

Modeling Asset Processes

Introduction

Over the last twenty five years significant advances have been made in the theory of asset processes and there now exist a variety of mathematical models, many of them computationally tractable, that provide a reasonable representation of their defining characteristics.

While the Geometric Brownian Motion model remains a staple of stochastic calculus theory, it is no longer the only game in town. Other models, many more sophisticated, have been developed to address the shortcomings in the original. There now exist models that provide a good explanation of some of the key characteristics of asset processes that lie beyond the scope of models couched in a simple Gaussian framework. Features such as mean reversion, long memory, stochastic volatility, jumps and heavy tails are now readily handled by these more advanced tools.

In this post I review a critical selection of asset process models that belong in every financial engineer's toolbox, point out their key features and limitations and give examples of some of their applications.

I. Geometric Brownian Motion

The starting point for this examination of stochastic processes is the ubiquitous GBM process, which is a continuous-time and continuous-state random process.

The state $x(t)$ of a geometric Brownian motion satisfies an Ito differential equation

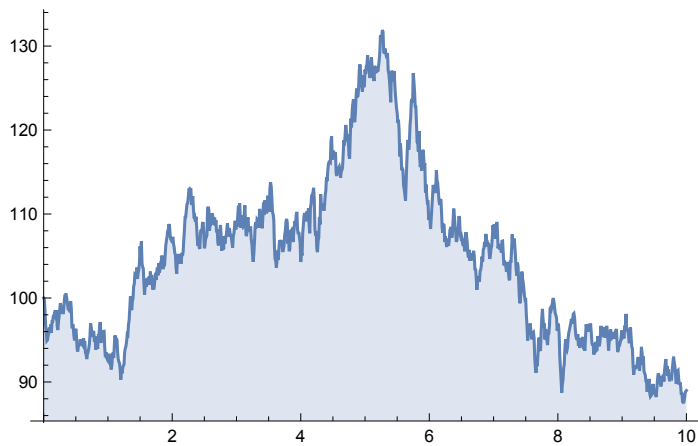
$dx(t) = \mu x(t) dt + \sigma x(t) d\omega(t)$, where $\omega(t)$ follows a standard Wiener process.

We begin by simulating a GBM process with zero drift and volatility of 10%:

```
data = RandomFunction[GeometricBrownianMotionProcess[0, .1, 100], {0, 10, .01}]
```

TemporalData[

```
ListLinePlot[data, Filling -> Axis]
```



Moments of a GBM Process

The mean of a GBM process is give by:

```
Mean[GeometricBrownianMotionProcess[μ, σ, x₀][t]] // Simplify
```

$$e^{t\mu} x_0$$

And its variance is given by:

```
Variance[GeometricBrownianMotionProcess[μ, σ, x₀][t]] // Simplify
```

$$e^{2t\mu} \left(-1 + e^{t\sigma^2} \right) x_0^2$$

Note that a GBM process is not weakly stationary:

```
WeakStationarity[GeometricBrownianMotionProcess[μ, σ, x₀]]
```

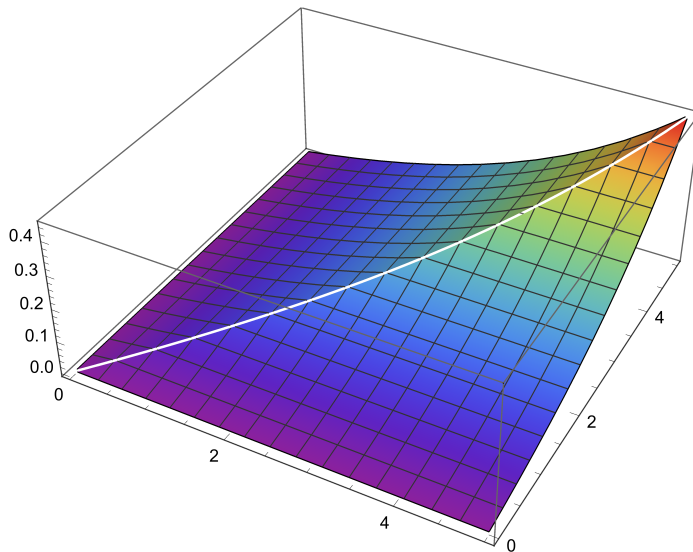
False

The process covariance is time-dependent:

```
CovarianceFunction[GeometricBrownianMotionProcess[μ, σ, x₀], s, t]
```

$$e^{(s+t)\mu} \left(-1 + e^{\sigma^2 \text{Min}[s,t]} \right) x_0^2$$


```
Plot3D[CovarianceFunction[GeometricBrownianMotionProcess[0, .6, .3], s, t],
{s, 0, 5}, {t, 0, 5}, ColorFunction -> "Rainbow"]
```



Modeling a Stock Process

Let's apply the GBM model concept for a specific stock, such as Apple. We'll use weekly closing prices since mid-2016, to capture the recent trend in the stock.

```
data = FinancialData["AAPL", "Close", {{2016, 7, 1}, {2017, 2, 8}, "Week"}];
tsAAPL = TimeSeries[data[[All, 2]], {data[[1, 1]], Automatic, "Week"}]
```

TimeSeries [ Time: 01 Jul 2016 to 10 Feb 2017
Data points: 33]

```
DateListPlot[tsAAPL, Filling -> Axis, Joined -> True]
```



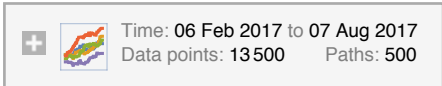
Estimating a GBM Process

We estimate a GBM process for AAPL stock, as follows:

```
Clear[ $\mu$ ,  $\sigma$ , x];
GBMproc =
  EstimatedProcess[tsAAPL["Values"], GeometricBrownianMotionProcess[ $\mu$ ,  $\sigma$ , x]]
GeometricBrownianMotionProcess[0.0107502, 0.0291338, 94.4766]
```

Next, we will simulate future paths for AAPL for the next half - year,
based on the estimated GBM process :

```
paths = RandomFunction[GBMproc,
  {tsAAPL["PathLength"], tsAAPL["PathLength"] + 26, 1}, 500];
td = TemporalData[paths["ValueList"], {data[[-1, 1]], Automatic, "Week"},
  ValueDimensions -> 1]
```

