Lab 2

Problem 1

a) $\Psi = \begin{bmatrix} R(\theta) & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_1 \\ \sin(\theta) & \cos(\theta) & t_2 \\ 0 & 0 & 1 \end{bmatrix}$

$$d\Psi = \begin{bmatrix} -\dot\theta\sin(\theta) & -\dot\theta\cos(\theta) & \dot t_1 \\ \dot\theta\cos(\theta) & -\dot\theta\sin(\theta) & \dot t_2 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow (0,0)$$

$$\boxed{d\Psi_{(0,0)} = \begin{bmatrix} 0 & -\dot\theta & \dot t_1 \\ \dot\theta & 0 & \dot t_2 \\ 0 & 0 & 0 \end{bmatrix}}$$

b) $\begin{pmatrix} \dot x \\ \dot y \\ \dot\theta \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\dot\varphi_r + \dot\varphi_l)\cos(\theta) \\ \frac{r}{2}(\dot\varphi_r + \dot\varphi_l)\sin(\theta) \\ \frac{r}{w}(\dot\varphi_r - \dot\varphi_l) \end{pmatrix} =$ at $(0,0)$ using $d\Psi$

$$= \boxed{\begin{pmatrix} 0 & -\frac{r}{w}(\dot\varphi_r - \dot\varphi_l) & \frac{r}{2}(\dot\varphi_r + \dot\varphi_l) \\ \frac{r}{w}(\dot\varphi_r - \dot\varphi_l) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}$$

c) $\boxed{X\dot\Omega(\dot\varphi_l, \dot\varphi_r)}$ Rotate the velocity at 0, 0 to the starting pose in our world frame

d) $X_{(\dot\varphi_l, \dot\varphi_r)}(X) = d(L_X)_I(\dot\Omega(\dot\varphi_l, \dot\varphi_r)) =$

$= X\dot\Omega(\dot\varphi_l, \dot\varphi_r)$. It is left invariant because $\dot\Omega$ was the velocity at the identify element, and to get the velocity elsewhere we would simply left multiply this velocity by the pose. This is the same as we would get if we use the rule for determining left invariance

e) $\boxed{\gamma(t) = X_0 \exp(t\,\Omega(\dot{\varphi}_L, \dot{\varphi}_r))}$

Question 2

a) $u = K_p(r-h) + \frac{mg}{4K_T}$

$\ddot{h} = \frac{4k_T}{m}\left(k_p(r-h) + \frac{mg}{4k_T}\right) - g$

$\ddot{h} = \frac{4k_T k_p r}{m} - \frac{4k_T k_p h}{m} + g - g$

$\frac{4k_T k_p r}{m} = \ddot{h} + \frac{4k_T k_p h}{m}$

The higher the proportional control constant, the faster the robot will oscillate around the desired state

b) $u = K_p(r-h) - K_d\dot{h} + \frac{mg}{4K_T}$

$\ddot{h} = \frac{4k_T}{m}\left(k_p(r-h) - k_d\dot{h} + \frac{mg}{4k_T}\right) - g$

$\ddot{h} = \frac{4k_T k_p r}{m} - \frac{4k_T k_p h}{m} - \frac{4k_T k_d \dot{h}}{m}$

$\frac{4k_T K_d}{m}\frac{1}{2w_n} = \zeta$

settling time is $\sim \frac{3}{\zeta w_n}$ so $\zeta w_n = 1$

$\omega_n^2 = \frac{4k_T}{m} k_p \longrightarrow \omega_n = \sqrt{\frac{4k_T}{m} k_p}$

