In [ ]:

```
f_m = lambda t: 0
fxy_x = lambda t,x,v: v
fxy_v = lambda t,x,v: f_m(t)-2*zeta*omega*v-((omega**2)*x)

def rk4_x(ti,xi,vi,dt):
    K1 = fxy_x(ti,xi,vi)
    K2 = fxy_x(ti+a2*dt,xi+b21*K1*dt,vi)
    K3 = fxy_x(ti+a3*dt,xi+b31*K1*dt+b32*K2*dt,vi)
    K4 = fxy_x(ti+a4*dt,xi+b41*K1*dt+b42*K2*dt+b43*K3*dt,vi)
    xip1 = xi+(c1*K1+c2*K2+c3*K3+c4*K4)*dt
    return xip1

def rk4_v(ti,xi,vi,dt):
    K1 = fxy_v(ti,xi,vi)
    K2 = fxy_v(ti+a2*dt,xi,vi+b21*K1*dt)
    K3 = fxy_v(ti+a3*dt,xi,vi+b31*K1*dt+b32*K2*dt)
    K4 = fxy_v(ti+a4*dt,xi,vi+b41*K1*dt+b42*K2*dt+b43*K3*dt)
    vip1 = vi+(c1*K1+c2*K2+c3*K3+c4*K4)*dt
    return vip1


zeta = 0.2
omega =1.0                          #initial time
tf = 75.0                           #final time
n = 1000                            #segments
dt = (tf-ti)/n                      #step size
t = np.zeros(n)
x = np.zeros(n)
v = np.zeros(n)
x_iv = 0.0                          #initial position
v_iv = 2.0                          #initial velocity
t[0] = ti
x[0] = x_iv
v[0] = v_iv



for i in range(1,n):
    t[i] = ti+i*dt #increases time
    x[i] = rk4_x(t[i-1], x[i-1], v[i-1],dt)
    v[i] = rk4_v(t[i-1], x[i-1], v[i-1],dt)


LinePlot111(t,x,"Time(sec)","Displacement(m)","Displacement vs. Time (Underdamped)","Dv
LinePlot111(t,v,"Time(sec)","Velocity(m/s)","Velocity vs. Time (Underdamped)","VvT2.png
```