CS 476/676: Assignment 3

Winter 2015

Instructor: W. Xu — Office: DC1312 — e-mail: *wei.xu@uwaterloo.ca*

Lecture Times: MWF 11:30-12:20 MC 2054

Office Hours: Until April 3, 2015: Wei Xu, Mondays 1:30-3:00pm, DC1312

Kai Ma, DC3594:

Wednesday 2:00-4:00pm, April. 2 (Thursday) 2:00-4:00pm,

Due April 6, in class

IMPORTANT: In this assignment, most of the marks for programming questions are allocated for explanations of algorithms and explanation of results. If all you hand in is the listing of the "Raw Code" or "Raw Output" by itself, you will get poor marks. All coding should be done in well documented (commented) Matlab. By default, you should submit listings of all Matlab code used in your assignment. Please ensure your code is tidy and well commented

Assignment Questions

1. (20 marks) Finite Difference: European Case, constant timesteps

Develop Matlab code for pricing European options under a Black Scholes model

$$\frac{dS_t}{S_t} = rdt + \sigma dZ_t$$

using a finite difference method as discussed in class. Use forward/backward/central differencing as appropriate to ensure a positive coefficient discretization. The code should be able to use fully implicit, Crank-Nicolson, and CN-Rannacher timestepping.

You will have to solve a tridiagonal linear system. DO NOT use the full matrix objects in Matlab. Use the Matlab sparse matrix function **spdiags** to set up a sparse diagonal matrix of the correct size and use **lu** Matlab function to solve linear systems. Avoid unnecessary LU factorization computation.

First, solve for a European put option, using the data given in the Table 2 , (with fully implicit, CN timestepping, and CN-Rannacher timestepping) and compare with the exact solution using **blsprice**. Use constant timestep sizes.

Show a convergence table, with a series of grids. Show the option value at $t = 0, S = 100$. Begin with a timestep of $\Delta\tau = T/25$, and the grid

```
S = [0:0.1*K:0.4*K,...
         0.45*K:0.05*K:0.8*K,...
         0.82*K:0.02*K:0.9*K,...
         0.91*K:0.01*K:1.1*K,...
         1.12*K:0.02*K:1.2*K,...
         1.25*K:.05*K:1.6*K,...
         1.7*K:0.1*K:2*K,...
         2.2*K, 2.4*K, 2.8*K,...
         3.6*K, 5*K, 7.5*K, 10*K];
```

where K is the strike, and we are interested in the solution near $S = K$.

Note: to carry out a convergence study, you should solve the pricing problem on a sequence of grids. Each grid has twice as many intervals as the previous grid (new nodes inserted halfway between the coarse grid nodes) and the timestep size is halved.

Assume that

$$\text{Error} \quad = \quad O((\Delta t)^2, (\Delta S)^2) \ ; \ \Delta S = \max_i (S_{i+1} - S_i) \tag{1}$$

Let

$$\begin{aligned} h &= C_1 \cdot \Delta S \\ h &= C_2 \cdot \Delta t \end{aligned}$$

Suppose we label each computation in the above sequence by a set of h values. Then the solution on each grid (at a given point) has the form

$$\begin{aligned} V(h) &= V_{exact} + A \cdot h^2 \\ V(h/2) &= V_{exact} + A \cdot (h/2)^2 \\ V(h/4) &= V_{exact} + A \cdot (h/4)^2 \end{aligned} \tag{2}$$

where we have assumed that the mesh size and timestep are small enough that the coefficient A in equation (2) is approximately constant. Now, equation (2) implies that

$$\frac{V(h) - V(h/2)}{V(h/2) - V(h/4)} \simeq 4 \tag{3}$$

Check the theory by examining the rate of convergence of your pricer.

Carry out the above tests using fully implicit, Crank Nicolson, and CN-Rannacher timestepping. Show a convergence table for each test.

Show plots of the option value, delta, gamma for the range $S = [50, 150]$, for your solution on the finest grid for CN-Rannacher timestepping.

Submit your matlab code.

2. (8 marks) SV, LVF, and Volatility Smile

The Heston-Nandi stochastic volatility model assumes that the correlation $\rho = -1$ in the Heston model, i.e.,

$$\begin{aligned} \frac{dS}{S} &= r dt + \sqrt{v} dZ \\ dv &= -\lambda(v - \bar{v}) dt - \eta \sqrt{v} dZ \end{aligned} \tag{4}$$

where λ is the speed of reversion of the variance v to its long-term mean $\bar{v}$.