with $k_p = 50$, $\omega_n = 1.274$

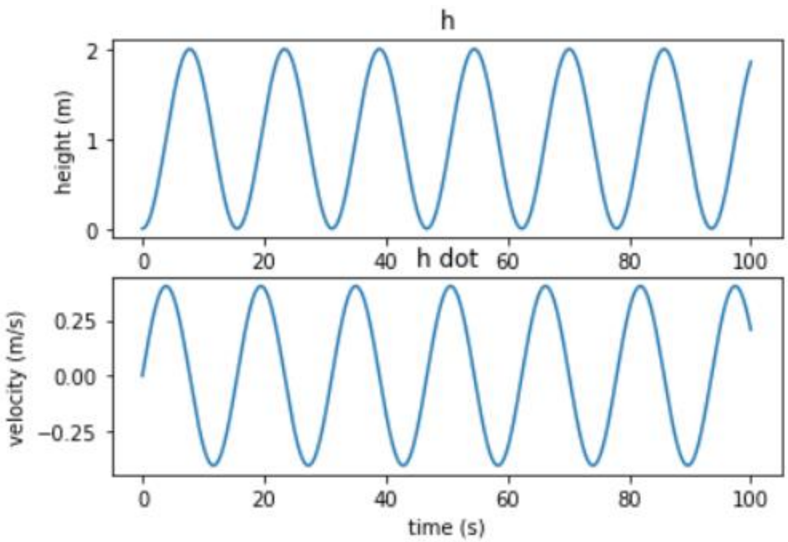
$\frac{1}{1.274} = 0.785 = \zeta$

$0.785 = \frac{4k_T}{m} k_d \cdot \frac{1}{2(1.274)}$

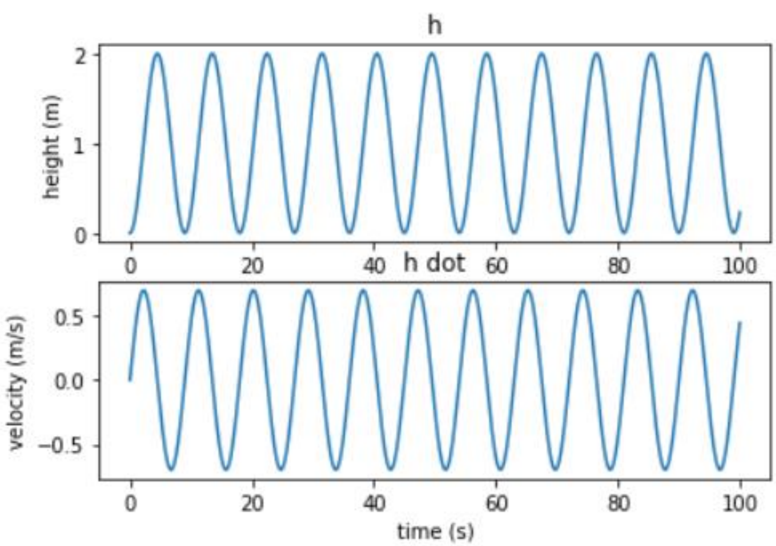
Choose $k_d = 62$

This system will be underdamped because ζ is between 0 and 1 and settling time will be 3 because $\frac{3}{\zeta \omega_n} \sim 3$

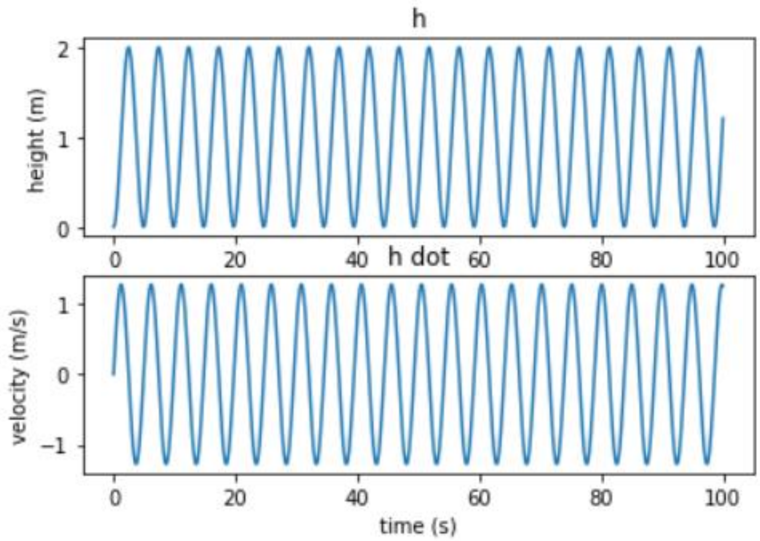
Plots for part a:


h
height (m)
h dot
velocity (m/s)
time (s)

Kp = 15:


h
height (m)
h dot
velocity (m/s)
time (s)

Kp = 50:



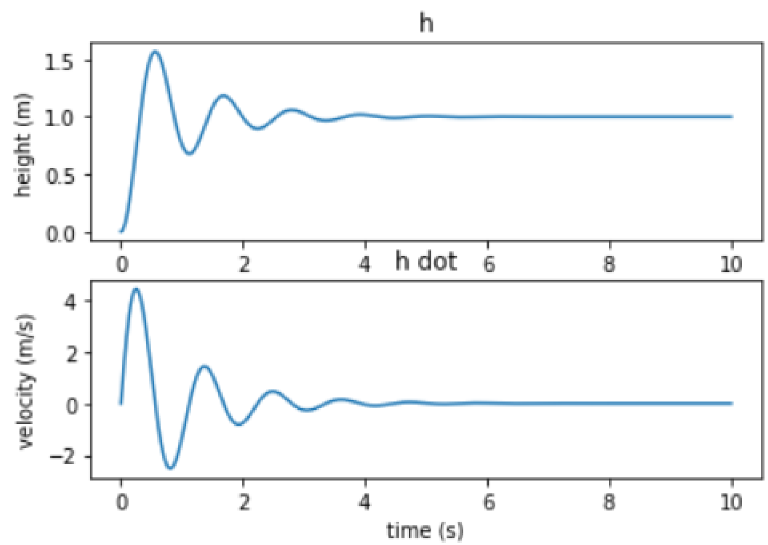
Code for part a:

```
import numpy
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

kt = 5.276 * 10 ** -4
m = 0.065
g = 9.81
Kp = 15

def x_dot(t, x):
    return [x[1], 4*kt*Kp/m - 4*kt*Kp*x[0]/m]

t = numpy.linspace(0, 100, 1000)

solution = solve_ivp(x_dot, [0, t[-1]], [0, 0], t_eval=t, vectorized=True)

t = solution['t']
h = solution['y'][0]
h_dot = solution['y'][1]

fig, ah = plt.subplots(2)
ah[0].plot(t, h)
ah[0].set_xlabel('time (s)')
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
```

```
ah[1].set_title('h dot')

plt.show()
```

The plots show that a higher Kp value results in more frequent oscillations around the reference point. This would indicate that the control responds more quickly to a displacement and continues to overcompensate.

