The main focus of the R Package Fitob is the PDE-based financial derivative pricing, where Fitob is able to solve not just the Black-Scholes PDE in a multidimensional setting, but also the more general Fokker-Planck (convection-diffusion) PDE. Fitob provides a general scripting interface that can describe almost any financial contract in an efficient and unique way.

Since Fitob provides an efficient convection-diffusion PDE solver in a multi-dimensional setting[1], it could be applied in other applications as well. (e.g., multi-dimensional distribution computations)

If the reader is already familiar with the theoretical background of the PDE-based option pricing, it is recommended to start with either the User Interface or Examples Sections.

[1] 1D-6D or even 12D

1.1 Mathematical and Numerical Background

In this section, we present the mathematical formulation of our general problem, which is to price a given financial contract based on the underlying PDE in a multidimensional setting. We emphasis here that, although, this was the main focus of our application, our can be applied in general to the time evolution of a given initial state under the influence of any multi-dimensional distribution (Fokker-Planck PDE).

The assumption is that the risk factors are modeled by stochastic differential equations (SDEs) that also define the distribution. In general fitob considers the following form of SDEs for $\mathbf{S} = \{S_i | i = 1, \ldots, D\}$ risk factors

$$dS_i = \mu_i^{'}(\mathbf{S}, t)dt + \sigma_i(\mathbf{S}, t)dW_i, \tag{1}$$

where $\mu_i^{'}(\mathbf{S}, t)$ is a general drift coefficient and $\sigma_i(\mathbf{S}, t)$ is the volatility. dW_i represents the stochastic increments which are assumed to be normally distributed. Furthermore the stochastic increments are allowed to be correlated $dW_i dW_j = \rho_{i,j}$ dt.

These factors influence a given function $P(S, t)$, where only the initial value $P(S, 0)$ of this function is known. The task is to find find the value $P(S, T)$ at given end time T, knowing that this function is transformed by the Fokker-Planck PDE. The boundary conditions (BCs) of this initial value problem (IVP) can also be different, but for the sake of simplicity we state this PDE with Dirichlet BCs.

In financial pricing, the time axis is reversed, since usually the payout of a given contract in the future is known $P(S, T)$, but not the current price $P(S, 0)$. Furthermore, the SDEs usually satisfy the non-arbitrage criterion, such that they can be transformed to different distribution measure where all SDEs have the risk-free interest rate drift. However this is not a necessary condition for a well-posed problem, and the PDE can be computed with with incomplete or even arbitrage markets.

$$\begin{aligned} &\frac{\partial P}{\partial t} + \sum_{i=1}^{D} \mu_i(\mathbf{S}, t) \frac{\partial P}{\partial S_i} - r(\mathbf{S}, t)P \\ &\frac{1}{2} \sum_{i=1}^{D} \sum_{j=1}^{D} \sigma_i(\mathbf{S}, t)\sigma_j(\mathbf{S}, t)\rho_{i,j} \frac{\partial^2 P}{\partial S_i \partial S_j} = 0, \end{aligned} \tag{2}$$

where $\mathbf{S}$ are the underlying risk factors that directly or indirectly impact the price. $\lambda_i(\mathbf{S}, t)$ are the market price of risks that for tradable assets are set such that $r(\mathbf{S}, t) = \mu_i(\mathbf{S}, t) = \mu_i^{'}(\mathbf{S}, t) - \lambda_i(\mathbf{S}, t)$, and for non-tradable assets is usually set to zero. It is important to mention here that this PDE for financial applications is solved backwards in time, since the payout is known, only the underlying's future value is unknown. Hence, the PDE is solved backwards in time.

The computational domain for this PDE can be estimated from the initial values of the risk factors and the SDEs, as it is described in [1] and illustrated on Figure 1.

