Worksheet 17

To accompany Chapter 6.4 Models of Discrete-Time Systems

Colophon

This worksheet can be downloaded as a PDF file (https://cpjobling.github.io/eg-247-textbook/worksheets/worksheet17.pdf). We will step through this worksheet in class.

An annotatable copy of the notes for this presentation will be distributed before the second class meeting as **Worksheet 9** in the **Week 4: Classroom Activities** section of the Canvas site. I will also distribute a copy to your personal **Worksheets** section of the **OneNote Class Notebook** so that you can add your own notes using OneNote.

You are expected to have at least watched the video presentation of Chapter 6.4 (https://cpjobling.github.io/eg-247-textbook/dt_systems/4/dt_models) of the notes (https://cpjobling.github.io/eg-247-textbook) before coming to class. If you haven't watch it afterwards!

After class, the lecture recording and the annotated version of the worksheets will be made available through Canvas.

Agenda

- Discrete Time Systems (Notes)

- Transfer Functions in the Z-Domain (Notes)

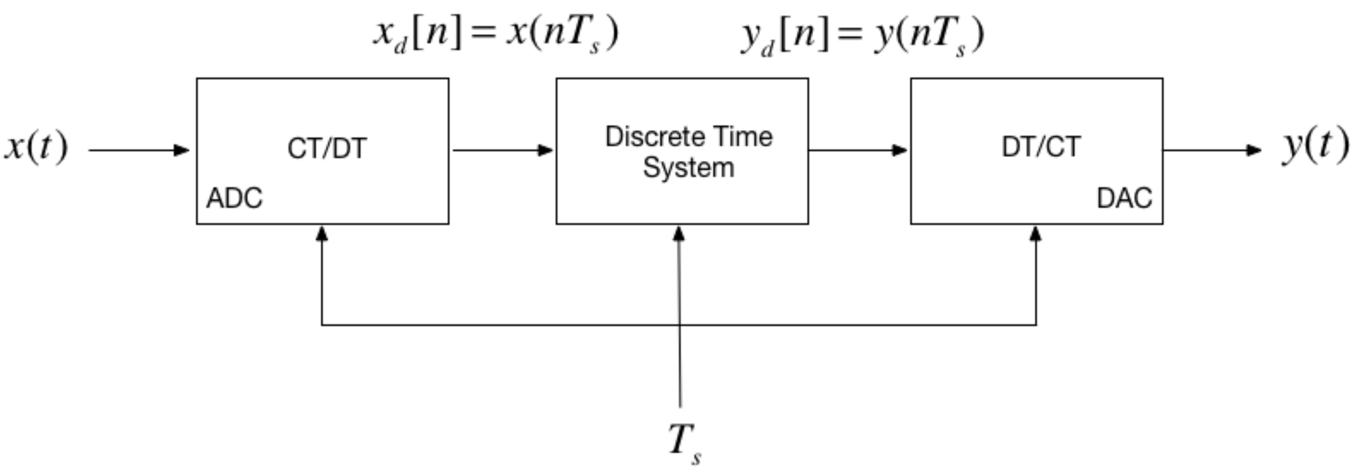
- Modelling digital systems in MATLAB/Simulink

- Continuous System Equivalents

- In-class demonstration: Digital Butterworth Filter

Discrete Time Systems

In the lecture that introduced the z-transform we talked about the representation of a discrete-time (DT) system by the model shown below:



In this session, we want to explore the contents of the central block.

Example 5

Karris Example 9.10:

The difference equation describing the input-output relationship of a DT system with zero initial conditions, is:

$$y[n] - 0.5y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

$$y[n] - 0.5y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

Compute:

1. The transfer function $H(z)$
2. The DT impulse response $h[n]$
3. The response $y[n]$ when the input $x[n]$ is the DT unit step $u_0[n]$

5.1. The transfer function

$$H(z) = \frac{Y(z)}{U(z)} = \ldots ?$$

$$H(z) = \frac{Y(z)}{U(z)} = \ldots ?$$

5.2. The DT impulse response

Start with:

$$\frac{H(z)}{z} = \frac{z-1}{z^2 + 0.5z + 0.125}$$

$$\frac{H(z)}{z} = \frac{z-1}{z^2 + 0.5z + 0.125}$$

MATLAB Solution

In [ ]:

```
clear all
cd matlab
pwd
format compact
```

See dtm_ex1_2.mlx (matlab/dtm_ex1_2.mlx). (Also available as dtm_ex1_2.m (matlab/dtm_ex1_2.m).)