Plots for part b:



Code for part b:

Code for plotting:

```
import numpy
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

kt = 5.276 * 10 ** -4
m = 0.065
g = 9.81
Kp = 1000
Kd = 62

def x_dot(t, x):
    return [x[1], 4*kt*Kp/m - 4*kt*Kp*x[0]/m - 4*kt*Kd*x[1]/m]

t = numpy.linspace(0, 10, 1000)

solution = solve_ivp(x_dot, [0, t[-1]], [0, 0], t_eval=t, vectorized=True)
```

```python
t = solution['t']
h = solution['y'][0]
h_dot = solution['y'][1]

fig, ah = plt.subplots(2)
ah[0].plot(t, h)
ah[0].set_xlabel('time (s)')
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
ah[1].set_title('h dot')

plt.show()
```

Code for finding Kp and Kd:

```python
import math
const = 4*(5.276*10**-4)/0.065
Kp = 1000
wn = math.sqrt(const*Kp)
zeta = 1/wn
print(wn)
print(zeta)
Kd = zeta*2*wn/const
print(Kd)
```

```
5.698042848881737
0.17549885575119076
61.59969673995451
```

The system is underdamped because the damping ratio (zeta) is between zero and one. The plots show this is the case as there is an oscillation around the reference point before the control finally settles.

c) overdamped: $\zeta > 1$

$\zeta \omega_n \sim 1$

choose $K_p = 30$, $K_d = 45$

ζ becomes 1.013 (> 1) ✓

$\omega_n = 0.987$ So overdamped

d) $x_1 = h$, $x_2 = \dot{h}$, $x_3 = \int (r - h)\,dt$

$\dot{x}_1 = x_2$, $\dot{x}_2 = \ddot{h} = \frac{4k_T u}{m} - g$, $\dot{x}_3 = r - x_1$

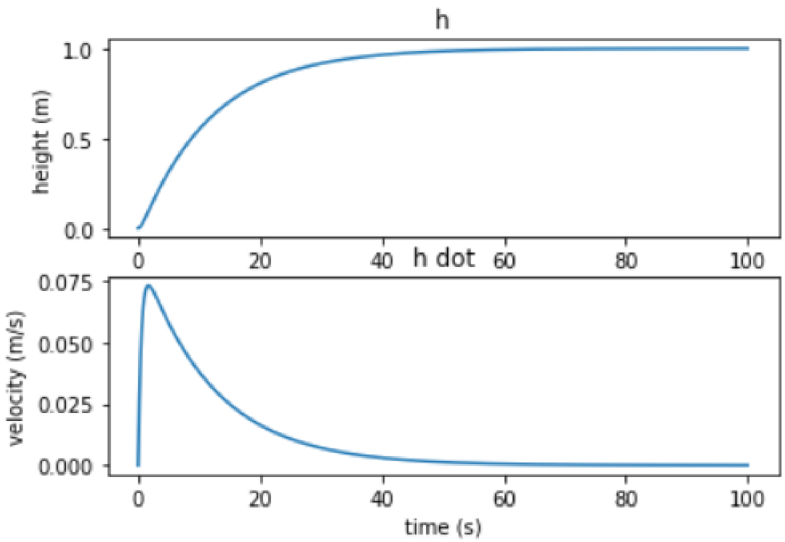
$u = K_p(r - x_1) + K_i x_3 - K_d x_2$

$$\dot{x}_2 = \left(K_p(r - x_1) + K_i x_3 - K_d x_2 + \frac{mg}{4k_T}\right)\frac{4k_T}{m} - g$$

$$\dot{x}_2 = \left(\frac{4k_T}{m}K_p r + \frac{4k_T K_i x_3}{m} - \frac{4k_T K_d x_2}{m} - \frac{4k_T K_p x_1}{m}\right)$$

0.95

Plots for part c:



Code for part c:

Code for plotting (after tuning):

```
import numpy
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

kt = 5.276 * 10 ** -4
m = 0.065
g = 9.81
Kp = 5
Kd = 62

def x_dot(t, x):
    return [x[1], 4*kt*Kp/m - 4*kt*Kp*x[0]/m - 4*kt*Kd*x[1]/m]

t = numpy.linspace(0, 100, 1000)

solution = solve_ivp(x_dot, [0, t[-1]], [0, 0], t_eval=t, vectorized=True)

t = solution['t']
h = solution['y'][0]
h_dot = solution['y'][1]

fig, ah = plt.subplots(2)
ah[0].plot(t, h)
ah[0].set_xlabel('time (s)')
```

```
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
ah[1].set_title('h dot')

plt.show()
```

Code for finding Kp and Kd:

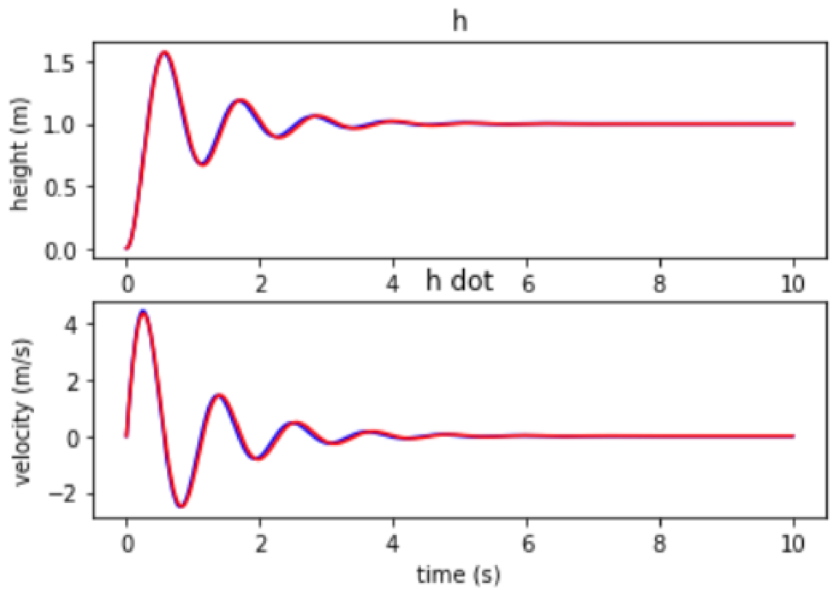
```
[57] import math
     const = 4*(5.276*10**-4)/0.065
     Kp = 5
     wn = math.sqrt(const*Kp)
     zeta = 1/wn
     print(wn)
     print(zeta)
     Kd = zeta*2*wn/const
     print(Kd)

     0.4029124737935791
     2.481928619842934
     61.59969673995451
```

This system is overdamped because the damping ratio (zeta) is greater than one. This is shown in the plots as the control takes a much slower ascent to the reference point.