During the maturity of a contract various transactions and decisions (such as early exercise) may take place. These operations, in a PDE-based evaluation, are either transformations on the P value or on one of the axis S_i [1]. The

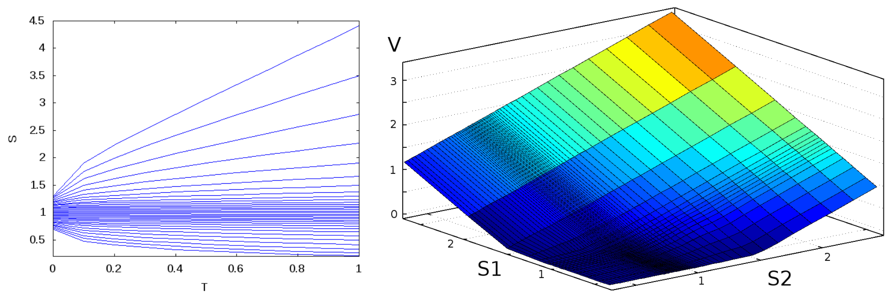

Figure 1: Generating the computational domain from an SDE (left) and a resulting regular computational grid.

main advantage of the PDE-based evaluation is that the expected value of the payoff is accessible at any given time t, and it is $P(\mathbf{S}, t)$. This value can be directly used for early exercise decisions. For other type of transactions, such as the average value computations for Asian options, the axis of the grid is transformed according to the transfer function (See Examples).

To handle the multi-dimensional space and the PDE-solving we employ combination technique based sparse grids (see Figure 2) and efficient multigrid solvers. The combination of these methods allow for computations in 1D-6D [2]. For more details we refer to [2].

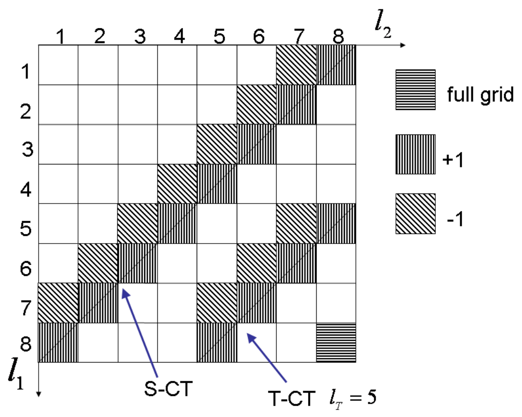

Figure 2: Two variants of the combination technique based sparse grid. (left) The Standard Combination Technique (S-CT) copes well with the curse of dimensionality, but fails for non-smooth functions. The Truncated Combination Technique (T-CT) can also handle severe discontinuities in higher dimensions [1,2].

[2]Or even higher in later publications

2 User Interface

Pricing a financial contract using the PDE-approach represents a complex numerical problem, where many different user interaction is required. The user has to provide several informations, which can be categorized in two groups.

1. **Financial contract**: This group contains informations that is strictly related to the structure of the financial contract. Since the structure of a financial contracts/derivatives might be complex, we use a script base description. In this script, each basic operations such as, waiting, transactions or exercise rights. More details are in Section 3. These informations are contained in a separate file.

2. **Numerical and modeling informations**: We separate between the informations regarding the financial contract and the modeling of the risk factors. The modeling informations, that is mainly the representation of the Eq. 1, is stored in an XML file. In addition to the modeling informations, we also store the numerical configurations (e.g., tolerance, type and level of the combination technique) in this XML file. These informations are described in Section 2.2.

Actually these informations are contained in the script and XML files. However, these could be argument of the Fitob R functions.

Following the above mentioned logic one can call the pricing functions of Fitob with two arguments, where the arguments represent the file names of these informations. In order to compute the price of an option one should type:

```
> resultPrice <- fitobPrice("XMLFile", "ScriptFile");
```

and the $resultPrice$ is the resulting value. The $XMLFile$ contains the modeling and numerical settings, whereas the *ScriptFile* contains the script describing the option.

The result of the evaluation could be not just a single value but also a regular D-dimensional grid, that contains the price of the option in the near vicinity of the strike. Furthermore, this regular grid could also be used very efficiently for hedging purposes. This functionality is done by the function:

```
> resultObject <- fitobPriceMesh("XMLFile", "ScriptFile", level);
```

where $XMLFile$ is the XML file containing the modeling and numerical settings, *ScriptFile* contains the script of the contract, and *level* is an integer value defining the resolution of the resulting regular grid.[3] *resultObject* represents the resulting list object that contains the regular mesh. This object can be plotted in 1D and 2D with the provided function.

```
> fitobMeshPlot(resultObject);
```

This list contains: 1) the actual price, 2) dimension of the problem 3) number of the points in the regular grid 4) array with $2^{D \cdot level}$ values of the regular grid 5) number of points on all axis $D \cdot 2^{level}$ 6) a $D \times 2^{level}$ array containing the values of the axis 7) list of strings containing the name of the axis.

