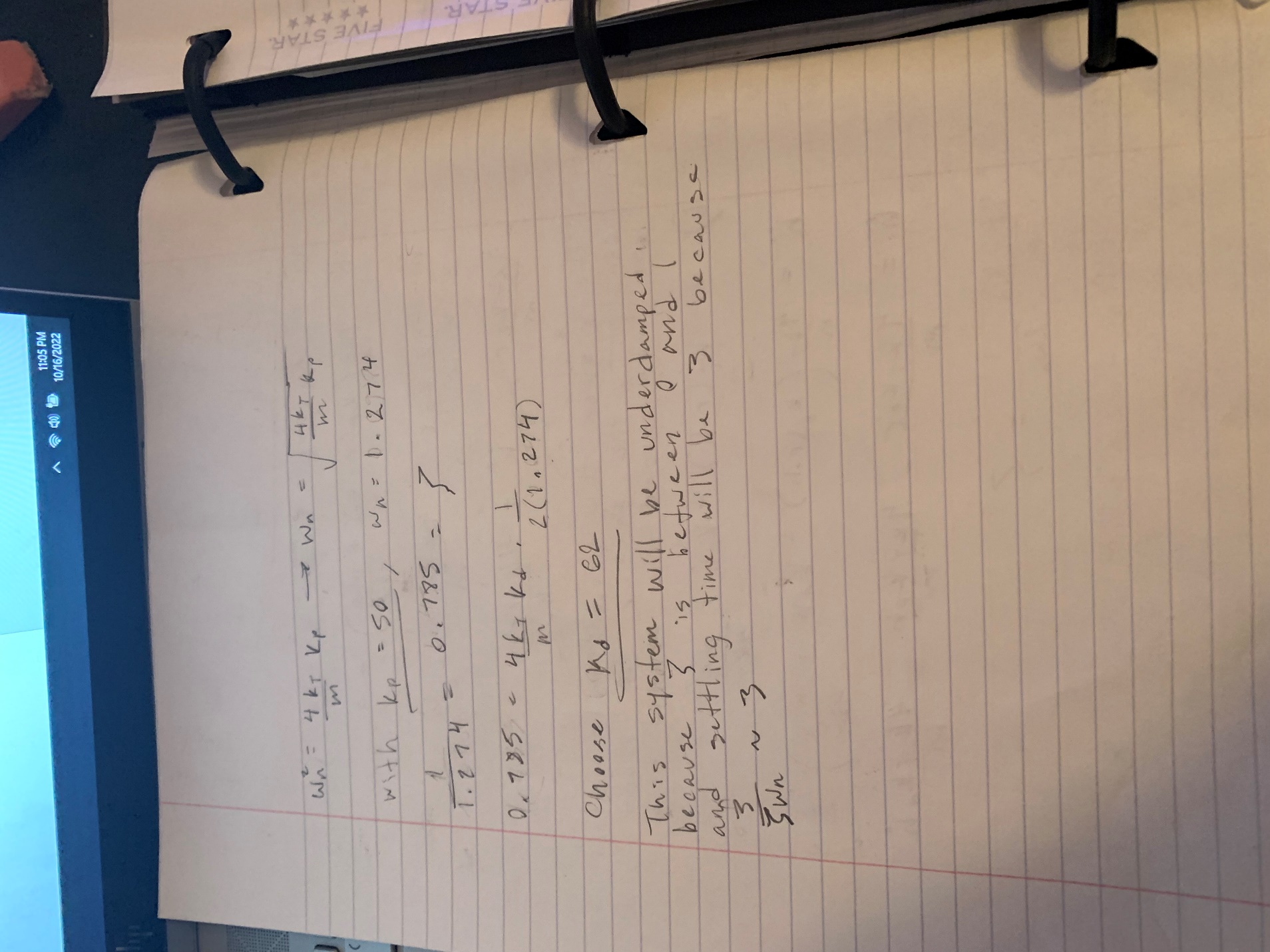
A picture containing diagram

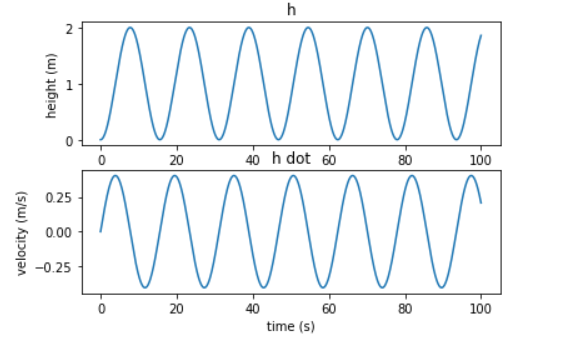
Description automatically generated

A picture containing text

Description automatically generatedA notebook with writing on it

Description automatically generated with low confidence

Plots for part a:



Kp = 15:

Histogram

Description automatically generated with medium confidence

Kp = 50:

Histogram

Description automatically generated with medium confidence

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:

Chart, line chart

Description automatically generated

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)

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:

Graphical user interface, text, application

Description automatically generated

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.

A piece of paper with writing on it

Description automatically generated

Plots for part c:

Diagram

Description automatically generated with medium confidence

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:

Text

Description automatically generated

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:

Chart, line chart

Description automatically generated

After adding Ki:

Chart, line chart

Description automatically generated

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()

A picture containing text

Description automatically generated

Diagram

Description automatically generated with low confidence

Diagram

Description automatically generated

Text

Description automatically generated with medium confidenceA picture containing diagram

Description automatically generated

A picture containing diagram

Description automatically generated

Plots for part k:

V(x):

Chart, surface chart

Description automatically generated

Indicator function:

Graphical user interface, chart

Description automatically generated

Phase portrait:

Chart

Description automatically generated

Level Sets:

Chart, radar chart

Description automatically generated