```
from future import division
 1
 2
     from math import sin,pi
 3
     from numpy import empty,array,arange
 4
     from pylab import plot,show
 5
 6
     q = 9.81
 7
     l = 0.1
 8
     theta0 = 179*pi/180
 9
10
     a = 0.0
     b = 10.0
11
    N = 100  # Number of "big steps"
H = (b-a)/N  # Size of "big steps"
delta = 1e-8  # Required position accuracy per unit time
12
13
14
15
16
     def f(r):
17
         theta = r[0]
18
         omega = r[1]
19
         ftheta = omega
20
         fomega = -(g/l)*sin(theta)
21
          return array([ftheta,fomega],float)
22
23
     tpoints = arange(a,b,H)
24
     thetapoints = []
25
     r = array([theta0,0.0],float)
26
27
     # Do the "big steps" of size H
28
     for t in tpoints:
29
30
         thetapoints.append(r[0])
31
32
         # Do one modified midpoint step to get things started
33
         n = 1
34
         r1 = r + 0.5*H*f(r)
35
         r2 = r + H*f(r1)
36
37
         # The array R1 stores the first row of the
38
         # extrapolation table, which contains only the single
39
         # modified midpoint estimate of the solution at the
40
         # end of the interval
41
         R1 = empty([1,2],float)
42
         R1[0] = 0.5*(r1 + r2 + 0.5*H*f(r2))
43
44
         # Now increase n until the required accuracy is reached
         error = 2*H*delta
45
46
         while error>H*delta:
47
```

Page 1, last modified

```
/Users/pjk/Dropbox (UToronto Physics)/teaching/PHY407/PHY407_2016/Lectures/Lecture07/bulirsch.py
```

```
48
             n += 1
49
             h = H/n
50
51
             # Modified midpoint method
52
             r1 = r + 0.5*h*f(r)
53
             r2 = r + h*f(r1)
54
             for i in range(n-1):
                 r1 += h*f(r2)
55
56
                  r2 += h*f(r1)
57
58
             # Calculate extrapolation estimates. Arrays R1 and R2
59
             # hold the two most recent lines of the table
60
             R2 = R1
61
             R1 = empty([n,2],float)
62
             R1[0] = 0.5*(r1 + r2 + 0.5*h*f(r2))
63
             for m in range(1,n):
                 epsilon = (R1[m-1]-R2[m-1])/((n/(n-1))**(2*m)-1)
64
65
                 R1[m] = R1[m-1] + epsilon
66
             error = abs(epsilon[0])
67
68
         # Set r equal to the most accurate estimate we have,
         # before moving on to the next big step
69
70
         r = R1[n-1]
71
72
     # Plot the results
73
     plot(tpoints, thetapoints)
74
     plot(tpoints,thetapoints,"b.")
75
     show()
76
```