For concrete problems we refer to the Example Section 3. In the following two subsections we provide a short manual of the XML file configuration and the scripting interface.

[3]It will have 2^{level} points in each dimensions so in all together $2^{D \cdot level}$ points.

2.1 Inputing a Financial Contract

Each financial contract can be represented as a sequence of basic operations and each operator represents a specific effect on the computational grid. This concept is also found in other commercial toolboxes, but here we use simple approach. These operations are contained in a *model* building block:

```
model Option1D;
   import S1;
   export P;
   ...
end;
```

where $Option1D$ is the name of the contract. The only underlying here is $S1$ expressed by the keyword *import*. *export* suggest that P is the output price of this financial contract. In the place of ... the operators follow that describe the contract:

- **Waiting time**: This represents the waiting time between various actions, where the SDEs follow stochastic paths. This waiting is represented by

  ```
   Theta dt
  ```

  where *dt* represents the time interval.

- **Transactions**: This represents the transfer of values, where the target of the transaction might be either the option price or one of the underlying axis. Such transaction is expressed in the script as an assignment expression, e.g., for 1D European Call option's payoff

- **Early exercise rights**: There is a special construct to implement constraint conditions that must hold and is done by the *loop* (inf) construct. In the body of this loop operator the conditions are enforced at each taken discrete time steps.

  ```
  P = MAX(S-K,0);
  ```

  In this way, a discrete dividend payments of 4 can also be simulated as $S = S - 4$;.

- **Control Operators**: These operators can either help the efficient description of the financial contract or describe decisions,

  ```
  if ( P < MAX(S-K,0) )
     P = S - K;
  end
  ```

  or describing a for loop

  ```
    loop (N)
       Theta (T/N);
    end;
  ```

For concrete examples we refer to Section 3

2.1.1 User Defined Models

In the script file one has the option to define user-specific models, by defining the three coefficients in an expression. To illustrate we consider that we have an underlying S1 that we defined previously as a normal Brownian motion and is defined as an *import* variable.

```
import S1;
```

One could simply redefine this variable as $dS_1 = 0.05\, S_1\, dt + 0.4\, S_1\, dW_1$ by the following construct in the script:

```
S1 = MODEL(0.05*S2,0.05*S2,0.4*S2);
```

where the general form is $S_i = MODEL(\mu_i, \mu_i^{'}, \sigma_i)$, and these coefficients can be defined arbitrarily. Usually it holds $\mu_i = \mu_i^{'}$. For a concrete example please see Section 3.

2.2 XML File Description

Since there are various numerical and modeling settings, Fitob uses a XML file to store them. Later on it would be useful to use some of the configurations as R function arguments. In the following, we enlist the most important configurations that the R users should be aware of by using R. In the section $< multigrid - solver >$ there are the following options important:

- $< mindtvalue = "0.0001"/ >$ the minimal time step during solving, this is the initial time step for the adaptive time step control.
- $< maxdtvalue = "0.01"/ >$ the maximal time step size. For American style contracts it is important to set this time step to a relative low level, since the continuous early exercise rights are checked at these small discrete time steps.
- $< timeStepControlvalue = "1e - 4"/ >$ This is the tolerance for the adaptive time step control routine.

The $< gridproperties >$ section contains information mainly about the **computational grid**, and the most important ones are the following:

- $< GRID\_TYPEvalue = "fullgrid\_WB"/ >$ specifies the type of the computational mesh. We recommend for 1D the $fullgrid\_WB$ option which is a regular grid without boundary points. This grid can also be employed for higher dimensions, but the curse of dimensionality is present. For most of the multi-dimensional problems we recommend the *extrap_truncated_combigrid_WB* grid, that is described, including its configurations, in [2].
- $< MAX\_LEVELvalue = "8"/ >$ specifies the level of the grid. Higher levels imply higher computational costs.

