PID Control of Fourth-Order Analog Plant

Digital Control System Design

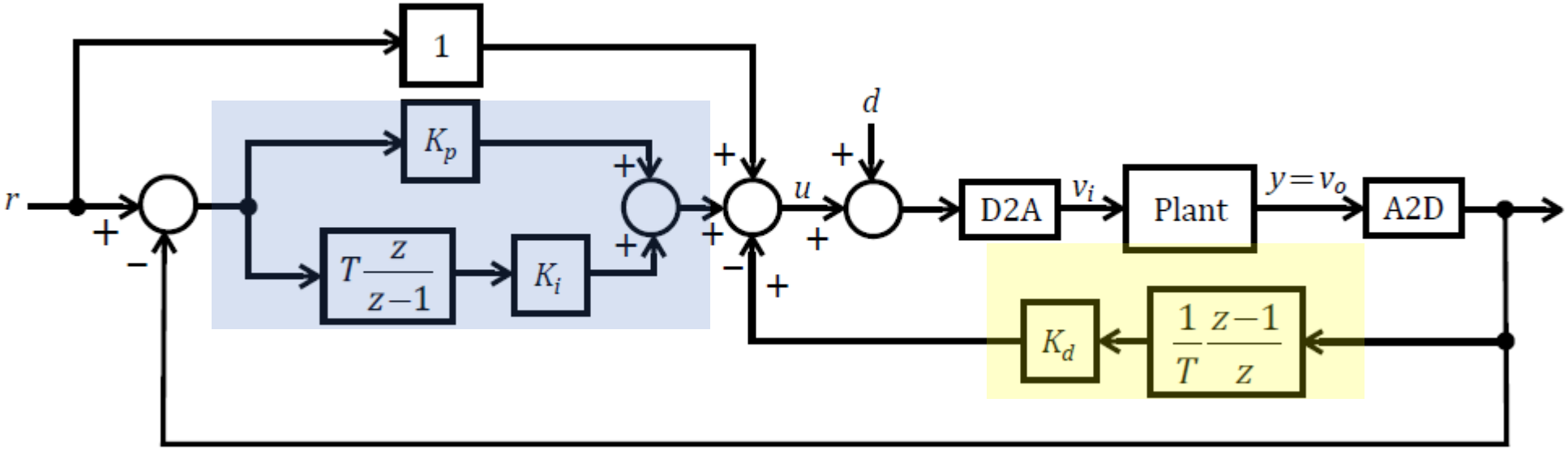
Professor Fuller

Project Assignment

Take note of “design objectives” and specific submission instructions “what to submit”.

What we're doing:

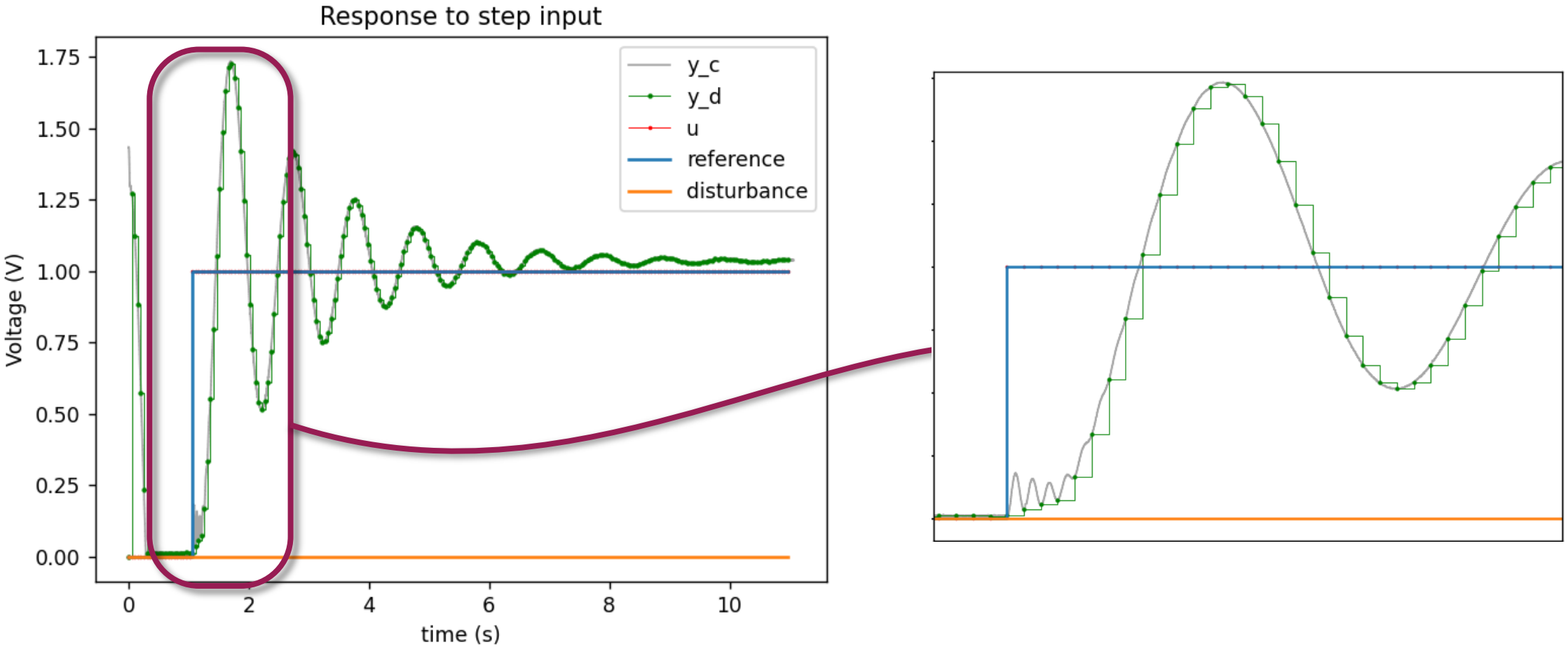
You are tasked with controlling a plant. To start, this plant was represented by a single vibration mode which you implemented electrically so that you could test hardware-in-the-loop (HIL). In previous labs you determined that only feedforward and derivative control were insufficient to control the plant output to within specifications. In order to add disturbance rejection and achieve better performance, you then implemented a PID design of your making. Now to increase the realism of the plant which we are modeling, in this next lab we add an additional vibration mode, making the plant 4th order. You will construct the circuit for this, building off your 2nd order plant, and you will perform a new PID design for the 4th order system. You will implement this block diagram on the microcontroller using CircuitPython and MicroPython!



