FRE 6951 – Selected Topics in Financial Engineering - GPU's

CUDA C Programming for Financial Models

Lecture 1 – Introduction

Parallel Processing for Pricing Financial Derivatives

Louis Scott

April 5, 2019

FRE 6951 – Lecture 1, Introduction

- What can you expect to learn in this computing lab?
  1. Programming in C/C++ to use massive parallel processing on GPU's, using CUDA C
  2. Using GPU's on a workstation in the Windows operating system (optional: on a server using the Linux operating system)
  3. Strategies for designing programs to take advantage of both multi-threading on CPU's and parallel processing on multiple GPU's
- Topics to include programming for Monte Carlo simulations and finite difference algorithms. We will cover several different random number generators and how to run random number simulation in parallel.
- Computers in the Bloomberg lab are equipped with a GPU, designed for running multiple monitors, but these GPU's can run CUDA C. The CUDA software has been downloaded so that one can develop and run small programs on the machines: Microsoft's Interactive Development Environment (IDE) Visual Studio Professional hooked up with Nvidia's CUDA C.
- Option: Student can use accounts on the Prince High Performance Computing (HPC) cluster. The Prince cluster has a number of powerful GPU's for running applications.

Parallel Processing for Financial Applications

- Steady development in the computing technology. **Faster CPU's** plus **massive parallel processing with GPU's** (graphical processing unit) and **FPGA's** (field programmable gate array)
- Applications include **machine learning**, artificial intelligence (**AI**), big data, driverless cars, graphics, medical imaging, gaming, and last, but not least, **numerical solutions to math problems in Finance**
- Financial firms are actively using machine learning to perform routine tasks previously performed by humans. Examples include trading algorithms, portfolio management for investors (robo advisors), automation of payment systems, and online services in which the user interacts with a computer. I have even heard that some banks are planning to use machine learning to automate some of the model validation required by regulators. In the future, robots may be teaching Finance in Business Schools. (Data from Star Trek Next Generation)
- There are applications of neural networks and machine learning for predicting stock returns, but I have yet to see any examples of strategies that really outperform the market. We now have data sources and computing power to explore models and investment techniques that were considered to be impossible 10 years ago.

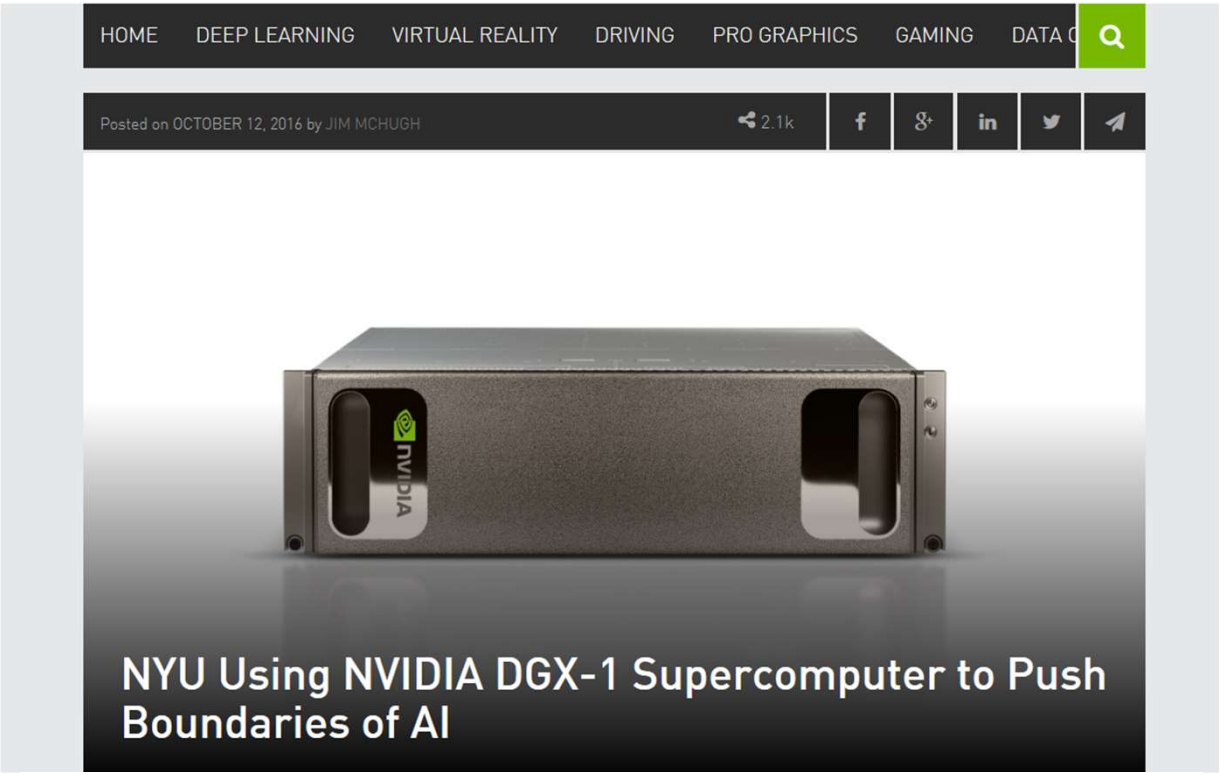
GPU Programming Languages

Numerical analytics	MATLAB Mathematica
Fortran	CUDA Fortran
C	CUDA C
C++	CUDA C++
Python	PyCUDA, Copperhead, Numba
F#	Alea.cuBase

Parallel Processing for Financial Derivatives

- **The most interesting finance problems typically require numerical solutions and are well suited for parallel processing: examples include derivative pricing, port-folio optimization, and algorithmic trading**
- **Massive parallel processing** coupled with the latest computing technology (**GPU's**) has made it possible to apply more robust, more complex financial models to capture the complexities observe in financial markets
- **Numerical techniques** like **Monte Carlo simulation** and **finite difference methods** are used by both academics in math finance and by quants at banks and trading firms
- **Vendor software** for financial applications incorporates **GPU's**: Murex, Numerix, Matlab, NAG, Quantifi
- And several of the **leading banks** incorporate **GPU's** for the daily processing in their **risk systems**

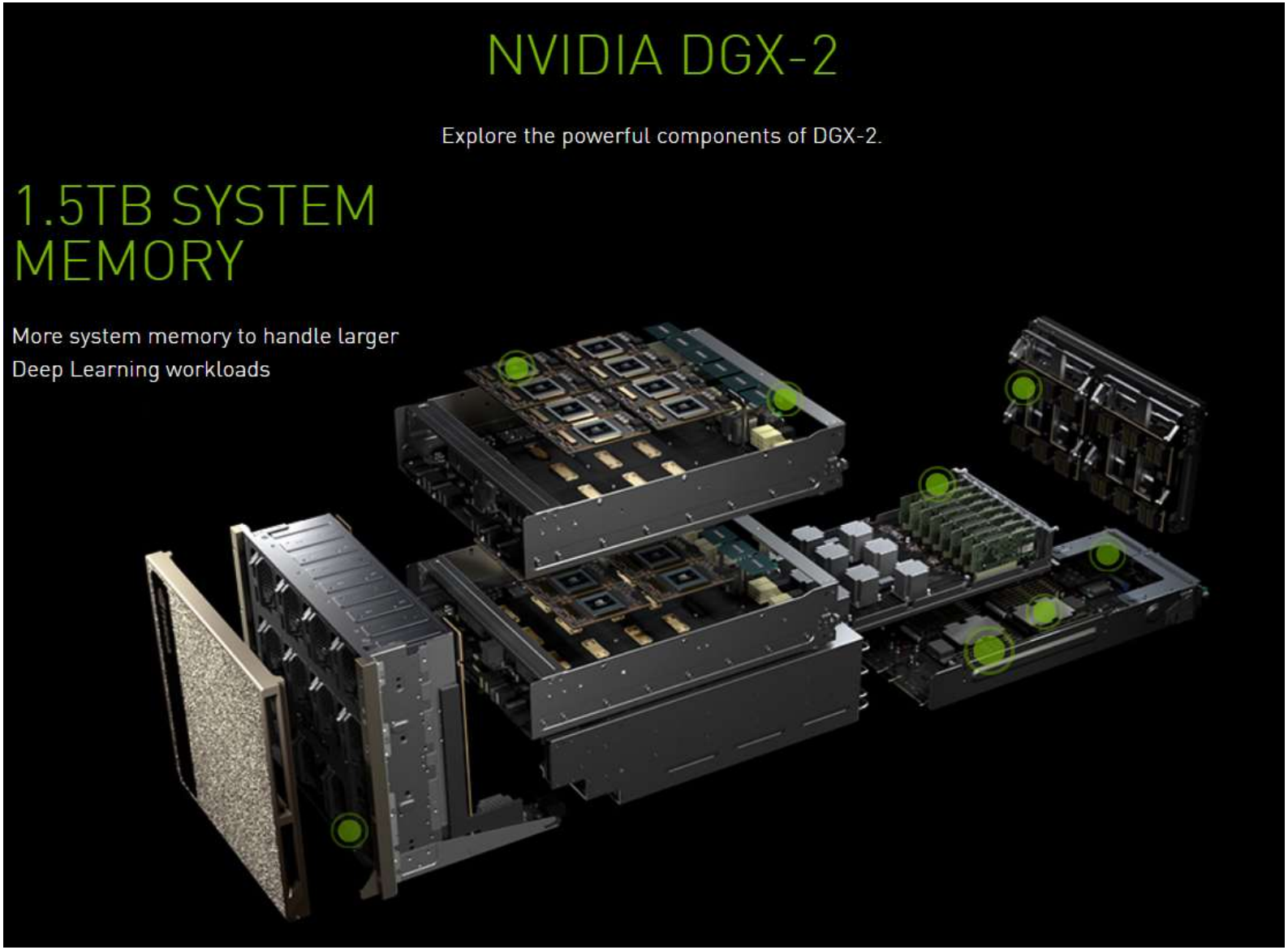
Parallel Processing Technology - 2017



New York University's Center for Data Science is at the cutting edge of fields with revolutionary implications such as machine learning, natural language processing, computer vision and intelligent machines.

Because computing speed is critical to accelerating experimentation and advancing research, the center's Computational Intelligence, Learning, Vision and Robotics (CILVR) lab recently acquired a NVIDIA DGX-1 AI supercomputer to fuel this work like never before.

Parallel Processing Technology - 2018



Parallel Processing Technology – June 2018

DESIGN > SUPERCOMPUTERS

IBM, Nvidia Build "World's Fastest Supercomputer" for US Government

The Summit supercomputer at ORNL, designed by IBM and Nvidia.

Computing Solutions on a Smaller Scale

Parallel Processing for Financial Derivatives

- This talk focuses on GPU applications for derivative pricing
- Implementing a derivative pricing model
  1) Choose a model and calibrate the model parameters to fit the model to market prices for actively traded option, including forwards and futures. Parameter calibration may include statistical estimation from historical time series, but typically it is based on calibration to a large data set of option prices.
  2) Use the model to structure and price derivative products that are not actively traded in financial markets. Use the model to manage hedges in the actively traded options and futures.
  3) Use the model to do relative value trading: buy underpriced options, sell overpriced options, and hedge exposure to market risk factors.
  4) If the model fails as a business tool, then use the model to publish an academic paper.
- Banks and trading firms have historically focused on simpler models that can be calculated quickly for managing a derivatives portfolio. For equity options, firms typically use the Black-Scholes model and the local volatility model.

Parallel Processing for Financial Derivatives

- Implementing more complex derivative pricing models – 2 examples

1) Calibrate a 3 Factor version of the Hull-White model and price interest rate derivatives: European swaptions and Eurodollar futures options.

2) Build a stochastic volatility model with jumps in the stock price and volatility and calibrate the model to SPX and SPY options.