As next the $< domain >$ section follows, that describes the **initial values** of the underlying's SDEs. Beside the initial value, one can specify an initial domain, where at the end the price of the financial contract will be defined. For each risk factor we have a separate line:

```
<variable name="S1" min="0.7" max="1.3" evaluation="1"/>
```

that specifies for the underlying S1 the initial value of the SDE 1.0 and the initial domain $[0.7, 1.3]$.

The next section $< thetaoperator >$ contains SDE specific information. Here one should specify the **model of the risk factors**, where one can choose among the build-in methods. However the Fitob user can overwrite this model by defining a user-specific model (See Examples). The model parameters need to be specified [4].

```
<DIFFUSION_VARIABLES>
<variable name="S" type="GB" drift="0.05" convec="0.05" sigma="0.4"/>
<DIFFUSION_VARIABLES\>
```

The lines above defines one geometrical Brownian (type="GB") $S_i, i = 1$

$$dS_i = \mu_i^{'} S\, dt + \sigma_i S\, dW_i,$$

where mu_i is the *convec* parameter and the $mu_i^{'}$ is the $drift$ value, whereas σ_i is specified by *sigma*. (see Eq. (2) and Eq. (1)). In the same way one can specify a Heston and a CIR model,

```
<variable name="V1" type="Heston-CIR" k="1e-4" theta="0.16" sigma="0.04"/>
<variable name="S1" type="Heston" drift="0.05" convec="0.05" />
```

$$\begin{array}{ll} dS(t) & = \mu\, S(t)\, dt + \sqrt{v(t)} dW^{(1)} \\ dv(t) & = \kappa(\tilde{v} - v(t))dt + \sigma \sqrt{v(t)} dW^{(2)}. \end{array} \tag{3}$$

where the fields represent the corresponding parameter from the model Eq 3. The second equation in Eq 3 can be used separately as an interest rate model. A fourth type of model is the normal Brownian model (type="NB") with the same parameters as the (type="GB"). The number of built-in model can rather easily extended, however for the Fitob R package users we recommend to define such specific models in the script (see Section 2.1.1).

The **interest rate** can be set as constant with the line

```
<RISK_FREE_RATE value="0.05" variable_coupled=""/>
```

or can be coupled to a risk factor by setting the field *variable_coupled* to the name of this interest rate model (see Examples).

The **correlation matrix** can be given by the following format:

```
<CORRELATIONS>
    <correlation value="1.0,0.0"/>
    <correlation value="0.0,1.0"/>
</CORRELATIONS>
```

In the last section, one can specify a so called *standard deviation factor* for each model that steers the domain estimation of the computations, we recommend to use values 3-5 for the computations [5]

```
<STANDARD_DEVIATION_FACTOR value="3"/>
```

[4]These options can also be overwritten in the script file by a user defined model.

[5]3 for lower accuracy 5 for higher accuracy and for higher levels

3 Examples

In the following section we illustrate the capabilities of the Fitob R package various computational finance examples.[6] The example files, which are presented in this section, are all found in the *data* subdirectory in the downloadable package. Before using the fitob's functions, please use the R command $require(fitob)$. Furthermore, in the examples, this tutorial assumes that the XMl and the script files are located (copied) into the actual R directory[7].

3.1 European Put Option in 1D

We start with a simple 1D example, where the underlying stock price is modeled by a geometrical Brownian motion with a drift rate of 5% and a volatility of 40%. In the XML configuration file we set the initial domain to

```
 <variable name="S" min="90" max="110" evaluation="100"/>
```

end we choose the following parameters configurations: $S = 100$, $\mu = r = 0.05$, $\sigma = 0.4$ expressed in XML configuration file:

```
<DIFFUSION_VARIABLES>
  <variable name="S" type="GB" drift="0.05" convec="0.05" sigma="0.4"/>
</DIFFUSION_VARIABLES>
<RISK_FREE_RATE value="0.05" variable_coupled=""/>
```

Furthermore we use a regular mesh with level 8 ($2^{8 \cdot 1}$ points) that is specified in the XML file:

```
 <MAX_LEVEL value="8"/>
```

This European Put option is described by a short script, where K represents the strike price and the maturity is one year:

```
model European1D
   import S;
   export P;

   Theta 1.0;
   P = MAX(100-S,0);
end;
```

Using the defined inputs the user can now price this simple European option. Assuming that the user started R in the directory where the files *euro1D.xml* and *euro1DScript* are located you can call the pricing function as:

```
> require(fitob);
> price <- fitobPrice("euro1D.xml","euro1DScript");
```

If you want the get the price on the initial domain then you should use the other pricing function.

