Towards efficient nonlinear option pricing with GPU computing

Student Computational Finance Day 2016

Shih-Hau Tan, Kevin Parrott, Choi-Hong Lai

University of Greenwich

May. 23, 2016

Outline

1. Motivation
2. Newton-based solver
3. GPU computing implementation
4. Conclusion

Incomplete Market

transaction costs

liquidity

market feedback

uncertain volatility

$$dS = rSdt + \sigma SdW$$

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

nonlinear option pricing

Nonlinear Option Pricing

- Advantages
  - More reasonable and accurate option price
  - Easier to do model calibration
  - Can be used to design better hedging strategies
- Types of problems
  - Nonlinear Black-Scholes Equation
  - Hamilton-Jacobi-Bellman (HJB) Equation
  - Backward Stochastic Differential Equation
- Challenge
  - Complicated to solve a single problem

Nonlinear Black-Scholes Equation

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0,\ S > 0,\ t \in [0, T)$$

with

- $\sigma = \sigma_0(1 + Le\, sign(V_{SS}))^{1/2}$ (Leland model (1985))
- $\sigma = \phi(x)$ (Barles and Soner model (1998))
- $\sigma = \sigma_0(1 - \rho S V_{SS})^{-1}$ (Frey-Patie model (2002))
- $\sigma \in [\sigma_{min}, \sigma_{max}]$ (Uncertain volatility model (1995))

where

$$V_{SS} = \frac{\partial^2 V}{\partial S^2}$$

Large-Scale Problems

Apple Inc. (AAPL) ★ Watchlist

105.19 -0.48 (0.45%) NASDAQ - As of 4:00PM EDT

After Hours: **105.00** -0.19 (-0.18%) 6:46AM EDT

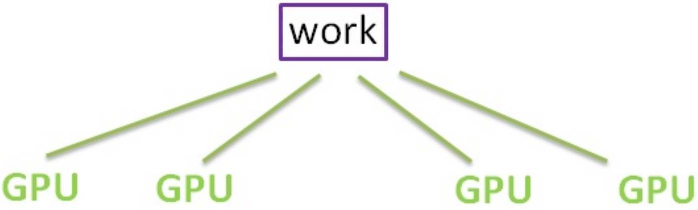
April 1, 2016 | In The Money | List | Straddle | Lookup Option

	Calls						Puts			
Last	Change	%Change	Volume	Open Interest	Strike	Last	Change	%Change	Volume	Open Interest
0.00	0.00	0.00%	0	0	65.00	0.04	0.00	0.00%	0	5
0.00	0.00	0.00%	0	0	70.00	0.06	0.00	0.00%	0	123
30.90	0.00	0.00%	18	18	75.00	0.03	0.00	0.00%	0	21
26.85	4.54	20.35%	10	7	80.00	0.01	0.00	0.00%	3	227
13.24	0.00	0.00%	0	1	85.00	0.01	-0.01	-50.00%	9	529
0.00	0.00	0.00%	0	0	85.50	0.01	-0.09	-90.00%	1	54
11.00	0.00	0.00%	0	13	86.00	0.01	-0.08	-88.89%	1	48
10.36	0.00	0.00%	0	2	86.50	0.04	0.00	0.00%	25	48
19.09	0.00	0.00%	1	1	87.00	0.05	0.00	0.00%	10	690
18.20	0.00	0.00%	150	96	87.50	0.01	0.00	0.00%	17	72
11.90	0.00	0.00%	0	3	88.00	0.01	0.00	0.00%	1	46
0.00	0.00	0.00%	0	0	88.50	0.01	0.00	0.00%	60	409
16.50	2.25	15.79%	1	10	89.00	0.02	-0.01	-33.33%	1	288

Objective

- Single nonlinear option pricing problem

- Large-scale nonlinear option pricing problems



Finite Difference Method

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0,\ S > 0,\ t \in [0, T) \tag{1}$$

with $\sigma = \sigma(V_t, V_S, V_{SS}, V)$.
Apply finite difference implicit scheme on equation (1)

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} + \frac{1}{2}(\sigma_i^{n+1})^2 S_i^2 \frac{V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}}{(\Delta S)^2} + rS_i \frac{V_{i+1}^{n+1} - V_{i-1}^{n+1}}{2\Delta S} - rV_i^{n+1} = 0,$$

which can be simplified as

$$a_i V_{i-1}^{n+1} + b_i V_i^{n+1} + c_i V_{i+1}^{n+1} = V_i^n, \tag{2}$$

where i represents the spatial discretization and n represents the temporal discretization.

