```
1  ##############################################################################
2  #############  Code online at github.com/tsbertalan/504hw4 #############
3  ##############################################################################
4  import numpy as np
5  import matplotlib.pyplot as plt
6  from math import sqrt
7
8  def do_IC(k1, kn1, k2, kn2, k3, x0, y0, dt=0.01, numsteps=256):
9      def xprime(x, y):
10         return k1 * (1 - x - y) - kn1 * x - k3 * x * y
11     def yprime(x, y):
12         return k2 * (1 - x - y)**2 - kn2 * y ** 2 - k3 * x * y
13     def pprime(x, y):
14         return k3 * x * y
15
16     xold = x0
17     yold = y0
18     tlist = []
19     xlist = []
20     ylist = []
21     plist = []
22     pold = 0
23     told = 0
24     tlist.append(told)
25     xlist.append(xold)
26     ylist.append(yold)
27     plist.append(pold)
28     for j in range(numsteps):
29         x = xold + dt * xprime(xold, yold)
30         y = yold + dt * yprime(xold, yold)
31         p = pold + dt * pprime(xold, yold)
32         t = told + dt
33         xlist.append(x)
34         xold = x
35         ylist.append(y)
36         yold = y
37         plist.append(p)
38         pold = p
39         tlist.append(t)
40         told = t
41     return (tlist, xlist, ylist, plist)
42
43 def make_ICs(resolution=32):
44     ICs = []
45     for x0 in np.arange(0, 1. + 1./resolution, 1./resolution):
46         ICs.append((x0, 0))
47         ICs.append((x0, 1))
48         ICs.append((0, x0))
49         ICs.append((1, x0))
50     return ICs
51
52 # Problem 2 ###################################################################
53 k2 = 1
54 k3 = 10
55 fk1 = lambda y: k2*k3*(1-y)/k3/y
56 fx = lambda y: fk1(y)*(1-y)/(fk1(y)+k3*y)
57 fr = lambda x,y: k3*x*y
58 l1 = list(np.arange(0.001,        1,   .001))
59 l2 = list(np.arange(0.0001,    .001,  .0001))
60 l3 = list(np.arange(0.00001,  .0001, .00001))
61 l4 = list(np.arange(0.000001,.00001,.000001))
62 ylist = l1
63 ylist.extend(l2)  # We need extra resolution at low y
64 ylist.extend(l3)  # to properly capture the behavior.
65 ylist.extend(l4)  # Only a couple dozen extra points.
66 ylist.sort()
```

```
 67 xlist = []
 68 k1list= []
 69 rlist = []
 70
 71 kn1 = 0.05
 72 fk1n = lambda y: (kn1+k3*y)*(sqrt(1+4*k2*(1-y)/k3/y)-1)/2
 73 fxn = lambda y: fk1n(y)*(1-y)/(fk1n(y)-kn1+k3*y)
 74 frn = fr
 75 ylistn = []
 76 xlistn = []
 77 k1listn= []
 78 rlistn = []
 79 for y in ylist:
 80     x = fx(y)
 81     xlist.append(x)
 82     k1list.append(fk1(y))
 83     rlist.append(fr(x,y))
 84
 85     ylistn.append(y)
 86     xn = fxn(y)
 87     xlistn.append(xn)
 88     k1listn.append(fk1n(y))
 89     rlistn.append(frn(xn,y))
 90
 91 # Make a plot comparing reversible and reversible
 92 fig2b = plt.figure(22, figsize=(12, 8))
 93
 94 ax2O = fig2b.add_subplot(3,1,1)
 95 ax2O.set_xlim((0,1))
 96 ax2O.plot(k1list, ylist, 'k--')
 97 ax2O.plot(k1listn, ylistn, 'k-')
 98 ax2O.set_ylabel(r'Adsorbed $O_1$')
 99
100 ax2CO = fig2b.add_subplot(3,1,2)
101 ax2CO.set_xlim((0,1))
102 ax2CO.plot(k1list,xlist, 'k--')
103 ax2CO.plot(k1listn,xlistn, 'k-')
104 ax2CO.set_ylabel(r'Adsorbed $CO_1$')
105
106 ax2r = fig2b.add_subplot(3,1,3)
107 ax2r.set_xlim((0,1))
108 ax2r.set_ylim((0,.5))
109 ax2r.plot(k1list, rlist, 'k--')
110 ax2r.plot(k1listn, rlistn, 'k-')
111 ax2r.set_ylabel(r'Rate of Product Formation, ${d[CO_2]}/{dt}$')
112 ax2r.set_xlabel(r'Partial Pressure of Unadsorbed $CO_2$, $k_1$')
113
114 ax2r.legend([r'$k_{-1}=0$, Irreversible', r'$k_{-1}=%.2f$, Reversible'%kn1])
115 plt.suptitle('Reversible vs. Irreversible Adsorption and Reaction\n'\
116              r'$k_2=%.2f$, $k_3=%.2f$, $k_{-2}=0$' % (k2, k3))
117 plt.savefig('hw4_2b.pdf')
118 #plt.show()
119