[6]Since Fitob is a general multi-dimensional convection-diffusion PDE solver could also be used for other purposes other than finance.

[7]where R has been started

```
> resObj <- fitobPriceMesh("euro1D.xml" , "euro1DScript",6);
```

We request to use level 6 for the regular resulting grid, and the list object *resObj* contains all the results that can be visualized as:

```
 > fitobMeshPlot(resObj);
```

and the resulting GNU plot is shown in Figure 3.

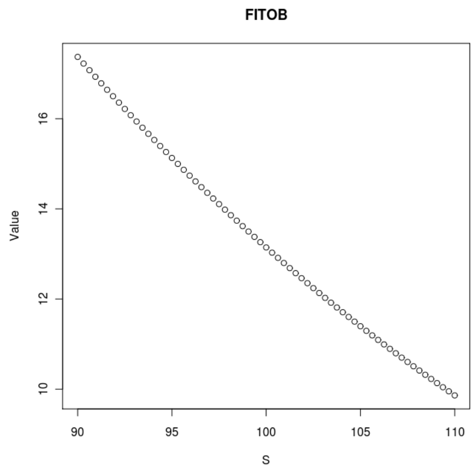

Figure 3: Resulting price curve for the 1D European Put Option.

The computed price of the option (variable *price* in R) is 13.14455 and the analytical solution is 13.14589.

3.2 American Put Option 1D and 3D

3.2.1 1D

In this example we extend the previous example with early exercise rights. There is a special construct to implement constraint conditions that must hold and is done by the *loop* (inf) construct. In the body of this loop operator the conditions are enforced discrete time steps, therefore we set the maximum time step in this case to $1e-3$ that means a continuous exercise interval of at least six hours for a maturity of one year. [8] and we use a smaller initial and time step $1e-4$. Furthermore we demonstrate the usage of the *loop* construct, where we split up the *Theta* 1.0 operator into N parts.

```
model americD1;
   import S;
   export P;

   loop (inf)
      P = MAX(100-S,P);
   end;
```

[8]Close to the maturity it will be less than 10 minutes

```
   N = 2;
   loop (N)
       Theta 1/N;
   end;
   P = MAX(100-S,0);
end;
```

Except the mentioned settings we use the presented configurations from the previous example.

Calling the Fitob functions the routine computes the price in about one second,

```
> require(fitob);
> pr <- fitobPrice("american1D.xml" , "american1DScript");
```

or using the script that outputs a regular mesh:

```
> lisMy <- fitobPriceMesh("american1D.xml" , "american1DScript", 6);
> fitobMeshPlot(lisMy)
```

The resulting price is 13.66277 (see Figure 4), where the reference value is 13.666. [9]

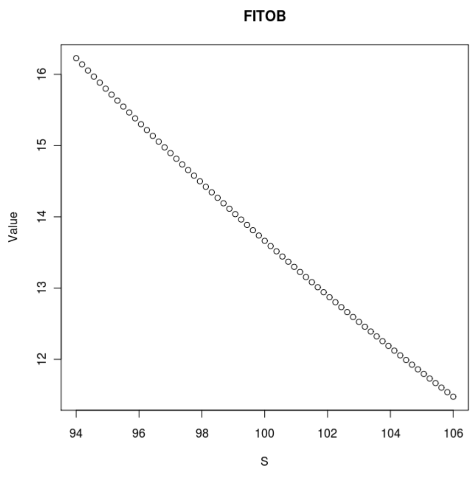

Figure 4: Resulting price curve of the 1D American Put Option

3.2.2 3D

For this example, we use the following settings: $S_i = 100$, $\mu_i = r = 0.05$, $\sigma_i = 0.4$ and $\rho = [1.0, 0.3, 0.5; 0.3, 1.0, -0.3; 0.5, -0.3, 1.0]$. These settings are specified in the XML file as:

```
<DIFFUSION_VARIABLES>
  <variable name="S1" type="GB" drift="0.05" convec="0.05" sigma="0.4"/>
```

