Nonlinear FEM Homework 5

Truman Ellis

The natural frequencies of the system can be computed by solving for the square root of the eigenvalues of $M^{-1}K = K$. A simple calculation yields the following eigenvalues,

$$\lambda_1 = \frac{10002 - \sqrt{10002^2 - 40000}}{2} \approx 0.999900$$

$$\lambda_2 = \frac{10002 + \sqrt{10002^2 - 40000}}{2} \approx 10001.0\,,$$

from which we can calculate

$$\omega_1 \approx 0.99995$$

$$\omega_2 \approx 100.005\,.$$

Therefore $t_1 = \frac{2\pi}{\omega_1} \approx 2\pi$.

Then we get the following solutions for Central Difference, Trapezoid rule, and Damped Newmark, respectively for Exercise 16.

For Exercise 17 we get $\omega_1 = 0.707098$, and $\omega_2 = 141.423$, and similar results follow below.

In both problems it appears that Central Differences are unstable, the Trapezoid rule keeps the spurious modes, but doesn't allow them to grow, and Damped Newmark kills the spurious modes.

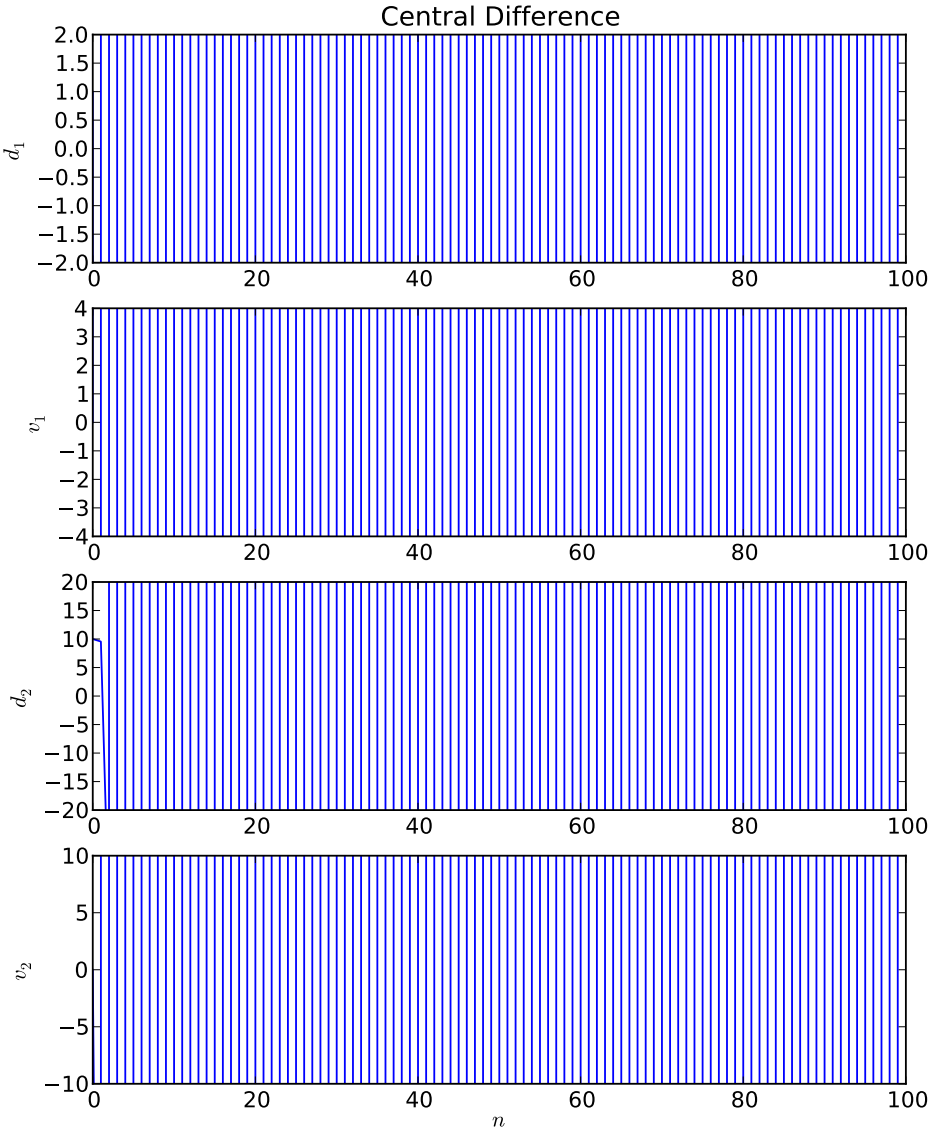

Figure 1: Exercise 16, Central Difference

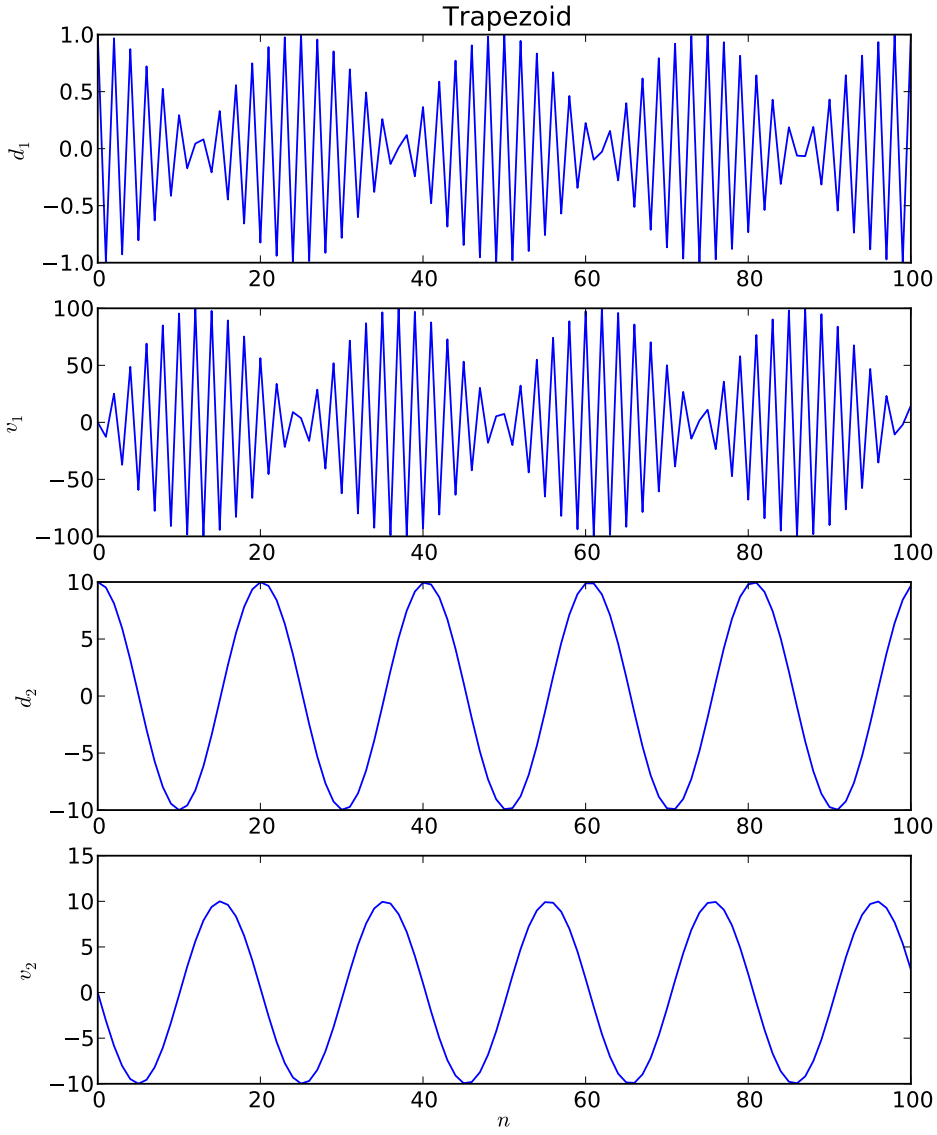

Figure 2: Exercise 16, Trapezoid Rule

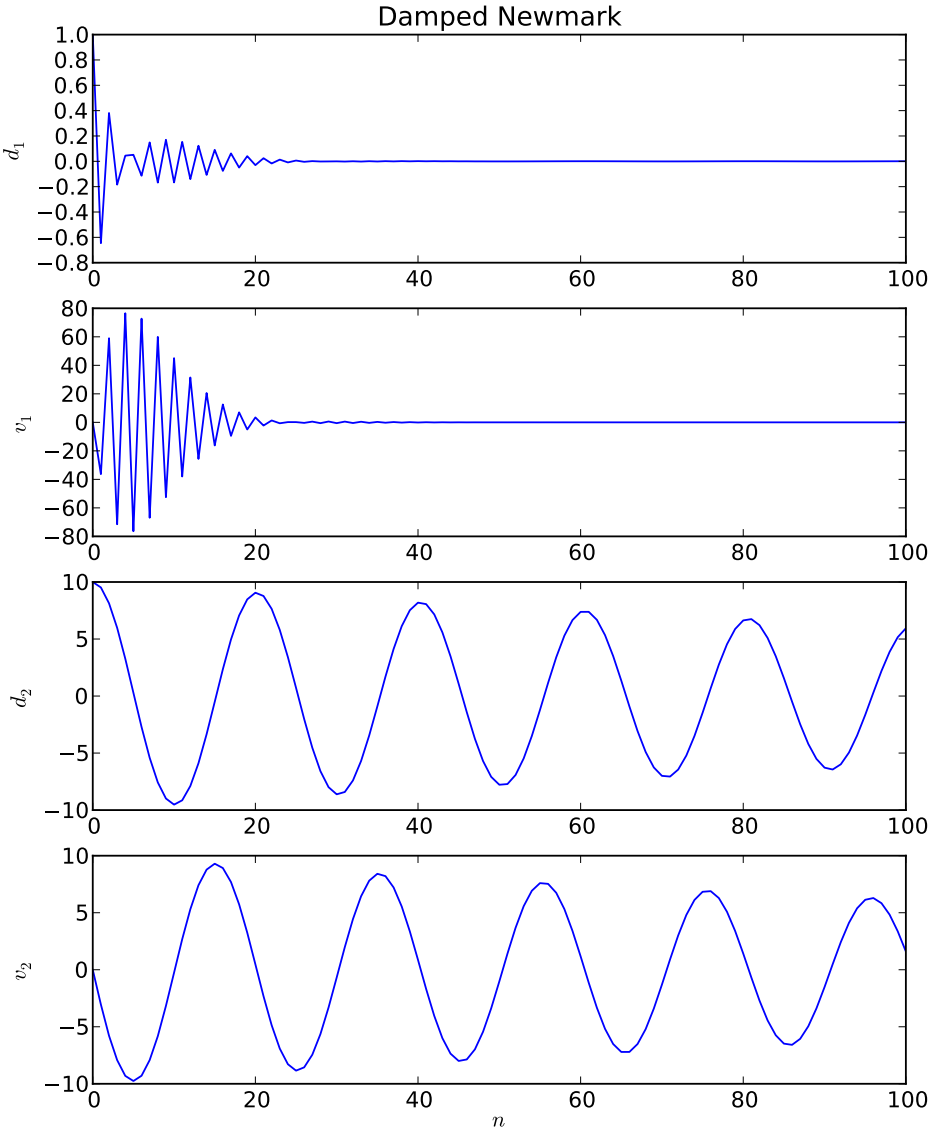

Figure 3: Exercise 16, Damped Newmark

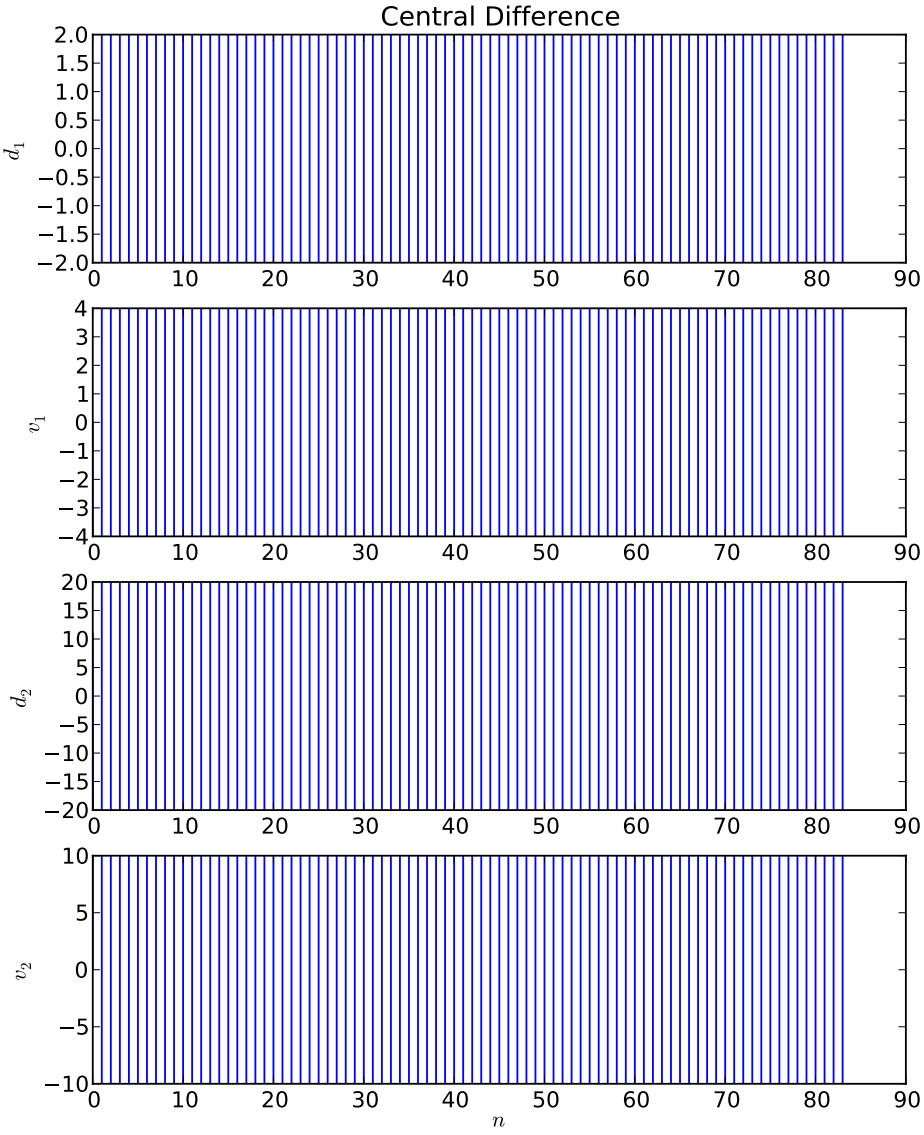

Figure 4: Exercise 17, Central Difference

