# SimulatingSupply

March 21, 2019

## 1 Simulating Supply Tutorial

Finance 5330: Advanced Derivative Markets Tyler J. Brough Last Updated: March 19, 2019

```
In [37]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
```

### 1.1 Simulating the Spot Prices for Heating Oil and Gasoline

To simulate a few values from the spot price equation, which is stated as follows:

$$\ln\left(\frac{S_{i,t}}{S_{i,t-1}}\right) = \alpha_i(\beta_i - S_{i,t-1}) + \varepsilon_{i,t}$$

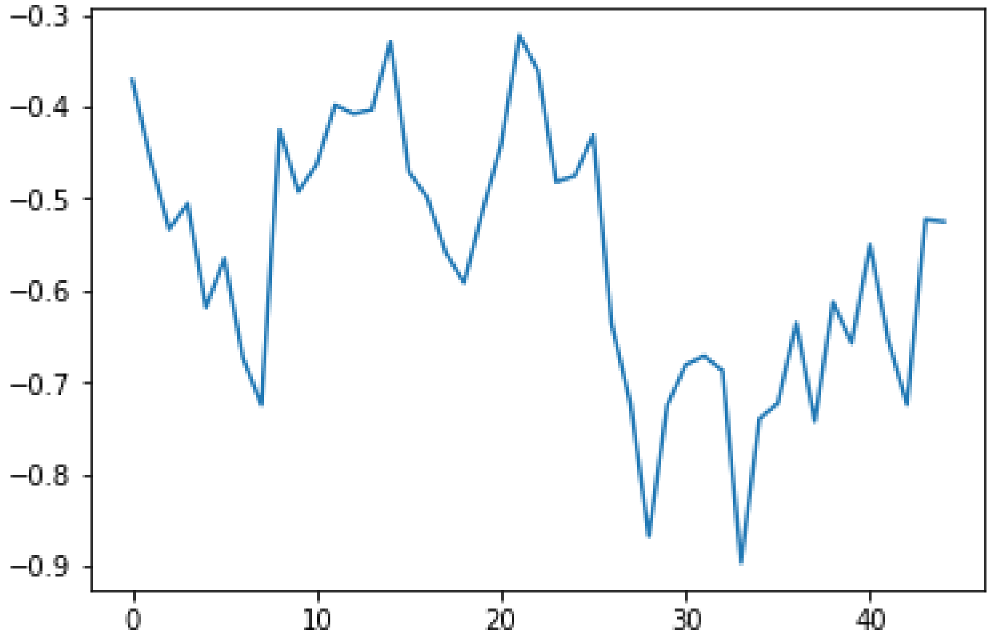
Where $\varepsilon_1, \varepsilon_2 \sim BVN(0, \sigma_1^2, 0, \sigma_2^2, \rho)$. For convenience we can rewrite this as:

$$\ln(S_{i,t}) = \ln(S_{i,t-1}) + \alpha_i(\beta_i - S_{i,t-1}) + \varepsilon_{i,t}$$

```
In [38]: a1 = 0.342
         b1 = 0.539
         s1 = 0.11
         S1 = 0.69
         numReps = 45

         lnSpot1 = np.zeros(numReps)
         lnSpot1[0] = np.log(S1)

         z1 = np.random.normal(size=numReps)

         for t in range(1, numReps):
             lnSpot1[t] = lnSpot1[t-1] + a1 * (b1 - np.exp(lnSpot1[t-1])) + z1[t] * s1
```

```
In [39]: plt.plot(lnSpot1)
```

```
Out[39]: [<matplotlib.lines.Line2D at 0x7f6a1a4d2080>]
```




### 1.2 Simulating Correlated Normals

Since the disturbance terms are distributed jointly Normal, we need a way to draw from the BVN distribution. It turns out this is pretty easy.

Here are the necessary steps to draw two correlated normal random variables.

1. Draw $z_1$ independently from $N(0, \sigma_{z_1})$
2. Next draw $z_{tmp}$ from a standard normal $N(0, 1)$
3. Create $z_2$ that is correlated with $z_1$ according to the correlation coeficient $\rho$

We can use the following equation to accomplish the third step:

$$z_2 = z_1 * \rho + \sqrt{(1 - \rho^2)} * z_{tmp} * \sigma_2$$

We can do this in Python as follows:

```
In [40]: s1 = 0.11
         s2 = 0.116
         rho = 0.705
         numReps = 10000

         z1 = np.random.normal(size=numReps) * s1
         ztmp = np.random.normal(size=numReps)
         z2 = z1 * rho + np.sqrt((1 - rho**2)) * ztmp * s2

In [41]: np.corrcoef(z1,z2)
```

```
Out[41]: array([[1.        , 0.68271331],
                [0.68271331, 1.        ]])
```