Block diagram showing disturbance input "d" affecting the control signal "u". The disturbance affects "u" before it is actuated by the D2A.

- PID+FF control. PI controller (blue) + derivative (yellow) + with feed-forward (FF). This particular feed-forward control element is represented by the unity-gain block at top, your feedforward gain may be different.

Open loop reference step response



Assignment overview

1. Design your PID controller in software to meet the performance and stability margin objectives below.
2. Construct the circuit. Get hardware components per Canvas announcement
3. Use the provided *pyboard.py* and *yourName_lab3.py* to implement the difference equation of your discrete controller design.
4. Perform experiments demonstrating your design following the “what to submit” and “design objectives” slides below.
5. Analyze your saved data and generate your own plots displaying the output, inputs, and control signal over time. Generate high quality figures.
6. Submit a report fulfilling the instructions on the next slide.

What to Submit:

1.

a) An "executive summary" that describes what you did, the results you obtained, and the problems/difficulties you ran into in your work to complete this project.

b) A description (accompanied by annotated figures) of the procedure you used to determine one combination of K_{FF}, K_P, K_I, and K_D values to simultaneously satisfy (or to nearly satisfy) all Performance and Stability Margin Objectives. You must define the plant component values in your simulation as instructed above.

c) Annotated figures that show the extent to which the one combination of K_{FF}, K_P, K_I, and K_D values you determined satisfies the Performance Objectives, when implemented [virtual]-hardware-in-the-loop with your analog plant. You must define the plant component values in your simulation as instructed below under "Implementation Instructions".

d) An annotated figure, or figures, that show the extent to which the one combination of K_{FF}, K_P, K_I,and K_D values you determined, when implemented with the theoretical model of your analog plant, satisfies the Stability Margin Objectives**.

e) A discussion (accompanied by annotated figures) of the extent to which the poles and zeros of your closed-loop system model's r-to-y transfer function predicts the y response of your hardware-in-the-loop closed-loop system to a unit step r input. ***

f) A discussion (accompanied by annotated figures) of the extent to which the poles and zeros of your closed-loop system model's d-to-y transfer function predict the y response of your hardware-in-the-loop closed-loop system to a unit step d input. ***

g) A brief discussion of the discrete controller's timing that you achieved in your HIL tests, and one plot showing the sampling intervals during a test.

2. Your detailed answer (accompanied by annotated figures) to the Question about doubling the sampling rate.

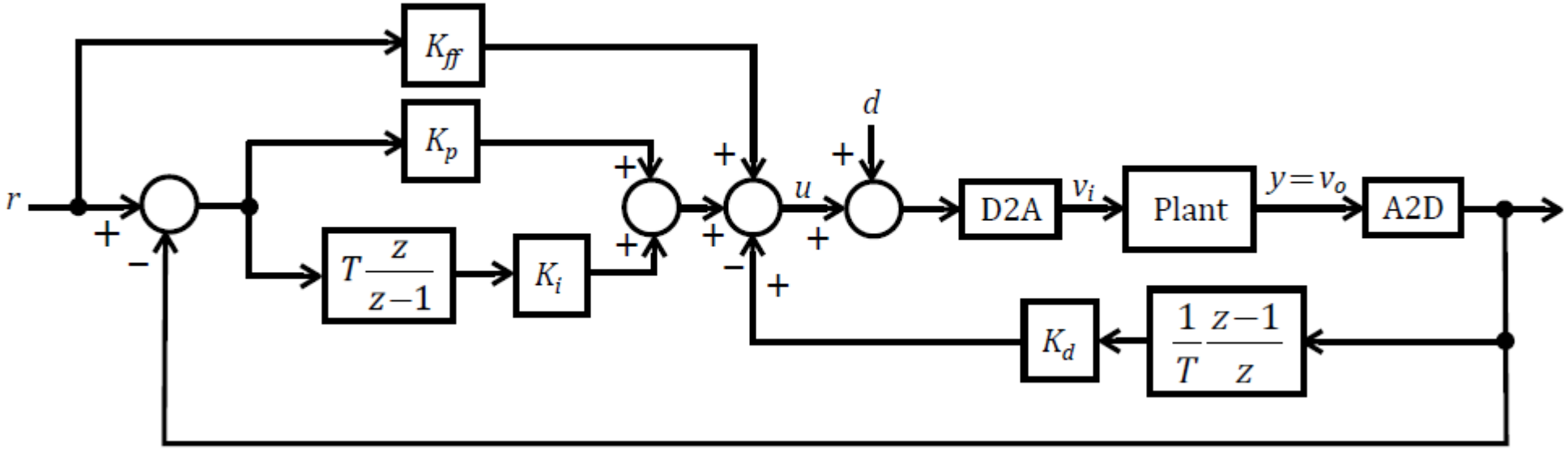
3. Your code and saved data files.

4. Photo(s) of your hardware sufficient to understand your wiring

** In practice you would likely also be required to show the extent to which the one combination of K_{FF}, K_P, K_I,and K_D values you determined satisfies each of these same Stability Margin Objectives when implemented hardware-in-the-loop with your analog plant (as was done in HW).

*** note that for this lab, you have the more realistic scenario that you do not know the exact values of the components, only that they are within +/-5% of their nominal values.

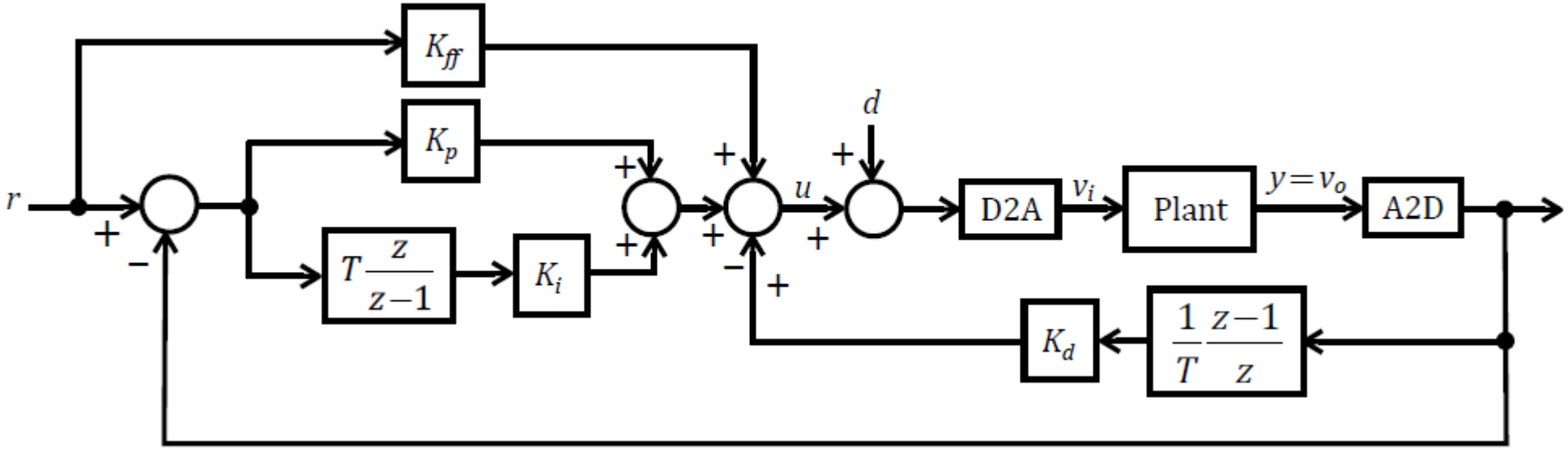
Design Objectives (same as previous lab):