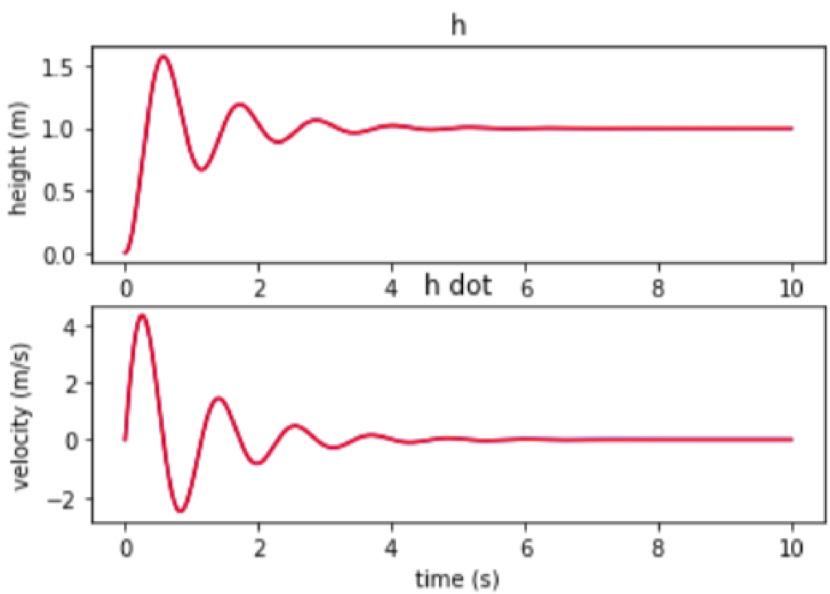
Part d:

Plots:

Before adding Ki:



After adding Ki:



The plots show that before adding the I control, there is some difference between the expected control and the actual control (with uncertainty). After adding in the I control, this uncertainty is eliminated and the two overlap.

Code:
Before adding Ki:

```
import numpy
import matplotlib.pyplot as plt
```

```
from scipy.integrate import solve_ivp

kt = 5.276 * 10 ** -4
m = 0.065
g = 9.81
Kp = 1000
Kd = 62
ucty = 0.95 #uncertainty for actuators

def x_dot_B(t, x):
    return [x[1], 4*kt*Kp/m - 4*kt*Kp*x[0]/m - 4*kt*Kd*x[1]/m]

t = numpy.linspace(0, 10, 1000)

solutionB = solve_ivp(x_dot_B, [0, t[-
1]], [0, 0], t_eval=t, vectorized=True)

t_B = solutionB['t']
h_B = solutionB['y'][0]
h_dot_B = solutionB['y'][1]

def x_dot_D(t, x):
    return [x[1], ucty*4*kt*Kp/m - ucty*4*kt*Kp*x[0]/m - ucty*4*kt*Kd*x[1]
/m - 0.05]

solutionD = solve_ivp(x_dot_D, [0, t[-
1]], [0, 0], t_eval=t, vectorized=True)

t_D = solutionD['t']
h_D = solutionD['y'][0]
h_dot_D = solutionD['y'][1]

overlapping = 1
fig, ah = plt.subplots(2)
ah[0].plot(t, h_B, color='blue', alpha=overlapping)
ah[0].set_xlabel('time (s)')
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot_B, color='blue', alpha=overlapping)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
ah[1].set_title('h dot')

ah[0].plot(t, h_D, color='green', alpha=overlapping)
ah[0].set_xlabel('time (s)')
```

```
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot_D, color='green', alpha=overlapping)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
ah[1].set_title('h dot')

plt.show()
```

After adding Ki:

```
import numpy
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

kt = 5.276 * 10 ** -4
m = 0.065
g = 9.81
Kp = 1000
Kd = 62
Ki = 0
ucty = 0.95 #uncertainty for actuators

def x_dot_D_no_Ki(t, x):
    return [x[1], ucty*4*kt*Kp/m - ucty*4*kt*Kp*x[0]/m - ucty*4*kt*Kd*x[1]
/m - 0.05]

t = numpy.linspace(0, 10, 1000)

solutionNoKi = solve_ivp(x_dot_D_no_Ki, [0, t[-
1]], [0, 0], t_eval=t, vectorized=True)

t_B = solutionNoKi['t']
h_B = solutionNoKi['y'][0]
h_dot_B = solutionNoKi['y'][1]

def x_dot_D_with_Ki(t, x):
    return [x[1], ucty*4*kt*Kp/m + ucty*4*kt*Ki*x[2]/m - ucty*4*kt*Kd*x[1]
/m - ucty*4*kt*Kp*x[0]/m - 0.05, 1 - x[0]]

solutionWKi = solve_ivp(x_dot_D_with_Ki, [0, t[-
1]], [0, 0, 0], t_eval=t, vectorized=True)

t_D = solutionWKi['t']
```

```
h_D = solutionWKi['y'][0]
h_dot_D = solutionWKi['y'][1]

overlapping = 1
fig, ah = plt.subplots(2)
ah[0].plot(t, h_B, color='blue', alpha=overlapping)
ah[0].set_xlabel('time (s)')
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot_B, color='blue', alpha=overlapping)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
ah[1].set_title('h dot')

ah[0].plot(t, h_D, color='red', alpha=overlapping)
ah[0].set_xlabel('time (s)')
ah[0].set_ylabel('height (m)')
ah[0].set_title('h')
ah[1].plot(t, h_dot_D, color='red', alpha=overlapping)
ah[1].set_xlabel('time (s)')
ah[1].set_ylabel('velocity (m/s)')
ah[1].set_title('h dot')

plt.show()
```