Example 1: 3 Factor Hull-White Model

- 3 factor Hull-White model with normally distributed interest rates, and volatilities as deterministic functions of time. Model has a short rate that mean reverts to 2 stochastic mean factors, y_1 and y_2. The y_1 factor is set up with a slow rate of mean reversion and serves as the long run mean. The y_2 factor is set up with a zero mean and a slower rate of mean reversion and it is a curvature factor

$$
\begin{aligned}
dr &= [\kappa_0(\theta_0(t) + y_1 + y_2 - r) - \lambda_0\sigma_0^2(t)]\,dt + \sigma_0(t)dz_0 \\
dy_1 &= [\kappa_1(\theta_1 - y_1) - \lambda_1\sigma_1^2(t)]\,dt + \sigma_1(t)dz_1 \\
dy_2 &= (-\kappa_2 y_2 - \lambda_2\sigma_2^2(t))\,dt + \sigma_2(t)dz_2
\end{aligned}
$$

- r is the instantaneous interest rate, which we will proxy with the overnight rate.
- $\theta_0(t)$ is a deterministic function of time, which is used to ensure that the model can be calibrated to match the initial term structure of forward rates. We will start with $\theta_0(t) = 0$ for all t. λ parameters are risk premia, and are set to 0 for this example.
- The model is a 3 factor version of the Vasicek-Hull-White model. The volatilities will be calibrated so that the model fits the current yield curve and either matches or comes close on market implied volatilities (ED futures options, caps/floors, swaptions). I use swaption volatilities and ED futures options in the example.

Example 1: 3 Factor Hull-White Model

- First, we can solve the model to get the discount bond pricing function.

$$D(0,T) = \hat{E}_0\left(\exp - \int_0^T r(s)ds\right)$$

- Because *r* is normally distributed, the integral (summation) of *r* is also normally distributed. The easiest method for solving the discount function is to solve the PDE for the pricing function.

$$0 = \frac{\partial D}{\partial t} + \frac{1}{2}\sigma_0^2(t)\frac{\partial^2 D}{\partial r^2} + \frac{1}{2}\sigma_1^2(t)\frac{\partial^2 D}{\partial y_1^2} + \frac{1}{2}\sigma_2^2(t)\frac{\partial^2 D}{\partial y_2^2} + \kappa_0(\theta_0(t) + y_1 + y_2 - r - \lambda_0\sigma_0^2(t))\frac{\partial D}{\partial r}$$
$$+ \kappa_1(\theta_1 - y_1 - \lambda_1\sigma_1^2(t))\frac{\partial D}{\partial y_1} + \kappa_2(-y_2 - \lambda_2\sigma_2^2(t))\frac{\partial D}{\partial y_2} - rD$$

- The solution is an exponential affine function:

$$D(t,T) = \exp(-A(t,T) - B_0(t,T)r(t) - B_1(t,T)y_1(t) - B_2(t,T)y_2(t))$$

Example 1: 3 Factor Hull-White Model

- Evaluate the partial derivatives using the proposed solution and plug into the PDE. Then organize the derivatives with respect to each one of the state variables and the constant. Start by picking up all of the terms in the PDE which include r:

$$0 = -\frac{\partial B_0}{\partial t} + \kappa_0 B_0(t,T) - 1$$

- Then collect the terms associated with y_1 and y_2, and the remaining terms which are associated with a constant (does not vary with any state variables)

y_1:
$$0 = -\frac{\partial B_1}{\partial t} + \kappa_1 B_1(t,T) - \kappa_0 B_0(t,T)$$

y_1:
$$0 = -\frac{\partial B_2}{\partial t} + \kappa_2 B_2(t,T) - \kappa_0 B_0(t,T)$$

Constant:

$$0 = -\frac{\partial A}{\partial t} + \frac{1}{2}\sigma_0^2(t)B_0^2(t,T) + \frac{1}{2}\sigma_1^2(t)B_1^2(t,T) + \frac{1}{2}\sigma_2^2(t)B_2^2(t,T)$$
$$(\lambda_0\sigma_0^2(t) - \kappa_0\theta_0(t))B_0(t,T) + (\lambda_1\sigma_1^2(t) - \kappa_1\theta_1)B_1(t,T) + \lambda_2\sigma_2^2(t)B_2(t,T)$$

Example 1: 3 Factor Hull-White Model

- These equations are ordinary differential equations (ODE) and they are easy to solve.

$$B_0(t,T) = \frac{1 - e^{-\kappa_0 (T-t)}}{\kappa_0}$$

$$B_1(t,T) = \int_t^T e^{-\kappa_1 (s-t)} \kappa_0 B_0(s,T)\, ds$$

$$B_1(t,T) = \frac{1 - e^{-\kappa_1 (T-t)}}{\kappa_1} - e^{-\kappa_1 (T-t)} \left(\frac{1 - e^{-(\kappa_0 - \kappa_1)(T-t)}}{\kappa_0 - \kappa_1} \right)$$

$$B_2(t,T) = \int_t^T e^{-\kappa_2 (s-t)} \kappa_0 B_0(s,T)\, ds$$

$$B_2(t,T) = \frac{1 - e^{-\kappa_2 (T-t)}}{\kappa_2} - e^{-\kappa_2 (T-t)} \left(\frac{1 - e^{-(\kappa_0 - \kappa_2)(T-t)}}{\kappa_0 - \kappa_2} \right)$$

Example 1: 3 Factor Hull-White Model

- The constant term involves an integral which can be solved analytically. In the interest of time, I solve it numerically on a computer using an ODE solver.

$$A(t,T) = -\int_t^T \left(\frac{1}{2}\sigma_0^2(s)B_0^2(s,T) + \frac{1}{2}\sigma_1^2(s)B_1^2(s,T) + \frac{1}{2}\sigma_2^2(s)B_2^2(s,T) + (\lambda_0\sigma_0^2(t) - \kappa_0\theta_0(t))B_0(s,T) + (\lambda_1\sigma_1^2(s) - \kappa_1\theta_1)B_1(s,T) + \lambda_2\sigma_2^2(t)B_2(s,T)\right) ds$$

- You can verify that this solution works for the discount bond pricing function by evaluating the partial derivatives and plugging back into the PDE.
- For a model application, I simplify the model by setting the $\theta_0(t) = 0$ and making all of the volatilities constants. The model will not provide an exact fit for the initial term structure. I will try to see how close I can get without $\theta_0(t)$. If you were managing an interest rate derivatives portfolio, you should incorporate the $\theta_0(t)$ term to produce the exact fit. Asset management firms would apply this model without the $\theta_0(t)$ term and look for relative value trading opportunities.
- The solution for the zero rates is a linear function in the state variables

$$R(0,T) = \frac{1}{T}\left(A(0,T) + B_0(0,T)\, r(0) + B_1(0,T)\, y_1(0) + B_2(0,T)\, y_2(0)\right)$$

Example 1: 3 Factor Hull-White Model

- I apply this model to OIS. And I treat 3m LIBOR as a deterministic spread over 3m OIS. From the model, 3m OIS is determined as follows

$$D(3m) = \exp(-A(3m) - B_0(3m)r(t) - B_1(3m)y_1(t) - B_2(3m)y_2(t))$$
$$= \frac{1}{1 + R_{3m}(t) \times \frac{Days\,(3m)}{360}} = \exp\left(-r_{3m}(t) \times \frac{Days\,(3m)}{365}\right)$$

- Here r_{3m} is the 3m zero rate for OIS, continuously compounded, and R_{3m} is the simple interest rate for 3m OIS. *T-t* for 3*m* will be close to 0.25. 3m LIBOR, the simple interest rate, is modeled as a fixed spread over 3m OIS.

$$R_{3mL}(t) = R_{3m}(t) + SPR_{3mL}$$

Example 1: 3 Factor Hull-White Model

- The simulation model for derivatives on 3 month LIBOR uses the following model for fixing the spread: $R_{3mL}(T) = R_{3m}(T)$ + Spread. I use the model to calculate the 3m OIS rate and calculate the spread between the actual market quote for 3m LIBOR and the model rate for 3m OIS. I do the same for all of the forward 3m LIBORs: calculate spread between forward 3m OIS and forward 3m LIBOR.

- First, I calibrate the model parameters to fit the initial OIS yield curve, and then I adjust the volatilities to get the model close on the valuation for a few at-the-money swaptions.

- I set up and run the model calculations for the discount bond coefficients in Excel. I use a C++ program to do the numerical integration for the constant terms, and to do the Monte Carlo simulations for the options. The C++ programs are compiled as dll's, which I can call from Excel. Separate C++ programs do the finite difference solutions for American options. The Monte Carlo simulations and the finite difference solutions have been programmed to run on the CPU or a GPU.

→ go to demo in Excel Workbook

→ Go to 3d finite difference solutions in Visual Studio

Example 2: Models for SPX and SPY

- The 2nd example is a model for pricing for options on the S&P 500 index (known by its ticker symbol, SPX) and options on the S&P 500 ETF (SPY). A variety of models are used in the industry for pricing and risk management.

- **Black-Scholes Model:** $dS = (r(t) - \delta(t))S\,dt + \sigma(t)\,dz\,.$

- **Local Volatility Model:** $dS = (r(t) - \delta(t))S\,dt + \sigma(t,S)\,dz\,.$

- Empirical studies in finance: (1) stock price volatility changes (randomly) over time and (2) changes in both stock prices and volatility have large changes which can be modeled with simultaneous jump processes. Hence the **curse of complexity**

- **Stochastic Volatility Models with Jump Processes:**

$$d\ln S(t) = [r(t) + \lambda(t) - \delta(t) - \mu_J^*(t) - \tfrac{1}{2}v(t)]\,dt + \sqrt{v(t)}\,dz_1 + dJ_1$$

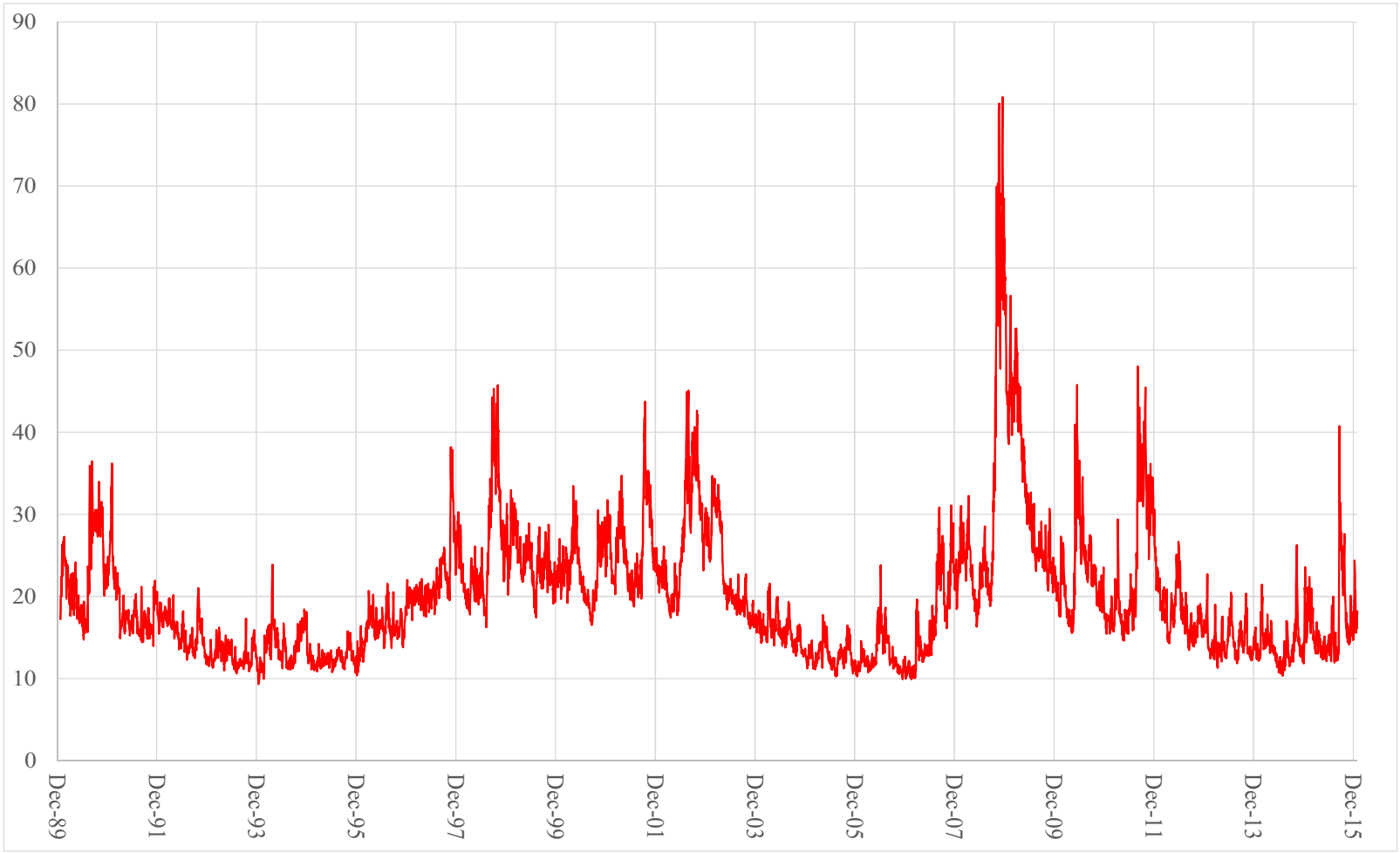
$$dv(t) = \mu_v(t,v)\,dt + \sigma_v(t,v)\,dz_2 + dJ_2$$