The difference equation describing the input-output relationship of the DT system with zero initial conditions, is:

$$y[n] - 0.5y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

$$y[n] - 0.5y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

Transfer function

Numerator $z + 1$ $z + 1$

In [ ]:

```
Nz = [0 1 1];
```

Denominator $z^2 - 0.5z + 0.125$ $z^2 - 0.5z + 0.125$

In [ ]:

```
Dz = [1 -0.5 0.125];
```

Poles and residues

In [ ]:

```
[r,p,k] = residue(Nz,Dz)
```

Impulse Response

In [ ]:

```
Hz = tf(Nz,Dz,1)
hn = impulse(Hz, 15);
```

Plot the response

In [ ]:

```
stem([0:15], hn)
grid
title('Example 5 - Part 2')
xlabel('n')
ylabel('Impulse response h[n]')
```

Response as stepwise continuous y(t)

In [ ]:

```
impulse(Hz,15)
grid
title('Example 5 - Part 2 - As Analogue Signal')
xlabel('nTs [s]')
ylabel('Impulse response h(t)')
```

5.3. The DT step response

$$Y(z) = H(z)X(z)$$

$$Y(z) = H(z)X(z)$$

$$u_0[n] \Leftrightarrow \frac{z}{z-1}$$

$$u_0[n] \Leftrightarrow \frac{z}{z-1}$$

$$\begin{aligned} Y(z) = H(z)U_0(z) &= \frac{z^2+z}{z^2+0.5z+0.125} \cdot \frac{z}{z-1} \\ &= \frac{z(z^2+z)}{(z^2+0.5z+0.125)(z-1)} \end{aligned}$$

$$\begin{aligned} Y(z) = H(z)U_0(z) &= \frac{z^2+z}{z^2+0.5z+0.125} \cdot \frac{z}{z-1} \\ &= \frac{z(z^2+z)}{(z^2+0.5z+0.125)(z-1)} \end{aligned}$$

$$\frac{Y(z)}{z} = \frac{z^2+z}{(z^2+0.5z+0.125)(z-1)}$$

$$\frac{Y(z)}{z} = \frac{z^2+z}{(z^2+0.5z+0.125)(z-1)}$$

Solved by inverse Z-transform.