10/12/2022

Problem 3

a) $$\dot{x} = \begin{pmatrix} x_2 \\ -\frac{g\sin(x_1)}{\ell} - \frac{\mu x_2}{m\ell^2} + \frac{\tau}{m\ell^2} \end{pmatrix}$$

b) $$\frac{df}{dx} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell}\cos(x_1) & \frac{-\mu}{m\ell^2} \end{bmatrix}, \quad \frac{df}{d\tau} = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}$$

$(\pi, 0, 0) \Rightarrow$

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{\mu}{m\ell^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix} \tau$$

w/ no torque

$$f(\lambda) = \det\left(\begin{bmatrix} 0 & 1 \\ g/\ell & -\frac{\mu}{m\ell^2} \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) =$$

$$= \det\begin{pmatrix} -\lambda & 1 \\ g/\ell & -\frac{\mu}{m\ell^2} - \lambda \end{pmatrix} = \lambda^2 + \frac{\mu}{m\ell^2}\lambda - \frac{g}{\ell}$$

$$\lambda = \frac{-\frac{\mu}{m\ell^2} \pm \sqrt{\left(\frac{\mu}{m\ell^2}\right)^2 - 4\left(-\frac{g}{\ell}\right)}}{2}$$

One λ will always be positive, so it is unstable

$$c(x^*) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

c) $$\dot{x} = c(x) = \begin{pmatrix} x_2 \\ \frac{-g\sin(x_1)}{l} - \frac{\mu x_2}{ml^2} + \frac{k_p \sin(x_1) + k_d x_2}{ml^2} \end{pmatrix}$$

d) $$\frac{dc}{dx} = \begin{bmatrix} 0 & 1 \\ \frac{-g}{l}\cos(x_1) + \frac{k_p}{ml^2}\cos(x_1) & \frac{-\mu}{ml^2} + \frac{k_d}{ml^2} \end{bmatrix}$$

at $x^* = (\pi, 0) \rightarrow \begin{bmatrix} 0 & 1 \\ \frac{g}{l} - \frac{k_p}{ml^2} & \frac{-\mu + k_d}{ml^2} \end{bmatrix}$

So it is a stationary point

$$f(\lambda) = \det\left(\begin{bmatrix} 0 & 1 \\ \frac{g}{l} - \frac{k_p}{ml^2} & \frac{-\mu + k_d}{ml^2} \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right)$$

$$= \det\begin{pmatrix} -\lambda & 1 \\ \frac{g}{l} - \frac{k_p}{ml^2} & \frac{-\mu + k_d}{ml^2} - \lambda \end{pmatrix}$$

$$= \lambda^2 + \frac{\mu - k_d}{ml^2}\lambda - \frac{g}{l} + \frac{k_p}{ml^2}$$

$$\lambda = \frac{\frac{-\mu + k_d}{ml^2} \pm \sqrt{\left(\frac{\mu - k_d}{ml^2}\right)^2 - 4\left(\frac{g}{l} + \frac{k_p}{ml^2}\right)}}{2}$$

$$\frac{-\mu - k_d}{ml^2} > 0 \quad \text{and} \quad \left(\frac{\mu - k_d}{ml^2}\right)^2 < 4\left(\frac{g}{l} + \frac{k_p}{ml^2}\right)$$

$$\boxed{k_d < \mu}$$

$$\left(\frac{\frac{\mu - k_d}{ml^2}}{4}\right)^2 < \frac{g}{l} + \frac{k_p}{ml^2}$$

$$\frac{k_p}{ml^2} > \left(\frac{\frac{\mu - k_d}{ml^2}}{4}\right)^2 - \frac{g}{l}$$

$$\boxed{k_p > \frac{(\mu - k_d)^2}{4ml^2} - gml}$$

$$\frac{-\mu + k_d}{ml^2} \pm \sqrt{\left(\frac{\mu - k_d}{ml^2}\right)^2 - 4\left(\frac{g}{l} + \frac{k_p}{ml^2}\right)} < 0$$

$$\left(\frac{\mu - k_d}{ml^2}\right)^2 - 4\left(\frac{g}{l} + \frac{k_p}{ml^2}\right) < \left(\frac{\mu + k_d}{ml^2}\right)^2$$

$$-4\left(\frac{g}{l} + \frac{k_p}{ml^2}\right) < \left(\frac{\mu + k_d}{ml^2}\right)^2 - \left(\frac{\mu - k_d}{ml^2}\right)^2$$

e) $\nabla V(x) = \begin{bmatrix} mgl\sin(x_1) + \alpha mgl\sin(2x_1) \\ ml^2 x_2 \end{bmatrix}$

$H(V(x)) = \begin{bmatrix} -mgl\cos(x_1) + 2\alpha mgl\cos(2x_1) & 0 \\ 0 & ml^2 \end{bmatrix}$

with $x^* = (\pi, 0) \longrightarrow V(x^*) = 0$ ✓

$\nabla V(x^*) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ✓

$H(V(x^*)) = \begin{bmatrix} mgl + 2\alpha mgl & 0 \\ 0 & ml^2 \end{bmatrix}$

$mgl + 2\alpha mgl > 0$

$2\alpha mgl > -mgl$

$2\alpha > -1$

$\boxed{\alpha > -\frac{1}{2}}$

f) $\nabla V(x) \cdot f(x, \tau) =$

$= \begin{bmatrix} mgl\sin(x_1) + \alpha mgl\sin(2x_1) \\ ml^2 x_2 \end{bmatrix} \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) - \frac{\mu x_2}{ml^2} + \frac{\tau}{ml^2} \end{bmatrix}$

