Measurement Systems Project

Alexander Tschinkel

December 14 2022

1 Spring Mass Damper

The Spring Mass damper system with $m = 10kg$, $k = 1N/m$, and initial conditions $x = -3$ and $v = 3.2$ was simulated under four different damping conditions. The results can be seen in figure 1.

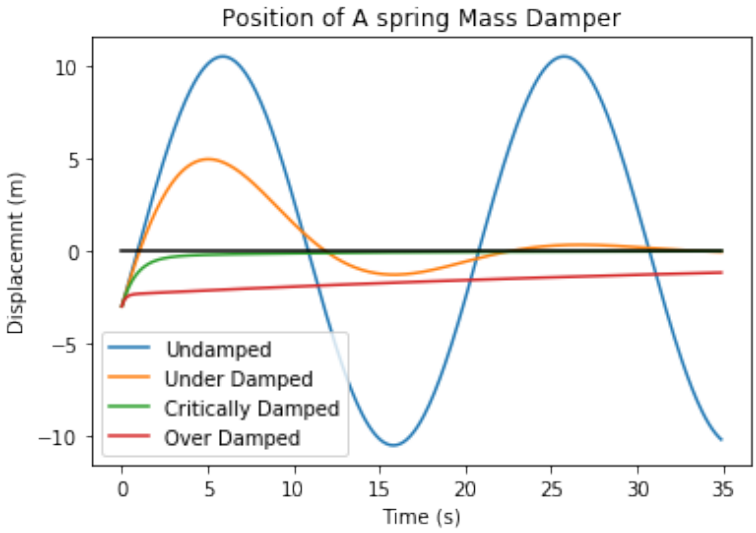

Figure 1: Spring Mass Damper System Under Different Conditions

2 Driven Oscillator

When the system was started at the origin with zero velocity, but with a driving force of $F(t) = 3\sin(10t + 87°)N$ and no damping, the following behavior was observed:

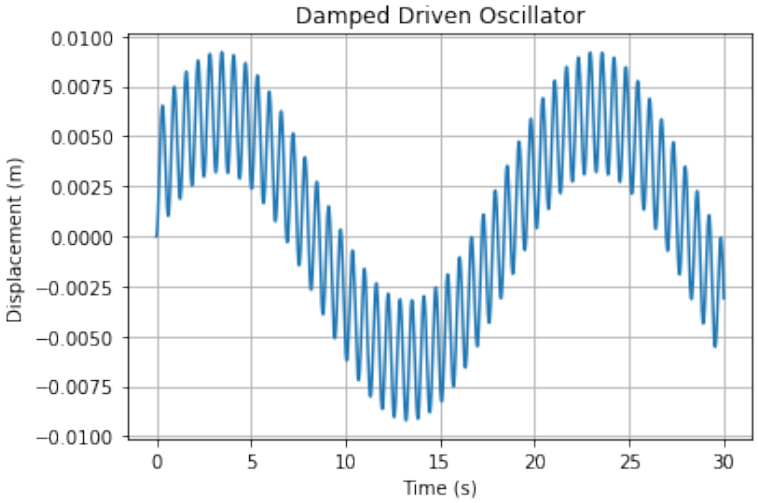

Figure 2: Driven Oscillator Started at Rest

3 Damped Driven Oscillator

Combining the systems above results in the following results for the same driving force and damping coefficients.