For the closed-loop system above, where "Plant" is the fourth-order analog plant with the component values assigned to you, and using 50 msec for the sampling period, determine one combination of K_{ff}, K_p, K_i and K_d values to simultaneously meet all of these Performance Objectives

1. Peak overshoot of the *y* response to a unit step *r* input ≤ 2 percent.
2. Time for the *y* response to a unit step *r* input to settle to within 2% of its steady-state value ≤ 1 sec.
3. Time for the *y* response to a unit step *d* input to settle to within 0.1 volts ≤ 30 sec.
4. Peak magnitude of the control signal *u* in response to a unit step *r* input as small as possible.
5. Zero steady-state tracking error (i.e., *y* = *r* in the steady state) in response to a unit step *r* input.
6. Zero steady-state tracking error (i.e., *y* = *r* in the steady state) in response to a unit step *d* input.

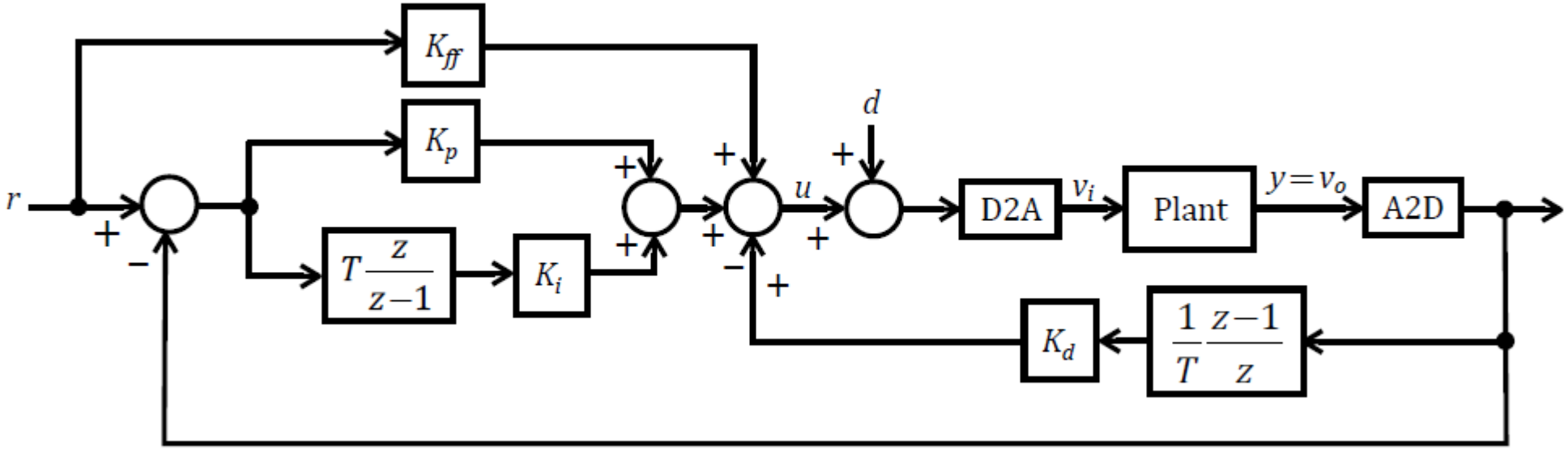
Design Objectives



and, for the loop-breaking point labeled "*u*" in the above diagram, to simultaneously meet all of these Stability Margin Objectives:

1. Positive gain margin ≥ 10 dB.
2. Negative gain margin = −∞ dB.
3. Phase margin ≥ 50 degrees.

Design Objectives



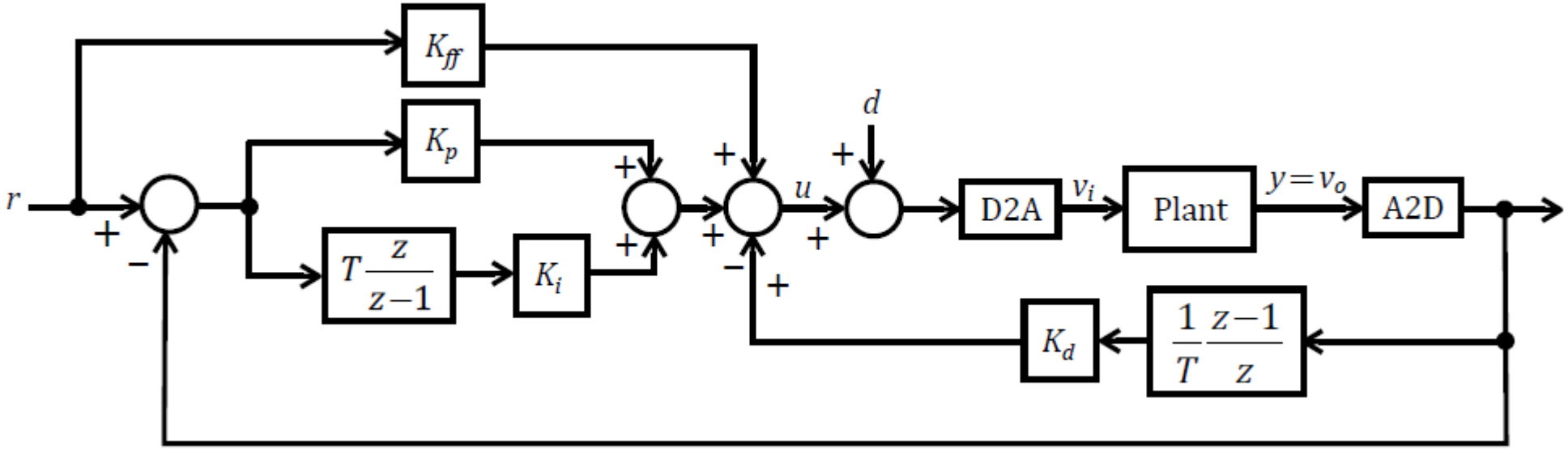
Question about doubling sampling rate

The above closed-loop system implements its discrete-time integral and derivative control actions via the standard backwards-difference approximations to the corresponding continuous-time integral and derivative control actions. So **doubling the sampling rate (with no change in the controller gains) will improve the closed-loop system's performance, right?**

Explain, in detail, why it is that, in this case, doubling the sampling rate *worsens* the closed-loop system's performance.