$= x_2\left[\left(\cancel{mgl\sin(x_1)} + \alpha mgl\sin(2x_1)\right) + \left(-\cancel{mgl\sin(x_1)} - \mu x_2 + \tau\right)\right]$

$= x_2\left[\alpha mgl\sin(2x_1) - \mu x_2 + \tau\right] =$

$\boxed{-\mu x_2^2 + (\alpha mgl\sin(2x_1) + \tau)x_2}$

when $c(x)$ or $\dot{x}$ goes to 0
will be a repeating / cyclic set

g) $c(x) = \begin{pmatrix} x_2 \\ -\frac{g\sin(x_1)}{l} - \frac{\mu x_2}{ml^2} - \frac{2\alpha g \sin(x_1)\cos(x_1)}{l} \end{pmatrix}$

h) $\frac{dc}{dx} = \cancel{\begin{bmatrix} 0 & 1 \\ \frac{-g\cos(x_1) + 2\alpha g\cos(x_1)}{l} & -\frac{\mu}{ml^2} \end{bmatrix}}$

x_2 must be 0

x_1 : $\sin(x_1) - \sin(x_1)\cos(x_1)$
when x_1 is $n\pi$, where n
is an integer

So S is the set of points x in
the form $x = (x_1, x_2)$ where x_1
is $n\pi$ with n being an integer
(e.g. $-1, 0, 1, 2, \ldots$) and x_2 is 0

i) $\dot{V} = -\mu x_2^2 + (2\alpha mgl\sin(x_1)\cos(x_1) + \tau)x_2$

with $\tau(x) = -2\alpha mgl\sin(x_1)\cos(x_1)$

$\downarrow$

$\dot{V} = -\mu x_2^2$

Since μ is a damping parameter due to
friction, it will not be negative, and
x_2 will be positive or 0 because it is
squared, so the negative sign in front
will ensure that μ times x_2^2 will be ≤ 0

j) $-\mu x_2^2 = 0 \longrightarrow x_2 = 0$

So I will be the set of points where x_2 is 0, which would indicate that our set S will be a subset of I

k)

Plots for part k:

V(x):