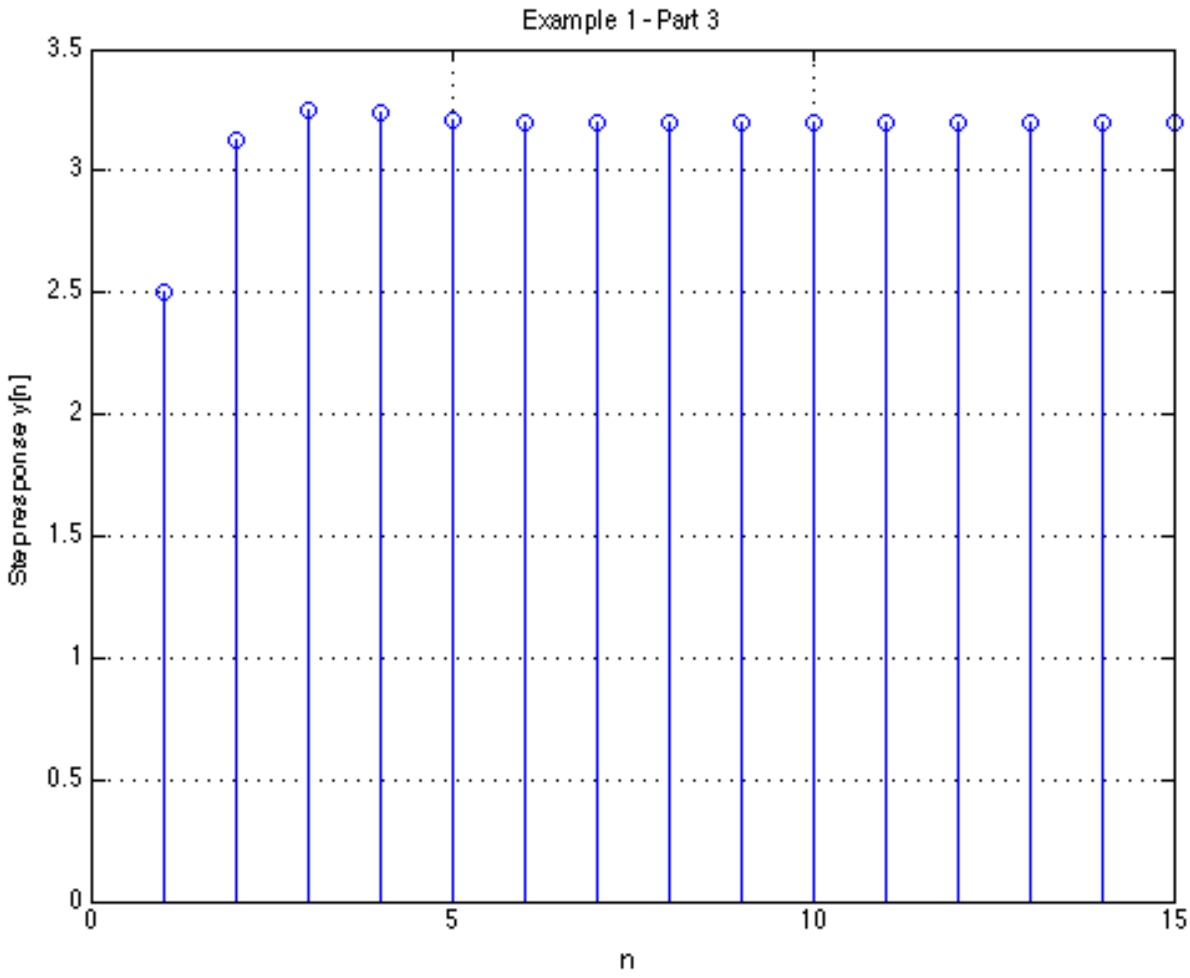
MATLAB Solution

See [dtm_ex1_3.mlx (matlab/dtm_ex1_3.mlx)](matlab/dtm_ex1_3.mlx). (Also available as [dtm_ex1_3.m (matlab/dtm_ex1_3.m)](matlab/dtm_ex1_3.m).)

In [ ]:

```
open dtm_ex1_3
```

Results



Modelling DT systems in MATLAB and Simulink

We will consider some examples in class

MATLAB

Code extracted from dtm_ex1_3.m (matlab/dtm_ex1_3.m):

In [ ]:

```
Ts = 1;
z = tf('z', Ts);
```

In [ ]:

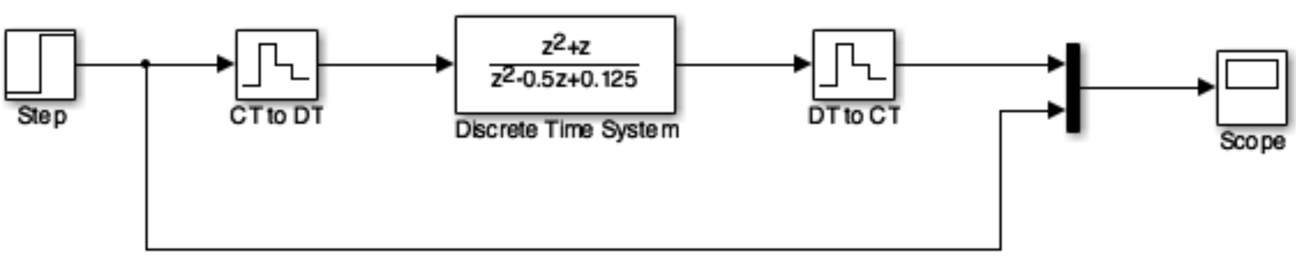
```
Hz = (z^2 + z)/(z^2 - 0.5 * z + 0.125)
```

In [ ]:

```
step(Hz)
grid
title('Example 1 - Part 3 - As Analogue Signal')
xlabel('nTs [s]')
ylabel('Step response y(t)')
axis([0,15,0,3.5])
```

