Practical 2
Converting an analog filter to digital filter

Ho Guo Xian
17WLR19013

January 15, 2019

1 Preliminaries

The specification frequencies for x, y and z are 20Hz, 200Hz and 2000Hz respectively. Figure 1 shows all the generated frequencies with different combination of x, y and z

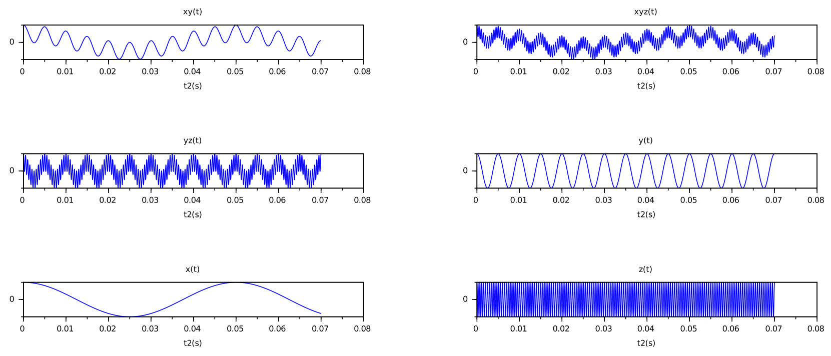

Figure 1: All generated waveform

For this practical, the chose analog filter will be high-pass filter, with aim of filtering signal *x, y* away from signal *xyz*.

2 The design

For designing a high pass filter, following equation will be used.

$$H(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{1}$$

With

$$\omega_n = natural frequency = 2\pi f_n \tag{2}$$

$$\zeta = dampingratio \tag{3}$$

The natural frequency f_n selected will be $500Hz$. Upon substituting with equation 1 and 2, following is the resulting transfer function.

$$f(t) = \frac{s^2}{9869604.4 + 6283.1853s + s^2} \tag{4}$$

The bode plot for the equation 4 is shown in figure 2. The cutoff frequency around *3dB* is around *770Hz*. This is fine since it will be filtering out signal x and y because both frequency (20 and 200 Hz) is lower than *770Hz*.

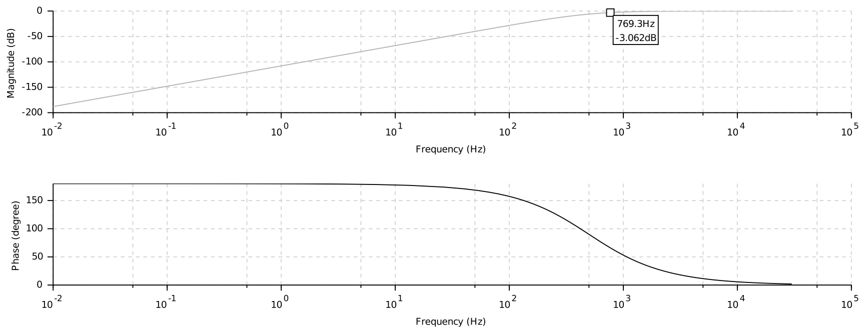

Figure 2: The resulting bode plot for equation 4

Following is the scilab code for constructing the transfer function and plot the bode plot.

```
// s transfer function
wn = 2*%pi*500
dampratio = 1
s = poly(0, "s");
Hs = s^2 / (s^2 + 2*dampratio*wn*s + wn^2)
disp(Hs)
h = syslin('c', Hs)
 clf();bode(h, 0.01, 30000)
```

Now the analog filter is ready to convert into digital equivalent filter. Among all the methods out there, pole-zero mapping method is chosen for the conversion. First the the root of the denominator is found by equating $9869604.4 + 6283.1853s + s^2 = 0$, we will get $s_1 = -3142 - 0.147i$ and $s_2 = -3142 + 0.147i$

Using the relationship of

$$z = e^{st} \tag{5}$$

We will get $z_1 = 0.961 - 1.767 \times 10^{-6}i$ and $z_2 = 0.961 + 1.767 \times 10^{-6}i$ where $t = \frac{1}{80kHz}$. The z transfer function is shown in equation 6.

$$T(z) = k\frac{(z-1)(z-1)}{(z - 0.961 - 1.767 \times 10^{-6}i)(z - 0.961 + 1.767 \times 10^{-6}i)} \tag{6}$$

The transfer function is then being plotted by taking z from $e^{i\pi n}$ where n is 1000 points from 0 to 1, and k is taken as $\frac{1}{max(abs(T(z)))}$. The result is shown in figure 3

The final z transfer function is shown below.

$$T(z) = \frac{0.9618619 - 1.9237238z + 0.9618619z^2}{0.9244652 - 1.9229823z + z^2} \tag{7}$$