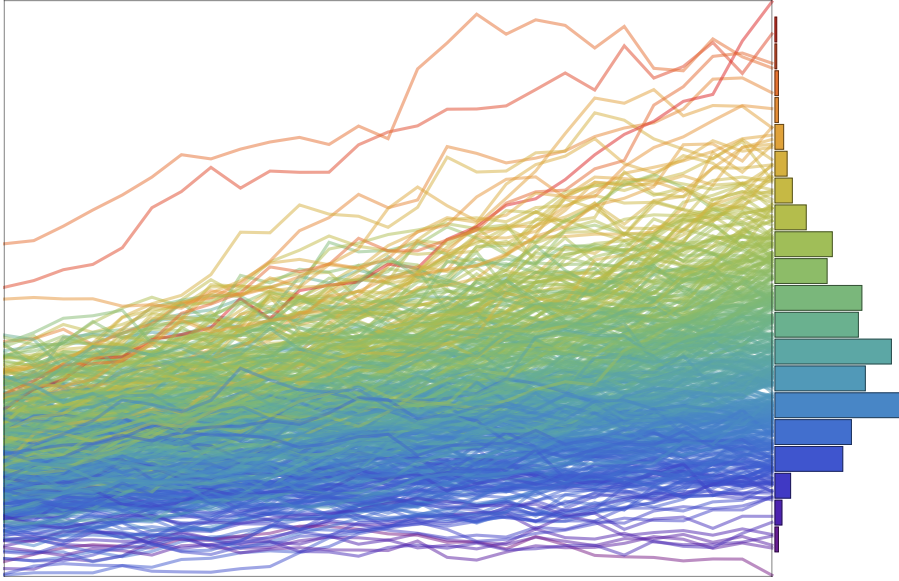
```
TemporalData[
```

Path Simulation for a GBM Process

We now calculate simulated stock price paths and chart our projections for the stock over the coming half-year. Notice the Log-Normal shape of the histogram of terminal stock prices.

```
sd = paths["SliceData", tsAAPL["PathLength"] + 26];
cf = ColorData["Rainbow"];
sliced =
  BarChart[Last[#], Axes -> None, BarOrigin -> Left, AspectRatio -> 4, ChartStyle ->
    {cf /@ Rescale[MovingAverage[First[#], 2], {Min[sd], Max[sd]}, {0, 1}}],
    ImageSize -> 74] & [HistogramList[sd,
    {Range[Min[sd], Max[sd], (Max[sd] - Min[sd]) / 20]}]];
chartpaths = DateListPlot[td, ImageSize -> 400, PlotRange -> All,
  AspectRatio -> 3/4, Epilog -> Inset[sliced, {3713212800, 0}],
  PlotStyle -> {cf /@ Rescale[sd]}, BaseStyle -> Directive[Thin, Opacity[0.5]],
  PlotRangePadding -> {{0, .25}, {.5, .5}}];
```

```
Row[{chartpaths, sliced}]
```



II. Heston Model

We can generalize the basic GBM model in a number of ways. One of the best-known is the Heston (1993) model, which introduces stochastic volatility in the form of a secondary, correlated random process.

We define an Ito process corresponding to a correlated 2D Wiener process:

$$\mathbf{cW}(\rho_-) := \text{ItoProcess}\left[\{\{0, 0\}, \text{IdentityMatrix}[2]\}, \begin{pmatrix} w_1 & w_2 \\ 0 & 0 \end{pmatrix}, t, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right];$$

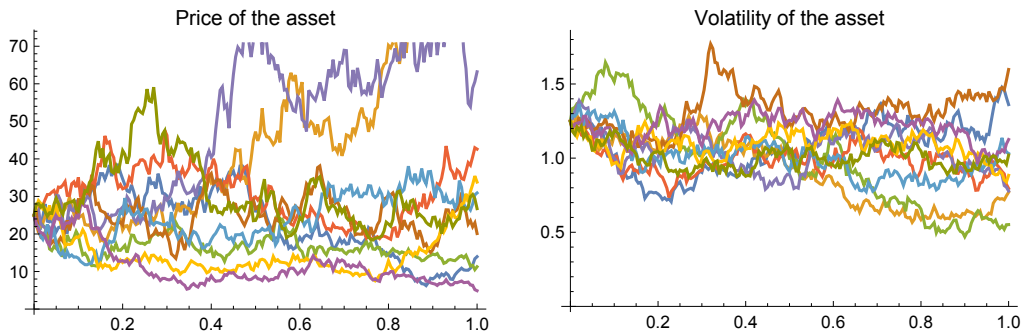
We then define the Heston Model using a system of stochastic differential equations driven by the 2D Wiener process, the first relating to the stock price, the second to the process volatility, which are correlated with parameter ρ . In the Heston model we assume that volatility reverts to a long term mean value θ , with speed of mean-reversion K , in a similar way to an Ornstein-Uhlenbeck process, or Cox-Ingersoll-Ross model.

$$\mathbf{hm} = \text{ItoProcess}\left[\left\{ds(t) = \sqrt{r(t)} s(t) dw_s(t) + \mu s(t) dt, dr(t) = \kappa dt (\theta - r(t)) + \xi \sqrt{r(t)} dw_v(t)\right\}, \{s(t), r(t)\}, \begin{pmatrix} s & r \\ s_0 & r_0 \end{pmatrix}, t, \{w_s, w_v\} \approx \mathbf{cW}(\rho)\right];$$

We can then simulate the model using a stochastic Runge-Kutta scheme:

$$\begin{aligned} \mathbf{td} &= \text{BlockRandom}\left[\text{SeedRandom}[1988]; \right. \\ &\quad \text{RandomFunction}\left[\mathbf{hm} /. \left\{\mu \rightarrow 0, \kappa \rightarrow 2, \theta \rightarrow 1, \xi \rightarrow \frac{1}{2}, \rho \rightarrow -\frac{1}{3}, s_0 \rightarrow 25, r_0 \rightarrow 1.25\right\}, \right. \\ &\quad \left. \left\{0, 1, 0.005\right\}, 10, \text{Method} \rightarrow \text{"StochasticRungeKutta"}\right]\left. \right]; \end{aligned}$$

```
Row[{ListLinePlot[td["PathComponent", 1], PlotLabel → "Price of the asset",
  ImageSize → 250], ListLinePlot[td["PathComponent", 2],
  PlotLabel → "Volatility of the asset", ImageSize → 250]}, Spacer[20]]
```