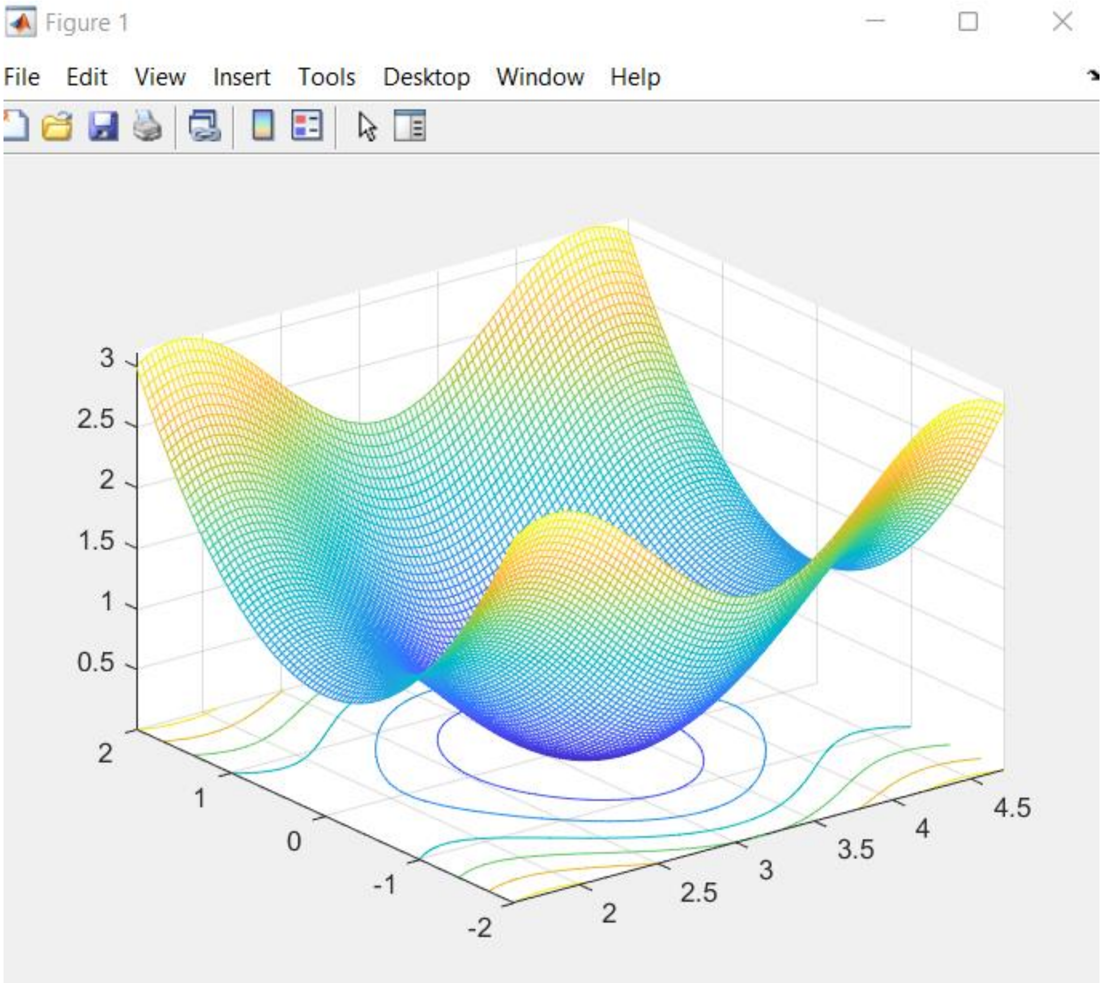
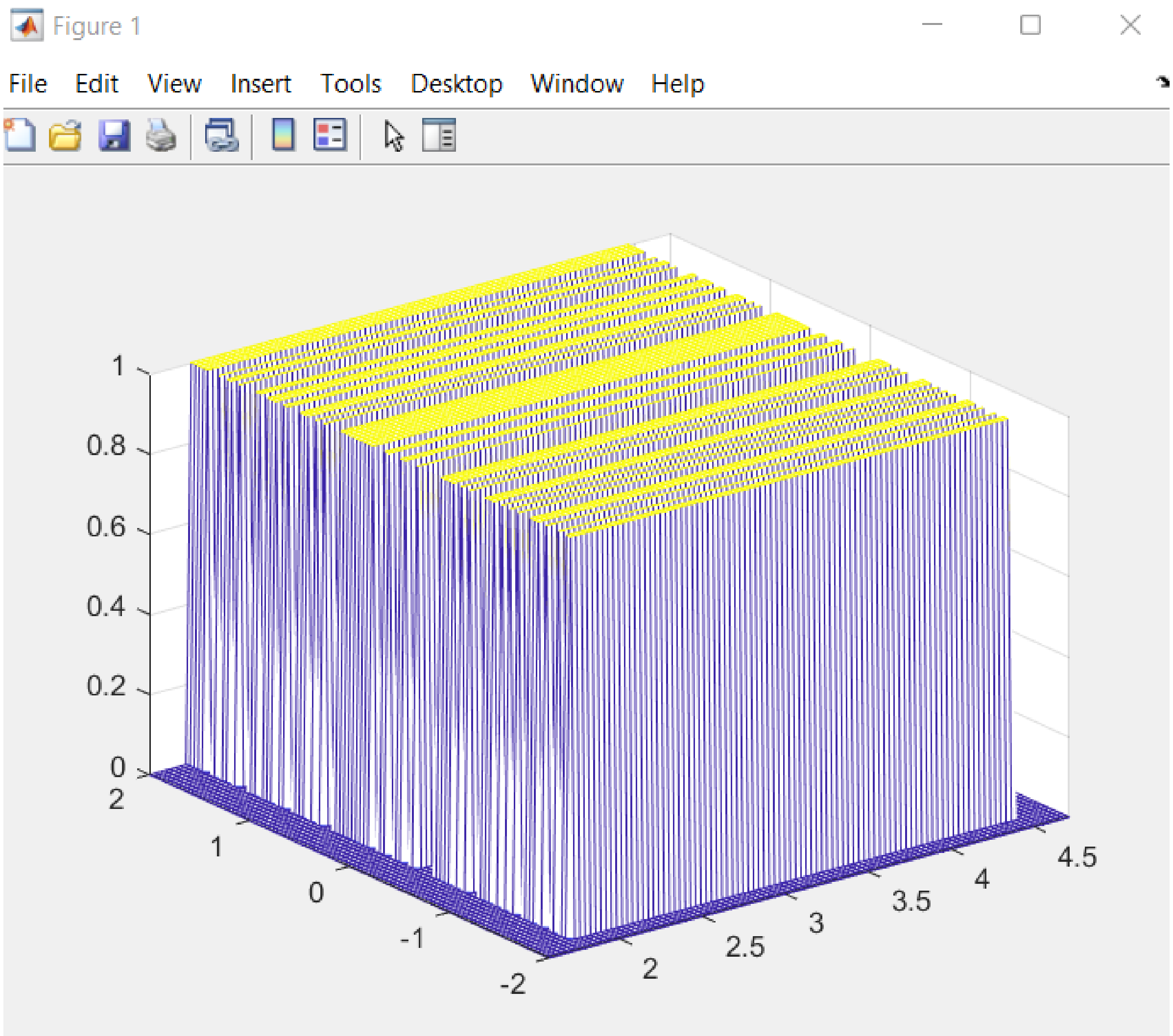
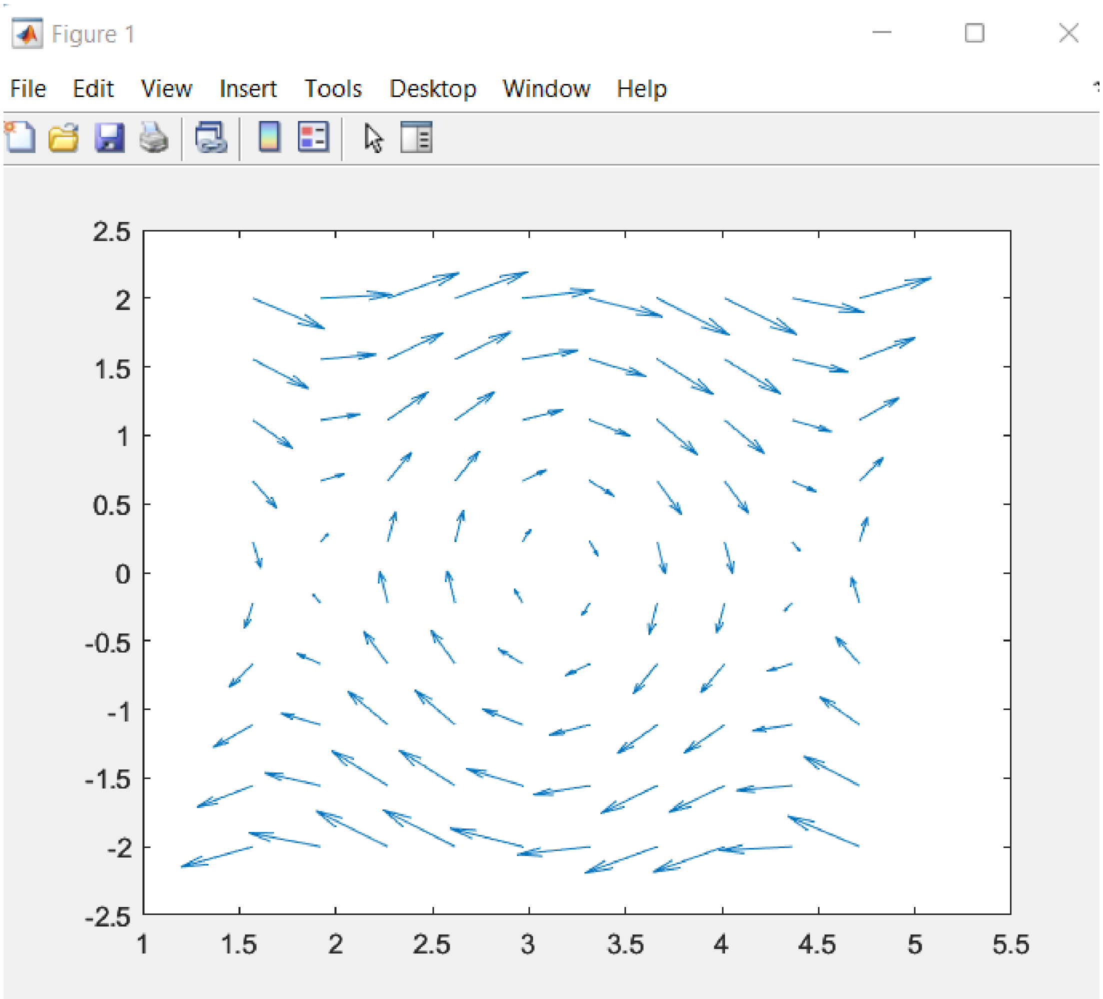