I prefer to create a function to handle it:

```
In [42]: def drawCorrelatedNormals(mn1 = 0.0, sd1 = 1.0, mn2 = 0.0, sd2 = 1.0, rho = 0.5, numRe
             z1 = np.random.normal(size=numReps, loc=mn1, scale=sd1)
             z2 = np.random.normal(size=numReps, loc=mn2, scale=sd2)
             z2 = rho * z1 + np.sqrt((1.0 - rho**2.0)) * z2

             return (z1,z2)
```

```
In [43]: x1, x2 = drawCorrelatedNormals(sd1=0.11, sd2=0.116, rho=0.705, numReps=45)
         np.corrcoef(x1, x2)
```

```
Out[43]: array([[1.        , 0.77321579],
                [0.77321579, 1.        ]])
```

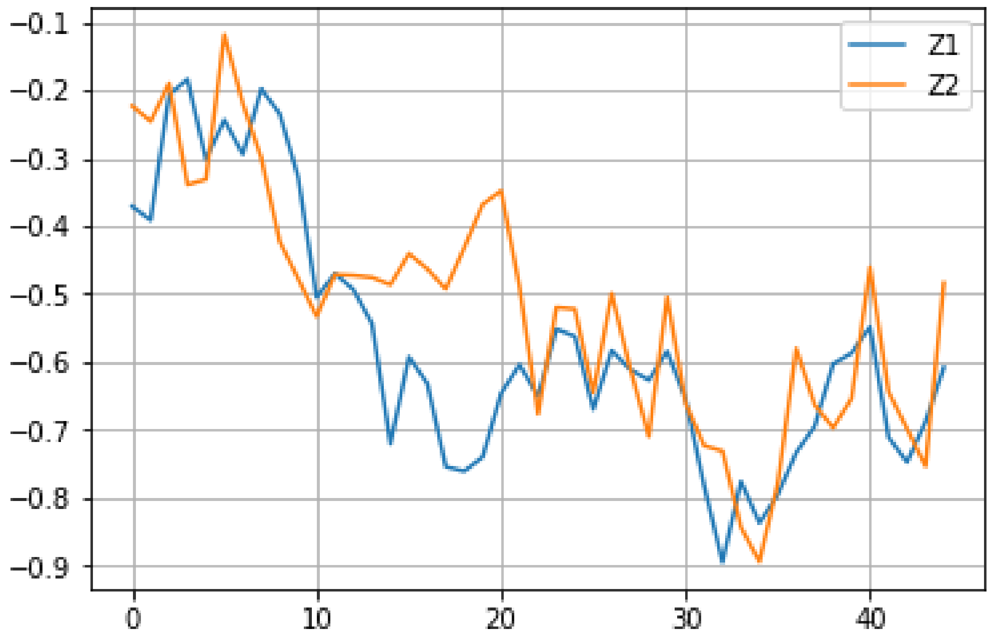
### 1.3 Simulating Sport Prices for Oil and Gasoline Jointly

With this in place, we can now simulate spot prices jointly as follows:

```
In [47]: a1 = 0.342
         b1 = 0.539
         s1 = 0.11
         S1 = 0.69
         a1 = 0.342
         b1 = 0.539
         s1 = 0.11
         S1 = 0.69
         a2 = 0.391
         b2 = 0.560
         s2 = 0.116
         S2 = 0.80
         numReps = 45

         lnSpot1 = np.zeros(numReps)
         lnSpot2 = np.zeros(numReps)

         lnSpot1[0] = np.log(S1)
         lnSpot2[0] = np.log(S2)


         z1, z2 = drawCorrelatedNormals(sd1=s1, sd2=s2, rho=rho, numReps=numReps)

         for t in range(1, numReps):
             lnSpot1[t] = lnSpot1[t-1] + a1 * (b1 - np.exp(lnSpot1[t-1])) + z1[t]
             lnSpot2[t] = lnSpot2[t-1] + a2 * (b2 - np.exp(lnSpot2[t-1])) + z2[t]

         ts = pd.DataFrame({'Z1' : lnSpot1, 'Z2' : lnSpot2})
```

```
In [48]: ts.plot(grid=True)
```

```
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6a1a3d4e80>
```




```
In [49]: ts.head()
```

```
Out[49]:          Z1        Z2
         0 -0.371064 -0.223144
         1 -0.391465 -0.245855
         2 -0.207017 -0.190089
         3 -0.184068 -0.339171
         4 -0.301766 -0.331299
```

```
In [50]: ts.tail()
```

```
Out[50]:           Z1        Z2
         40 -0.548103 -0.460639
         41 -0.710941 -0.644121
         42 -0.747535 -0.698347
         43 -0.688057 -0.753505
         44 -0.607892 -0.483634
```

```
In [ ]:
```