[9]The reference value is provided by a Least-Square Monte-Carlo Method with high accuracy and high computational cost.

```
  <variable name="S2" type="GB" drift="0.05" convec="0.05" sigma="0.4"/>
  <variable name="S3" type="GB" drift="0.05" convec="0.05" sigma="0.4"/>
</DIFFUSION_VARIABLES>
<RISK_FREE_RATE value="0.05" variable_coupled=""/>
<CORRELATIONS>
    <correlation value="1.0,0.3,0.5"/>
    <correlation value="0.3,1.0,-0.3"/>
    <correlation value="0.5,-0.3,1.0"/>
</CORRELATIONS>
```

The level of the grid is specified in the *¡MAX_LEVEL value="5"/¿*. The script is similar to the 1D case, but we have 3 underlyings:

```
model American3D
   import S1;
   import S2;
   import S3;
   export P;

   K = 300;
   loop (inf)
     P = MAX(K-(S1+S2+S3),P);
   end;
   Theta 1.0;
   P = MAX(K-(S1+S2+S3),0);
end;
```

A convergence analysis of the Fitob's approach is presented in [1,2], where the reference value (also validated by Least-Square Monte-Carlo) is 26.21. We use the pricing script (for 3D there is no visualization)

```
> require(fitob);
> pr <- fitobPrice("american3D.xml" , "american3D_Script");
```

For *MAX_LEVEL value="4"* Fitob prices in around 3 seconds and gives the price 25.77, and for *MAX_LEVEL value="5"* after $\sim$ 20 seconds the resulting price is 25.96. But for *MAX_LEVEL value="6"* the resulting price was 26.18 after $\sim$ 180 seconds.

3.3 Asian Option in 1D

The next example demonstrates the axis transformations, that was mentioned for transaction. This Asian option builds the average value of the stock price over one year, where it takes samples on monthly basis. This average is stored in the A variable that becomes the additional dimension of this problem, beside the underlying S, which is a geometrical Brownian motion. The describing script has the form of:

```
model asian;
    import S;
    export P;
    K = 1;
    A = S;
    N = 12;
    loop (N)
        Theta 1/N;
        A = (A*K+S)/(K+1);
        K = K+1;
    end;
    P = MAX(100-A,0);
end;
```

where K is a simple counter, A the mentioned average value, and N the sample frequency in one year. The script computes the accumulated average by the formula $A = (A * K + S)/(K + 1)$;. The drift rate and the interest rate is set to 5%, whereas the volatility of S is now 40%:

```
<DIFFUSION_VARIABLES>
  <variable name="S" type="GB" drift="0.05" convec="0.05" sigma="0.4"/>
</DIFFUSION_VARIABLES>
<RISK_FREE_RATE value="0.05" variable_coupled=""/>
```

This examples uses a sparse grid to price the product with level 6:

```
<GRID_TYPE value="extrap_truncated_combigrid_WB"/>
<MAX_LEVEL value="6"/>
```

After the user calls the pricing methods (assuming that "asian1D.xml" and "asian1DScript" are in the actual directory)

```
> require(fitob);
> pr <- fitobPrice("asian1D.xml" , "asian1DScript");
```

or

```
> liM <- fitobPriceMesh("asian1D.xml" , "asian1DScript",6);
> fitobMeshPlot(liM)
```

The resulting price after ~ 5 seconds is 7.558 that also fits the value of Monte-Carlo simulation.