Hint: Carefully compare the poles and zeros of Gzas(z) and your closed loop system for the two sampling rates. It is also helpful to consider the Bode plot of the plant provided above. Since you have access to all the signals in the simulated system (such as *u*'s, x_1, x_2, etc), it is useful to carefully evaluate these signals while doing this "detective work" to discover what has happened to your system.

A note on the meaning of the design Objectives



The instruction to design "Peak magnitude of the control signal *u* in response to a unit step *r* input as small as possible." is there to minimize the effort that the controller has to exert on your system. Realistic actuators can be damaged by high magnitude or rapidly changing input signals.
Taking this goal one step further, it is certainly possible to quantify this 'controller effort' and even to minimize *u* by utilizing a *cost* function which takes this quantity into account. This is the subject of subsequent course content such as *LQR* design.

For meeting this lab performance objective, it is sufficient to keep this control signal *u* in mind and to avoid controller designs which would result in excessive effort, such as saturation of controller outputs against the voltage limitations of the DAC. A design which required less controller effort in order to achieve the same performance and stability margin objectives would be superior to a design which unnecessarily 'abused' the actuator (the DAC) by exerting large magnitude control *u*.

Feedforward design

Spring stiffness $K_{spring} = \frac{10N}{m}$

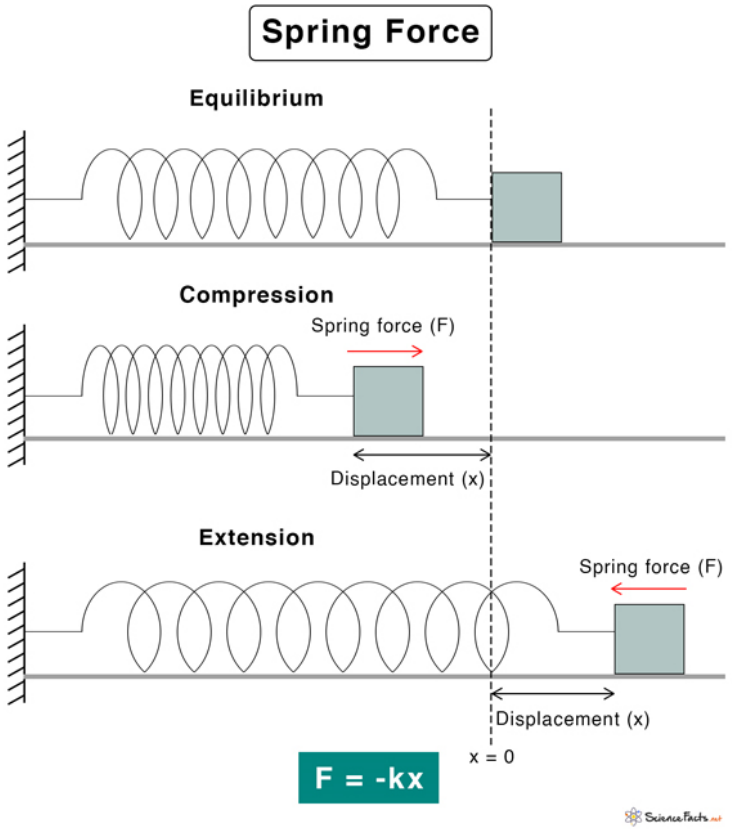
Desired position $r = 5m$

What is the force to apply? $u = K_{spring} r = \frac{10N}{m} 5m = 50N$

Generalize: we inverted the plant to determine the control effort ***u*** as a function of the reference ***r***; we didn't have to do anything fancy like closed loop control – we precomputed this in open loop (feedforward) using our knowledge of the plant.

$$u_{ff} = k_{ff}\ r$$

Analogy: spring



Feedforward design

```
dcgain = ctm.dcgain(plant[0,0])
print('DC gain of plant r input =', dcgain)
Kff = 1/dcgain
```

Spring stiffness $K_{spring} = \frac{10N}{m}$

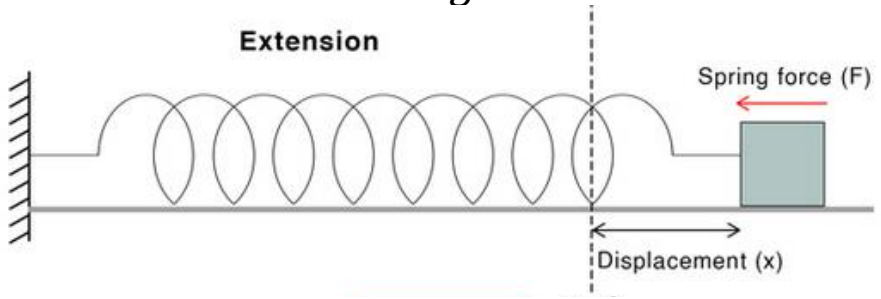
Desired position $r = 5m$

What is the force to apply? $u = K_{spring} r = \frac{10N}{m} 5m = 50N$

Generalize: we inverted the plant to determine the control effort ***u*** as a function of the reference ***r***; we didn't have to do anything fancy like closed loop control – we precomputed this in open loop (feedforward) using our knowledge of the plant.

$$u_{ff} = k_{ff}\ r$$

$$k_{ff} = \frac{1}{DCgain}$$



$$u_{ff} = k_{ff}\, r = Kx$$

$DCgain$ is a measure of the plant output in static steady state to a constant input. In the case of the spring mass it is just the inverse of the stiffness (units of $\frac{meter}{newton}$) a.k.a the compliance.

