

```

1 #####
2 ###Plots the true solution and the approximation on n (and 2*n) time-points for the
3 ###Geometric brownian motion
4 ###Euler-Maruyama, Milstein (optional: Wagner-Platen)
5 ###SDE_ApproximationGBM.py
6 ###Python 2.7
7 #####
8
9 import numpy as np
10 import matplotlib.pyplot as plt
11 from NumericalSDE import *
12
13 #####
14 ###Geometric brownian motion (SDE)
15 ###  $dX_t = a(X_t)dt + b(X_t)dW_t$ 
16 ###  $X_0 = x_0$ 
17 ###  $a(x) = \mu x$ ,  $b(x) = \sigma x$ 
18 ###  $\mu$ ,  $\sigma$  constants
19 ### True solution:
20 ###
21 #####
22 #Parameter
23 sigma = 2.0
24 mu = 1.0
25 #functions a, b
26 def a(x):
27     return mu*x
28 def b(x):
29     return sigma*x
30 #derivatives of a, b
31 def a_dv(x):
32     return mu
33 def b_dv(x):
34     return sigma
35 def a_dv dv(x):
36     return 0
37 def b_dv dv(x):
38     return 0
39 #starting value x0
40 x0 = 1
41 #####
42 # Number of steps.
43 n = 32
44 #Wiener process w, discretization of [0,T] t
45 w = wiener(n)
46 t = timegrid(n)
47 #Finer versions(using refinement)
48 w2 = refineWiener(w)
49 t2 = timegrid(2*n)
50 #####
51 #####Numerical solutions#####
52 #####
53 #Numerical solution: Euler-scheme
54 Yt_euler = sde_euler(x0,a,b,w)
55 Yt2_euler = sde_euler(x0,a,b,w2)
56 #Numerical solution: Milstein-scheme
57 Yt_milstein = sde_milstein(x0, a, b, b_dv, w)
58 Yt2_milstein = sde_milstein(x0, a, b, b_dv, w2)
59 #Numerical solution: Wagner-Platen-scheme
60 #Yt_wagnerplaten = sde_wagnerplaten(x0, a, b, a_dv, b_dv, a_dv dv, b_dv dv, w)
61 #Yt2_wagnerplaten = sde_wagnerplaten(x0, a, b, a_dv, b_dv, a_dv dv, b_dv dv, w2)
62
63 #Analytical solution (Geometric brownian motion) (adapt this for other SDEs.)
64 Xt=np.zeros(n+1)
65 Xt2=np.zeros(2*n+1)
66 Xt[0] = x0
67 Xt2[0] = x0

```

```

68 for k in range(0,n):
69     Xt[k+1] = Xt[0]*np.exp((mu-sigma**2/2)*(t[k+1]) + sigma*w[k+1])
70 for k in range(0,2*n):
71     Xt2[k+1] = Xt2[0]*np.exp((mu-sigma**2/2)*(t2[k+1]) + sigma*w2[k+1])
72
73 #plot
74 plt.figure(1)
75 plt.scatter(t, Yt_euler, 1, c='b', label="Euler-method")
76 plt.scatter(t, Xt, 1, c='k', label="Analytical solution")
77 plt.fill_between(t, Xt, Yt_euler, color='b',alpha=.1, interpolate=False)
78 plt.legend(loc='upper left')
79 ax = plt.gca()
80 plt.text(0.025, 0.80,'n= ' + str(n), bbox=dict(facecolor='white'), transform =
81 ax.transAxes)
82 plt.xlabel('t', fontsize=16)
83 plt.ylabel('x', fontsize=16)
84
85 plt.figure(2)
86 plt.scatter(t, Yt_milstein, 1, c='m', label="Milstein-method")
87 plt.scatter(t, Xt, 1, c='k', label="Analytical solution")
88 plt.fill_between(t, Xt, Yt_milstein, color='m',alpha=.1, interpolate=False)
89 plt.legend(loc='upper left')
90 ax = plt.gca()
91 plt.text(0.025, 0.80,'n= ' + str(n), bbox=dict(facecolor='white'), transform =
92 ax.transAxes)
93 plt.xlabel('t', fontsize=16)
94 plt.ylabel('x', fontsize=16)
95
96 ##plt.figure(3)
97 ##plt.scatter(t, Yt_wagnerplatten, 1, c='forestgreen', label="Wagner-Platen-method")
98 ##plt.scatter(t, Xt, 1, c='k', label="Analytical solution")
99 ##plt.fill_between(t, Xt, Yt_wagnerplatten, color='forestgreen',alpha=.1,
100 interpolate=False)
101 ##plt.legend(loc='upper left')
102 ##ax = plt.gca()
103 ##plt.text(0.025, 0.80,'n= ' + str(n), bbox=dict(facecolor='white'), transform =
104 ax.transAxes)
105 ##plt.xlabel('t', fontsize=16)
106 ##plt.ylabel('x', fontsize=16)
107
108 plt.figure(4)
109 plt.scatter(t2, Yt2_euler, 1, c='b', label="Euler-method")
110 plt.scatter(t2, Xt2, 1, c='k', label="Analytical solution")
111 plt.fill_between(t2, Xt2, Yt2_euler, color='b',alpha=.1, interpolate=False)
112 plt.legend(loc='upper left')
113 ax = plt.gca()
114 plt.text(0.025, 0.80,'n= ' + str(2*n), bbox=dict(facecolor='white'), transform =
115 ax.transAxes)
116 plt.xlabel('t', fontsize=16)
117 plt.ylabel('x', fontsize=16)
118
119 plt.figure(5)
120 plt.scatter(t2, Yt2_milstein, 1, c='m', label="Milstein-method")
121 plt.scatter(t2, Xt2, 1, c='k', label="Analytical solution")
122 plt.fill_between(t2, Xt2, Yt2_milstein, color='m',alpha=.1, interpolate=False)
123 plt.legend(loc='upper left')
124 ax = plt.gca()
125 plt.text(0.025, 0.80,'n= ' + str(2*n), bbox=dict(facecolor='white'), transform =
126 ax.transAxes)
127 plt.xlabel('t', fontsize=16)
128 plt.ylabel('x', fontsize=16)
129
130 ##plt.figure(6)
131 ##plt.scatter(t2, Yt2_wagnerplatten, 1, c='forestgreen', label="Wagner-Platen-method")
132 ##plt.scatter(t2, Xt2, 1, c='k', label="Analytical solution")
133 ##plt.fill_between(t2, Xt2, Yt2_wagnerplatten, color='forestgreen',alpha=.1,
134 interpolate=False)

```

```
128     ##plt.legend(loc='upper left')
129     #ax = plt.gca()
130     #plt.text(0.025, 0.80, 'n= ' + str(2*n), bbox=dict(facecolor='white'), transform =
ax.transAxes)
131     ##plt.xlabel('t', fontsize=16)
132     ##plt.ylabel('x', fontsize=16)
133
134     plt.show()
135
```