Figure 1
File Edit View Insert Tools Desktop Window Help
3
2.5
2
1.5
1
0.5
2
1
0
-1
-2
2
2.5
3
3.5
4
4.5

Indicator function:



Phase portrait:



Level Sets:

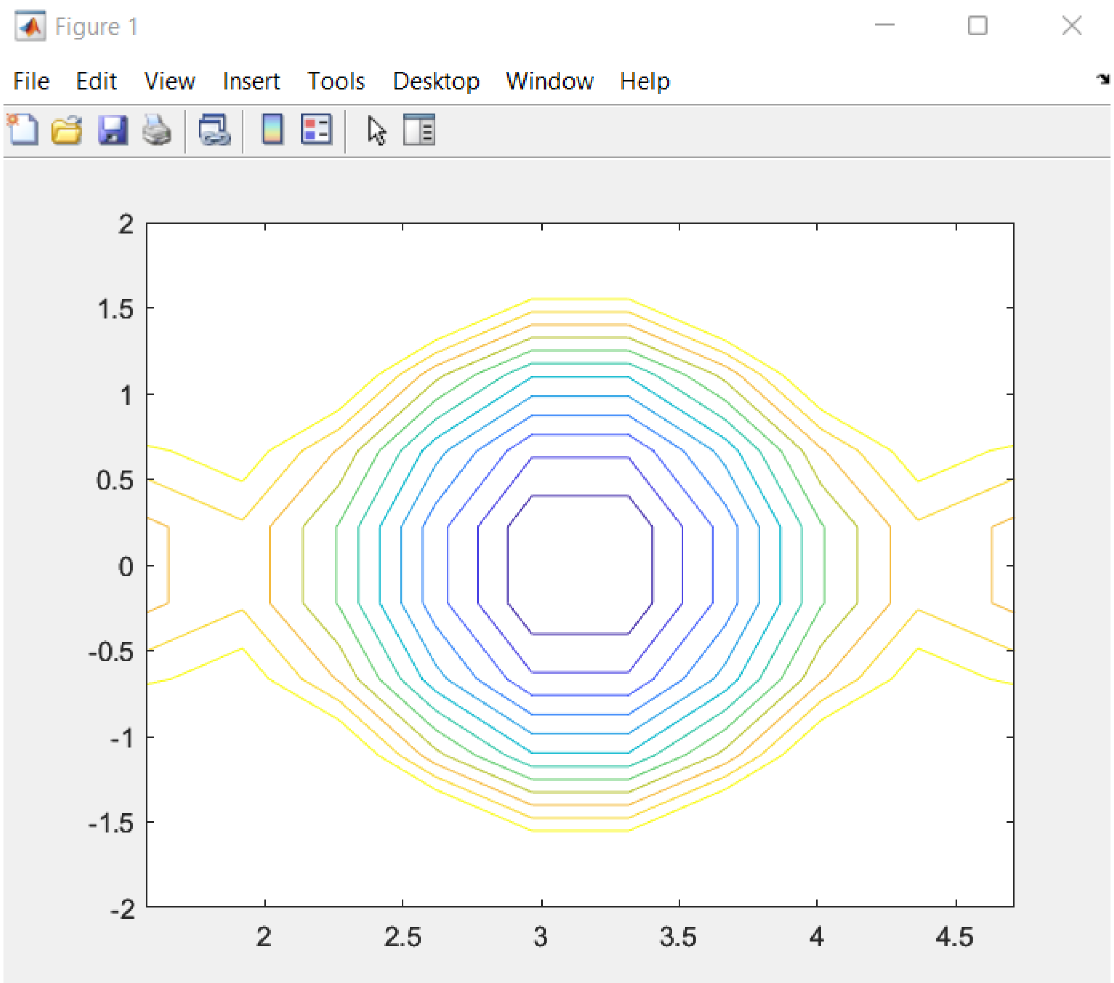


Figure 1
File Edit View Insert Tools Desktop Window Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
2
2.5
3
3.5
4
4.5