Simulink Model

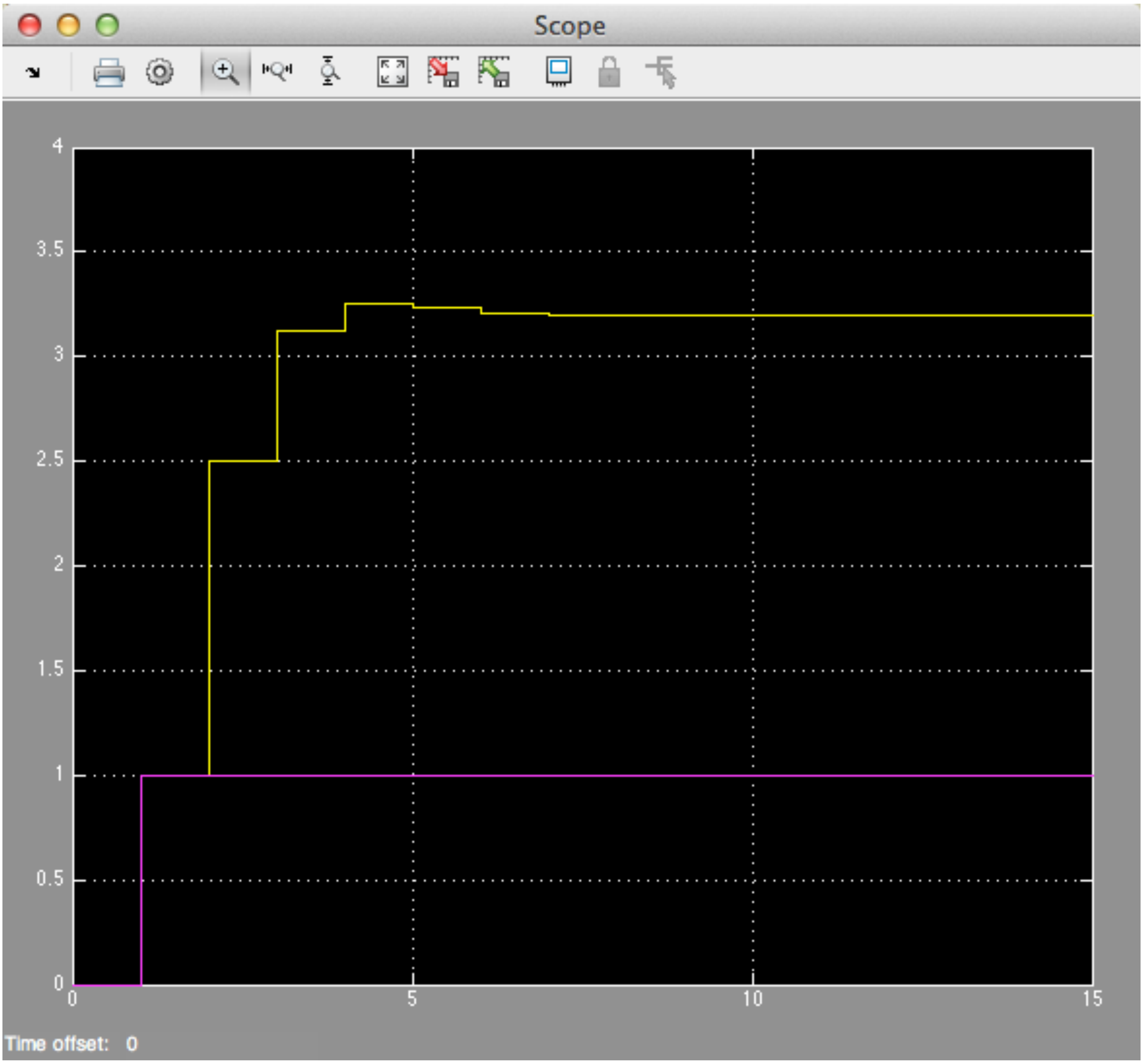
See dtm.slx (matlab/dtm.slx):



In [ ]:

```
dtm
```

Results



Converting Continuous Time Systems to Discrete Time Systems

Continuous System Equivalents

- There is no digital system that uniquely represents a continuous system
- This is because as we are sampling, we only have knowledge of signals being processed at the sampling instants, and need to *reconstruct* the inter-sample behaviour.
- In practice, only a small number of transformations are used.
- The derivation of these is beyond the scope of this module, but here we'll demonstrate the ones that MATLAB provides in a function called `c2d`

MATLAB c2d function

Let's see what the help function says:

In [ ]:

```
help c2d
```

In [ ]:

```
doc c2d
```

Example 6

- Design a 2nd-order butterworth low-pass anti-aliasing filter with transfer function $H(s)$ for use in sampling music.
- The cut-off frequency $\omega_c = 20$ kHz and the filter should have an attenuation of at least -80 dB in the stop band.
- Choose a suitable sampling frequency for the audio signal and give the transfer function $H(z)$ and an algorithm to implement $h[n]$

Solution

See digi_butter.mlx (matlab/digi_butter.mlx).