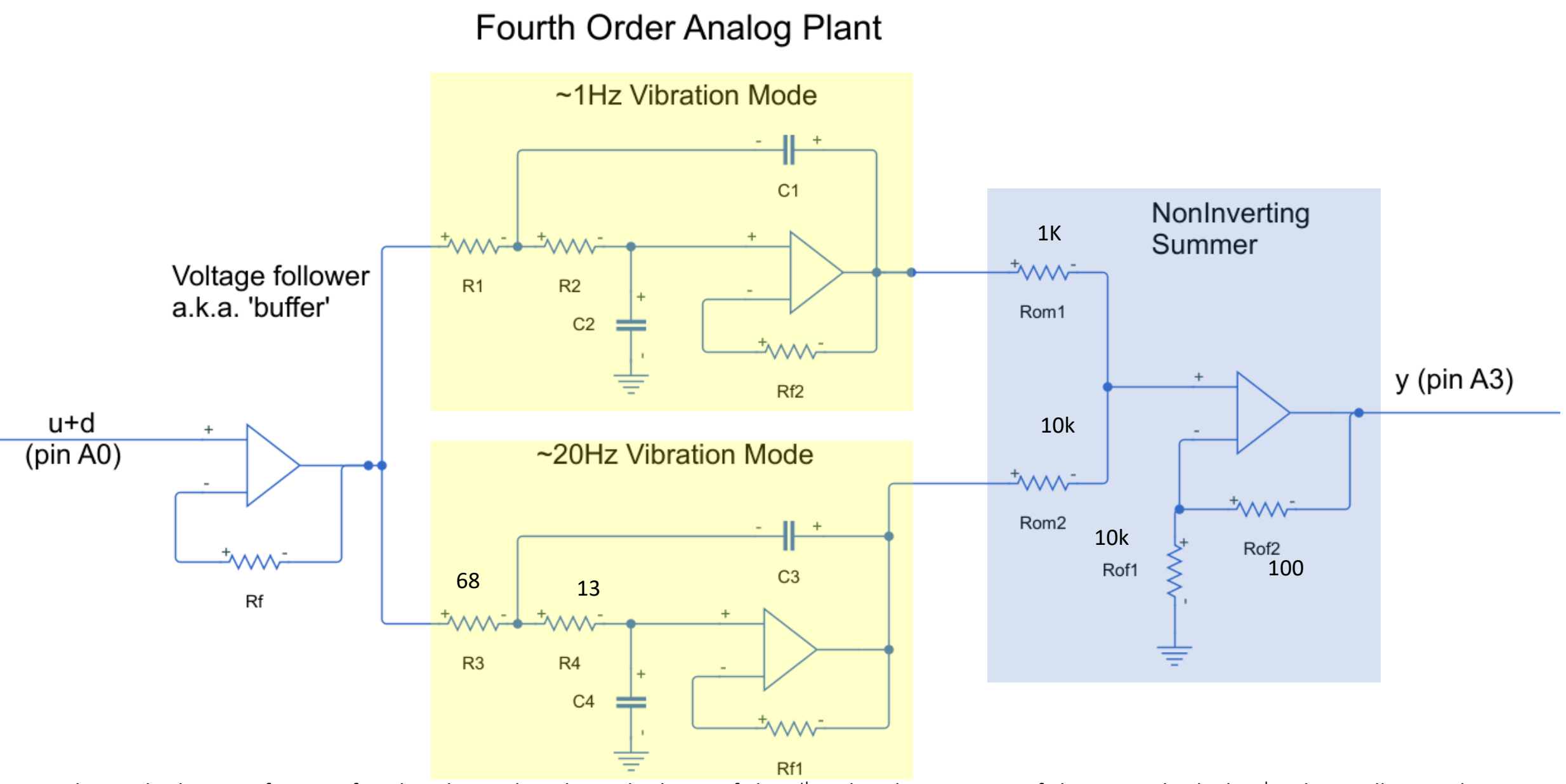
Feedforward design

```
dcgain = ctm.dcgain(plant[0,0])
print('DC gain of plant r input =', dcgain)
Kff = 1/dcgain
```

How to determine?

1. Measure/Model your plant really well
2. Empirically determine

$DCgain$ is a measure of the plant output in static steady state to a constant input. In the case of the spring mass it is just the inverse of the stiffness (units of $\frac{meter}{newton}$) a.k.a the compliance.



Electrical schematic for your fourth-order analog plant. The heart of the 4th order plant consists of the two individual 2nd order oscillators. The inverting summer circuit (blue) has a transfer function which achieves the addition operation of the outputs from each of the vibration modes, weighted according to the R_{om2} and R_{om1} resistors. Two input signals, v_D and v_U are summed together in the software before the DAC. The 2nd order oscillators each have their own characteristics as shown in the expanded form of the plant transfer function below.

Fourth-Order Analog Plant

Constituents of the transfer function, in Laplace domain

$$V_y(s) = \left[1 + \frac{R_{of2}}{R_{of1}}\right]\left[\left(\frac{R_{om1}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n1}^2}{s^2 + 2\zeta_1\omega_{n1}s + \omega_{n1}^2}\right) + \left(\frac{R_{om2}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n2}^2}{s^2 + 2\zeta_2\omega_{n2}s + \omega_{n2}^2}\right)\right][\,V_u(s) + V_d(s)]$$

y (output)

u (input)

d (input)

Fourth-Order Analog Plant

Constituents of the transfer function, in Laplace domain

$$V_y(s) = \left[1 + \frac{R_{of2}}{R_{of1}}\right]\left[\left(\frac{R_{om1}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n1}^2}{s^2 + 2\zeta_1\omega_{n1}s + \omega_{n1}^2}\right) + \left(\frac{R_{om2}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n2}^2}{s^2 + 2\zeta_2\omega_{n2}s + \omega_{n2}^2}\right)\right][\,V_u(s) + V_d(s)]$$

$$summer\ gain\ \approx 1$$

Fourth-Order Analog Plant

Constituents of the transfer function, in Laplace domain

$$V_y(s) = \left[1 + \frac{R_{of2}}{R_{of1}}\right]\left[\left(\frac{R_{om1}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n1}^2}{s^2 + 2\zeta_1\omega_{n1}s + \omega_{n1}^2}\right) + \left(\frac{R_{om2}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n2}^2}{s^2 + 2\zeta_2\omega_{n2}s + \omega_{n2}^2}\right)\right][\,V_u(s) + V_d(s)]$$

$Vibration\ mode$
$1\ Hz$

$Vibration\ mode$
$20\ Hz$

Fourth-Order Analog Plant

Constituents of the transfer function, in Laplace domain

$$V_y(s) = \left[1 + \frac{R_{of2}}{R_{of1}}\right]\left[\left(\frac{R_{om1}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n1}^2}{s^2 + 2\zeta_1\omega_{n1}s + \omega_{n1}^2}\right) + \left(\frac{R_{om2}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n2}^2}{s^2 + 2\zeta_2\omega_{n2}s + \omega_{n2}^2}\right)\right][\,V_u(s) + V_d(s)]$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{\underline{residues}}\quad \underbrace{\qquad\qquad\qquad\qquad}_{influence\ factors}$$