(a). (4 marks) Modify your Matlab code (in Assignment 2) for pricing options under a Heston model using Milstein scheme so that it now returns European call option values under a Heston-Nandi model. Using data in Table 1 with $\Delta t = \frac{1}{1000}$ and $M = 100,000$, compute and plot call option values against $\log K$, with $\log K$ in the range $[-\mathbf{0.5}, \mathbf{0.2}] + \log \mathbf{S^0}$ (using 10 equally spaced log strikes).

Table 1: Data for Heston-Nandi Model

λ	10
$\bar{v}$	0.04
η	1
v_0	0.04
r	0
Time to expiry (T)	1.0 years
Initial Price S^0	\$100
Type	Call

(b). (4 marks) Modify your pricer in Problem 1 so that it also returns option values when the local volatility function depends on both S and t. Assume that the local volatility function is given below,

$$\sigma^2(S,t) = \max\left((v_0 - \hat{\bar{v}})e^{-\hat{\lambda}t} + \hat{\bar{v}} - \eta\log(\frac{S}{S^0})\left(\frac{1-e^{-\hat{\lambda}t}}{\hat{\lambda}t}\right),\ 0\right) \tag{5}$$

where

$$\hat{\lambda} = \lambda + \frac{\eta}{2},\ \hat{\bar{v}} = \bar{v}\frac{\lambda}{\hat{\lambda}}$$

Using data in Table 1 and four refinements from the initial grid in Problem 1, compute and plot call option values against $\log K$, with $\log K$ in the range $[-\mathbf{0.5}, \mathbf{0.2}] + \log \mathbf{S^0}$ (using 10 equally spaced log strikes).

Now compute and plot implied volatilities from European option values computed from the above LVF and using MC method in (a). Comment on your observations.

3 (10 marks) Finite Difference: American option

Modify your code to use the penalty method for American options. Carry out a convergence study on an American put with a constant volatility as in Problem 1 (and other data in Table 2). Use CN-Rannacher with both constant and variable timesteppings described in the course notes. For the variable timestepping, use $dnorm = .1$ and an initial timestep of $\Delta\tau = T/25$. On each grid refinement, reduce the initial timestep by 4 and reduce $dnorm$ by 1/2. Be sure that your timestep selector stops at the pricer at t=T exactly.

Show the usual convergence table.

Show plots of the price and delta for the finest grid using CN-Rannacher timestepping. Explain what you see.

Submit your matlab code and a short pseudo-code description.

4. (10 marks) Model Calibration

A typical practice in finance is to estimate an option pricing model from market prices of options that are traded frequently (in other words, liquid options). The calibrated model can be used to hedge liquid options or price and hedge other (illiquid) exotic options on the same underlying.

Table 2: Data for Put Example

σ	.4
r	.02
Time to expiry (T)	1.0 years
Strike Price	\$100
Initial asset price S^0	\$100

Stochastic volatility model and LVF model have both been considered to model volatility smile. In this assignmen we calibrate a local quadratic volatility function model (6) below,

$$\sigma(S,t) = \max(0.0, x_1 + x_2 S + x_3 S^2) \tag{6}$$

The main purpose of this exercise is to familiarize you with the calibration problem and its challenges.

Assume that the initial market prices $V_0^{\text{mkt}}(K_j, T_j),\ j = 1, 2, \cdots, m$, are available; here (K_j, T_j) denotes the strike and expiry of the j-th option.

Assume that $V_0(K_j, T_j; x)$ denotes the initial value of an option with strike K_j and T_j under a local volatility model described by a set of parameters $x = (x_1, x_2, \cdots, x_n)$. In the quadratic local volatility model (6), the model parameters are (x_1, x_2, x_3).

The model which best fits the market option prices can be estimated by solving the following *nonlinear least squares problem*

$$\min_{x \in \Re^n} \frac{1}{2} \sum_{j=1}^{m} \Big(V_0(K_j, T_j; x) - V_0^{\text{mkt}}(K_j, T_j) \Big)^2 . \tag{7}$$

Let $F(x)$ denote the vector of model option value errors from the market prices:

$$F(x) = \begin{bmatrix} V_0(K_1, T_1; x) - V_0^{\text{mkt}}(K_1, T_1) \\ \vdots \\ V_0(K_m, T_m; x) - V_0^{\text{mkt}}(K_m, T_m) \end{bmatrix}$$

Vector $F(x) : \Re^n \to \Re^m$ is a nonlinear function of the model parameter $x \in \Re^n$ and the calibration problem (7) is equivalently formulated as

$$\min_{x \in \Re^n} \left(f(x) \stackrel{\text{def}}{=} \frac{1}{2} \|F(x)\|_2^2 \right) \tag{8}$$

A nonlinear least squares problem (8) can be solved by a special optimization method such as the Levenberg-Marquardt method, the Gauss-Newton method, or a general method for nonlinear unconstrained optimization problems.