3.4 A User Defined Model in 2D

This sections demonstrates the generality of Fitob in defining a user-specific model. We consider a simple FX rate model with simple mean-reverting stochastic volatility model:

$$dV = a(b - V)dt + \sigma dW_1$$

$$dFX = \mu dt + V dW_2$$

where $a = 0.001$, $b = 0.1$, $\sigma = 0.2$, $\mu = 0.02$, and $\rho_{1,2} = -0.2$. We set a small drift for the FX rate and the mean value of the FX's volatility is 0.1. In order to define such model in the Fitob R package, the first step is to define the initial domain and factors in the XML (FX.XML) file such as:

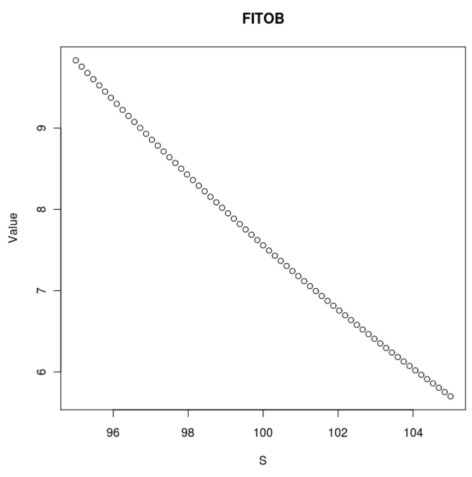

Figure 5: Resulting price curve of the 1D Asian Option

```
<domain>
   <variable name="V" min="0.12" max="0.2" evaluation="0.16"/>
   <variable name="FX" min="0.7" max="1.3" evaluation="1"/>
</domain>
<DIFFUSION_VARIABLES>
  <variable name="V" type="NB" drift="0.05" convec="0.05" sigma="0.4"/>
  <variable name="FX" type="NB" drift="0.05" convec="0.05" sigma="0.4"/>
</DIFFUSION_VARIABLES>
<RISK_FREE_RATE value="0.05" variable_coupled=""/>
<CORRELATIONS>
    <correlation value="1.0,-0.2"/>
    <correlation value="-0.2,1.0"/>
</CORRELATIONS>
```

The script defines these factors as normal Brownian motions, however these options are overwritten in the script file (FX_Script) by the lines

```
  V = MODEL(0.001*(0.1-V),0.001*(0.1-V),0.2);
  FX = MODEL(0.02*FX,0.02*FX,V*FX);
```

that specify exactly the mathematical model described in the SDE. The $MODEL$ pragma has three arguments μ, $\mu^{'}$ and σ (see Eq. (2) and Eq. (1)). In this concrete case, we set for both factors $\mu = \mu^{'}$. We mention here that one could easily define also a user specific interest model and use in the PDE. Assuming that V would be a short rate one would just need to specify in the XML file

```
<RISK_FREE_RATE value="0.05" variable_coupled="V"/>
```

and it could use V instead of 5% as short rate.

Using this model we price an European Call option on a FX strike rate. The script for this purpose is simple, and also includes the presented user-specific model declarations. The maturity of the option is $T = 1.0$ year.

```
model MyModelD2;
```

```
    import V;
    import FX;
    export P;

    V = MODEL(0.001*(0.1-V),0.001*(0.1-V),0.2);
    FX = MODEL(0.02*FX,0.02*FX,V*FX);

    T = 1.0;
    K = 1.0;
    Theta T;
    P = MAX(FX-K, 0.0 );
end;
```

Using the presented pricing R scripts the price is calculated as

```
> require(fitob);
> priceObj <- fitobPriceMesh("FX.xml" , "FX_Script",6);
> priceObj[1]
> fitobMeshPlot(priceObj);
```

The resulting surface plot (price surface for different FX rate and volatility V) is shown in Fig. 6. One can observe that the price, as expected, varies more along the FX axis as along the volatility axis V. The price at the initial values is 0.08384.

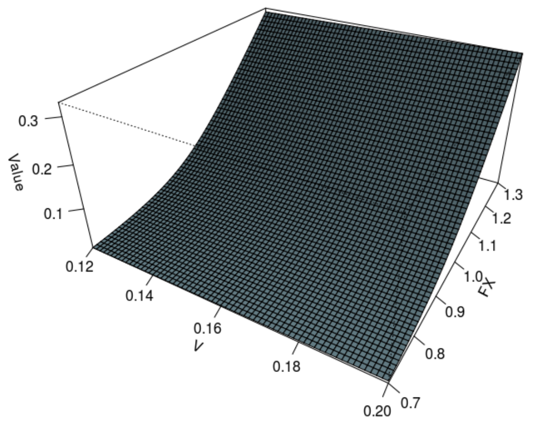

Figure 6: Surface plot of the price for the FX option. The price is more dependent on the FX than on the underlying volatility.