Empirical Models for SPX and VIX

Figure 1. S&P 500 Stock Price Index, SPX, 1990-2015

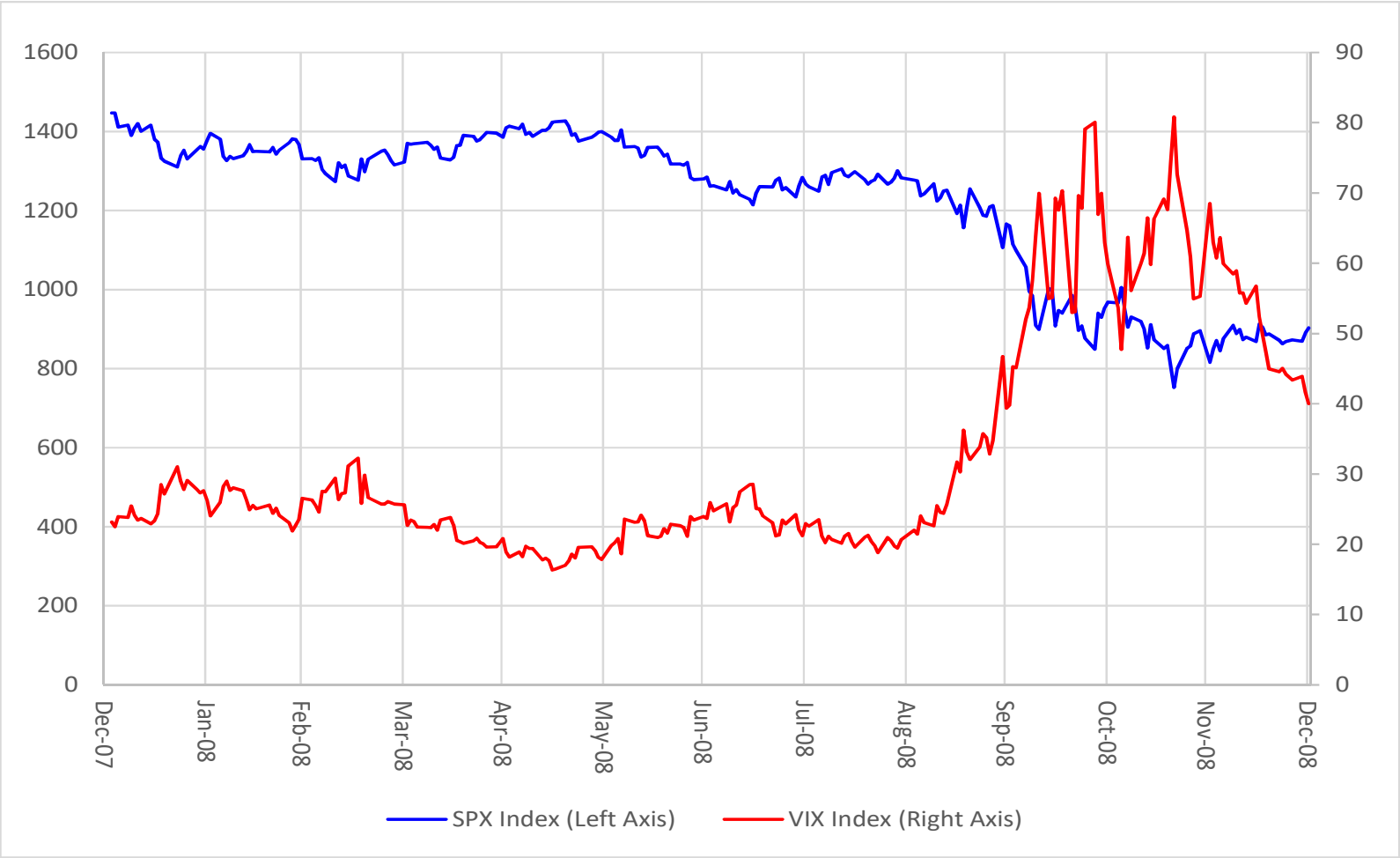
Empirical Models for SPX and VIX

Figure 2. VIX Time Series, 1990-2015



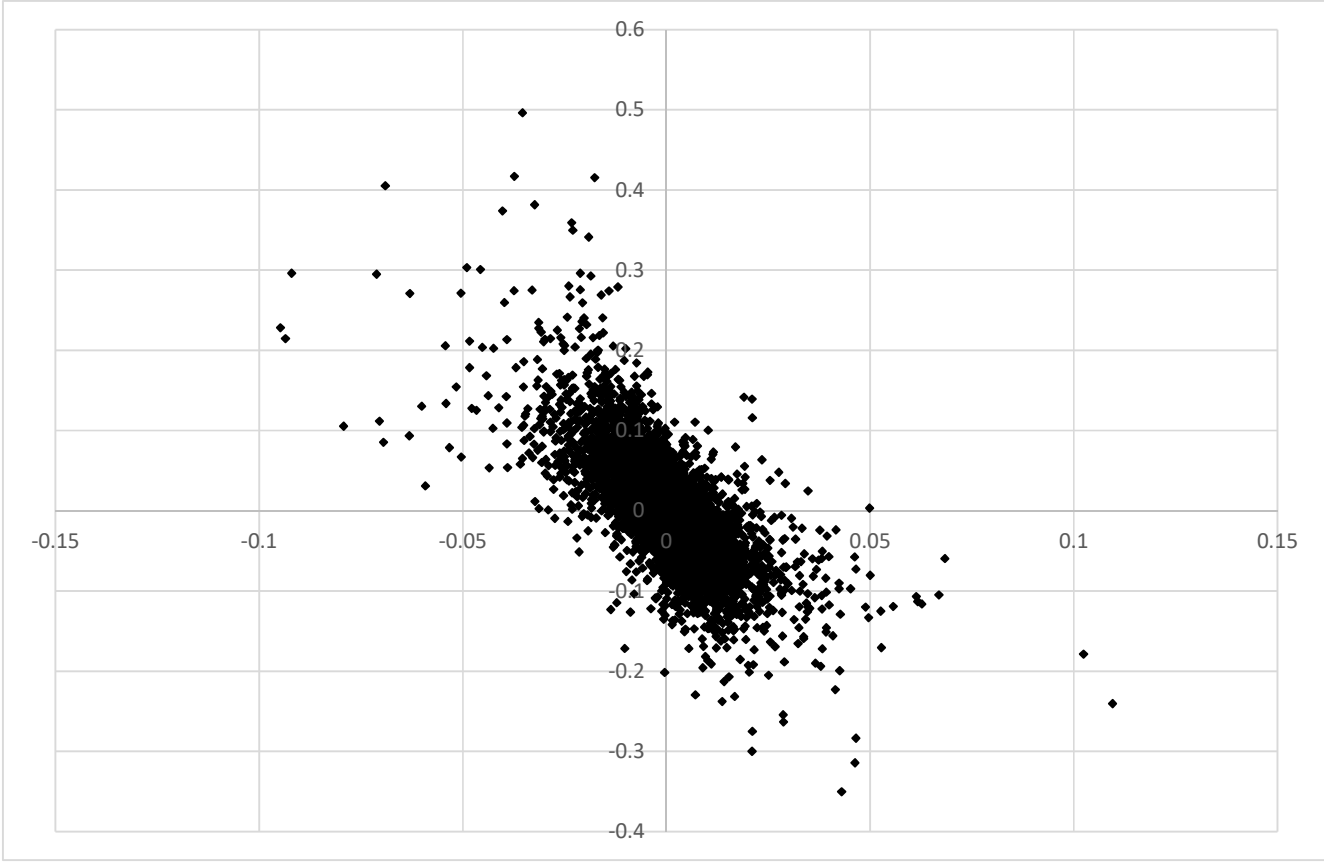
Empirical Models for SPX and VIX

Figure 3. SPX and VIX, during the financial crisis



Empirical Models for SPX and VIX

Figure 4. Scatter Diagram, Change in log VIX versus Change in log SPX



Empirical Models for SPX and VIX

- **Processes for the Derivative Pricing Model**

$$d\ln S = [r(t)+\lambda(t)-\delta(t)-\mu_J^*(t)-\tfrac{1}{2}v]\,dt+\sqrt{v}\,dz_1+dJ_1$$

$$dv = \kappa[\theta(t)-v]\,dt+\sigma v^{\beta}dz_2+dJ_2$$

- **SPX: a GARCH model with jumps and VIX squared in the conditional variance**

$$\Delta\log S_t = r_t\,\Delta t + \lambda + \sqrt{h_t}\,\varepsilon_t + \Delta J_t$$

$$h_t = a_0 + a_1 h_{t-1} + a_2(\sqrt{h_t}\,\varepsilon_{t-1} + \Delta J_{t-1})^2 + b\,Var_{t-1}$$

- **VIX: a GARCH model with jumps**

$$\Delta v_t = c_0 + c_1 v_{t-1} + c_2 v_{t-2} + v_{t-1}^{\beta}(\sqrt{h_{vt}}\,\varepsilon_{vt} + \Delta J_{vt})$$

$$h_{vt} = a_{v0} + a_{v1} h_{v,t-1} + a_{v2}(\sqrt{h_{vt}}\,\varepsilon_{vt} + \Delta J_{vt})^2$$

Empirical Model for SPX – MLE Results

- **Summary Results for Maximum Likelihood Estimation of SPX empirical model**
  - The GARCH parameters are not statistically significant.
  - The *b* coefficient for VIX squared is significant and less than 1.
  - The jump parameters are significant.
  - The standard likelihood ratio tests indicate reveal that the empirical model needs to include the VIX squared proxy and the jump processes

$$\Delta \log S_t = r_t \, \Delta t + \lambda + \sqrt{h_t} \, \varepsilon_t + \Delta J_t$$

$$h_t = a_0 + b \, Var_{t-1}$$

Empirical Model for VIX – MLE Results

- **Summary Results for Maximum Likelihood estimation of VIX empirical model**
  - The ML estimate for the beta coefficient is close to 1.
  - The GARCH parameters are statistically significant, indicating that there is some stochastic vol of vol.
  - The jump parameters are statistically significant.
  - The log model fits the data as well as the beta model.

$$\Delta v_t = c_0 + c_1 v_{t-1} + c_2 v_{t-2} + v_{t-1}^{\beta} (\sqrt{h_{vt}}\ \varepsilon_{vt} + \Delta J_{vt})$$

$$h_{vt} = a_{v0} + a_{v1} h_{v,t-1} + a_{v2} (\sqrt{h_{vt}}\ \varepsilon_{vt} + \Delta J_{vt})^2$$

Derivative Pricing Models for SPX

- **Stochastic Processes for Derivative Pricing Model with log process for stochastic volatility**

$$d \log S(t) = [r(t) - \delta(t) - \mu_J^*(t) - \tfrac{1}{2} v(t)]\, dt + \sqrt{v(t)}\, dz_1 + dJ_{1S} + dJ_{3S}$$

$$v(t) = \exp(y(t))$$

$$dy(t) = \kappa [\, \theta(t) - y(t)\,]\, dt + \sigma(t)\, dz_2 + dJ_{1y} + dJ_{3y}$$

- **General solution for European options, calls and puts on SPX**

$$V_C = \hat{E}_0 \left(\exp\left(- \int_0^T r(s)\, ds \right) \max[\, 0,\, S(T) - K\,] \right)$$

$$V_P = \hat{E}_0 \left(\exp\left(- \int_0^T r(s)\, ds \right) \max[\, 0,\, K - S(T)\,] \right)$$

- Prices for the European options (no early exercise) can be computed by simulating (Monte Carlo) the dynamic equations above. The model has time dependencies which can be easily handled in the simulation.