Root-finding Problem

Equation (2) can be simplified as

$$H(V^{n+1})V^{n+1} = V^n,$$

where H is a tridiagonal matrix. Introduce

$$G(V^{n+1}) = H(V^{n+1})V^{n+1} - V^n = 0,$$

then the problem becomes to use Newton's method to solve the root-finding problem of the function G.

Algorithm (NM)

Given initial guess V^*, tolerance tol , and terminal condition $V^N = V(S, t = T)$

For n = N-1 to 1

1. Calculate a_i, b_i, c_i to construct H
2. Calculate G

If $\|G(V^*)\| < tol$, stop and $V^n = V^*$;

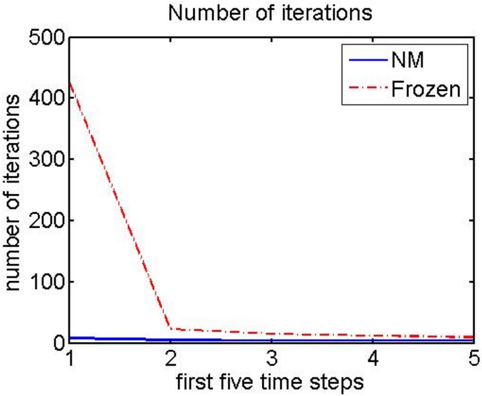
else $V^* = V^* - \left(Jac(G)\right)^{-1} G$ and go back to 1.

end

Newton's iteration

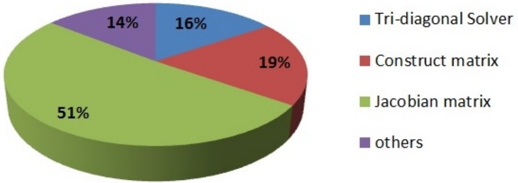
Features

- Can be applied to different cases with different schemes
- Complexity = #iterations×(linear model) + evaluation of Jacobian matrix
- Quadratic convergent rate



Complexity Analysis

$$G(V^{n+1}) = H(V^{n+1})V^{n+1} - V^n = 0$$



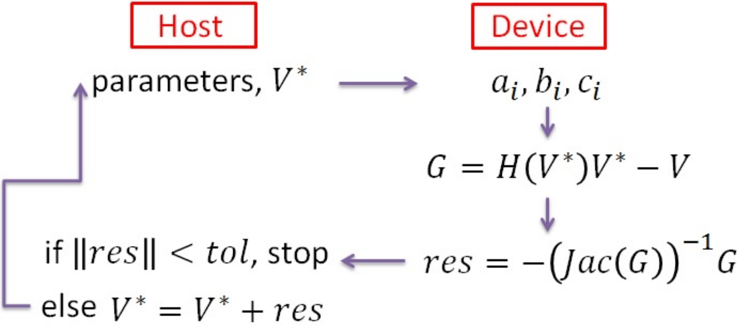
- Evaluate a_i, b_i, c_i for H at each iteration and time step
- Consider $H(V^{n+1}) = \Sigma^{n+1} H_1 + H_2,\ \Sigma^{n+1} = Diag((\sigma_i^{n+1})^2)$, then

$$Jac(G(V^{n+1})) = \frac{\partial [H(V^{n+1})V^{n+1}]}{\partial V^{n+1}} = H(V^{n+1}) + Diag(H_1 V^{n+1}) \nabla (\Sigma^{n+1})$$

- Tridiagonal solver for $(Jac(G))^{-1}$

Parallel Computing

At each time step, we have the Newton's iteration as:



Batch Operation

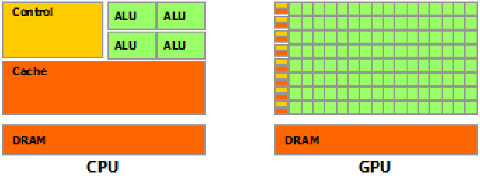
- Different option pricing problems such as different r, T, K, σ_0

