# MF4056/MF6013 Lab 3 - Numerical methods for SDE models

Open a new Jupyter notebook and import the following packages:

```
import numpy as np
import math as m
import matplotlib.pyplot as plt
```

# 1 Exact versus approximate trajectories

## 1.1 Path generation

To show the propagation of error along a single trajectory of a GBM we will reproduce Figure 3 from page 534 of Higham's review article.

**Exercise 1** *Using the parameters*

$$T = 1,\ N = 2^8,\ S_0 = 1,\ \lambda = 2,\ \sigma = 1,$$

*use the code from Demo 1 to generate and store a single trajectory of a standard Brownian motion, stored in an array variable* `B`*, followed by a single trajectory of the geometric Brownian motion*

$$S(t) = S_0 \exp\left((\lambda - \sigma^2/2)t + \sigma B(t)\right), \quad t \in [0, 1].$$

*stored in an array variable* `S`*.*

Next, we can re-use our sampled Brownian trajectory in `B` to generate an approximate solution via the Euler-Maruyama method. First, instantiate an array to hold the approximate values

```
EM=np.ones(N+1)
```

Now populate it using the scheme:

```
for i in range(N):
EM[i+1]=EM[i]*(1+lambda*dt+sigma*(B[i+1]-B[i]));
```

Finally, overlay the two plots:

```
fig, ax=plt.subplots; plt.plot(t,S); plt.plot(EM);
```

## 1.2 Pathwise error

We can visualise the propagation of error along this trajectory by computing the vector of errors at each point in time and plotting it:

```
err=np.abs(S-EM); plt.plot(t,err)
```

**Exercise 2** *Repeat this process over 100 different trajectories to generate an ensemble of error plots overlaid on each other.*

It should be clear from Exercise 2 that the error plot is random, since it changes for each trajectory. However if we stay on a single trajectory and increase the resolution of the mesh we should be able to observe *pathwise* convergence.

**Exercise 3**

1. *Generate an exact trajectory using the parameters from Exercise 1 and store in a variable* `S`. *Keep the Brownian path in a variable* `B`.
2. *Repeat the process described above to generate an Euler-Maruyama approximation of the same trajectory on the same mesh and store in a variable* `EM`.
3. *Use a Brownian bridge to increase the resolution of the Brownian path by a factor of 2, so that $N = 2^9$. Store in a variable* `B2`.
4. *Repeat steps 1 and 2 above on the denser mesh and store the exact and approximate solutions in the variables* `S2` *and* `EM2`.
5. *Compare the plot of* `err=np.abs(S-EM)` *to* `err2=np.abs(S2-EMs)`. *Remember to adjust the time set for the denser mesh and only compare at points common to both meshes.*
6. *Double the resolution again, so that $N = 2^10$ and repeat the above.*

# 2 The Heston stochastic volatility model

## 2.1 Uncorrelated noise terms

The Cox-Ingersoll-Ross process has also been used to describe a generalisation of the Black-Scholes model where the volatility is allowed to vary stochastically. An example of this kind of model, where the randomness driving the volatility is fully independent of the randomness driving the asset, is given by the following Heston model:

$$dS(t) = rS(t)dt + \sqrt{V(t)}S(t)dB_1(t) \tag{1}$$

$$dV(t) = \lambda(\mu - V(t))dt + \sigma\sqrt{V(t)}dB_2(t), \tag{2}$$

for $t \in [0,1]$, $S(0) = S_0 > 0$,$V(0) = V_0 > 0$, and $B_1$ and $B_2$ are independent standard Brownian motions.

**Exercise 4** *For the parameter values $S_0 = 1$, $V_0 = 0.5$, $r = 0.02$, $\lambda = 1$, $\mu = 0.3$, $\sigma = 0.1$, use your answer to Q4 of Problem Set 2 to write code to generate an individual trajectory of $V$ as described by the Heston model given above. Use $N = 2^8$ steps. Then generate a trajectory of $S$ by*

```
S=S0*np.exp((r-V/2)*t+np.sqrt(V)*B)
```

## 2.2 Correlated noise terms

In practice we would expect the pair of standard Brownian motions $B_1$ and $B_2$ in (1) and (2) to be correlated, since the volatility is affected by the same random flow of information as the spot price. In this case $V$ no longer has a closed form distribution from which we can directly sample, and we must use numerical approximation methods.

Denote $\Delta B_{n+1} = B(t_{n+1}) - B(t_n)$. For this lab we will consider the following three methods, implemented on a mesh with constant stepsize $h = T/N$:

1. A *truncated* Euler-Maruyama method:
$$V_{n+1} = V_n + \lambda(\mu - V_n)h + \sigma\sqrt{\max\{V_n, 0\}}\Delta B_{n+1};$$

2. A *reflected* Euler-Maruyama method:
$$V_{n+1} = V_n + \lambda(\mu - V_n)h + \sigma\sqrt{|V_n|}\Delta B_{n+1};$$

3. A method based on the implicit discretisation of the Lamperti transform $\sqrt{V}$:
$$V_{n+1} = \left(\frac{Y_n + \gamma\Delta B_{n+1}}{2(1-\beta h)} + \sqrt{\frac{(Y_n + \gamma\Delta B_{n+1})^2}{4(1-\beta h)^2} + \frac{\alpha h}{1-\beta h}}\right)^2,$$
where $\alpha = (4\lambda\mu - \sigma^2)/8$, $\beta = -\lambda/2$, $\gamma = \sigma/2$.

**Exercise 5** *Implement these numerical methods in the case where $B_1$ and $B_2$ are uncorrelated:*

1. *Repeat Exercise 4 using each of the three methods listed above to generate the stochastic volatility process $V$ approximately, generating a trajectory of the spot price $S$.*

2. *Recall from class that when $\lambda\mu > 2\sigma^2$, Feller's condition is said to hold, and this is the case when $\lambda = 1$, $\mu = 0.3$, $\sigma = 0.1$. However Feller's condition is viewed as an unrealistic requirement for stochastic volatility models. A more realistic choice might be $\sigma = 0.5$. Does this have any noticeable effect on your simulation?*

3. *When $\lambda = 1$, $\mu = 0.3$, $\sigma = 0.5$, we have $\alpha = 0.95 > 0$. Increase $\sigma$ further so that $\alpha < 0$, say to $\sigma = 1.2$. What happens now?*

**Exercise 6** *Can we investigate pathwise error in the same way as in Exercise 3? Explain.*

**Exercise 7** *Suppose that $B_1$ and $B_2$ in* (1) *and* (2) *are correlated with $\rho = -0.5$. Write code incorporating the Cholesky factorisation method that generates trajectories of* (1) *and* (2) *using each of the three approximation methods described above.*