Derivative Pricing Models for SPY

- **General solution for American options, calls and puts on SPY**

for $0 \le t < T$

$$V_c(t) = \hat{E}_t\left(\exp\left(-\int_t^{t+\Delta t} r(s)\,ds\right)V_c(t+\Delta t)\right),\ V_c(t) \ge \max[0, S(t)-K]$$

$$V_p(t) = \hat{E}_t\left(\exp\left(-\int_t^{t+\Delta t} r(s)\,ds\right)V_p(t+\Delta t)\right),\ V_p(t) \ge \max[0, K-S(t)]$$

at expiration

$$V_c(T) = \max[0, S(T)-K] \qquad V_p(T) = \max[0, K-S(T)]$$

- Grid based solutions in which the valuation is solved by working backwards from expiration. Grid based techniques: finite difference, lattice using approximations for transition probabilities.

Derivative Pricing Models for SPY

- **Valuation function** must satisfy the following **partial integral differential equation (P.I.D.E.)**, subject to boundary conditions

$$0 = \frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial x^2}v(t) + \frac{1}{2}\frac{\partial^2 V}{\partial y^2}\sigma^2(t) + \rho\sigma(t)\sqrt{v(t)}\frac{\partial^2 V}{\partial x\,\partial y}$$

$$+ \frac{\partial V}{\partial x}[r(t)-\delta(t)-\lambda_J(t)\mu_J^* - \tfrac{1}{2}v(t)] + \frac{\partial V}{\partial y}\kappa(\theta(t)-y) - rV$$

$$+ \lambda_{J1}(t)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}[V(t,x+u_0,y+u_1)-V(t,x,y)]\frac{\exp\left(-\frac{1}{2}\frac{(u_0-\mu_{J1}-\rho_J u_1)^2}{\sigma_{J1}^2} - \frac{1}{2}\frac{(u_1-\mu_{J2})^2}{\sigma_{J2}^2}\right)}{2\pi\,\sigma_{J1}\,\sigma_{J2}}du_0du_1$$

$$+ \lambda_{J3}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}[V(t,x+u_0,y+u_1)-V(t,x,y)]\frac{\exp\left(-\frac{1}{2}\frac{(u_0-\mu_{J3}-\rho_{J34}u_1)^2}{\sigma_{J3}^2} - \frac{1}{2}\frac{(u_1-\mu_{J4})^2}{\sigma_{J4}^2}\right)}{2\pi\,\sigma_{J3}\,\sigma_{J4}}du_0du_1$$

Derivative Pricing Models for SPY

Techniques for solving this P.I.D.E.

- For the jump processes, integrate across the finite difference grid using the bivariate normal probability function
- For the diffusion parts of the equation, use time splitting. Solve explicitly on the covariance term (normally one can transform equation to eliminate cross term, but not in this case)
- Use the alternating directions implicit method (ADI) for the 2 state variables
- I solve the terms for y (stochastic volatility) explicitly, and then I solve the terms for $x = \log S$ implicitly. This method is marginally faster than solving implicitly (ADI) on both y and x

Calibrating the Model Parameters

Some Finance for the Equity Option Model

- The model was calibrated to SPX option prices from the close on September 2, 2016. Some additional detail at end of presentation. (Monte Carlo simulation using 4 GPU's)

- A term structure of dividend yields was calculated from the put-call parity formula applied to the at-the-money calls and puts for SPX. Dividends are paid daily on the stocks in the SPX index.

- The same model parameters were applied for the SPY ETF options. The dividend payments for the SPY ETF are discrete quarterly payments. The starting stock price and the dividend adjustments are different for the SPY options. At-the-money puts and calls for the SPY have been used to estimate an initial discrete dividend for September 2016, and subsequent discrete dividend yields.

Calibrating the Model Parameters

Table 1. SPX Options used in calibration, from close on September 2, 2016

Expiration	Days to Expiration	Number of Options	Strike Range
9/16/2016	14	95	1725 – 2300
10/21/2016	49	190	1000 – 2450
11/18/2016	77	146	800 – 2500
12/16/2016	105	149	700 – 2600
1/20/2017	140	78	650 – 2600
3/17/2017	196	68	400 – 2700
6/16/2017	287	79	300 – 2750
12/15/2017	469	92	200 – 3500
12/21/2018	840	94	100 – 3500

Note: Put options for strikes below the forward price, call options for strikes above the forward price, plus both the put and the call for the strike closest to the forward price. The calibration uses only options with bid prices > 0.

Time Tests

- **Two sets of time tests:** one on a desktop computer equipped with GPU's and one using Nvidia's V100 GPU on AWS.
- **First Set (Desktop Computer):**
  - Compare Monte Carlo simulations run sequentially on the CPU versus Monte Carlo simulations run in parallel on the GPU: Speedup factors with the GPU vary between 35 and 500
  - Compare the finite difference solution run on the CPU versus the finite difference solution run in parallel on the GPU: Speedup factors with the GPU vary between 95 and 125
- **Second Set (V100 on AWS):** compare the finite difference solution run on the CPU versus the finite difference solution run in parallel on the GPU
  - The Monte Carlo simulations run slightly faster on the CPU using the Linux operating system. The speedup factors on the V100 are much greater: between 125 and 700 times.
  - The speedup factors for the finite difference solutions on the V100 are also much greater: between 500 and 1000 times.

Table - Computing Times for Monte Carlo Simulation, SPX Options (Windows 7 – Desktop, Visual Studio and Cuda 7)

A. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
			Compute Time (seconds)	Speedup Factor
Days to Expiration	105	CPU	27.862	
Number of Options	4	GPU	0.593	47.0
		GPU, w/o mem. time	0.390	71.4
B. 2,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	105	CPU	56.566	
Number of Options	4	GPU	0.905	62.5
		GPU, w/o mem. time	0.734	77.1
C. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	105	CPU	28.017	
Number of Options	149	GPU	0.787	35.6
		GPU, w/o mem. time	0.587	47.7
D. 2,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	105	CPU	58.453	
Number of Options	149	GPU	1.315	44.5
		GPU, w/o mem. time	1.125	52.0
E. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	287	CPU	81.863	
Number of Options	79	GPU	1.099	74.5
		GPU, w/o mem. time	1.001	81.8
F. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	840	CPU	234.152	
Number of Options	94	GPU	2.702	86.7
		GPU, w/o mem. time	2.567	91.2

Table - Computing Times for Monte Carlo Simulation, SPX Options (Windows 7 – Desktop, Visual Studio and Cuda 7) (using optimized function for normal random number generation on GPU)

A. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)

			Compute Time (seconds)	Speedup Factor
Days to Expiration	105	CPU	27.862	
Number of Options	4	GPU	0.290	96.1
		GPU, w/o mem. time	0.100	278.6

B. 2,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)

			Compute Time	Speedup Factor
Days to Expiration	105	CPU	56.566	
Number of Options	4	GPU	0.395	143.2
		GPU, w/o mem. time	0.200	282.8

C. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)

			Compute Time	Speedup Factor
Days to Expiration	105	CPU	28.017	
Number of Options	149	GPU	0.542	51.7
		GPU, w/o mem. time	0.345	81.2

D. 2,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)

			Compute Time	Speedup Factor
Days to Expiration	105	CPU	58.453	
Number of Options	149	GPU	0.790	74.0
		GPU, w/o mem. time	0.620	94.3

E. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)

			Compute Time	Speedup Factor
Days to Expiration	287	CPU	81.863	
Number of Options	79	GPU	0.470	174.2
		GPU, w/o mem. time	0.270	303.2

F. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)

			Compute Time	Speedup Factor
Days to Expiration	840	CPU	234.152	
Number of Options	94	GPU	0.655	357.5
		GPU, w/o mem. time	0.470	498.2

Table - Computing Times for Finite Difference Solutions, SPY Options (Windows 7 – Desktop, Visual Studio and Cuda 7)

A. Finite Difference Method (1 time step per trading day)

			Compute Time (seconds)	Speedup Factor
Days to Expiration	**77**	CPU	143.870	
1 put, $K = 217$		GPU	1.419	101.4
		GPU, w/o mem. time	1.276	112.8
			Dimension	Range
Valuation - CPU	4.668	x grid (log S)	752	3.2615 – 5.7066
Valuation - GPU	4.668	y grid (log v)	219	-35.0 – 12.0

B. Finite Difference Method (1 time step per trading day)

			Compute Time	Speedup Factor
Days to Expiration	**105**	CPU	146.547	
1 put, $K = 217$		GPU	1.531	95.7
		GPU, w/o mem. time	1.340	109.4
			Dimension	Range
Valuation - CPU	6.207	x grid (log S)	827	3.0210 – 5.7244
Valuation - GPU	6.206	y grid (log v)	183	-35.0 – 12.0

C. Finite Difference Method (1 time step per trading day)

			Compute Time	Speedup Factor
Days to Expiration	**196**	CPU	341.718	
1 put, $K = 217$		GPU	3.110	109.9
		GPU, w/o mem. time	2.934	116.5
			Dimension	Range
Valuation - CPU	9.908	x grid (log S)	1002	2.6415 – 5.7919
Valuation - GPU	9.907	y grid (log v)	183	-35.0 – 12.0

D. Finite Difference Method (1 time step per trading day)

			Compute Time	Speedup Factor
Days to Expiration	**287**	CPU	510.887	
1 put, $K = 217$		GPU	4.263	119.8
		GPU, w/o mem. time	4.095	124.8
			Dimension	Range
Valuation - CPU	13.125	x grid (log S)	1028	2.4093 – 5.8046
Valuation - GPU	13.123	y grid (log v)	183	-35.0 – 12.0

Table - Computing Times for Monte Carlo Simulation, SPX Options (Ubuntu Linux - Desktop and AWS, using optimized norminv function on GPU)