Fourth-Order Analog Plant

$$V_y(s) = \left[1 + \frac{R_{of2}}{R_{of1}}\right]\left[\left(\frac{R_{om1}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n1}^2}{s^2 + 2\zeta_1\omega_{n1}s + \omega_{n1}^2}\right) + \left(\frac{R_{om2}}{R_{om1}+R_{om2}}\right)\left(\frac{\omega_{n2}^2}{s^2 + 2\zeta_2\omega_{n2}s + \omega_{n2}^2}\right)\right][\,V_u(s) + V_d(s)]$$

influence factors

residues

$$\left.\begin{aligned} R_1 &= 160\ \text{K}\Omega \\ R_2 &= 200\ \text{K}\Omega \\ C_1 &= 10\ \mu\text{F} \\ C_2 &= 0.082\ \mu\text{F} \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} \omega_{n_1} &= 6.1733\ \text{rad/sec} \\ \zeta_1 &= 0.091118 \end{aligned}\right.$$

$\omega_{n_1} \approx 1$ Hz & $\zeta_1 \approx 10\%$

$$\left.\begin{aligned} R_3 &= 68\ \text{K}\Omega \\ R_4 &= 13\ \text{K}\Omega \\ C_3 &= 6.8\ \mu\text{F} \\ C_4 &= 0.01\ \mu\text{F} \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} \omega_{n_2} &= 128.98\ \text{rad/sec} \\ \zeta_2 &= 0.052237 \end{aligned}\right.$$

$\omega_{n_2} \approx 20$ Hz & $\zeta_2 \approx 5\%$

Nominal component values

$$R_{of1} = 10k\Omega$$
$$R_{om1} = 1k\Omega$$
$$R_{om2} = 10k\Omega$$
$$R_{of2} = 100\Omega$$

$$R_1 = 160k\Omega$$
$$R_2 = 200k\Omega$$
$$C_1 = 10\mu F$$
$$C_2 = 82nF$$

$$R_3 = 68k\Omega$$
$$R_4 = 13k\Omega$$
$$C_3 = 6.8\mu F$$
$$C_4 = 10nF$$

Input Resistor selection for the summing amplifier

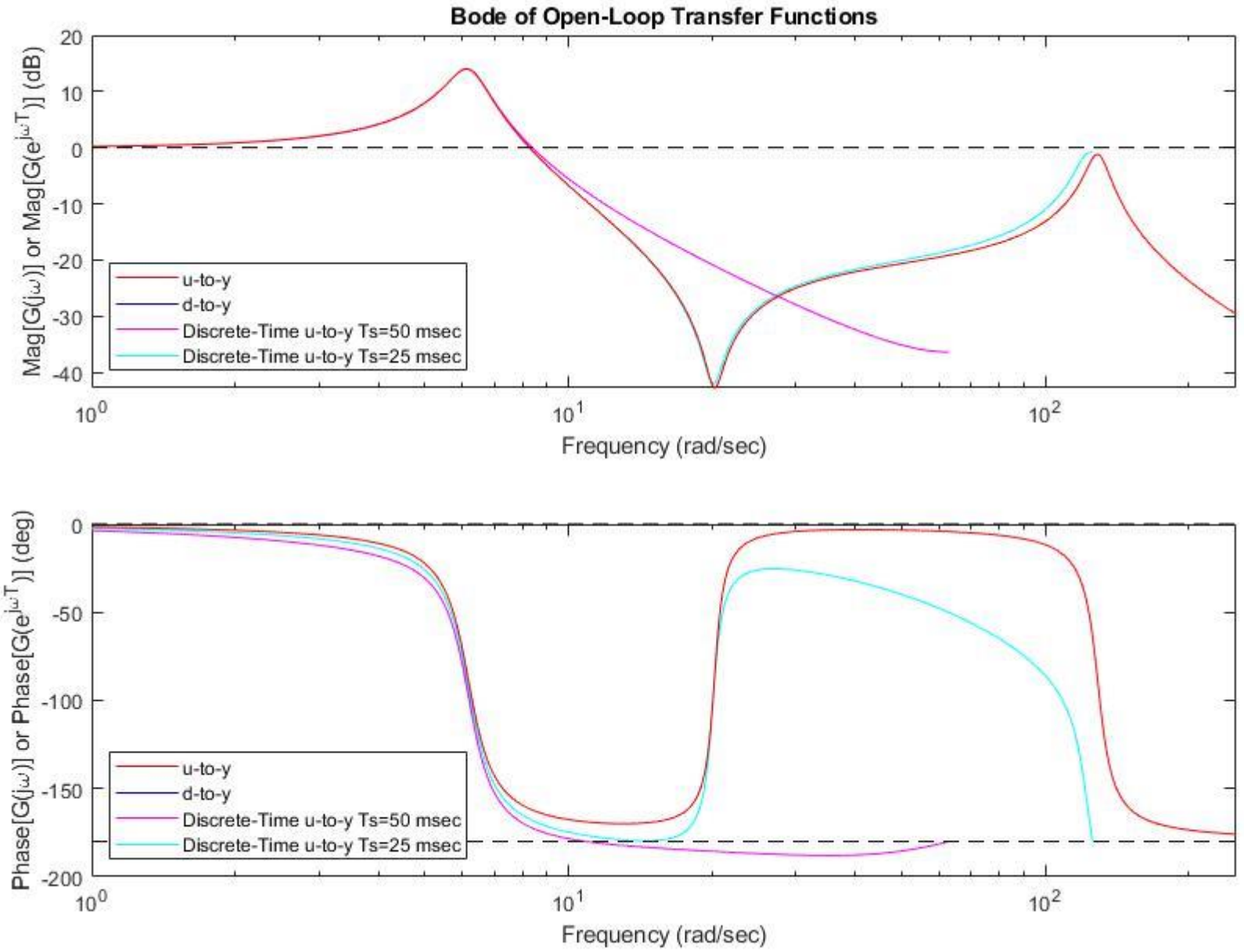
$$k = 10$$
$$R_{om1} = 1k\Omega$$
$$R_{om2} = 10k\Omega$$

Relative contribution of first vibration mode to the output V_y(s) , compared to the second vibration mode.
For our plant we want this to be 10x
The summer is not doing a pure summation!
it is a ***weighted*** summation.

Feedback Resistor selection for the summing amplifier

$$k \approx 1$$
$$R_{of1} = 10k\Omega$$
$$R_{of2} = 100\Omega$$

Output gain of the summing amplifier. We want to be close to 1.