Slava Solganik has contributed some code in the Wolfram Demonstrations Project, which I shall reproduce here, that creates an interactive model that the user can to display the volatility surface in the Heston model, varying the model parameters dynamically to examine the sensitivity of the surface to changes in volatility, mean reversion and volatility of volatility:

```
Hhat[k_, V_, τ_, θ_, ξ_, ω_, ρ_] :=
Module[{f1, f2, d, g, h, ttheta, t, tomega, tc}, c =  $\frac{1}{2} (k^2 - \mathfrak{i} k)$ ;

t =  $\frac{1}{2} \xi^2 \tau$ ;
tomega =  $\frac{2}{\xi^2} \omega$ ;
tc =  $\frac{2}{\xi^2} c$ ;
ttheta =  $\frac{2}{\xi^2} (\mathfrak{i} k \rho \xi + \theta)$ ;
d =  $\sqrt{ttheta^2 + 4 tc}$ ;
g =  $\frac{1}{2} (ttheta + d)$ ;
h =  $\frac{ttheta + d}{ttheta - d}$ ;
f1 = tomega  $\left( t g - \text{Log} \left[ \frac{1 - h \text{Exp}[d t]}{1 - h} \right] \right)$ ; f2 = g  $\left( \frac{1 - \text{Exp}[d t]}{1 - h \text{Exp}[d t]} \right)$ ;
Exp[f1 + f2 V]]
kim = 0.6; Δ = 125;
HestonCallLewis[S_, K_, τ_, r_, d_, V0_, MR_, ρ_, Vinf_, VolVol_] :=
Module[{Integral, X, θ, ξ, ω, V, k}, X =  $\text{Log} \left[ \frac{S}{K} \right] + (r - d) \tau$ ;

k = kre +  $\mathfrak{i} kim$ ;
Integral =  $\frac{1}{\pi}$ 
NIntegrate[Re[Exp[- $\mathfrak{i} k X$ ]]  $\left( \frac{\text{Hhat}[k, V0, \tau, MR, \text{VolVol}, Vinf MR, \rho]}{k^2 - \mathfrak{i} k} \right)$ ], {kre, θ, Δ},
```

```

Method → {"LobattoKronrodRule", "Points" → 25}, PrecisionGoal → 4];
S Exp[-d τ] - K Exp[-r τ] Integral]

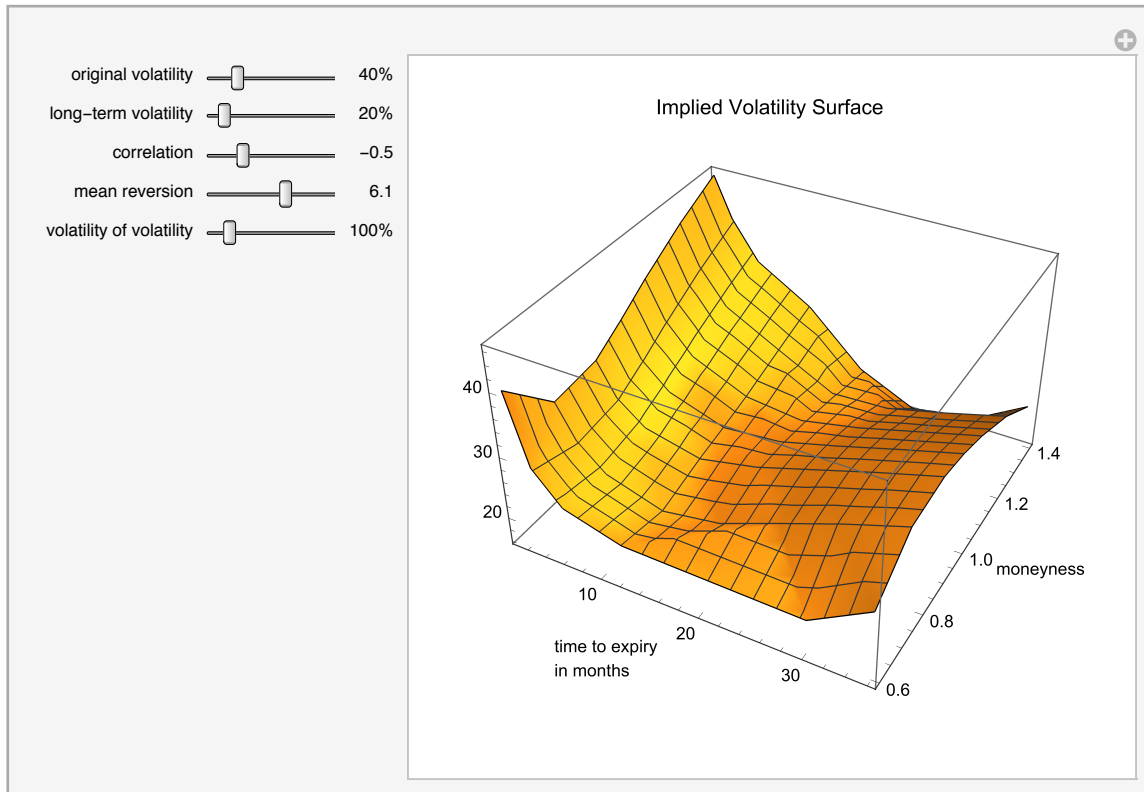
Moneyness = {.6, .75, .9, 1, 1.1, 1.25, 1.4};
Tenors = {1, 3, 6, 12, 18, 24, 30, 36};
m = Length[Moneyness]; t = Length[Tenors]; size = t m;
S = 100; r = 0.03; d = 0.;
Ncdf = Compile[{{x, _Real}}, (Erf[x/√2] + 1)/2];
BSCall =
  Compile[{{S, _Real}, {K, _Real}, {σ, _Real}, {T, _Real}, {r, _Real}, {d, _Real}},
    Module[{d1, d2}, d1 = 
$$\frac{\text{Log}\left[\frac{S}{K}\right] + (r - d) T + \sigma^2 T / 2}{\sigma \sqrt{T}}$$
;
      d2 = d1 - σ √T;
      S e-d T Ncdf[d1] - K e-r T Ncdf[d2]]];
ImpliedVolCall = Compile[{{p, _Real}, {S, _Real},
  {K, _Real}, {T, _Real}, {r, _Real}, {d, _Real}, {precision, _Real}},
  Module[{Vol0, Vol1, Vol2, Price0, Price1, Price2}, Vol2 = 0.8;
    Vol0 = 0.;
    Price0 = S Exp[-d T] - K Exp[-r T];
    If[Price0 < 0, Price0 = 0.];
    If[p < Price0, 0.000001, Vol1 = Vol2;
      Price2 = BSCall[S, K, Vol1, T, r, d];
      While[Price2 < p, Price2 = BSCall[S, K, Vol1, T, r, d];
        Vol1 = 2 Vol1];
      Vol2 = Vol1; Price1 = BSCall[S, K, Vol1, T, r, d];
      While[Abs[Price1 - p] > precision, Vol1 = (Vol0 + Vol2)/2.;
        Price1 = BSCall[S, K, Vol1, T, r, d];
        If[Price1 < p, Vol0 = Vol1, Vol2 = Vol1];];
      Vol1]]];

```

```

Manipulate[
  Quiet@Module[{i, j, x, k, T, Price, pr, Volatility, vol},
    Price = Array[pr, {m, t}];
    Volatility = Array[vol, size];
    For[i = 1, i < t + 1, i++, T = Tenors[[i]] / 12;
      For[j = 1, j < m + 1, j++, k = S / Moneyness[[j]];
        pr[i, j] = HestonCallLewis[S, k,
          T, r, d, ( $\sigma_0$  / 100) ^ 2, MR,  $\rho$ , ( $\sigma_{inf}$  / 100) ^ 2, VolVol / 100];
        x = ImpliedVolCall[pr[i, j], S, k, T, r, d, 0.0001];
        If[x  $\neq$  0.000001, vol[(i - 1) m + j] = {Tenors[[i]], Moneyness[[j]], 100 x},
          vol[(i - 1) m + j] = {Null, Null, Null}];];];
    ListPlot3D[Volatility, PlotRange  $\rightarrow$  {Full, Full, Full},
      PlotLabel  $\rightarrow$  "Implied Volatility Surface",
      AxesLabel  $\rightarrow$  {"time to expiry\nin months", "moneyness", None},
      ImageSize  $\rightarrow$  {350, 350}, BoxRatios  $\rightarrow$  {1, 1, .6}, ImagePadding  $\rightarrow$  25]],
  Item[
    Grid[{
      {"original volatility", Manipulator[Dynamic[ $\sigma_0$ ], {10, 200, 10}, ImageSize  $\rightarrow$  Tiny],
        Row[{Dynamic[ $\sigma_0$ ], "%"}]}, {"long-term volatility", Manipulator[
          Dynamic[ $\sigma_{inf}$ ], {10, 200, 10}, ImageSize  $\rightarrow$  Tiny], Row[{Dynamic[ $\sigma_{inf}$ ], "%"}]},
      {"correlation", Manipulator[Dynamic[ $\rho$ ], {- .9, .9, .1}, ImageSize  $\rightarrow$  Tiny],
        Row[{Dynamic[ $\rho$ ]}]},
      {"mean reversion", Manipulator[Dynamic[MR], {.1, 10, 0.1},
        ImageSize  $\rightarrow$  Tiny], Dynamic[MR]},
      {"volatility of volatility", Manipulator[Dynamic[VolVol],
        {10, 1000, 10}, ImageSize  $\rightarrow$  Tiny], Row[{Dynamic[VolVol], "%"}]}},
      Alignment  $\rightarrow$  {{Right, Left, Right}}, ItemSize  $\rightarrow$  {{12, 8, 4}, Automatic}]],
    {{ $\sigma_0$ , 40}, 10, 200, 10, ControlType  $\rightarrow$  None},
    {{ $\sigma_{inf}$ , 20}, 10, 200, 10, ControlType  $\rightarrow$  None},
    {{ $\rho$ , - .1}, - .9, .9, .1, ControlType  $\rightarrow$  None},
    {{MR, 1.5}, .1, 10, .1, ControlType  $\rightarrow$  None},
    {{VolVol, 80}, 10, 1000, 10, ControlType  $\rightarrow$  None}, SynchronousUpdating  $\rightarrow$  False,
    SaveDefinitions  $\rightarrow$  True, ControlPlacement  $\rightarrow$  Left, TrackedSymbols  $\rightarrow$  Manipulate]

```

III. Ornstein-Uhlenbeck Process

An Ornstein-Uhlenbeck process, also known as a Vasicek process, is widely used in the modeling of interest rate processes and related derivative assets. It is a continuous state, continuous-time process in which the state $X(t)$ follows an Ito stochastic differential equation of the form:

$dx(t) = \theta(\mu - x(t))dt + \sigma dw(t)$ where $w(t)$ follows a standard Wiener process.