First determine the cut-off frequency ω_c ω_c

$$\omega_c = 2\pi f_c = 2 \times \pi \times 20 \times 10^3 \text{ rad/s}$$

$$\omega_c = 2\pi f_c = 2 \times \pi \times 20 \times 10^3 \text{ rad/s}$$

In [ ]:

```
wc = 2*pi*20e3
```

$$\omega_c = 125.66 \times 10^3 \text{ rad/s}$$

$$\omega_c = 125.66 \times 10^3 \text{ rad/s}$$

From the lecture on filters, we know the 2nd-order butterworth filter has transfer function:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{\omega_c^2}{s^2 + \omega_c \sqrt{2}\, s + \omega_c^2}$$

$$H(s) = \frac{Y(s)}{U(s)} = \frac{\omega_c^2}{s^2 + \omega_c \sqrt{2}\, s + \omega_c^2}$$

Substituting for $\omega_c = 125.6637 \times 10^3$ $\omega_c = 125.6637 \times 10^3$ this is ...?

In [ ]:

```
Hs = tf(wc^2,[1 wc*sqrt(2), wc^2])
```

$$H(s) = \frac{15.79 \times 10^9}{s^2 + 177.7 \times 10^3 s + 15.79 \times 10^9}$$

$$H(s) = \frac{15.79 \times 10^9}{s^2 + 177.7 \times 10^3 s + 15.79 \times 10^9}$$

Bode plot

MATLAB:

In [ ]:

```
bode(Hs,{1e4,1e8})
grid
```

Sampling Frequency

From the bode diagram, the frequency at which $|H(j\omega)|$ is -80 dB is approx 12.6×10^6 rad/s.

To avoid aliasing, we should choose a sampling frequency twice this = ?

$\omega_s = 2 \times 12.6 \times 10^6$ rad/s.

In [ ]:

```
ws = 2* 12.6e6
```

So

$\omega_s = 25.2 \times 10^6$ rad/s.

Sampling frequency (f_s) in Hz = ?

$$f_s = \omega_s/(2\pi) \text{ Mhz}$$

$$f_s = \omega_s/(2\pi) \text{ Mhz}$$

In [ ]:

```
fs = ws/(2*pi)
```

$$f_s = 40.11 \text{ Mhz}$$

$$f_s = 40.11 \text{ Mhz}$$

Sampling time $T_s = ?$

$T_s = 1/fs$ s

In [ ]:

```
Ts = 1/fs
```

$$T_s = 1/f_s \approx 0.25 \;\mu\text{s}$$

$$T_s = 1/f_s \approx 0.25 \;\mu\text{s}$$

Digital Butterworth

zero-order-hold equivalent

In [ ]:

```
Hz = c2d(Hs, Ts)
```

Step response

In [ ]:

```
step(Hz)
```

Algorithm

From previous result:

$$H(z) = \frac{Y(z)}{U(z)} = \frac{486.6 \times 10^{-6} z + 476.5 \times 10^{-6}}{z^2 - 1.956 z + 0.9567}$$

$$H(z) = \frac{Y(z)}{U(z)} = \frac{486.6 \times 10^{-6} z + 476.5 \times 10^{-6}}{z^2 - 1.956 z + 0.9567}$$

Dividing top and bottom by z^2 z^2 ...

$$H(z) = \frac{Y(z)}{U(z)} = \frac{486.6 \times 10^{-6} z^{-1} + 476.5 \times 10^{-6} z^{-2}}{1 - 1.956 z^{-1} + 0.9567 z^{-2}}$$

$$H(z) = \frac{Y(z)}{U(z)} = \frac{486.6 \times 10^{-6} z^{-1} + 476.5 \times 10^{-6} z^{-2}}{1 - 1.956 z^{-1} + 0.9567 z^{-2}}$$

expanding out ...

$$Y(z) - 1.956z^{-1}Y(z) + 0.9567z^{-2}Y(z) =$$
$$486.6 \times 10^{-6}z^{-1}U(z) + 476.5 \times 10^{-6}z^{-2}U(z)$$

$$Y(z) - 1.956z^{-1}Y(z) + 0.9567z^{-2}Y(z) =$$
$$486.6 \times 10^{-6}z^{-1}U(z) + 476.5 \times 10^{-6}z^{-2}U(z)$$

Inverse z-transform gives ...

$$y[n] - 1.956y[n-1] + 0.9567y[n-2] =$$
$$486.6 \times 10^{-6}u[n-1] + 476.5 \times 10^{-6}u[n-2]$$