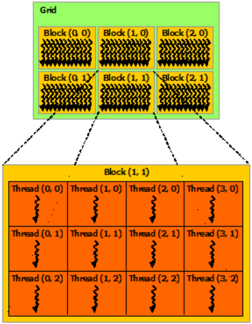
$$
\begin{pmatrix} b_1 & c_1 & \\ & \cdots & \\ & a_M & b_M \end{pmatrix} \longrightarrow \begin{pmatrix} b_1^1 & c_1^1 & \\ & \cdots & \\ & a_M^1 & b_M^1 \end{pmatrix} \cdots \begin{pmatrix} b_1^k & c_1^k & \\ & \cdots & \\ & a_M^k & b_M^k \end{pmatrix}
$$

- Combine all problems together

$$
\begin{pmatrix}
\begin{pmatrix} b_1^1 & c_1^1 & \\ & \cdots & \\ & a_M^1 & b_M^1 \end{pmatrix} & \mathbf{0} & \\
\mathbf{0} & \begin{pmatrix} b_1^2 & c_1^2 & \\ & \cdots & \\ & a_M^2 & b_M^2 \end{pmatrix} & \mathbf{0} \\
& \mathbf{0} & \begin{pmatrix} b_1^3 & c_1^3 & \\ & \cdots & \\ & a_M^3 & b_M^3 \end{pmatrix}
\end{pmatrix}
$$

GPU Computing





Implementation

- OpenACC
  - easier to start
    - #pragma acc kernels
      {
        code to be parallelised
      }
  - need to be careful on data movement
- CUDA library
  - **cusparseSgtsv** for tridiagonal solver
  - enable users to get good performance without writing too many codes
- Kernel functions
  - evaluate $a[i], b[i], c[i]$
  - Jacobian matrix
    - $Jac(G(V^{n+1})) = H(V^{n+1}) + Diag(H_1 V^{n+1})\nabla(\Sigma^{n+1})$
    - contains 2 matrix constructions and 2 level-2 function evaluations

Numerical Experiment

Consider Frey-Patie model with nonlinear volatility $\sigma = \sigma_0(1 - \rho S V_{SS})^{-1}$. Parameters are chosen to be

$$S \in [0, 300],\ K = 100,\ T = 1/12, \sigma_0 = 0.4,\ \rho = 0.01,\ r = 0.03,$$

and tolerance for Newton's iteration is $tol = 10^{-8}$ for double precision. The grid points for space and time are $M = N = 1000$. We calculate $64, 128, 256$ option pricing problems.

System information:

Processor: Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz

Memory: 4096MB RAM

Compiler: gcc 4.7, CUDA 7.0, PGI 15.9

Graphic card : **Quadro K2100M** (Kepler microarchitecture, compute capability 3.0)

Computation Time

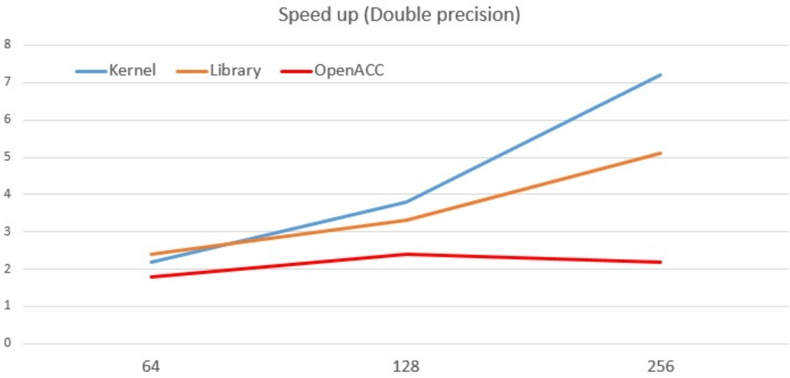
Table: Computation time (s) for double precision.

#Options	CPU	OpenACC	Library	Kernel
$n = 64$	24.8	13.5	10.2	10.9
$n = 128$	49.7	25.5	14.7	13.0
$n = 256$	122	66.5	23.9	17.0