The scilab code for converting to z transform and the plot is shown in listing 1

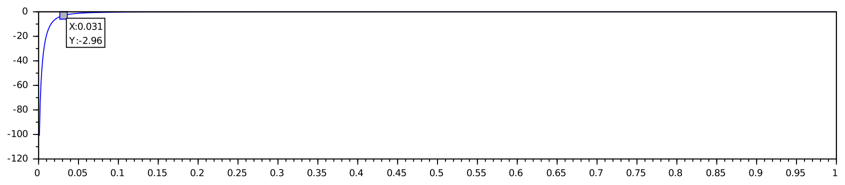

Figure 3: The frequency response of digital filter

Listing 1: The scilab code for z transform

```
      s_den_root1 = -3141.593- %i*0.147
      s_den_root2 = -3141.593+ %i*0.147
      z_den_1 = exp(s_den_root1 * (1/80000))
      z_den_2 = exp(s_den_root2 * (1/80000))
      number = 0:0.001:1
      z = exp(%i*%pi*number)
      ztf = ((z-1).*(z-1))./((z-z_den_1) .* (z-z_den_2))
      k = 1 / max(abs(ztf))
      ztfk = ztf * k
      clf();plot(number, 20 * log(abs(ztfk)))

      z = poly(0, 'z')
      ztf = ((z-1)*(z-1))/((z-z_den_1) * (z-z_den_2)) * k
```

The transfer function in equation 7 is then undergo long division to get the actual filter signal, listing 2 shows the scilab code and figure 4 shows the plot of the filter.

Listing 2: scilab code for long division

```
      ld = ldiv(ztf.num, ztf.den, 25);
      clf();plot(1:length(ld), (ld))
```

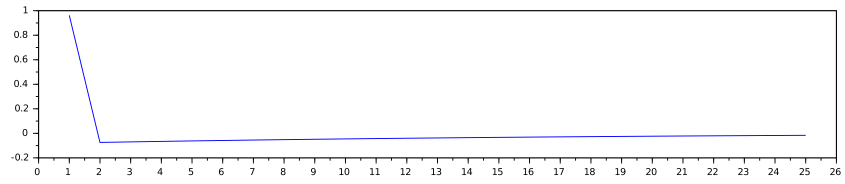

Figure 4: The plot of the digital filter

3 The result

For the digital filter shown in equation 7, it will be convolute with signal xyz. The scilab code is shown in listing 3. The resulting graph is shown in figure 5

Listing 3: scilab code for convolution with signal xyz

```
convd = conv(ld, xyzl);
 clf();subplot(2,1,1), plot(1:length(convd), convd)
subplot(2,1,2), plot(t2, xyzl), title('xyz(t)'), xlabel('t2(s)');
```

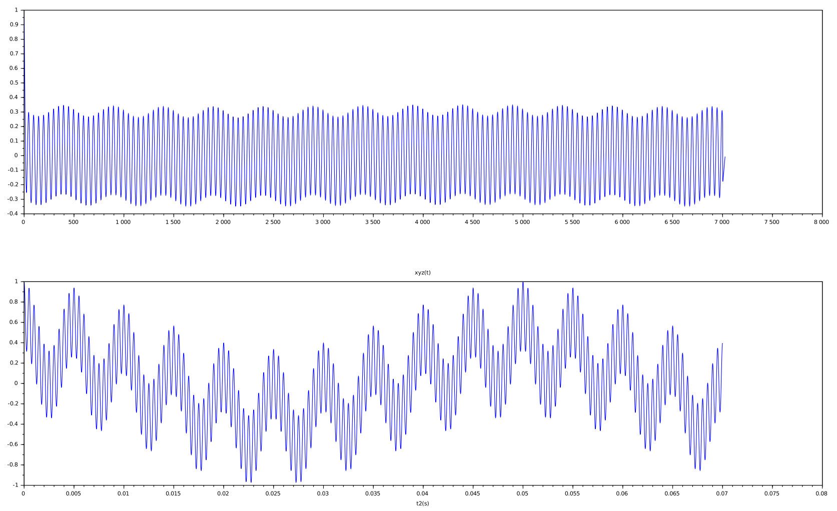

Figure 5: convolution of equation 7 with signal xyz, top is resulting signal, bottom is signal xyz

In the figure 5, it's clearly seen that the signal x can't be seen from the result, while the frequency of y and z still visible.

4 The discussion

In figure 3, the cutoff frequency obtained is $0.031 \times 40kHz = 1240Hz$ where $nyquist = 40Khz$. 1240Hz is way higher than the cutoff frequency for analog transfer function shown in figure 2 which is around 769Hz. But since 1240Hz is still able to filter 200Hz and 20Hz while pass 2000Hz through, the equivalent digital filter is still performed the job required.