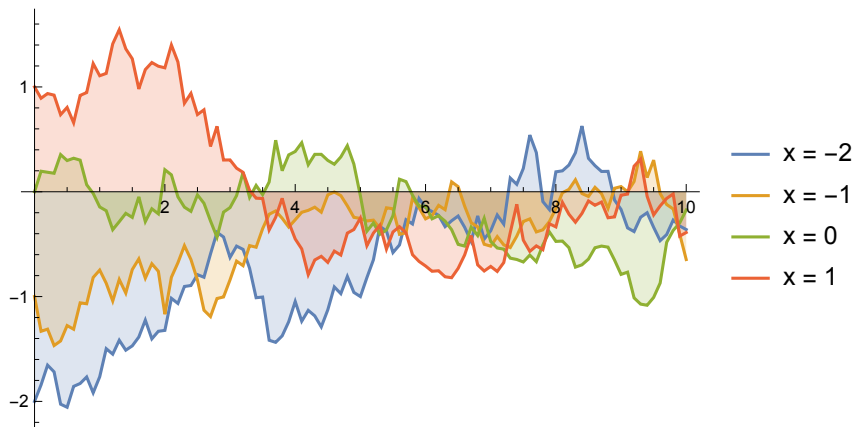
The process with parameters $[\mu, \sigma, \theta, x_0]$ is mean reverting, to long-term mean value μ (in the above formulation,) with rate of mean-reversion θ , volatility σ and starting value x_0 .

Here are a few sample paths from an O-U process with mean zero, with various randomized starting points:

```
points = {-2, -1, 0, 1};
```

```
sample[x_] := RandomFunction[OrnsteinUhlenbeckProcess[0, .5, .3, x], {0, 10, 0.1}];
```

```
ListLinePlot[sample[#] & /@points, PlotRange → All, Filling → Axis,
  PlotLegends → (StringJoin["x = ", ToString[#]] & /@points)]
```



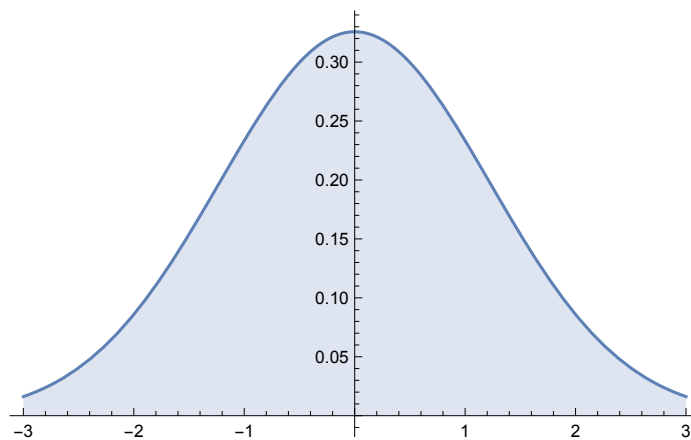
Stationarity of the Ornstein-Uhlenbeck Process

If we take a slice of the process at time t , we find that the process is stationary with a normal distribution that is independent of time:

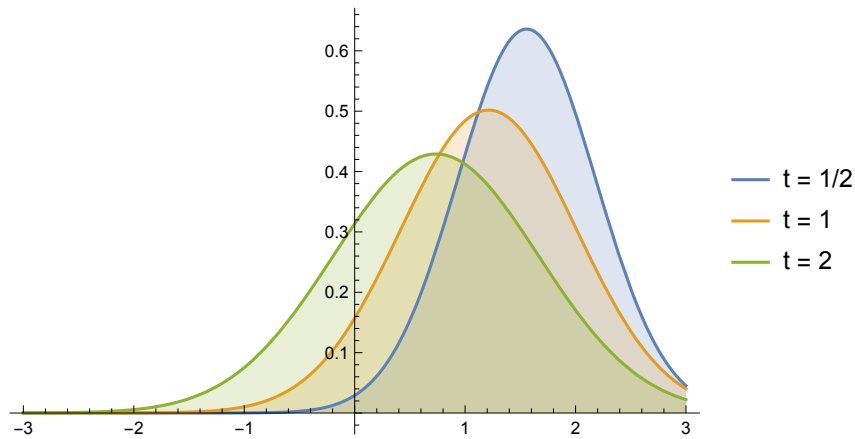
```
StationaryDistribution[OrnsteinUhlenbeckProcess[ $\mu$ ,  $\sigma$ ,  $\theta$ ]]
```

```
NormalDistribution[ $\mu$ ,  $\frac{\sigma}{\sqrt{2} \sqrt{\theta}}$ ]
```

```
Plot[PDF[OrnsteinUhlenbeckProcess[0, 1, 1/3][t], x], {x, -3, 3}, Filling → Axis]
```

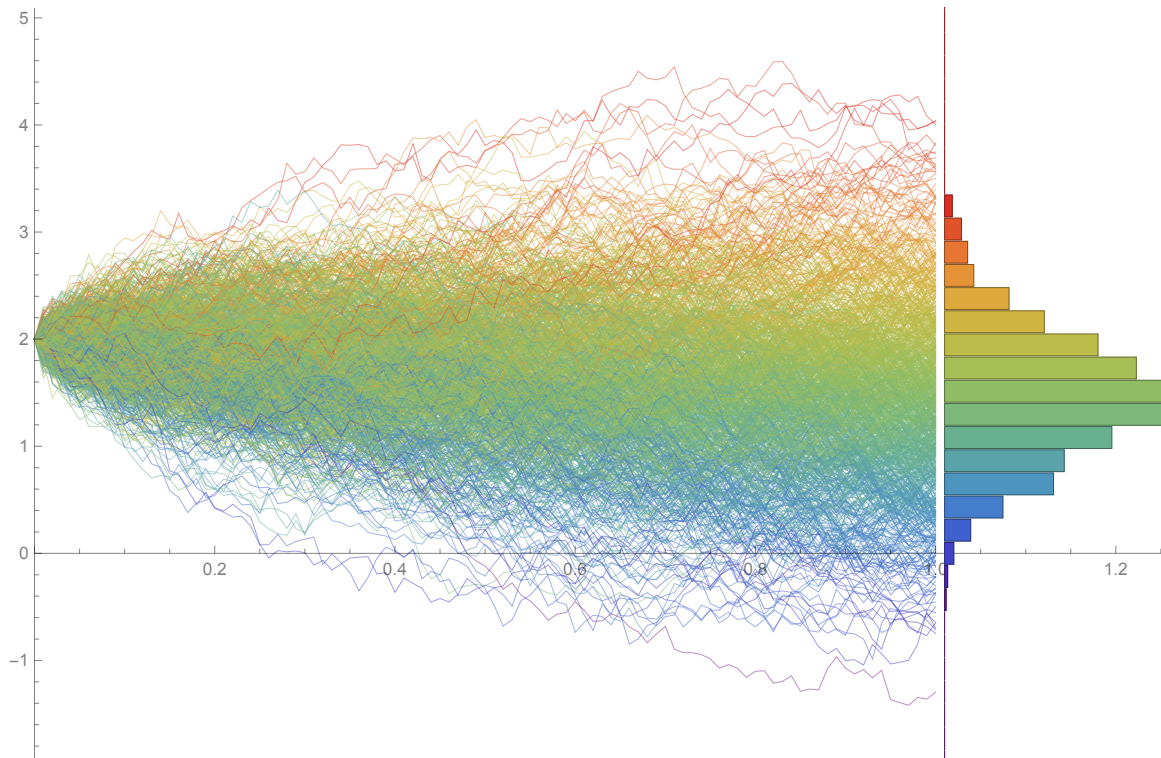


```
Plot[
  Evaluate@Table[PDF[OrnsteinUhlenbeckProcess[0, 1, .5, 2][t], x], {t, {1/2, 1, 2}}],
  {x, -3, 3}, Filling -> Axis, PlotLegends -> {"t = 1/2", "t = 1", "t = 2"}]
```