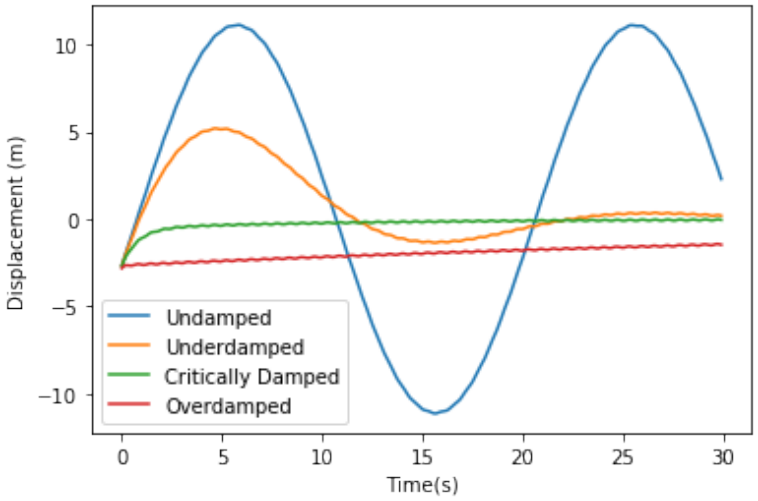

Figure 3: Damped Driven Oscillator Under Different Conditions

4 DC Servo Motor

Upon running the simulation files given, the following graph was observed:

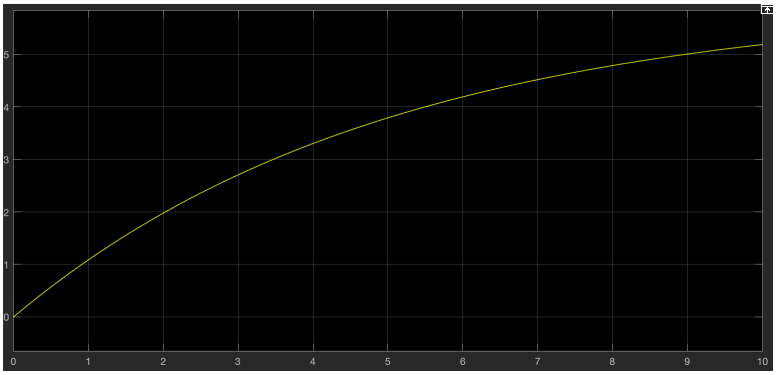

Figure 4: Caption

Assuming the final value of $\dot{\theta}$ is $5.25 rad/s$, the time constant τ will be reached when $\dot{\theta}$is $3.318 rad/s$, which happens at $\tau = 4s$. Given the input is $1V$, the gain in the system K is 5.25.

5 Code Screenshots

```
time=np.arange(0,35,0.1)
xinit=[-3,3.2]
m=10
k=1
c=0
f=0
def d2(x,m,k,c,f):
    return (f-c*x[1]-k*x[0])/m
def differentiator(x,t):
    d1=x[1]
    D2=d2(x,m=m,k=k,c=c,f=f)
    diffs=[d1,D2]
    return diffs
def spring(xinit,m,k,c,f,time):
    return odeint(differentiator,xinit,time)
position=spring(xinit,m,k,c,f,time)[:,0]
c=2.5
positiondamped=spring(xinit,m,k,c,f,time)[:,0]
c=6.32*2
positioncrit=spring(xinit,m,k,c,f,time)[:,0]
c=50
positionover=spring(xinit,m,k,c,f,time)[:,0]
plt.plot(time,position,label='Undamped')
plt.plot(time,positiondamped,label='Under Damped')
plt.plot(time,positioncrit,label='Critically Damped')
plt.plot(time,positionover,label='Over Damped')
xinit=[0,0]
positionrest=spring(xinit,m,k,c,f,time)[:,0]
plt.plot(time,positionrest,color='black')
plt.xlabel('Time (s)')
plt.ylabel('Displacemnt (m)')
plt.title('Position of A spring Mass Damper')
plt.legend()
```

```
def f(k,x):
    restoring_force = -k*x
    return restoring_force
def d(c,v):
    drag_force = -c*v
    return drag_force
def driving(a, f, t):
    driving_force = a*sin(f*t+87*2*pi/360)
    return driving_force
def simulate_oscillations(driving_force_amplitude, driving_frequency, drag_coefficient, k, m, dt, timelim):
    x = -3
    t,v = 0,3.2
    x_list = [0,0,0]
    displacement = []
    time = []
    amplitude_list = []
    time1 = []
    while t <= timelim:
        r = f(k,x) + d(drag_coefficient,v) + driving(driving_force_amplitude, driving_frequency, t)
        a = r / m
        v = v + (a*dt)
        x = x + (v*dt)
        time.append(t)
        displacement.append(x)
        t = t + dt
    return time,displacement
```