			Compute Time (seconds)	Speedup Factor
A. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)				
Days to Expiration	105	CPU – i7 5630K	23.651	
Number of Options	4	GPU – Desktop, 980 Ti	0.412	57.4
		GPU – Tesla K80	0.169	139.9
		GPU – Tesla V100	0.143	165.4
		w/o mem. time	0.067	353.0
B. 2,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)			Compute Time	Speedup Factor
Days to Expiration	105	CPU – i7 5630K	47.222	
Number of Options	4	GPU – Desktop, 980 Ti	0.451	104.7
		GPU – Tesla K80	0.241	195.9
		GPU – Tesla V100	0.177	266.8
		w/o mem. time	0.101	467.5
C. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)			Compute Time	Speedup Factor
Days to Expiration	105	CPU – i7 5630K	25.402	
Number of Options	149	GPU – Desktop, 980 Ti	0.646	39.3
		GPU – Tesla K80	0.616	41.2
		GPU – Tesla V100	0.197	128.9
		w/o mem. time	0.120	211.7
D. 2,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)			Compute Time	Speedup Factor
Days to Expiration	105	CPU – i7 5630K	47.715	
Number of Options	149	GPU – Desktop, 980 Ti	0.889	53.72
		GPU – Tesla K80	0.875	54.5
		GPU – Tesla V100	0.263	181.4
		w/o mem. time	0.190	251.1
E. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)			Compute Time	Speedup Factor
Days to Expiration	287	CPU – i7 5630K	66.193	
Number of Options	79	GPU – Desktop, 980 Ti	0.599	110.5
		GPU – Tesla K80	0.373	177.5
		GPU – Tesla V100	0.276	239.8
		w/o mem. time	0.200	331.0
F. 1,000,000 Simulations on 1 Thread (at least 2 time steps per trading day)			Compute Time	Speedup Factor
Days to Expiration	840	CPU – i7 5630K	188.385	
Number of Options	94	GPU – Desktop, 980 Ti	0.799	235.8
		GPU – Tesla K80	0.671	280.8
		GPU – Tesla V100	0.327	576.1
		w/o mem. time	0.257	733.0

Table - Computing Times for Finite Difference Solutions, SPY Options (Ubuntu Linux - Desktop and AWS)

A. Finite Difference Method (1 time step per trading day)				
			Compute Time (seconds)	Speedup Factor
Days to Expiration	**77**	CPU – i7 5630K	140.835	
1 put, $K = 217$		GPU – Desktop, 980 Ti	17.521	8.0
		GPU – Tesla K80	2.642	53.3
		GPU – Tesla V100	0.211	667.5
		w/o mem. time	0.136	1035.6
B. Finite Difference Method (1 time step per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	**105**	CPU – i7 5630K	145.725	
1 put, $K = 217$		GPU – Desktop, 980 Ti	23.000	6.3
		GPU – Tesla K80	3.368	43.3
		GPU – Tesla V100	0.268	543.8
		w/o mem. time	0.193	755.1
C. Finite Difference Method (1 time step per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	**196**	CPU – i7 5630K	337.379	
1 put, $K = 217$		GPU – Desktop, 980 Ti	53.420	6.3
		GPU – Tesla K80	8.206	41.1
		GPU – Tesla V100	0.427	790.1
		w/o mem. time	0.358	942.4
D. Finite Difference Method (1 time step per trading day)				
			Compute Time	Speedup Factor
Days to Expiration	**287**	CPU – i7 5630K	459.234	
1 put, $K = 217$		GPU – Desktop, 980 Ti	76.585	6.0
		GPU – Tesla K80	12.016	38.2
		GPU – Tesla V100	0.584	786.4
		w/o mem. time	0.514	893.5

Model Calibration for SPX and SPY Options

- The model was calibrated to SPX option prices for September 2, 2016. The options for the calibration are included back in Table 1. State variables and model parameters were set to minimize the sum of squared errors across all of the option prices.
- The next slide shows the calibrated parameter values. Slides at the end of the presentation include graphical displays of the fit of the model to bid and ask prices for SPX options and SPY options. The options included for each calibration are the out-of-the-money calls and puts plus the at-the-money put and call for each expiration.
- The model generally fits the option prices by falling inside the bid and ask prices. Most of the exceptions (mispricings) occur with the very short dated options. For example, the model overprices out-of-the-money calls with 2 weeks to expiration; or one can say that the market underprices these options relative to the model.

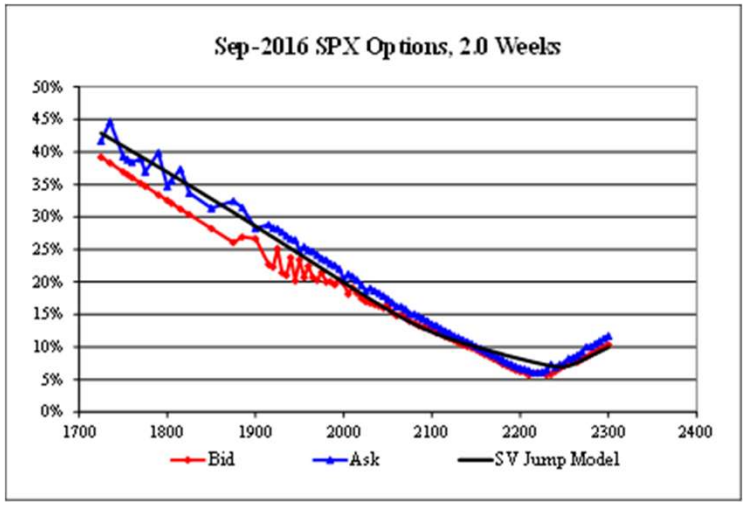
Model Parameters from the Calibration

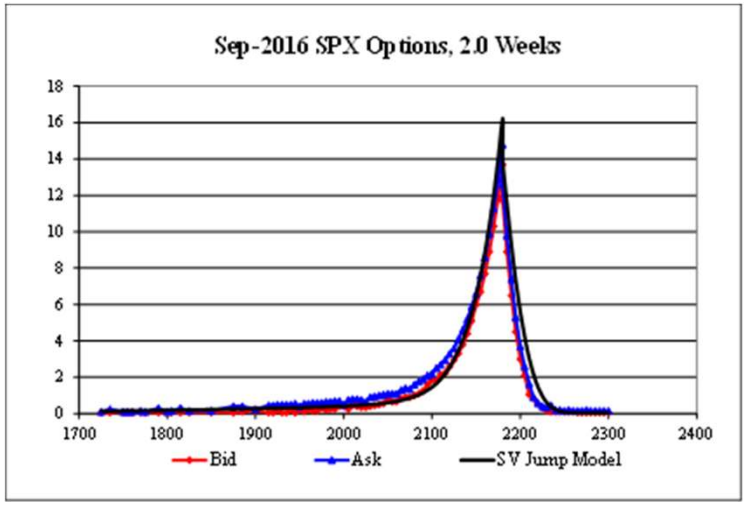
Table 2. Calibrated Parameters, September 2, 2016

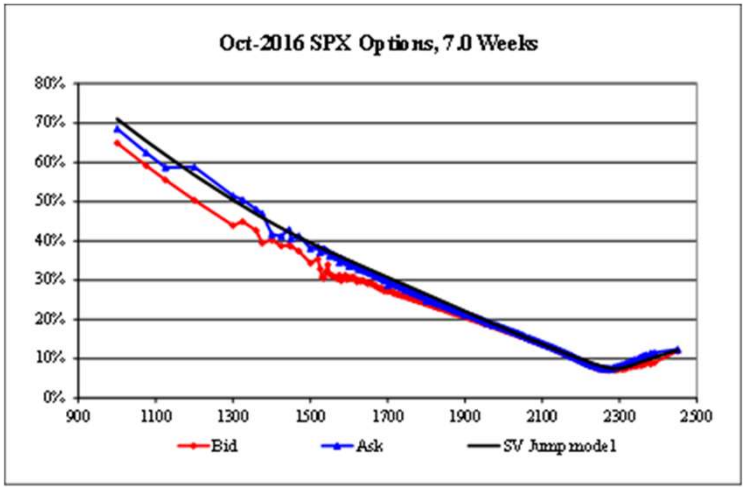
SPX	2179.98	SPY	218.33
$\sqrt{v(0)}$	10.4121%	log $v(0)$	-4.524407
κ	1.19048	λ_{J3}	0.00192339
ρ	-0.9040	μ_{J3}	-0.25
μ_{J1}	-0.01	σ_{J3}	0.05
σ_{J1}	0.0735348	μ_{J4}	2.0
μ_{J2}	0.768372	σ_{J4}	0.65
σ_{J2}	0.81901447	ρ_{J34}	-0.90
ρ_J	-0.16328		
$\theta(0)$	-3.63256	$\sqrt{\exp(\theta(0))}$	16.263%
$\bar{\theta}$	-3.79424	$\sqrt{\exp(\bar{\theta})}$	15.00%
κ_θ	0.01		
$\sigma(0)$	4.0495336	$\lambda_J(0)$	0.048114
$\bar{\sigma}$	2.010363	$\bar{\lambda}_J$	0.453853
κ_{σ^2}	5.000307	κ_{λ_J}	8.0

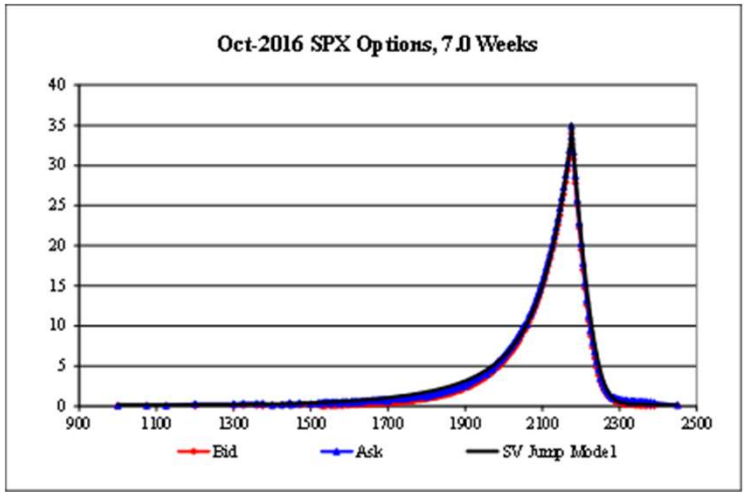
Note: The parameters θ, σ^2, and λ_J are deterministic functions of time that change daily by a model of the following form: $u(t+\Delta t) = \exp(-\kappa\Delta t)(u(t)-\bar{u})+\bar{u}$, where Δt = 1/365.