The normality of the time-sliced distribution can be illustrated using simulated process paths:

```
data = RandomFunction[OrnsteinUhlenbeckProcess[0, 1, 1/3, 2], {0, 1, .01}, 1000];
sd = data["SliceData", 1];
cf = ColorData["Rainbow"];
sliced =
  BarChart[Last[#], Axes -> False, BarOrigin -> Left, AspectRatio -> 4, ChartStyle ->
    (cf /@ Rescale[MovingAverage[First[#], 2], {Min[sd], Max[sd]}, {0, 1}]),
    ImageSize -> 128] & [HistogramList[sd, {Range[-4, 8, .3]}]];
ListLinePlot[data, ImageSize -> 600, PlotRange -> All, AspectRatio -> 2/3,
  Epilog -> Inset[sliced, {1.01, 0}, {0, 12}], PlotStyle -> (cf /@ Rescale[sd]),
  BaseStyle -> Directive[Thin, Opacity[0.5]], PlotRangePadding -> {{0, .25}, {.5, .5}}]
```



Moments of the Ornstein-Uhlenbeck Process

For an O-U process with zero as the fixed initial condition, both the mean and variance are constant:

$$\text{Mean}[\text{OrnsteinUhlenbeckProcess}[\mu, \sigma, \theta][t]]$$

$$\mu$$

$$\text{Variance}[\text{OrnsteinUhlenbeckProcess}[\mu, \sigma, \theta][t]]$$

$$\frac{\sigma^2}{2\theta}$$

But for an O-U process with fixed initial state x , the mean and variance are time-dependent:

$$\text{Mean}[\text{OrnsteinUhlenbeckProcess}[\mu, \sigma, \theta, x][t]]$$

$$e^{-t\theta}(x - \mu) + \mu$$

$$\text{Variance}[\text{OrnsteinUhlenbeckProcess}[\mu, \sigma, \theta, x][t]]$$

$$\frac{(1 - e^{-2t\theta})\sigma^2}{2\theta}$$

The skewness and kurtosis are constant, however, regardless of the starting point of the process:

$$\text{Skewness}[\text{OrnsteinUhlenbeckProcess}[\mu, \sigma, \theta][t]]$$

$$0$$

```
Skewness[OrnsteinUhlenbeckProcess[μ, σ, θ, Subscript[x, 0]][t]]
0

Kurtosis[OrnsteinUhlenbeckProcess[μ, σ, θ][t]]
3

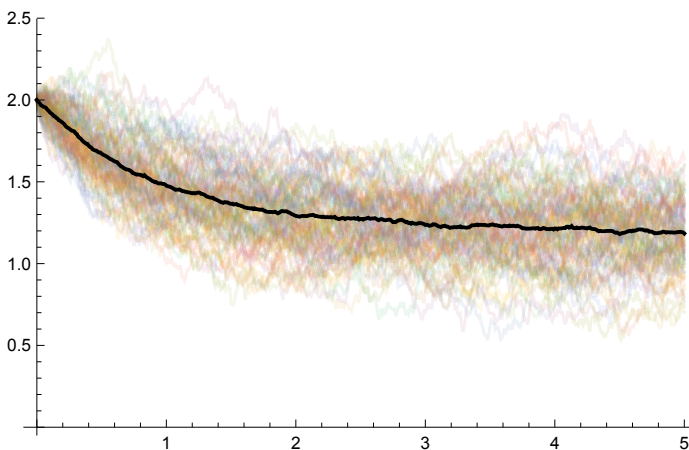
Kurtosis[OrnsteinUhlenbeckProcess[μ, σ, θ, Subscript[x, 0]][t]]
3
```

Mean Reversion in the Ornstein-Uhlenbeck Process

The inherent tendency of an O-U process to long term mean reversion explains its popularity in modeling interest rate and volatility processes. The following example demonstrates the tendency of the process towards the long-term mean (of 1.2, in this example), with speed governed by the parameter $\theta=1$:

```
μ = 1.2; σ = 0.3; θ = 1; T = 5;
noisysol = RandomFunction[OrnsteinUhlenbeckProcess[μ, σ, θ, 2], {0, T, 0.01}, 10^2];
meanFunction = TimeSeriesThread[Mean, noisysol];

Show[ListLinePlot[noisysol, PlotStyle → Directive[Opacity[.1]]],
ListLinePlot[meanFunction, PlotStyle → Directive[Black, Thick]]]
```



IV. Fractal Brownian Motion

The fractal Brownian motion process is another continuous-time, continuous state model that generalizes the standard GBM process by introducing a parameter h , known as the Hurst exponent, that determines how the volatility of the process scales with time. It is a Gaussian process with mean μt and covariance $\sigma^2 (s^{2h} + t^{2h} - |t-s|^{2h})/2$. For $h = \frac{1}{2}$ it reduces to a standard Wiener process.

The time-slice properties of the process show how the Hurst exponent comes into play:

```
Clear[μ, σ, θ, T];
PDF[FractionalBrownianMotionProcess[μ, σ, h][t], x]

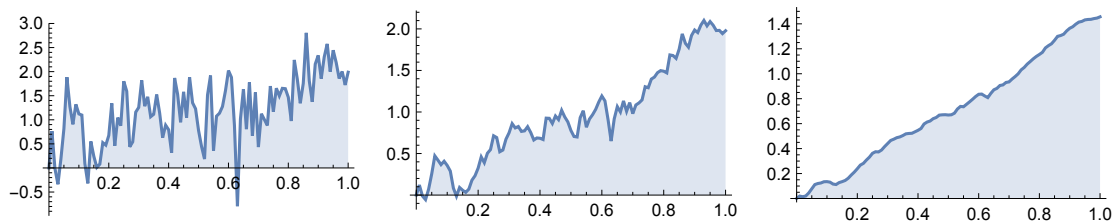
$$\frac{e^{-\frac{t^{-2h}(x-t\mu)^2}{2\sigma^2}} t^{-h}}{\sqrt{2\pi}\sigma}$$

```

The effect of the Hurst exponent parameter is perhaps best understood by means of an illustrative example. Below we have charted the evolution of three otherwise identical processes that differ only in the value of the Hurst parameter, which ranges from 0.1 to 0.5 to 0.9.

```
sample[h_] := (SeedRandom[3];
  RandomFunction[FractionalBrownianMotionProcess[h], {0, 1, 0.01}])

GraphicsRow[ListLinePlot[#, Filling -> Axis] & /@
  {sample[.1], sample[.5], sample[.9]}, ImageSize -> Large]
```



The middle process, with $h=0.5$, illustrates a benchmark GBM process. The first chart shows an anti-persistent process, with $h=0.1$, in which volatility scales at much less than the square root of time, while the third chart shows a persistent process with $h=0.9$ that has a tendency to exhibit trending behavior, a feature of processes known as “long memory” processes. A long memory process is one in which serial autocorrelations die away very slowly, leading to a tendency for long term trends to develop. It has been quite widely adopted for modeling volatility processes, and with some success.