Bilinear method is carried out to find out whether the digital equivalent filter is more accurate than pole-zero mapping method, and surprisingly, the result we gathered is almost the same with the digital filter have cutoff frequency almost twice the analog filter.

Listing 4 shows the scilab code for converting to analog filter using bilinear transformation.

Listing 4: scilab code for bilinear transformation

```
slss=tf2ss(h_lin);  //Now in state-space form
sl1=cls2dls(slss,1/80000);  //sl1= output of cls2dls
sl1t=ss2tf(sl1) // Converts in transfer form
disp(sl1t)
```

The resulting z transfer function is shown below in equation 8. The frequency response of the z transfer function is then plotted out using the same method in pole-zero mapping. Figure 6 shows the result. The cutoff frequency in this case will be $0.031 \times 40Khz = 1240Hz$ which is exactly the same with pole-zero mapping method.

$$T(z) = \frac{0.9618571 - 1.9237143z + 0.9618571z^2}{0.9244559 - 1.9229726z + z^2} \tag{8}$$

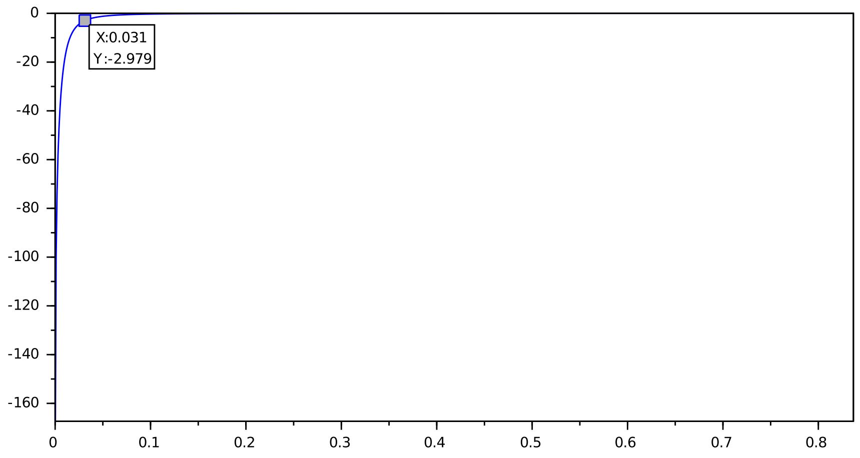

Figure 6: The frequency response of z transfer function in equation 8

The plot after long division is shown in figure 7, and it's similar to figure 4.

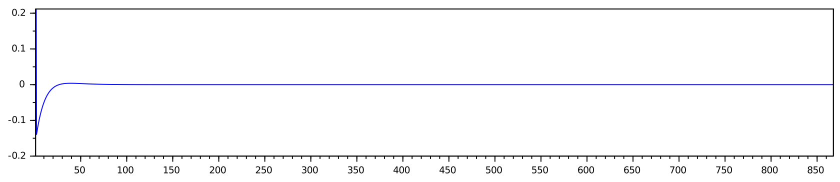

Figure 7: Plot after long division

After bilinear transformation, the filter is then convolute with signal xyz and figure 8 shows the result after convolution. Again it can clearly see that the result is similar with figure 5.

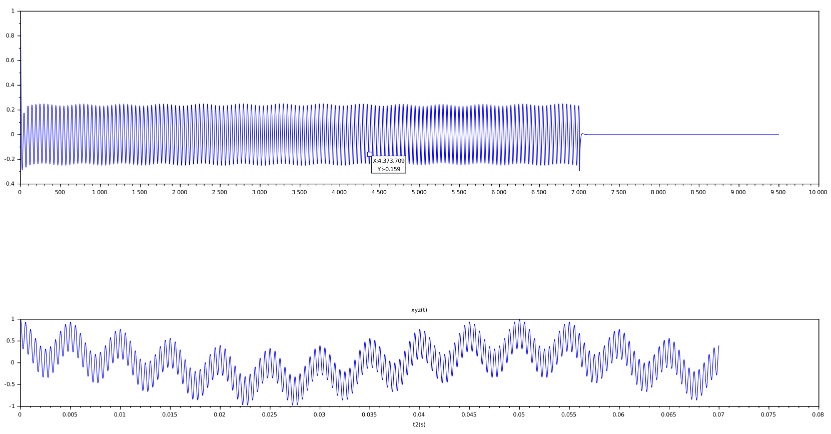

Figure 8: Convolution with signal xyz, top is result, bottom is signal xyz.

5 The conclusion

Even though the analog to digital transformation is not accurate, but because of the resulting digital filter still meet the requirement of filtering 20Hz and 200Hz out of the signal, so the result shown is still technically correct. Perhaps direct design of z transfer function will be more suitable for simple filter like this. To conclude, the a high pass digital filter is built and succesfully filtered all low frequency signals.