Model Calibrations – SPX Options

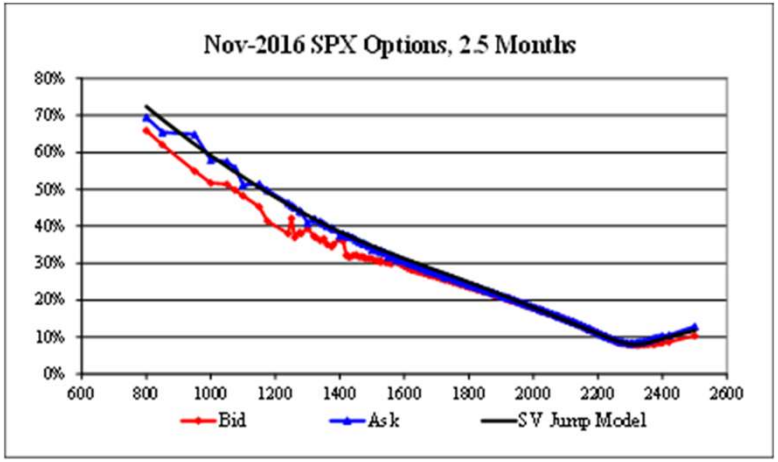


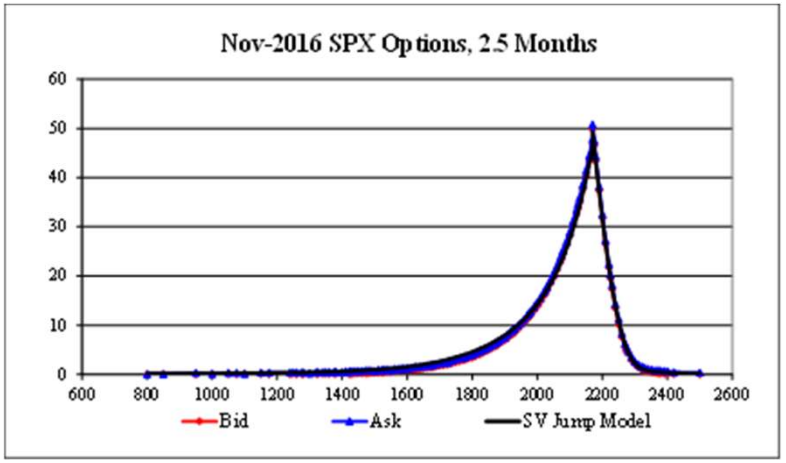


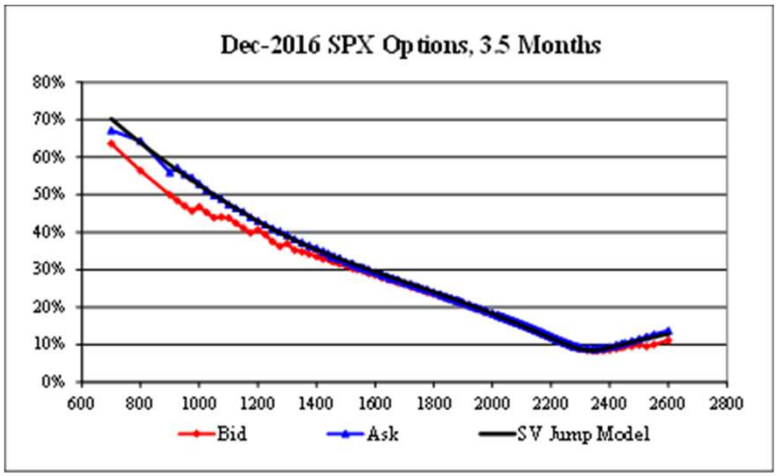


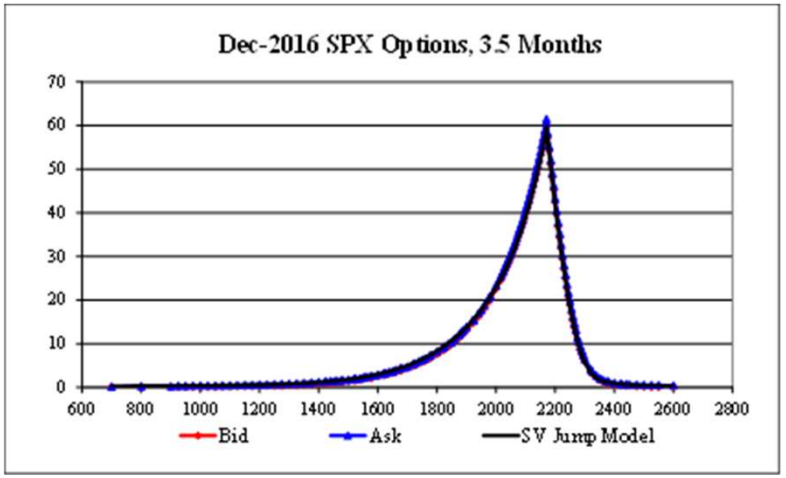


Model Calibrations – SPX Options









Additional graphs for SPX and SPY options at end of presentation

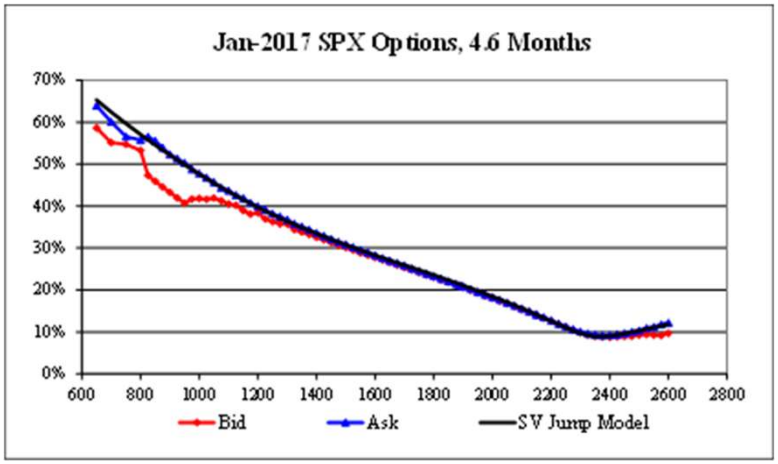
Summary & Observations

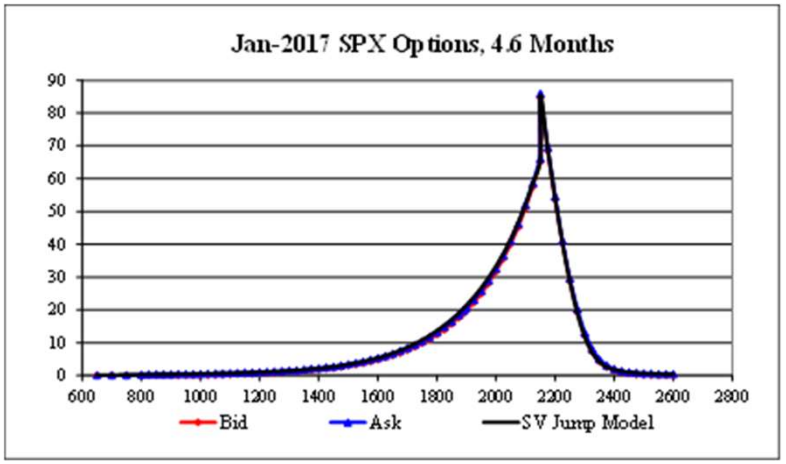
- The computing experiments demonstrate significant reductions in compute time required for both the Monte Carlo solutions and the finite difference solutions. For the StochasticVolatility - Jump model applied to SPX and SPY options, **the speedup factors with the GPU vary between 50 and 1000.** The laptop GPU also produces a significantly faster finite difference solution for American options in the 3 factor Hull-White model.

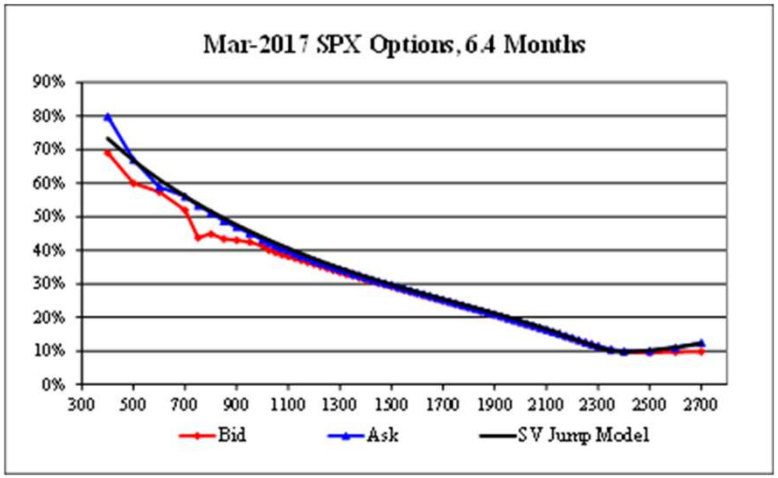
- When jump processes are included in this option pricing model, massive parallel processing on a GPU is necessary to reduce the computing time for practical application. Empirical analysis in finance has shown that more complex models are necessary: (1) multiple factors are necessary for modeling the term structure of interest rates, (2) stochastic volatility and jump processes serve to capture important features of stock price volatility.

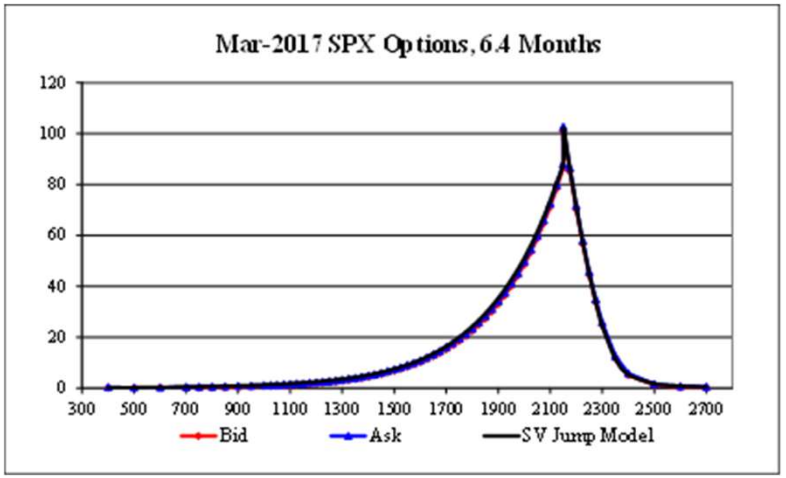
- **Primary Conclusion:** one can use massive parallel processing on GPU's to run more complex financial models that capture the fundamentals, and complexities, observed in financial markets. With massive parallel processing on GPU's, compute time is no longer an obstacle for the implementation of more robust, complex financial models.