3.5 Guaranteed Minimum Income Benefit (GMIB)

As next we consider a product with considerably longer maturity, namely one GMIB. To keep things simple [10], we consider that the underlying S of this variable annuity follows a Geometrical Brownian path, and the interest rate is modeled by a CIR process.

$$dS = 0.02\,S\,dt + 0.3\,S\,dW_1$$

[10]Complex products could be set-up in the same way as well, see [1]

$$dr = 3.0(0.05 - r)\,dt + \sigma\,dW_2$$

that is correlated with r as $\rho_{1,2} = 0.2$. The model in S is specified in the GIMB_Script file as:

```
S = MODEL(0.0,0.02*S,0.3*S);
```

and r is defined as CIR model in the GMIB.xml file

```
<variable name="r" min="0.005" max="0.12" evaluation="0.01"/>
```

The flow of this contract is simple. In the initial state a face value 1.0 is invested in the risk-averse asset. After 10 years a constant payment of $CI = 0.15$ is payed out for 10 years, that represents a simplified view of a general GMIB. At the beginning of the payment period, the invested value is transfered to a risk less bank account. Therefore, in the second loop, we only sum up the payments P whereas the discounting by the short rate r takes place implicitly by the PDE (Eq. 2). The key point in the script is the line $P = MAX(S - P, 0)$;, where we state that the price of the product is the difference of the invested face value S and the discounted payments P. This value is saturated at 0, since the constant payments need to be made in all cases.

```
model GMIB;
   import S;
   import r;
   export P;
   S = MODEL(0.0,0.02*S,0.3*S);
   CI = 0.15;
   N1 = 10;
   N2 = 10;
   loop (N1)
     Theta 1.0;
   end;
   P = MAX(S-P,0);
   P = P+CI;
   loop (N2)
     Theta 1.0;
     P = P+CI;
   end;
   P = 0;
end;
```

This product can be priced with the script in less than 10 seconds and the price of this product is $liMy[[1]] + 1.0 = 1.254544$. The pricing with the Fitob R package is called as:

```
> require(fitob);
> liMy <- fitobPriceMesh("GMIB.xml" , "GMIB_Script",5);
> fitobMeshPlot(liMy);
```

The resulting price surface on Fig. 7 shows a greater variance along the S axis. The variance along the r axis is also measurable.

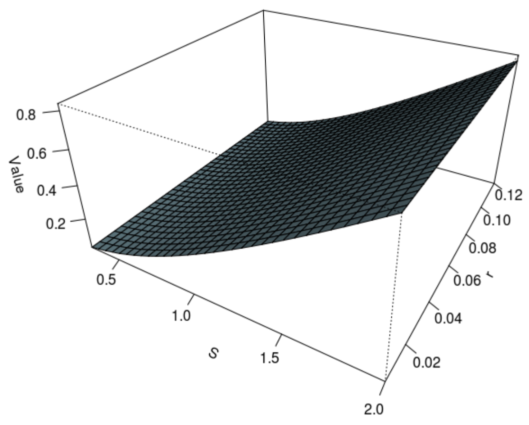

Figure 7: The price surface of the presented GMIB. The price is varies more in the direction of S, although in the r direction the change is also measurable.

3.6 Setting up your own example

Recommendations:

1. Start from an existing working example and change it incrementally.
2. If the fitob call fails (error handling is under constructions) than this might be cased by some black spaces in the script file.
3. If you have any problems, do not hesitate to write to benkjanos@gmail.com

4 Refrences

[1] J.Benk, D. Plueger: Hybrid Parallel Solutions of the Black-Scholes PDE with the Truncated Combination Technique. In Proceedings of the HPCS conference, 2012 Madrid.

[2] J. Benk, H.-J. Bungartz, A.-E. Nagy und S. Schraufstetter: Variants of the Combination Technique for Multi-Dimensional Option Pricing. In Progress in Industrial Mathematics at ECMI 2010, Oktober 2010.