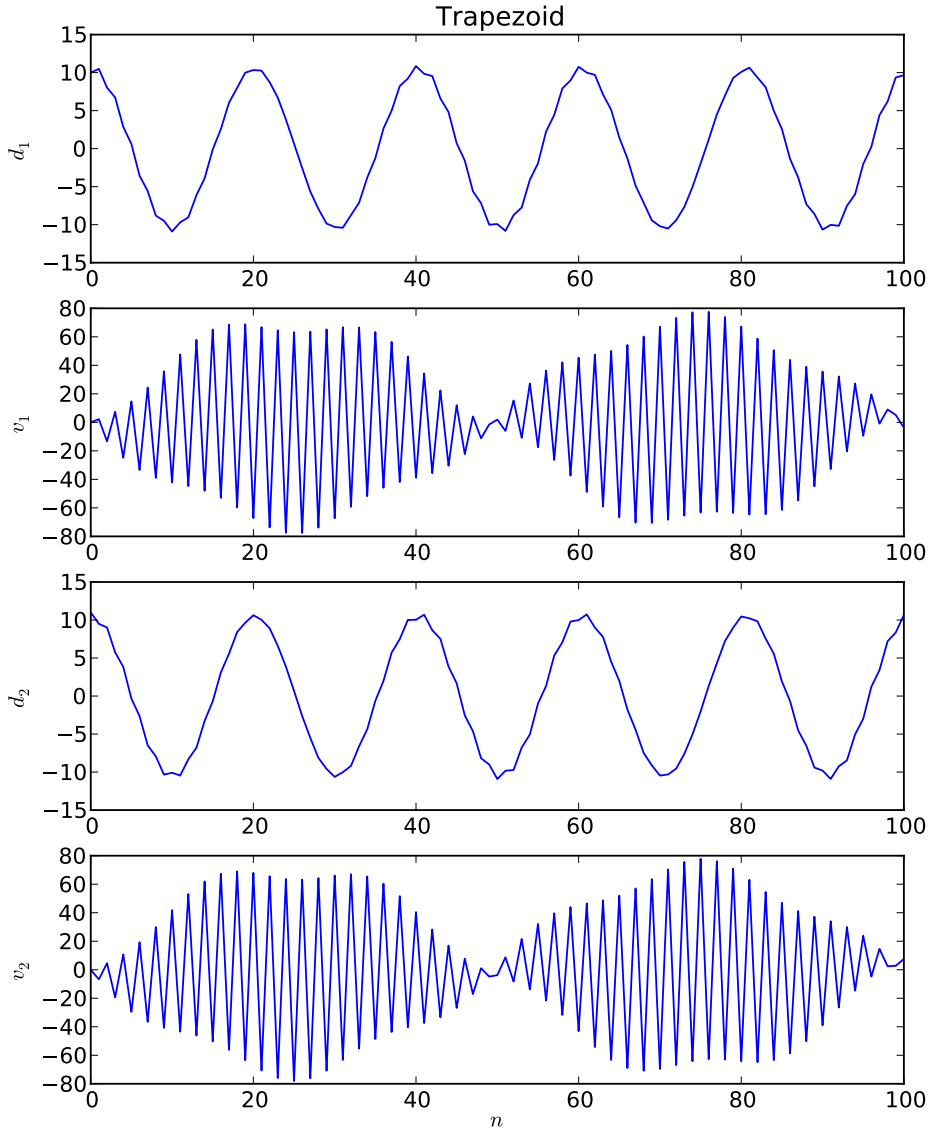

Figure 5: Exercise 17, Trapezoid Rule

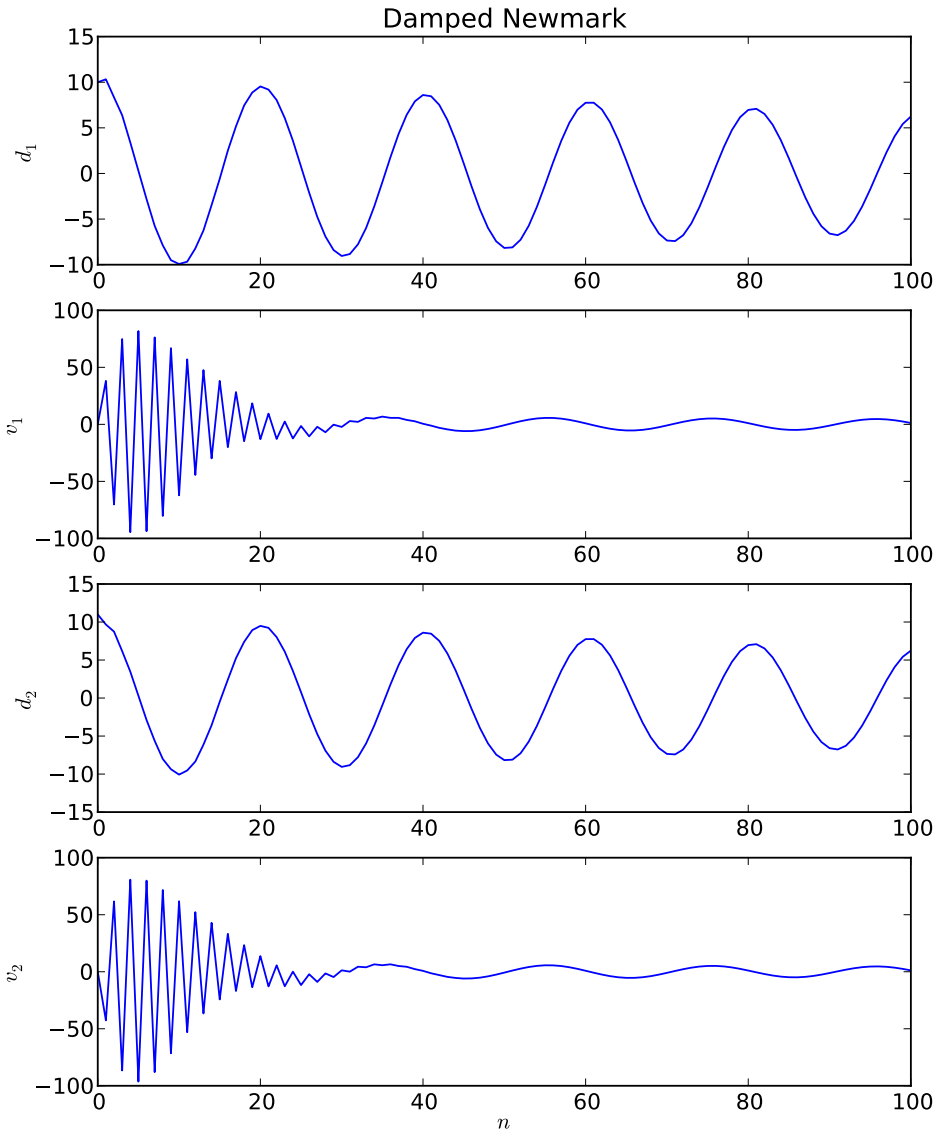

Figure 6: Exercise 17, Damped Newmark

HW10.py

```
from pylab import *
from scipy import optimize
close('all')

# Number of data points
N = 101

# System Definition
#T1 = 2*pi/sqrt((10002 - sqrt(10002**2-40000))/2)
T1 = 2*pi/sqrt((20001 - sqrt(20001**2-40000))/2)
#k1 = 1e4
#k2 = 1.
k1 = 1.
k2 = 1e4
M = array([[1,0],[0,1]])
K = array([[(k1+k2), -k2], [-k2, k2]])
#d0 = array([1, 10])
d0 = array([10, 11])
v0 = array([0, 0])
a0 = linalg.solve(M, -dot(K, d0))

# Method Parameters
# Central Difference
beta_cd = 0