Model Calibrations – SPX Options

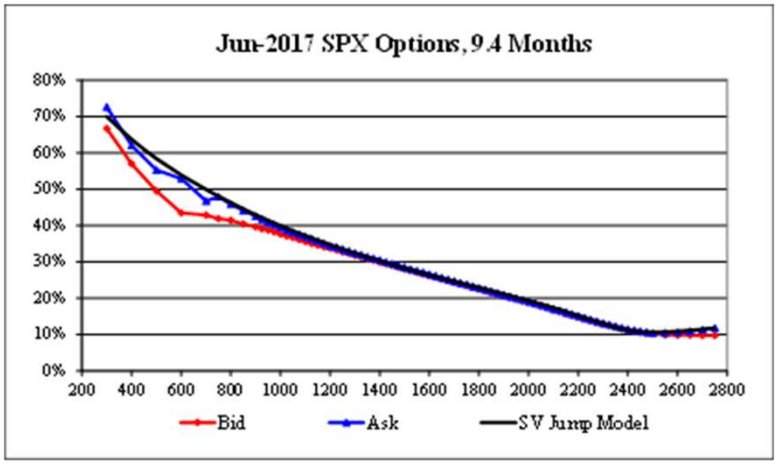


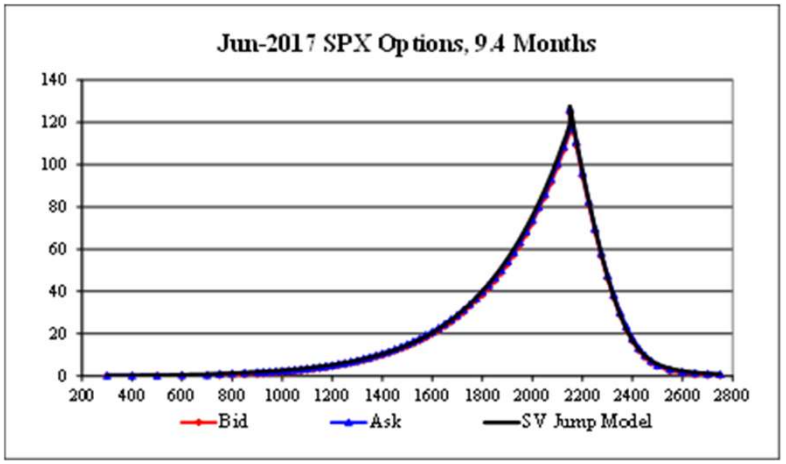


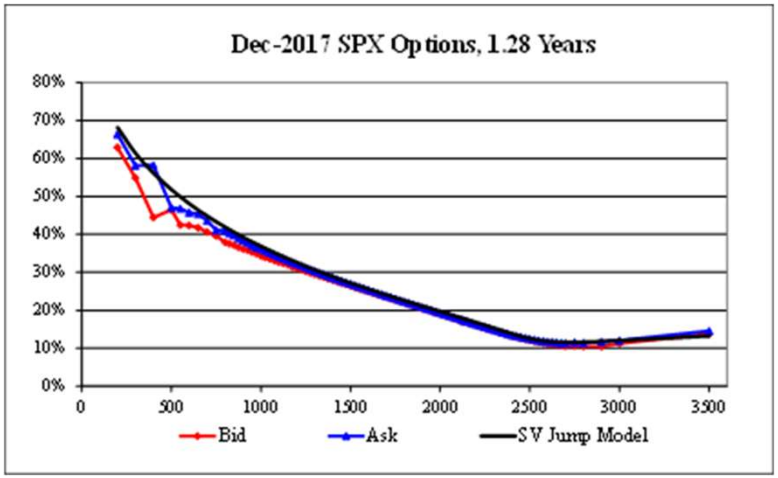


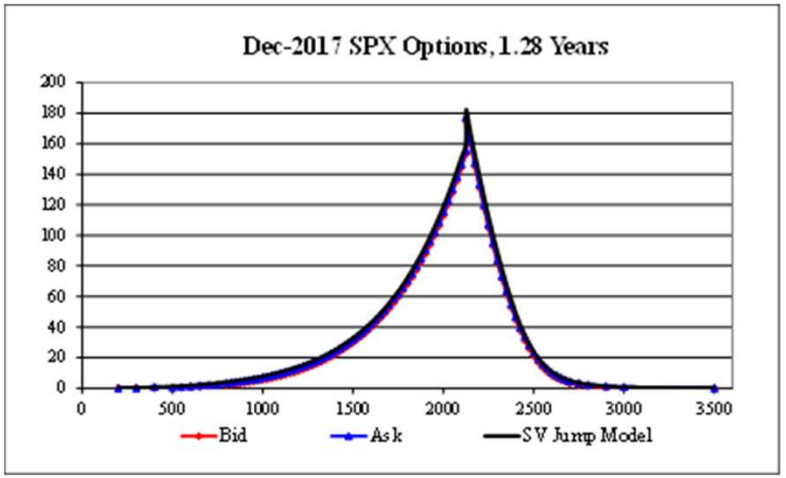


Model Calibrations – SPX Options

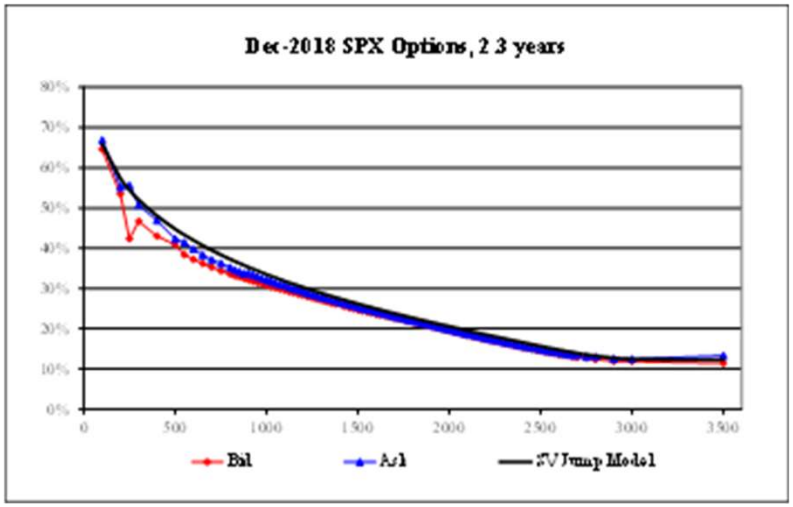


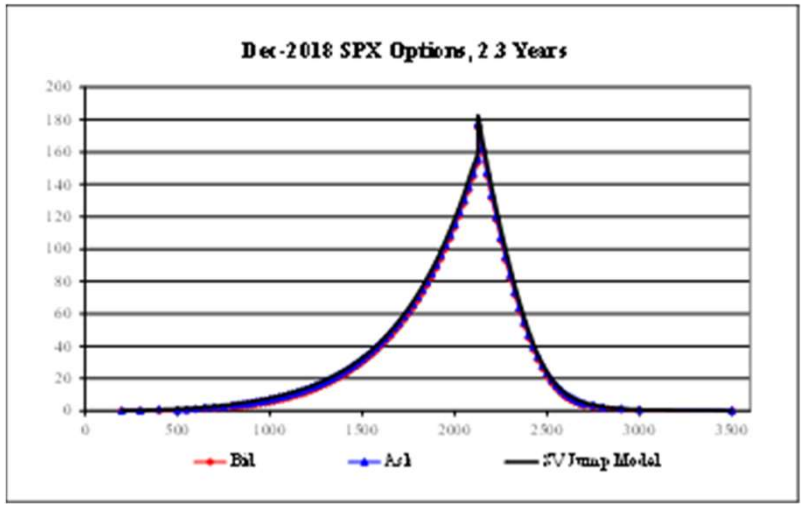




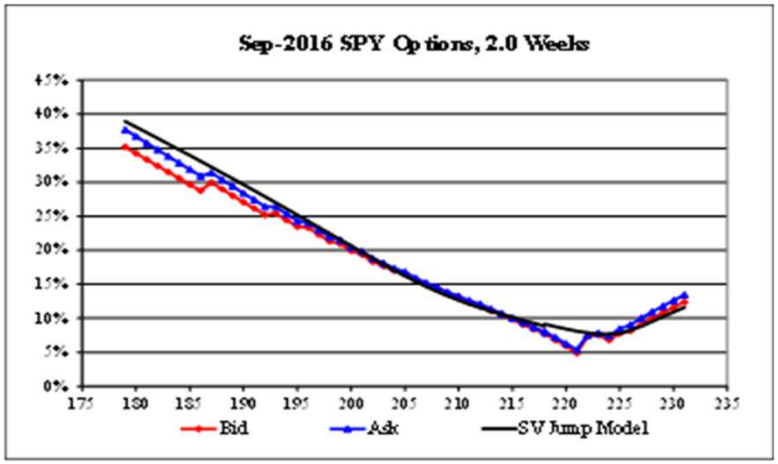


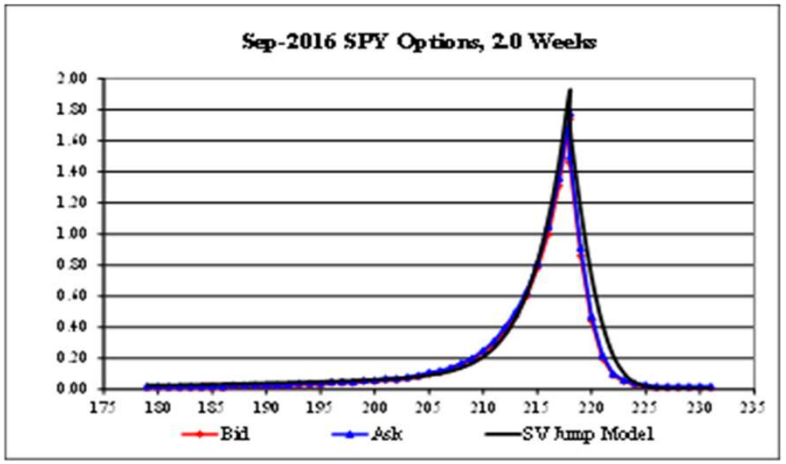
Model Calibrations – SPX Options

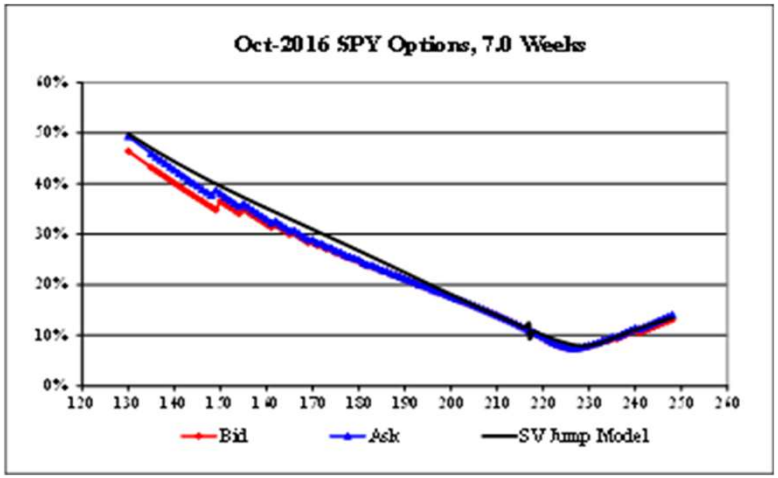


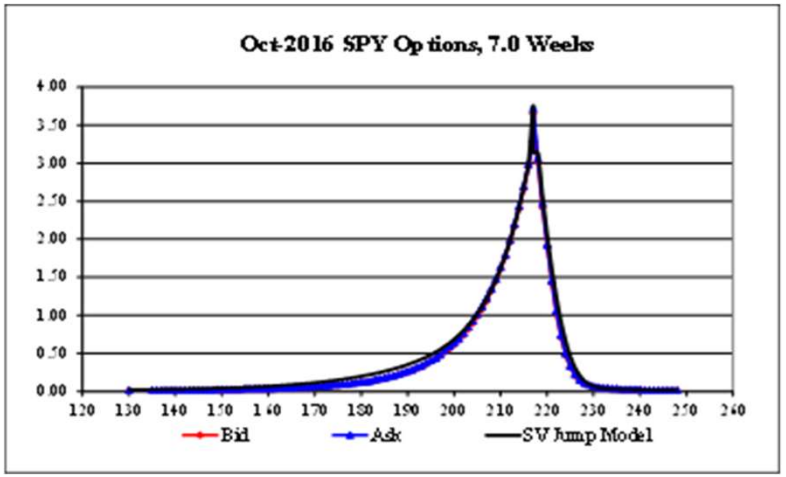


Model Calibrations – SPY Options

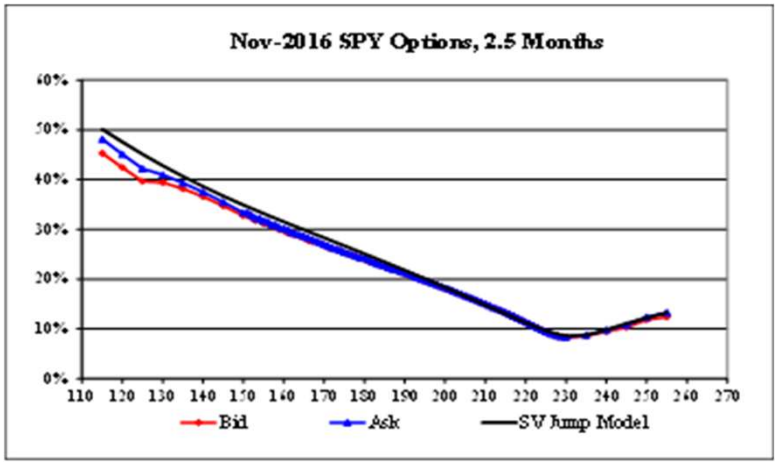


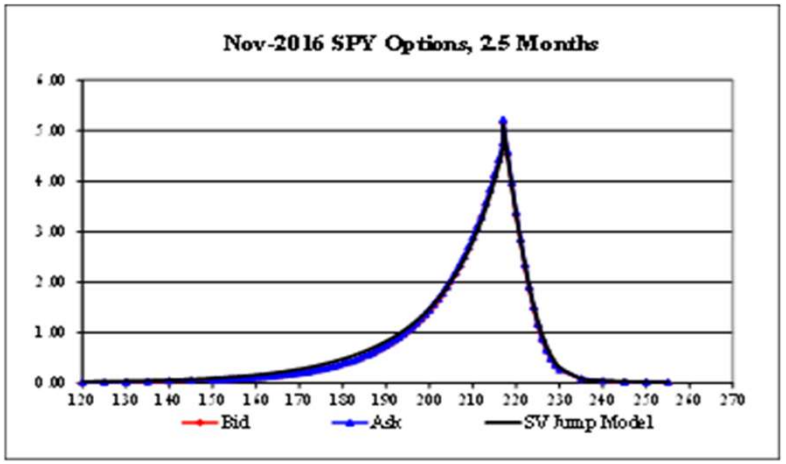


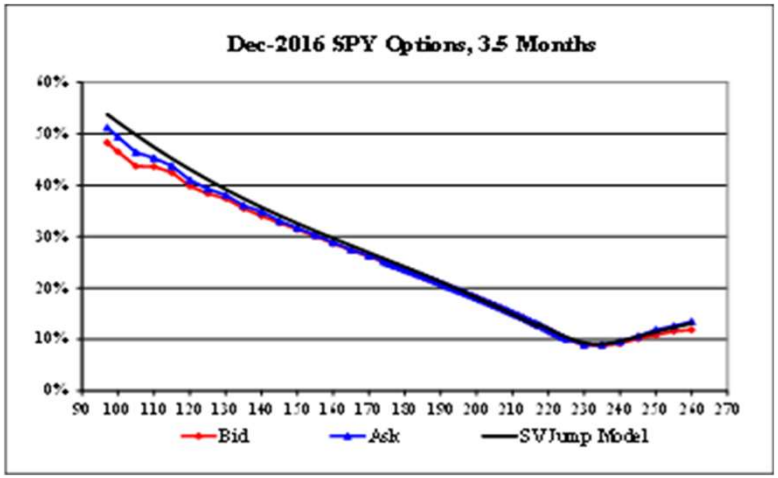


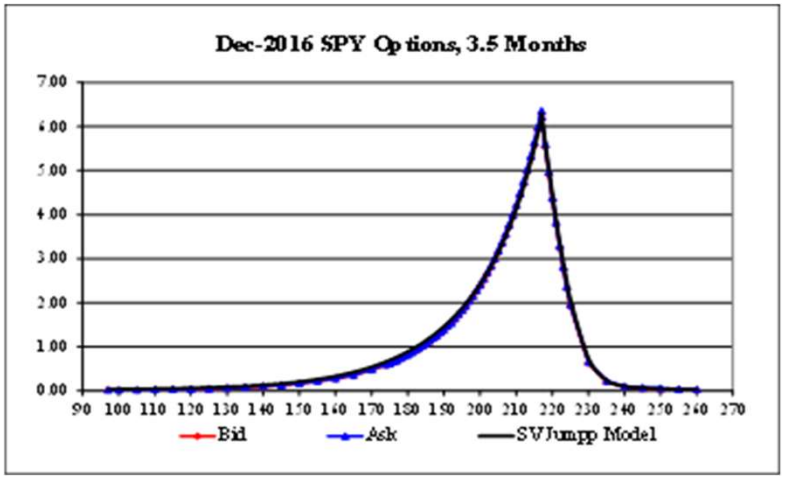


Model Calibrations – SPY Options

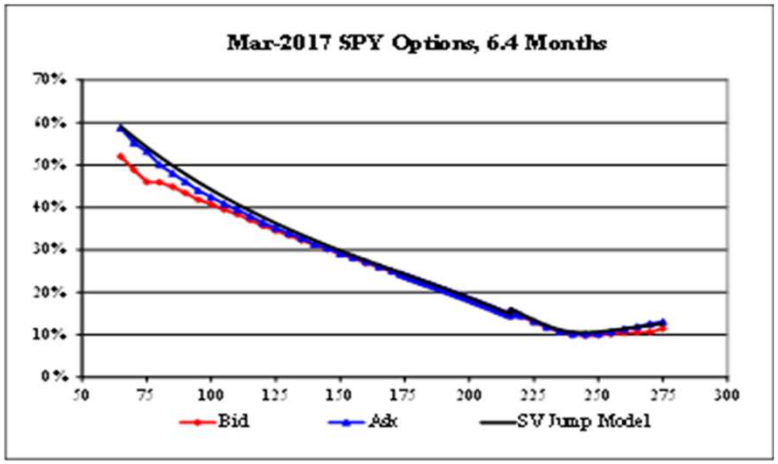


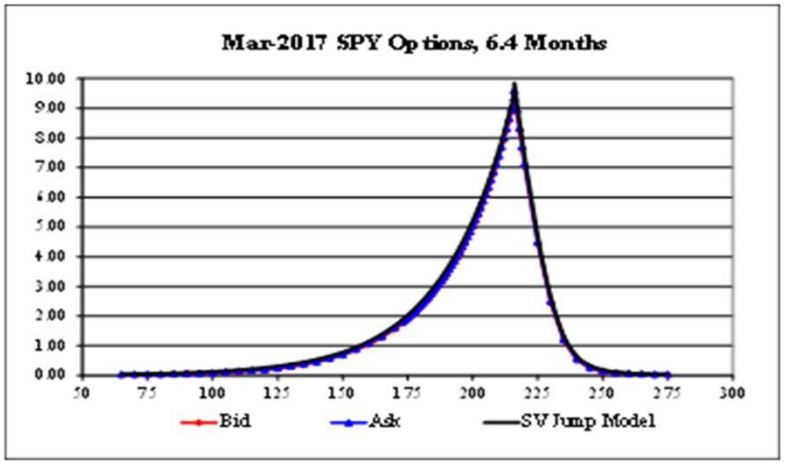


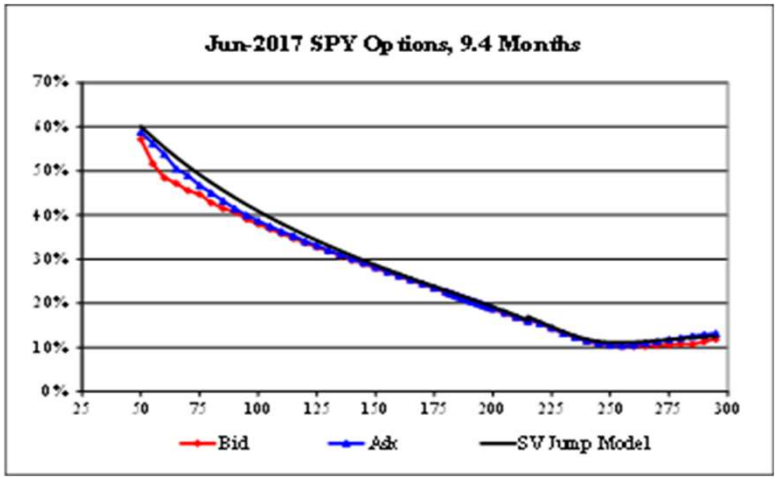


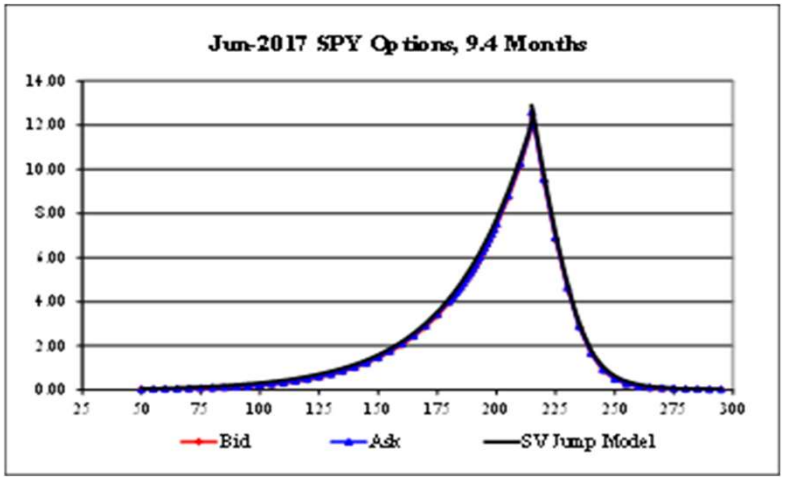


Model Calibrations – SPY Options









References

Scott, Louis (2017), "A Stochastic Volatility Jump Model for Pricing SPX and SPY Options with Massive Parallel Processing," Working Paper, December 2017, available on request. Send requests for paper to LOScott@outlook.com

References for The Econometric Work

Bandi, F.M., and R. Reno (2016), "Price and volatility co-jumps," *Journal of Financial Economics* 119, 107-146.

Christoffersen, P., K. Jacobs, and K. Mimouni (2010). Volatility dynamics for the S&P500: Evidence from realized volatility, daily returns, and option prices. *Review of Financial Studies* 23, 3141–3189.

Christoffersen, Peter, Kris Jacobs, and Chayawat Ornthanalai (2012), "Dynamic jump intensities and risk premiums: Evidence from S&P500 returns and options," *Journal of Financial Economics* 106, 447-472.

Cont, Rama (2001), "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative Finance* 1, 223-236.

Duan Jin-Chuan, and Chung-Ying Yeh, "Price and Volatility Dynamics Implied by the VIX Term Structure," June 2012.

References

Durham, Garland B. (2013), "Risk-neutral Modeling with Affine and Nonaffine Models," *Journal of Financial Econometrics*, Vol. 11, No. 4, 650-681.

Durham, G. and Y.-H. Park (2013). Beyond stochastic volatility and jumps in returns and volatility. *Journal of Business & Economic Statistics* 31, 107–121.