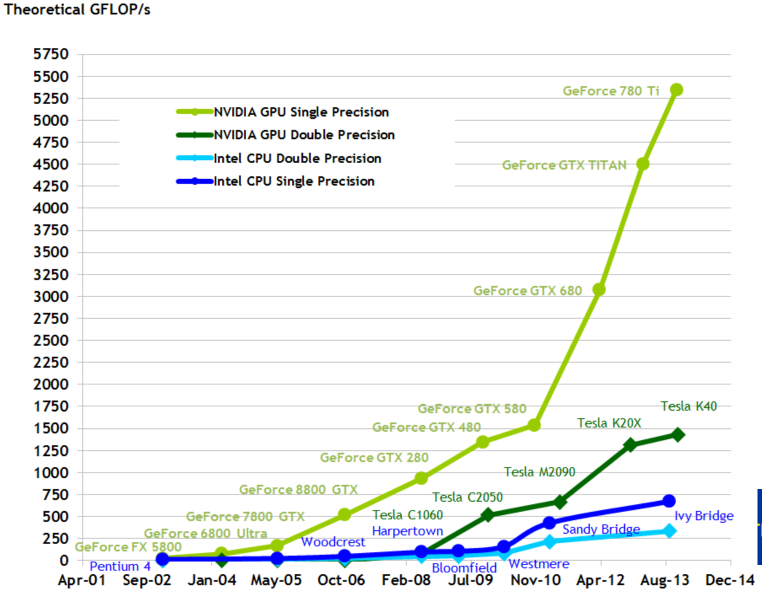
Table: Speed-up for double precision.

#Options	CPU	OpenACC	Library	Kernel
$n = 64$	1.0x	1.8x	2.4x	2.2x
$n = 128$	1.0x	1.9x	3.3x	3.8x
$n = 256$	1.0x	1.8x	5.1x	7.2x

Speed-up

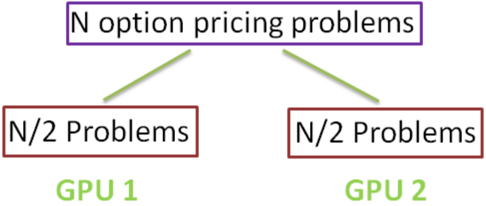

Speed up (Double precision)
Kernel
Library
OpenACC
8
7
6
5
4
3
2
1
0
64
128
256

Further Improvement



Multi-GPU Computing

- Split the work

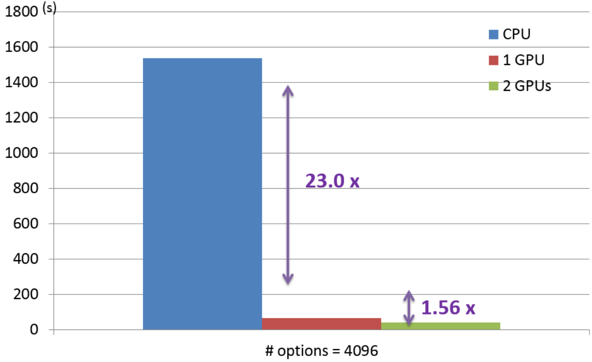


- Use OpenMP

```
#pragma omp parallel for
for (int i=0;i<Num_GPU;i++){
    cudaSetDevice(i);
    main code;
}
```

Multi-GPU Computing

- System information:
  Graphic cards : **GTX Titan Black** (Kepler microarchitecture, compute capability 3.5)
  Memory: 6143MB RAM
- Numerical result



Summarize

- Newton-based solver for nonlinear option pricing
- Batch operation for dealing with large-scale problems
- Comparison of different implementations of doing GPU computing
- Work in progress
    - Multi-asset problems with Alternating Direction Implicit (ADI) method
    - Asian option pricing problem with semi-Lagrangian scheme

- Reference

[1] M. Ehrhardt (edt): **Nonlinear Models in Mathematical Finance**, Nova Science Publishers, Inc. New York, 2008.
[2] J. Guyon, P. Henry-Labordère, **Nonlinear Option Pricing**, CRC Press, 2013.
[3] M.B. Giles, E. Laszlo, I. Reguly, J. Appleyard, J. Demouth, GPU implementation of finite difference solvers, Seventh Workshop on High Performance Computational Finance (WHPCF'14), IEEE, 2014.

- Acknowledgement
  Special thanks to
  - Mr. Lung-Sheng Chien, NVIDIA, USA.
  - Mr. Alvaro Leitao, CWI, the Netherlands.
  - Prof. Carlos Vázquez Cendón and Prof. Jose Antonio García Rodríguez, Universidade da Coruña, Spain.

Shih-Hau Tan
email: shihhau.tan@gmail.com
web: http://staffweb.cms.gre.ac.uk/~ts73/

Thank you very much!