gamma_cd = 0.5
# Trapezoid
beta_t = 0.25
gamma_t = 0.5
# Damped Newmark
beta_dn = 0.3025
gamma_dn = 0.6

# Data Arrays
# Central Difference
d_cd = zeros((2,N))
v_cd = zeros((2,N))
a_cd = zeros((2,N))
# Trapezoid
d_t = zeros((2,N))
v_t = zeros((2,N))
a_t = zeros((2,N))
# Damped Newmark
d_dn = zeros((2,N))
v_dn = zeros((2,N))
a_dn = zeros((2,N))

# Initialize
d_cd[:,0] = d_t[:,0] = d_dn[:,0] = d0
v_cd[:,0] = v_t[:,0] = v_dn[:,0] = v0
a_cd[:,0] = a_t[:,0] = a_dn[:,0] = a0

dt = T1/20.
t = linspace(0, 5*T1, N)
ns = range(0, N)

for n in range(0,N-1):
```

```
  # Central difference predictor stage
  d_cd_p = d_cd[:,n] + dt*v_cd[:,n] + 0.5*dt**2*(1-2*beta_cd)*a_cd[:,n]
  v_cd_p = v_cd[:,n] + (1-gamma_cd)*dt*a_cd[:,n]
  # Central difference corrector stage
  a_cd[:,n+1] = linalg.solve(M + beta_cd*dt**2*K, -dot(K, d_cd_p))
  d_cd[:,n+1] = d_cd_p + beta_cd*dt**2*a_cd[:,n+1]
  v_cd[:,n+1] = v_cd_p + gamma_cd*dt*a_cd[:,n+1]

  # Trapezoid predictor stage
  d_t_p = d_t[:,n] + dt*v_t[:,n] + 0.5*dt**2*(1-2*beta_t)*a_t[:,n]
  v_t_p = v_t[:,n] + (1-gamma_t)*dt*a_t[:,n]
  # Central difference corrector stage
  a_t[:,n+1] = linalg.solve(M + beta_t*dt**2*K, -dot(K, d_t_p))
  d_t[:,n+1] = d_t_p + beta_t*dt**2*a_t[:,n+1]