Durham, G., J. Geweke, and P. Ghosh (2015), "A Comment on Christoffersen, Jacobs, and Ornthanalai, Dynamic jump intensities and risk premiums: Evidence from S&P 500 returns and options," *Journal of Financial Economics* 115, 210-214.

Eraker, B. (2004). Do stock prices and volatility jump? Reconciling evidence from spot and option prices. *Journal of Finance* 59, 1367–1403.

Eraker, B., M. Johannes, and N. Polson (2003), "The Impact of Jumps in Volatility and Returns," *Journal of Finance* 58, 1269-1300.

Kaeck, A. and C. Alexander (2012). Volatility dynamics for the S&P 500: Further evidence from non-affine, multi-factor jump diffusions. *Journal of Banking and Finance* 36, 3110–3121.

Wu, Liuren (2011), "Variance Dynamics: Joint Evidence from Options and High-Frequency Returns," *Journal of Econometrics* 160, 280-287.

References

Additional Math Finance References

Bates, D.S. (1996), "Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options," *Review of Financial Studies* 9, 69-107.

Black, F., and M. Scholes (1973), "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy* 81, 637-659.

Chicago Board of Options Exchange, "The CBOE Volatility Index – VIX," 2009. The VIX Whitepaper available at https://www.cboe.com/micro/vix/vixwhite.pdf

Cox, J., J. Ingersoll, and S. Ross, 1985a, An Intertemporal General Equilibrium Model of Asset Prices, *Econometrica* 53, 363-384.

D'Halluin, Y., P.A. Forsyth, and K.R. Vetzal (2005), "Robust Numerical Methods for Contingent Claims under Jump-Diffusion Processes," *IMA Journal of Numerical Analysis* 25, 87-112.

Dupire, B. (1994), "Pricing with a Smile," *Risk* 7 (January), 18-20.

Duffie, Darrell. (2001), *Dynamic Asset Pricing Theory*, 3rd ed., Princeton: Princeton University Press.

Duffie, D., J. Pan, and K. Singleton (2000), "Transform Analysis and Asset Pricing for Affine Jump-Diffusions," *Econometrica* 68, 1343-1376.

References

Heston, S.L. (1993), "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options," *Review of Financial Studies* 6, 327-343.

Hull, John C. (2012), *Options, Futures, and Other Derivatives*, 8th ed., Boston: Prentice Hall.

Merton, R.C. (1976), "Option Pricing When Underlying Stock Returns are Discontinuous," *Journal of Financial Economics* 3, 125-144.

Windcliff, H., P.A. Forsyth and K.R. Vetal (2006), "Pricing Methods and Hedging Strategies for Volatility Derivatives," *Journal of Banking and Finance* 30, 409-431.