120 # Make a plot showing irreversible only, since it has
121 # qualitatively interesting large-k1 behavior that won't
122 # fit on the previous plot
123 fig2c = plt.figure(23, figsize=(12, 8))
124
125 ax2O = fig2c.add_subplot(3,1,1)
126 ax2O.set_xlim((0,100))
127 ax2O.plot(k1list, ylist, 'k--')
128 ax2O.set_ylabel(r'Adsorbed $O_1$')
129
130 ax2CO = fig2c.add_subplot(3,1,2)
131 ax2CO.set_xlim((0,100))
132 ax2CO.plot(k1list,xlist, 'k--')
```

```
133 ax2CO.set_ylabel(r'Adsorbed $CO_1$')
134
135 ax2r = fig2c.add_subplot(3,1,3)
136 ax2r.set_xlim((0,100))
137 ax2r.plot(k1list, rlist, 'k--')
138 ax2r.set_ylabel(r'Rate of Product Formation, ${d[CO_2]}/{dt}$')
139 ax2r.set_xlabel(r'Partial Pressure of Unadsorbed $CO_2$, $k_1$')
140
141 ax2r.legend([r'$k_{-1}=0$, Irreversible'])
142 plt.suptitle('Irreversible Adsorption and Reaction\n'\
143              r'$k_2=%.2f$, $k_3=%.2f$, $k_{-2}=0$' % (k2, k3))
144 plt.savefig('hw4_2c.pdf')
145 #plt.show()
146
147
148 # Problem 3 ###################################################################
149 k1=.5
150 kn1=.05
151 k2=1
152 kn2=0
153 k3=10
154 # roots of the cubic polynomial in y that results from setting
155 #    dx/dt = 0 = dy/dt
156 # coefficients of said polynomial:
157
158 coeffs = [
159      k2*k3**2,
160      k1*k3**2 - 2*k2*k3**2 + 2*k2*k3*kn1,
161      k1**2*k3 - k1*k3**2 + k1*k3*kn1 + k2*k3**2 - 4*k2*k3*kn1 + k2*kn1**2,
162      -k3*k1**2 - k1*k3*kn1 +2*k2*k3*kn1 - 2*k2*kn1**2,
163      k2*kn1**2
164      ]
165
166 yroots = np.roots(coeffs)
167 fx = lambda y: k1*(1-y)/(k1+kn1+k3*y)
168 xroots = map(fx, yroots)
169
170 fig3 = plt.figure(3, figsize=(12, 8))
171 ax3 = fig3.add_subplot(1,1,1)
172 ax3.set_xlim((0, 1))
173 ax3.set_ylim((0, 1.2))
174 ICs_ss = zip(xroots, yroots)
175 print '#### Problem 3, steady-states ####'
176 print "Steady-states (x, y) are:"
177 for (x, y) in ICs_ss:
178     print '(%.3f, %.3f)' % (x, y)
179 repeats = 12
180 for (x0,y0) in ICs_ss * repeats:
181     x0 = x0 + np.random.rand()*1/8. - 1/16.
182     y0 = y0 + np.random.rand()*1/8. - 1/16.
183     (tlist, xlist, ylist, plist) = do_IC(k1, kn1, k2, kn2, k3, x0, y0)
184     ax3.plot(xlist, ylist, 'k')
185 for (x0,y0) in ICs_ss:
186     ax3.scatter([x0], [y0], color='k')
187 ax3.set_title('Small random perturbations from steady-states')
188 ax3.set_xlabel(r'x, adsorbed $CO_1$')
189 ax3.set_ylabel(r'y, adsorbed $O_1$')
190 plt.savefig('hw4_3.pdf')
191
192 fig3b = plt.figure(32, figsize=(12, 8))
193 ax3b = fig3b.add_subplot(1,1,1)
194 ICs = make_ICs(resolution=128)
195
196 for (x0,y0) in ICs:
197     (tlist, xlist, ylist, plist) = do_IC(k1, kn1, k2, kn2, k3, x0, y0)
198     ax3b.plot(xlist, ylist, 'k')
```

```
199     ax3b.set_xlim((0,1))
200     ax3b.set_ylim((0,1))
201 for (x0,y0) in ICs_ss:
202     ax3b.scatter([x0], [y0], color='k')
203 ax3b.set_xlim((0, 1))
204 ax3b.set_ylim((0, 1))
205 ax3b.set_title('Phase plot for many ICs.')
206 ax3b.set_xlabel(r'x, adsorbed $CO_1$')
207 ax3b.set_ylabel(r'y, adsorbed $O_1$')
208 plt.savefig('hw4_3b.pdf')
209 
210 # Problem 3, Linear stability analysis ########################################
211 f1x = lambda x,y: -k1 -kn1 - k3*y
212 f2x = lambda x,y: -2*k2+2*k2*x+2*k2*y-k3*y
213 f1y = lambda x,y: -k1-k3*x
214 f2y = lambda x,y: -2*k2+2*k2*x+2*k2*y-k3*x
215 print ''
216 print '#### Problem 3, Linear stability analysis ####'
217 for (x,y) in ICs_ss:
218     jacobian = np.array([[f1x(x,y), f1y(x,y)],
219                          [f2x(x,y), f2y(x,y)]])
220     print ''
221     print 'At (', x, ',', y, '), the Jacobian is:'
222     print jacobian
223     [a,b,c,d] = jacobian.flatten().tolist()
224     coeffs = [1, -d-a, a*d-c*b]
225     eigv = np.roots(coeffs)
226     print 'Eigenvalues are:', eigv
227 
228 
229 
```