Frequency response of the open-loop plant.
Notes:
The analogplant4thClass.m model generator provided produces this plot.

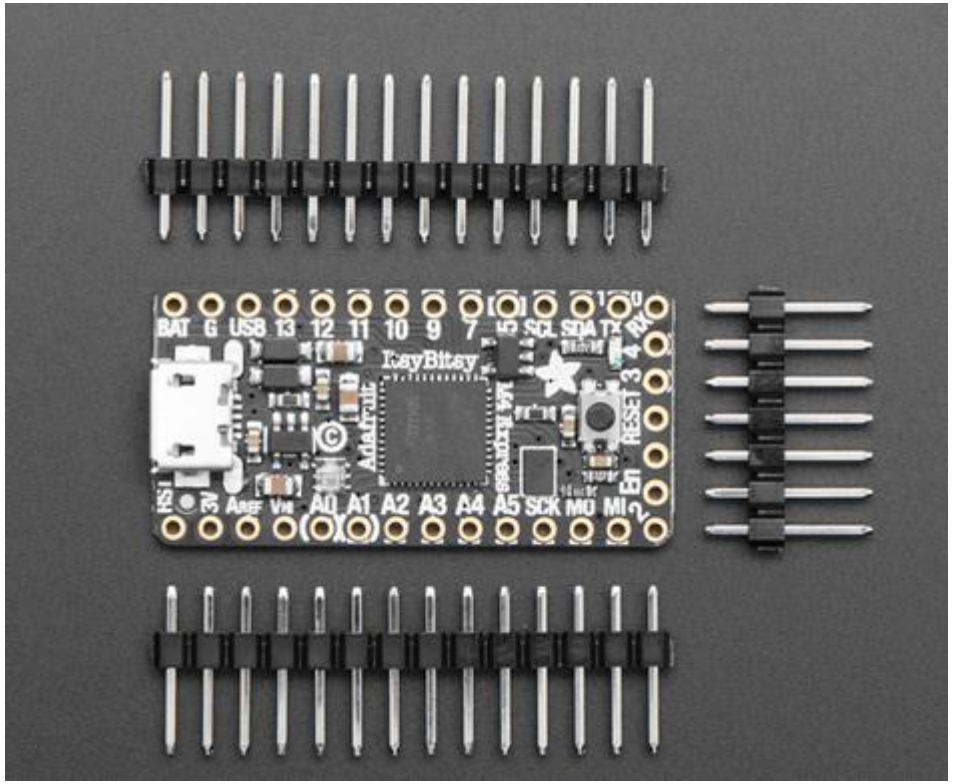
The sampling rate of 50ms corresponds to a Nyquist frequency which is below the natural frequency of the second oscillator within the plant! That has resulted in the 50ms discretized plant's transfer function (purple) terminating at lower frequency than 129 rad/sec. This means that that higher frequency mode is not in the primary strip at that sampling frequency.

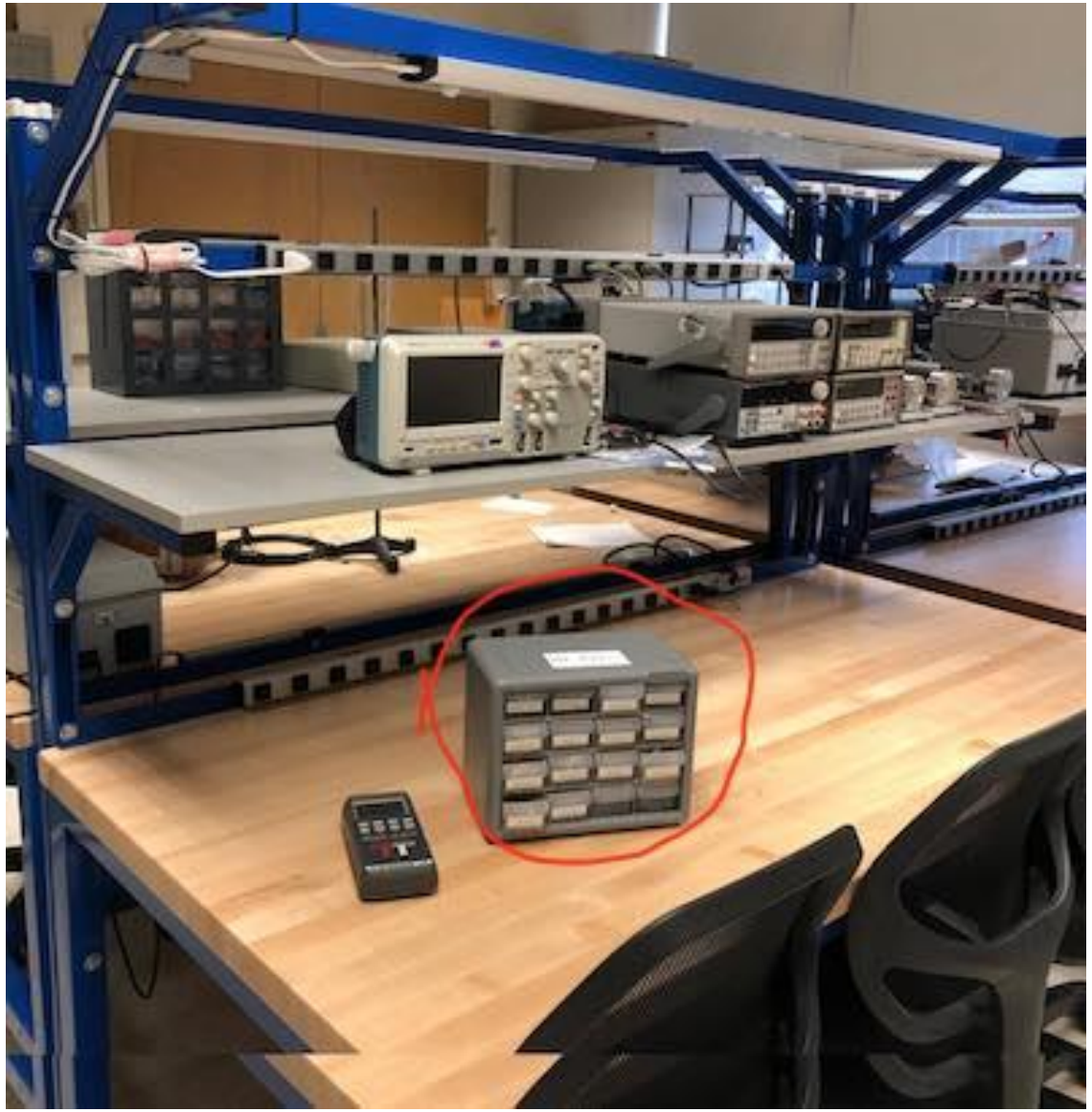
The continuous-time u-to-y and d-to-y transfer functions (red and blue) are superimposed because they enter into the system in the same way, as can be seen in the block diagram below and in the fact that the inverting summer 'weights' these terms equally.