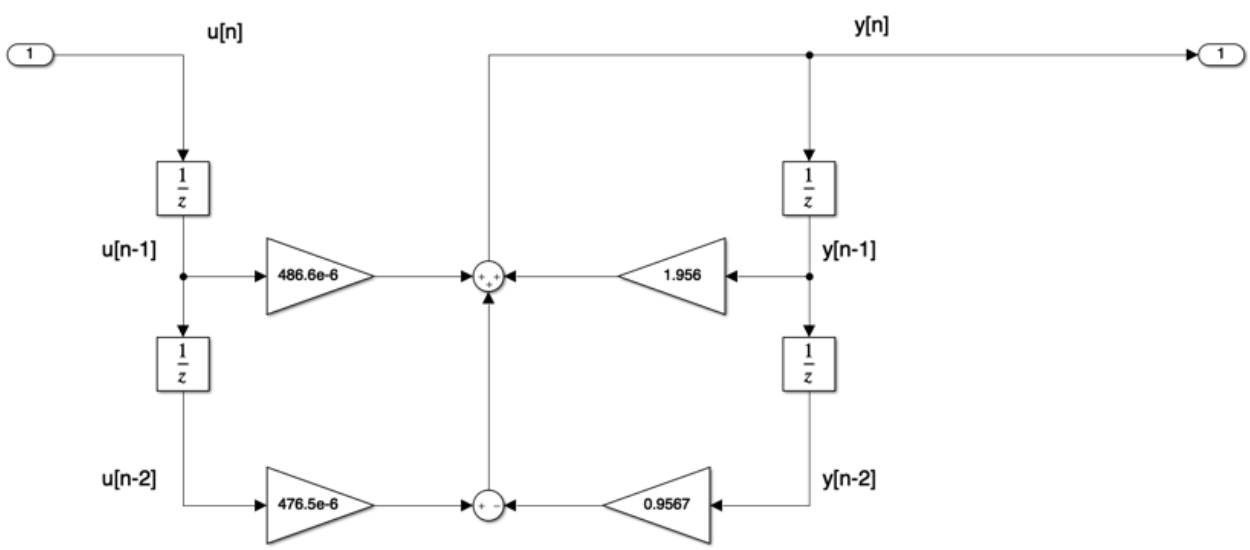
$$y[n] - 1.956y[n-1] + 0.9567y[n-2] =$$
$$486.6 \times 10^{-6}u[n-1] + 476.5 \times 10^{-6}u[n-2]$$

in algorithmic form (compute $y[n]y[n]$ from past values of uu and yy) ...

$$y[n] = 1.956y[n-1] - 0.9567y[n-2] + 486.6 \times 10^{-6}u[n-1] + \ldots$$
$$476.5 \times 10^{-6}u[n-2]$$

$$y[n] = 1.956y[n-1] - 0.9567y[n-2] + 486.6 \times 10^{-6}u[n-1] + \ldots$$
$$476.5 \times 10^{-6}u[n-2]$$

Block Diagram of the digital BW filter



As Simulink Model

[digifilter.slx (matlab/digifilter.slx)](matlab/digifilter.slx)

In [ ]:

```
open digifilter
```

Convert to code

To implement:

$y[n] = 1.956y[n-1] - 0.9567y[n-2] + 486.6 \times 10^{-6}u[n-1] + 476.5 \times 10$

$$y[n] = 1.956y[n-1] - 0.9567y[n-2] + 486.6 \times 10^{-6}u[n-1] + 476.5 \times 10^{-6}u[n-2]$$

```
/* Initialize */
Ts = 2.4933e-07; /* more probably some fraction of clock speed */
ynm1 = 0; ynm2 = 0; unm1 = 0; unm2 = 0;
while (true) {
    un = read_adc;
    yn = 1.956*ynm1 - 0.9567*ynm2 + 486.6e-6*unm1 + 476.5e-6*unm2;
    write_dac(yn);
    /* store past values */
    ynm2 = ynm1; ynm1 = yn;
    unm2 = unm1; unm1 = un;
    wait(Ts);
}
```

Comments

PC soundcards can sample audio at 44.1 kHz so this implies that the anti-aliasing filter is much sharper than this one as $f_s/2 = 22.05$ kHz.

You might wish to find out what order butterworth filter would be needed to have $f_c = 20$ kHz and f_{stop} of 22.05 kHz.