For those interested in learning more about long memory processes, see my post on Long memory and Regime Shifts in Asset Volatility (<http://jonathankinlay.com/2011/03/long-memory-and-regime-shifts-in-asset-volatility/>).

V. Jump Diffusion Processes

The standard GBM process assumes that the price process follows a LogNormal distribution, or equivalently, that returns are normally distributed. However, it is widely recognized that empirical distributions exhibit heavier tails than seen in a Gaussian framework. In other words, empirical processes are characterized by a higher frequency of very large moves in either direction than would be expected if returns were normally distributed.

We have already described two approaches to generalizing the standard GBM model that go some way to addressing its shortcomings. The first, the Heston model, incorporates a correlated, stochastic volatility process that will produce greater weight in the tails of the returns distribution and, as a consequence, volatility smiles in the option implied volatility curves.

A second approach, the fraction Brownian motion process, can also produce a fat-tailed distribution in the returns process, but is used primarily to model the phenomenon known as “long memory”.

We turn next to another very popular approach to modeling the empirical phenomenon of heavy tails, in which the process is assumed to be subject to random “jumps”. Jumps are generally modelled with a Poisson process with rate λ which defines the jump frequency. The higher the value of λ , the more likely a jump is to occur. Jumps occur independently of one another and when a jump does occur, the underlying process increases by the jump size.

The issue with using a simple Poisson model is that it assumes that all jumps are of uniform size, which is manifestly not the way that asset process behave empirically. So, in addition to modeling the jump arrival process, we also need to a means of modeling the magnitude of jumps.

Merton's Jump Diffusion Model

In Merton's jump diffusion model, the stock price follows the random process

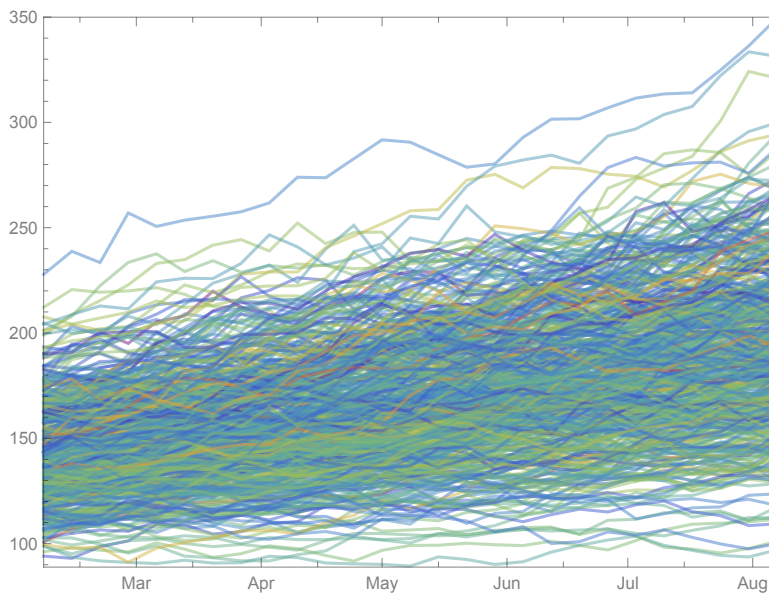
$dS_t / S_t = \mu dt + \sigma dW_t + (J - 1) dN(t)$, which comprises, in order, drift, diffusive, and jump components. The jumps occur according to a Poisson distribution and their size follows a log-normal distribution, or, equivalently, the size of jumps in the returns process follows a normal distribution, allowing for both positive and negative jumps in asset returns. The model is characterized by the diffusive volatility σ , the average jump size J (expressed as a fraction of S_t), the frequency of jumps λ , and the volatility of jump size v .

Let's apply this concept to create a jump diffusion version of the GBM process we originally estimated for AAPL stock:

```
Clear[t, x, y]
estJGBMproc = TransformedProcess[x[t] + y[t],
  {x ≈ GBMproc, y ≈ CompoundPoissonProcess[0.1, LogNormalDistribution[1, 0.5]]}, t];

simJGBMproc = RandomFunction[estJGBMproc,
  {tsAAPL["PathLength"], tsAAPL["PathLength"] + 26, 1}, 500];
tJd = TemporalData[simJGBMproc["ValueList"],
  {data[{-1, 1}], Automatic, "Week"}, ValueDimensions → 1];
```

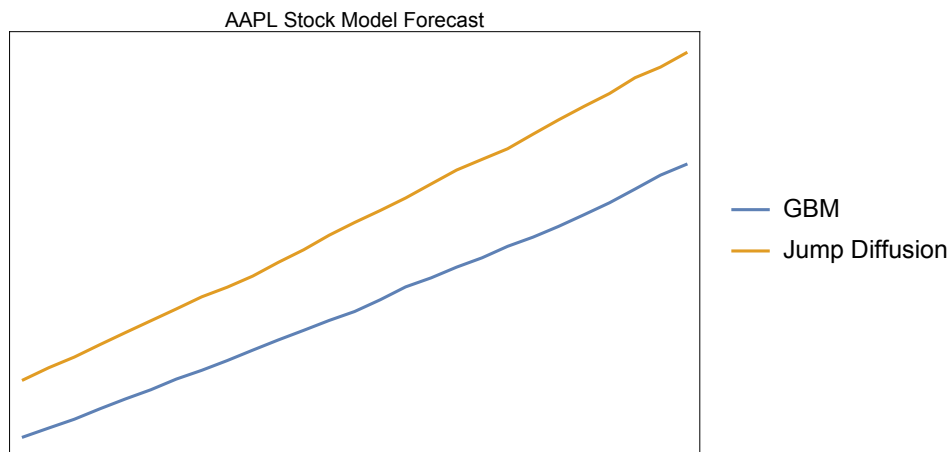
```
DateListPlot[tJd, ImageSize → 400, PlotRange → All, AspectRatio → 3/4,
  Epilog → Inset[sliced, {3 713 212 800, 0}], PlotStyle → (cf /@ Rescale[sd]),
  BaseStyle → Directive[Thin, Opacity[0.5]], PlotRangePadding → {{0, .25}, {.5, .5}}]
```



The chart of the simulated paths for the jump diffusion process may not look too dissimilar from the original GBM process, but the differences become more apparent when we compare the evolution of the means of the two processes:

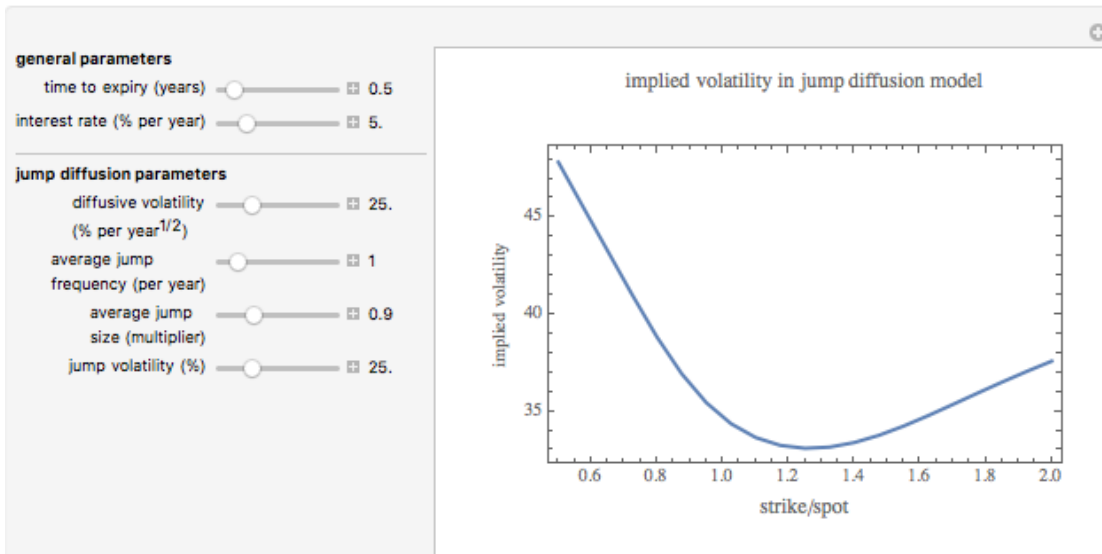
```
meantd = TimeSeriesThread[Mean, td];
meantJd = TimeSeriesThread[Mean, tJd];

DateListPlot[{meantd, meantJd}, PlotLegends → {"GBM", "Jump Diffusion"},
  PlotLabel → "AAPL Stock Model Forecast"]
```

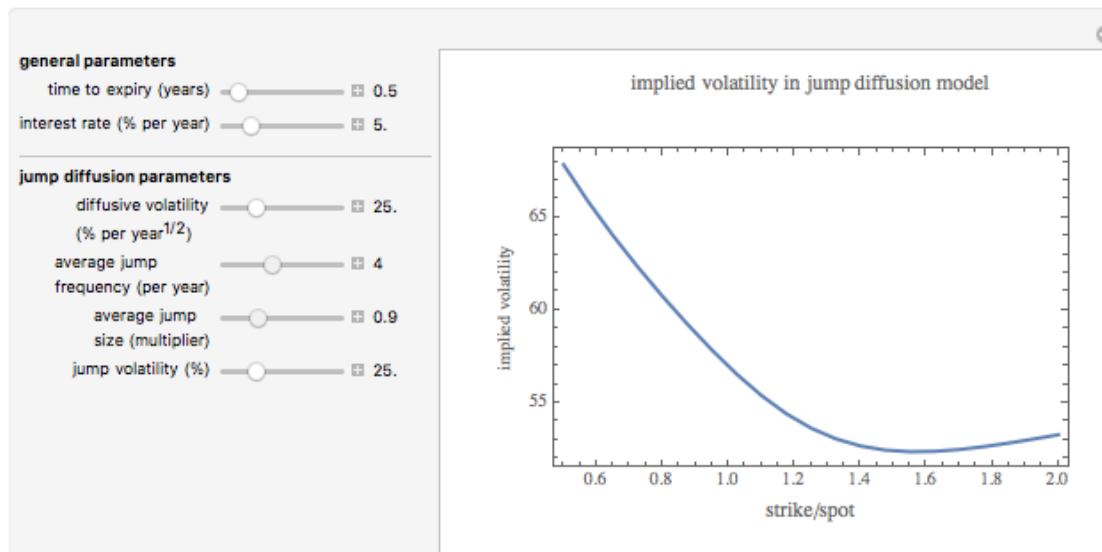


The Volatility Surface in Merton's Jump Diffusion Model

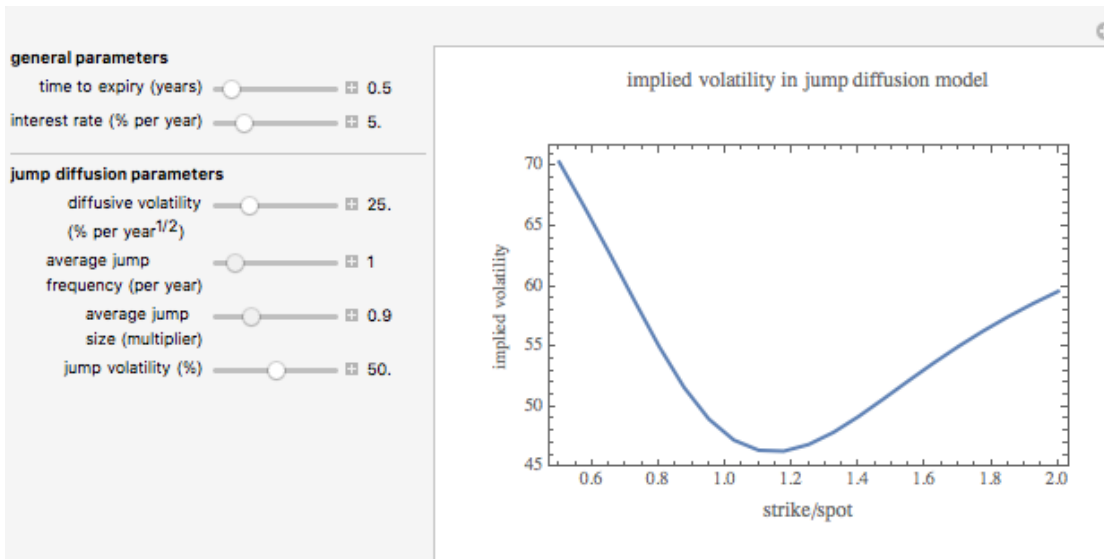
One of the principal effects of introducing stochastic jumps into the standard GBM framework is to produce distributions with heavier tails than predicated in a Gaussian framework. This in turn results in implied volatility smiles of varying shapes, depending on the values chosen for the model parameters. For example, if we consider 6-month options with an average jump frequency of one jump a year of average relative jump size 0.9 and volatility of volatility of 25%, we find a characteristic smile shape for the implied volatility curve, as follows:



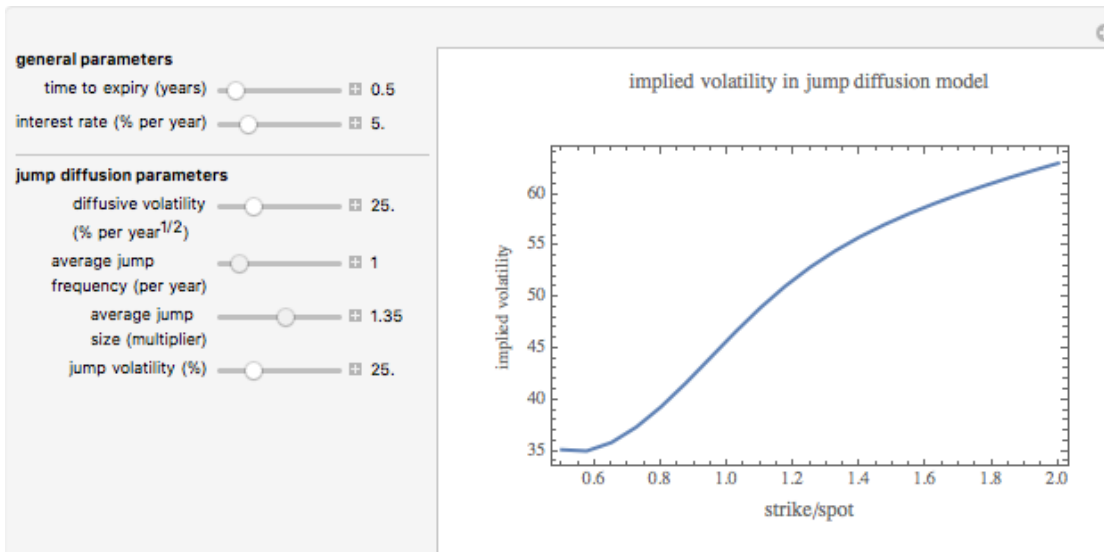
Some of the effects of varying the model parameters are intuitively obvious. For instance, if we increase the average jump frequency to 4 times a year, while keeping all the other model parameters unchanged, we find that the entire implied volatility curve is shifted upwards, with greater curvature in the leftmost portion:



Increasing the jump volatility likewise shifts the entire implied volatility curve upwards, but now accentuates the curvature in both halves:

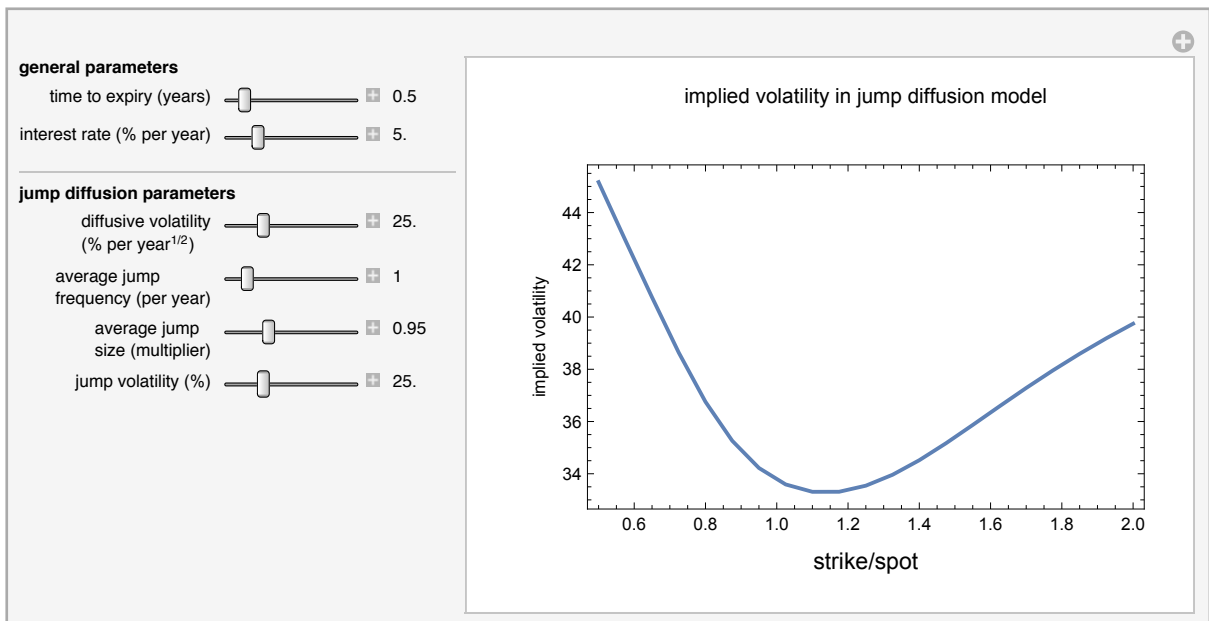


On the other hand, an increase in the average relative jump size changes the shape of the entire implied volatility curve:



A Mathematica demonstration project developed by Peter Falloon, reproduced below, allows the user to explore how varying the parameters of the Merton jump diffusion model affects option implied volatility and the shape of the implied volatility curve.

```
Manipulate[
Module[{type = "call", S = $spot, q = 0, r = 0.01 rate,
   $\sigma$  = 0.01 vol,  $\nu$  = 0.01 nu, xMin = 0.5 $spot, xMax = 2. $spot}, Quiet@
  ListLinePlot[Table[{k/S, 100. impliedVolJD[type, S, k, r, q,  $\sigma$ , T,  $\lambda$ , m,  $\nu$ ]}, {k,
    xMin, xMax, ControlActive[0.25, 0.05] (xMax - xMin)}], Evaluate[plotOpts]]],
Style["general parameters", Bold],
{{T, 0.5, "time to expiry (years)"},
  0.25, 4, 0.25, Appearance → "Labeled", ImageSize → Tiny},
{{rate, 5., "interest rate (% per year)"}, 0., 25.,
  0.25, Appearance → "Labeled", ImageSize → Tiny},
Delimiter,
Style["jump diffusion parameters", Bold],
{{vol, 25.,
  Row[{"diffusive volatility\n(% per ", Superscript["year", "1/2"], ")"]},
  1., 100., 1., Appearance → "Labeled", ImageSize → Tiny},
{{ $\lambda$ , 1, "average jump \nfrequency (per year)"}, 0, 10,
  1, Appearance → "Labeled", ImageSize → Tiny},
{{m, 0.9, "average jump\nsize (multiplier)"}, .5, 2.,
  0.05, Appearance → "Labeled", ImageSize → Tiny},
{{nu, 25., "jump volatility (%)"}, 0., 100., 1.,
  Appearance → "Labeled", ImageSize → Tiny},
ControlPlacement → Left, AutorunSequencing → {1, 3, 4, 5, 6}, SaveDefinitions → True]
```



Conclusion

The basic Geometric Brownian Motion model has served the quantitative research and financial engineering community well over the last several decades, providing a coherent framework for modeling many different types of asset processes and leading to tractable models for the pricing of assets and their derivatives.

Over time, however, shortcomings in the model have become more apparent as discrepancies have been revealed between the empirical characteristics of asset processes compared to those assumed within a theoretical, Gaussian setting.

Thanks to advances in the theory of asset process models over the last few decades, however, we are no longer constrained by the unrealistic assumptions of purely Gaussian techniques. We can, instead, incorporate a variety of different types of nonlinear behavior into our models, leading to more plausible explanations of some of the important characteristics we see in the empirical data and providing the means to price assets and their derivatives more consistently.

References

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

P. Falloon, "Implied Volatility in Merton's Jump Diffusion Model" from the Wolfram Demonstrations Project
<http://demonstrations.wolfram.com/ImpliedVolatilityInMertonsJumpDiffusionModel/>

J. Gatheral, *The Volatility Surface: A Practitioner's Guide*, Hoboken: John Wiley & Sons, Inc., 2006.

P. Jackel and C. Kahl, "Not-So-Complex Logarithms in the Heston Model", *Wilmott Magazine*, **19**, 2005 pp. 94–103.

M. Joshi, *The Concepts and Practice of Mathematical Finance*, Cambridge: Cambridge University Press, 2003.

A. L. Lewis, *Option Valuation under Stochastic Volatility: With Mathematica Code*, Newport Beach: Finance Press, 2000.

R. Merton, *Continuous-Time Finance*, Oxford: Blackwell, 1998.

S. Solganik, "Volatility Surface in the Heston Model" from the Wolfram Demonstrations Project
<http://demonstrations.wolfram.com/VolatilitySurfaceInTheHestonModel/>

Web Reference

Stochastic Volatility at Wikipedia