Microcontroller

controller, actuator (D2A converter “DAC”), and sensor (A2D converter “ADC”)

- Arm core M4 ATSAMD51 “Adafruit Itsy Bitsy”
  - 120 MHz, FPU, 12-bit DAC and ADC with plenty of pins
  - (documentation pdf on canvas)
- Circuit Python
  - Runs simplified Python! Can import some libraries like Numpy.
  - Easy to use with Pyboard.py tool to execute your .py code
    - Command line or scripted call of pyboard.py (demo in class)





Component location

MEB115 – Mechatronics lab
Door code: 2468
(also announced in Canvas)

Get these new components

Nominal component values

$R_{of1} = 10k\Omega$

$R_{om1} = 1k\Omega$

$R_{om2} = 10k\Omega$

$R_{of2} = 100\Omega$

$R_3 = 68k\Omega$

$R_4 = 13k\Omega$

$C_3 = 6.8\mu F$

$C_4 = 10nF$

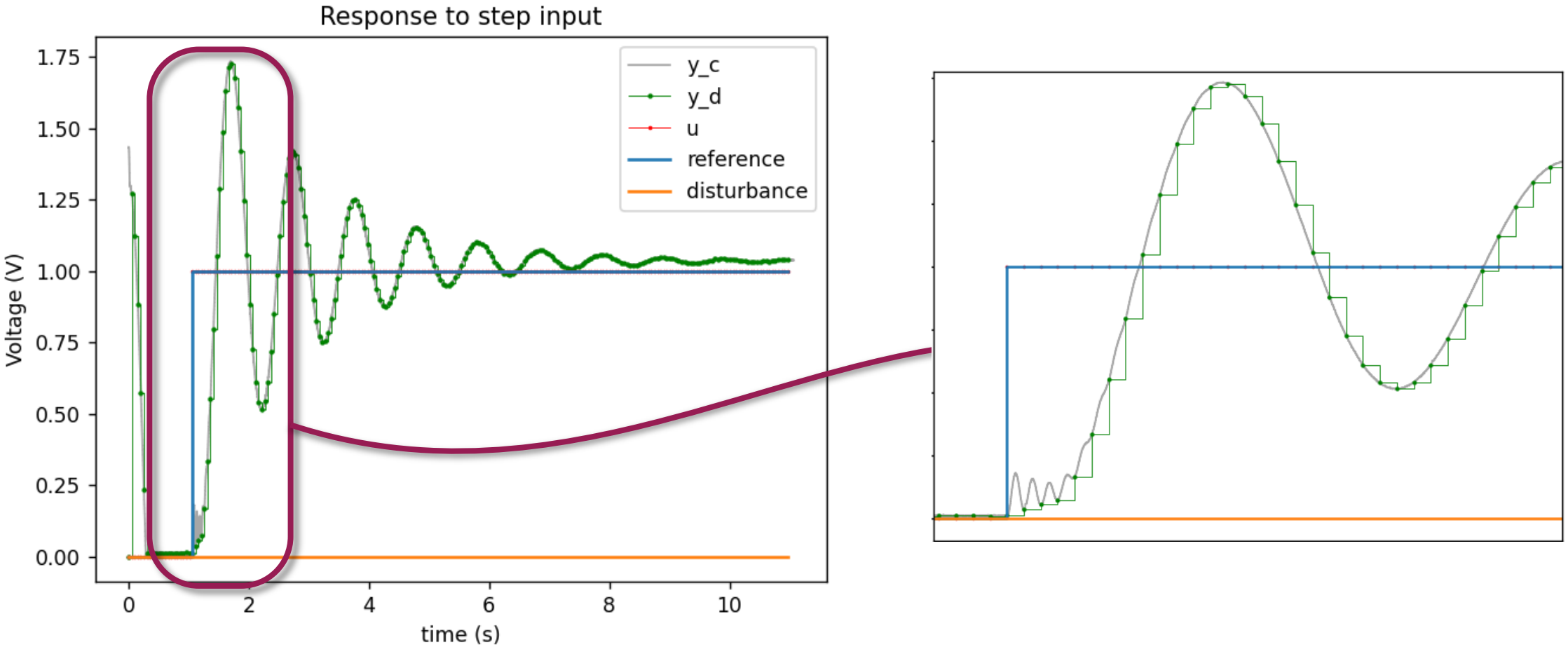
Note:
To make the design stage more realistic, do not use measured component values for your PID design. The actual values are within $\pm 5\%$ of nominal value. This uncertainty is intended realism for the lab.

(Optional) visualize the plant output using the LED

Recorded demo in class. Steps:

- Plug in the ItsyBitsy and open the CIRCUITPY drive
- Download the compressed “lib.zip” folder from Canvas and extract it.
- Copy the folder “lib” into CIRCUITPY
  - This folder contains the .mpy files with new libraries. You can add many other libraries in the same way.
- Uncomment the indicated lines of code on the yourName_lab3.py

Open loop reference step response



Reading the saved data and plotting

Example commands to help get started. Demo in recorded lecture

csv columns are in order of: $k, t_c, y_c, t_d, y_d, r, d, u, u_{sat}$

Create another Python script that loads this data into an array using, for example,

```
import numpy as np
data = np.loadtxt('rawdata.csv', delimiter=',' , encoding='utf-16')
```

then use Matplotlib (or your favorite plotting system) to plot various parameters.
For example (the indices may be different for you)

```
t_c = data[:, 1]/1e9
y_c = data[:,2]
```

To **remove duplicate samples from the discrete data**, index only the non-repeat elements, for ex:

```
ddata = np.squeeze(data[np.nonzero(np.diff(k)),:])
t_d = ddata[:,3]/1e9
y_d = ddata[:,4]
plt.step(t_c, y_c, where='post')
```

Make sure to label your plot axes.
Save your plot using, e.g. `plt.savefig('myfigname.pdf')`

And similarly in Matlab:

```
data = table2array( readtable('rawdata.csv') );
ddata = data(find(diff(data(:,1)) > 0 ), :);
t_c = data(:,2)/1e9;
y_c = data(:,3);
t_d = ddata(:,4)/1e9;
y_d = ddata(:,5);
```