Optimization methods for nonlinear least squares problems typically exploit the special structure of the gradient and Hessian of the function $\|F(x)\|_2^2$. They typically require computation

of the Jacobian matrix of $F(x)$

$$J(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{pmatrix}$$

One difficulty in solving the nonlinear least squares calibration problem (7) is that it is not easy to compute the Jacobian matrix $J(x)$ explicitly. Instead, a finite difference approximation of the Jacobian matrix can be computed.

(a) Modify your Matlab code in Problem 2 (b) so that it also computes efficiently the option value when the volatility function depends on S only.

(b) Write a matlab function to return F, and J, if necessary, for any given x. Your Matlab function must return, in the second output argument, the Jacobian matrix J at x. Note that, by checking the value of **nargout**, you can avoid computing J when your Matlab function is called with only one output argument (in the case where the optimization algorithm only needs the value of F but not J).

```
function [F,J] = myfun(x)
F = ...                    % Objective function values at x
if nargout > 1             % Two output arguments
   J = ...                 % Jacobian of the function evaluated at x
End
```

Compute J using finite difference approximation as described in class.

(c) Use Matlab function **lsqnonlin** with LevenbergMarquardt as the choice of the optimization method to estimate the unknown coefficients x for the LVF model from the set of call option prices $\{V_0^{\text{mkt}}(K_j, T_j)\}$ whose implied volatilities are givien below:

K	80	85	90	95	100	105	110	115	120
σ_{impv}	0.2369	0.2236	0.2161	0.2055	0.1945	0.1848	0.1752	0.1648	0.1550

All of these options have expiry of one year (and they are generated from the data for Problem 2 under a Heston-Nandi stochastic volatility model).

You can set the options for optimization as follows

```
options = optimset('Jacobian','on','LevenbergMarquardt','on','Display','iter','MaxIter',20);
```

Perform the computation with the following starting points respectively:

- $x0 = [0.2;\ 0.0;\ 0.0]$;
- $x0 = [0.2;\ 0.1;\ 0.01]$;

For each starting point, report the estimated parameter set x^* and the calibration error $\|F(x^*)\|_2^2$. For each computed x^*, plot the given implied volatilities and the implied volatilities of the corresponding option values from the computed x^*. Comment on the extent to which the matlab program is successful in solving the calibration problem.

5. (12 marks) (**PDE Discretization, Stability**)

Let V^* be the payoff and

$$\mathcal{L}V \equiv \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} + (r-q)S\frac{\partial V}{\partial S} - rV \tag{9}$$

then the price of an American option on a dividend paying stock (dividend rate $q > 0$, $q < r$) is given by $\min(V_\tau - \mathcal{L}V, V - V^*) = 0$

Assume the above problem is discretized using a penalty method, as in the course notes in the form $(1 \le i \le imax - 1)$

$$\begin{aligned} V_i^{n+1} &= V_i^n + \Delta\tau\alpha_i V_{i-1}^{n+1} + \Delta\tau\beta_i V_{i+1}^{n+1} - (\alpha_i + \beta_i + r)\Delta\tau V_i^{n+1} + P_i^{n+1}(V_i^* - V_i^{n+1}) \\ V_i^* &= \text{ call payoff} \\ P_i^{n+1} &= \text{ penalty term} \end{aligned} \tag{10}$$

Assume that the following boundary conditions are specified at $i = 0, i = imax$

$$\begin{aligned} V_0^{n+1} &= 0 \\ V_{imax}^{n+1} &= S_{imax} \end{aligned}$$

with the payoff

$$\begin{aligned} &V_i^0 = \max(S_i - K, 0) \;\; ; \;\; 0 \le i \le imax - 1 \\ &V_{imax}^0 = S_{imax} \end{aligned}$$