  v_t[:,n+1] = v_t_p + gamma_t*dt*a_t[:,n+1]

  # Damped Newmark predictor stage
  d_dn_p = d_dn[:,n] + dt*v_dn[:,n] + 0.5*dt**2*(1-2*beta_dn)*a_dn[:,n]
  v_dn_p = v_dn[:,n] + (1-gamma_dn)*dt*a_dn[:,n]
  # Damped Newmark corrector stage
  a_dn[:,n+1] = linalg.solve(M + beta_dn*dt**2*K, -dot(K, d_dn_p))
  d_dn[:,n+1] = d_dn_p + beta_dn*dt**2*a_dn[:,n+1]
  v_dn[:,n+1] = v_dn_p + gamma_dn*dt*a_dn[:,n+1]

figure(1, figsize=(8,10))
subplot(411)
title('Central Difference')
plot(ns, d_cd[0,:])
ylabel(r'$d_1$')
ylim(-2, 2)
subplot(412)
plot(ns, v_cd[0,:])
ylabel(r'$v_1$')
ylim(-4, 4)
subplot(413)
plot(ns, d_cd[1,:])
ylabel(r'$d_2$')
ylim(-20, 20)
subplot(414)
plot(ns, v_cd[1,:])
ylabel(r'$v_2$')
ylim(-10, 10)
xlabel(r'$n$')

figure(2, figsize=(8,10))
subplot(411)
title('Trapezoid')
plot(ns, d_t[0,:])
ylabel(r'$d_1$')
subplot(412)
plot(ns, v_t[0,:])
ylabel(r'$v_1$')
subplot(413)
plot(ns, d_t[1,:])
ylabel(r'$d_2$')
subplot(414)
plot(ns, v_t[1,:])
```

```
ylabel(r'$v_2$')
xlabel(r'$n$')

figure(3, figsize=(8,10))
subplot(411)
title('Damped Newmark')
plot(ns, d_dn[0,:])
ylabel(r'$d_1$')
subplot(412)
plot(ns, v_dn[0,:])
ylabel(r'$v_1$')
subplot(413)
plot(ns, d_dn[1,:])
ylabel(r'$d_2$')
subplot(414)
plot(ns, v_dn[1,:])
ylabel(r'$v_2$')
xlabel(r'$n$')

show()
```