Fourth Order Runge-Kutta

Colt Bradley

1 Description

We use a fourth order Runge-Kutta method to solve a second order equation for the damped driven pendulum. We'll use a convergence test to check our answer. We know that the equation

$$\frac{N_{\Delta t} - N_{\Delta t/2}}{N_{\Delta t/2} - N_{\Delta t/4}} = 2^n \tag{1}$$

For RK4, $n = 4$, so $2^4 = 16$. The numbers are 14.5546, 15.3481, 15.6911, 15.8491, so they converge to 16. It looks like the answer is accurate up to four decimal places for 1600 time steps, and up to five decimal places for 3200 time steps.

2 Code

```
#Colt Bradley
#Lesson 18
#4.5.16

#import essential modules
import numpy as n
import pylab as p

def f(u,v,t):
    g = 0.8
    w = n.pi*2
    w0 = 1.5*w
    b = w0/4.
    return -w0**2*n.sin(u)-2*b*v+g*w0**2*n.cos(w*t)
```

```
#Initial Values and Setup
time = []
N = []
R = []

for r in range(1,10):
    t=n.linspace(0,20,100*2.**r)
    R.append(100*2.**r)
    dt = t[1]-t[0]
    u = 0.
    v = 0.
    g = 0.8
    w = n.pi*2
    w0 = 1.5*w
    b = w0/4.
    U = []
    V = []
    for i in range (0,len(t)-1):
        ti = t[i]
        th = ti +dt/2.
        tf = t[i+1]

        va = v +u*dt/2.
        ua = u +f(v,u,ti)*dt/2.
        vb = v +ua*dt/2.
        ub = u +f(va,ua,th)*dt/2.
        vc = v +ub*dt
        uc = u +f(vb,ub,th)*dt
        vd = v +uc*dt
        ud = u +f(vc,uc,tf)*dt

        v = (va+2*vb+vc+(vd/2.))/3-v/2
        u = (ua+2*ub+uc+(ud/2.))/3-u/2

        V.append(v)
        U.append(u)
    N.append(V[-1])


print (N[-6]-N[-5])/(N[-5]-N[-4])
print (N[-5]-N[-4])/(N[-4]-N[-3])
print (N[-4]-N[-3])/(N[-3]-N[-2])
print (N[-3]-N[-2])/(N[-2]-N[-1])
print
print R
print N
```