These boundary conditions imply that the discrete equations (at nodes $i = 0, imax$) are

$$V_i^{n+1} = V_i^n \;\; ; \;\; i = 0, imax \tag{11}$$

Following the steps below to obtain the sharp bound

$$0 \le V_i^{n+1} \le S_i \tag{12}$$

(a) Show that $V_i^{n+1} \ge 0, \forall n$. Hint: write the discrete equations as

$$\begin{aligned} &\mathcal{Q}V^{n+1} = V^n + \text{Rest}^{n+1} \\ &\text{Rest}_i^{n+1} = \begin{cases} P_i^{n+1}(V_i^* - V_i^{n+1}), & 1 \le i \le imax - 1 \\ 0 \;\; , & \text{otherwise} \end{cases} \end{aligned}$$

where $\mathcal{Q}$ is an $\mathcal{M}$ matrix. Show that $\text{Rest}^{n+1} \ge 0$ and then use induction.

(b) Show that $(1 \le i \le imax - 1)$

$$S_i = S_i + qS_i\Delta\tau + \Delta\tau\alpha_i S_{i-1} + \Delta\tau\beta_i S_{i+1} - (\alpha_i + \beta_i + r)\Delta\tau S_i + P_i^{n+1}(S_i - S_i) \tag{13}$$

and of course

$$S_i = S_i \;\; ; \;\; i = 0, imax \tag{14}$$

Hint: The finite difference expressions for V, V_S, V_{SS} are exact if $V(S,t) = S$, i.e.

$$\mathcal{L}_i^{n+1} S = \mathcal{L}V\left(S_i, t^{n+1}\right)$$

where $\mathcal{L}_i^{n+1}$ is the finite difference approximation to $(\mathcal{L}V)(S_i, t^{n+1})$ in (9).

(c) Using equations (10,11,13,14) develop an equation for $E_i^{n+1} = S_i - V_i^{n+1}$ of the form

$$\begin{aligned}
&\mathcal{T}E^{n+1} = E^n + \overline{\text{Rest}}^{n+1} \\
&\overline{\text{Rest}}_i^{n+1} = \begin{cases} P_i^{n+1}(S_i - V_i^*) + qS_i\Delta\tau \ , & 1 \le i \le imax - 1 \\ 0 \ , & \text{otherwise} \end{cases}
\end{aligned}$$

where $\mathcal{T}$ is an $\mathcal{M}$ matrix. Show that $\overline{Rest}^{n+1} \ge 0$, and then use induction.

6 (10 marks) Graduate Student Question

Markowtiz's mean variance portfolio optimization is typically used to allocate stocks. When the asset returns are not normal, the standard deviation may not be an appropriate risk measure. Assume that n risky assets returns are $r_1, \cdots, r_n$. Consider the following CVaR risk constrained allocation problem:

$$\begin{aligned}
\max_{x_1,\cdots,x_n} \quad & \mathbf{E}\left(\sum_{i=1}^{n} r_i x_i\right) \\
& 0 \le x_i \le 1, \quad i = 1, \cdots, n \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i \le 1 \\
& \text{CVaR}_\beta\left(-r^T x\right) \le \rho
\end{aligned} \tag{15}$$

where x_i is the portfolio weight of the asset i, r_i is the(random) rate of return of asset i and r is the column vector $(r_1, \cdots, r_n)$.

Assume that we have M independent samples of the asset returns, the above CVaR minimization can be approximated by the simulation CVaR problem below :

$$\begin{aligned}
& \max_{(x,y,\alpha)} \ \sum_{i=1}^{n} \mathbf{E}(r_i) x_i \\
& 0 \le x \le 1 \\
& \textstyle\sum_i^n x_i \le 1 \\
\text{subject to} \quad & \alpha + \frac{1}{M(1-\beta)} \sum_{j=1}^{M} y_j \le \rho \\
& y_j \ge -x^T (r)_j - \alpha, \quad j = 1, \cdots, M \\
& y_j \ge 0, \ j = 1, \cdots, M
\end{aligned} \tag{16}$$

where $(r)_j$ denote the jth sample of vector of returns, $\mathbf{E}(r_i)$ can be computed as the average from M samples, and ρ is an upper bound on the CVaR risk.

You are provided here with data contains the monthly returns of the 218 hedge funds from August 1993 to Feburary 1996. Use this data to generate an efficient frontier with horizontal axis representing CVaR value and the vertical axis representing expected return. The efficient frontier should be generated by computing optimal portfolio with the upper bound $\rho = \text{linspace}(-0.01, 0.3, 100)$. Describe how the optimal holdings change as ρ increases.