L.EEC025 - Fundamentals of Signal Processing

Academic year 2025-2026, week 8
P2P exercises

Topics: Peer-to-peer learning/teaching exercises on the analysis of a second-order system

Peer-to-peer learning/teaching (P2P L/A) exercises

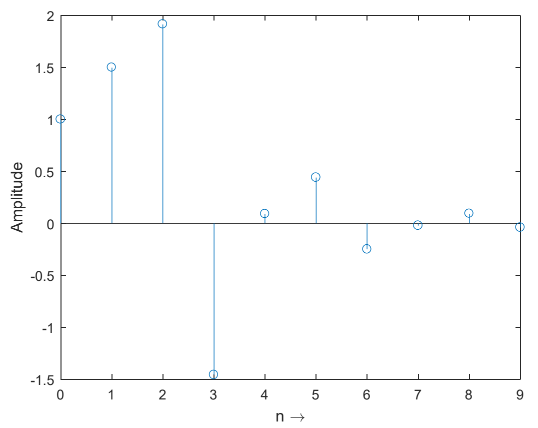
Preliminary considerations

All four exercises indicated here involve the same second-order discrete-time system: a causal discrete-time system that has two poles, one at $z = \alpha e^{j\beta}$, and another one at $z = \alpha e^{-j\beta}$, and two zeros at $z = 0$. Some of the challenges discussed here have been addressed already in past PL classes, or lectures.

In addition to a few Matlab functions we are already familiar with (`roots()`, `zplane()`,`freqz()`, `xcorr()`), during this week we will use two other signal processing-related Matlab functions: `impz()` and `filter()`.

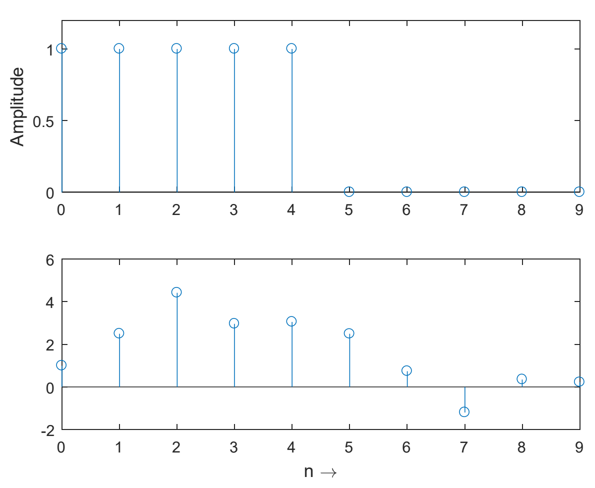
Regarding `impz()`: it computes the impulse response of a discrete-time system that is described by a transfer function whose numerator coefficients are defined in a vector `b`, and whose denominator coefficients are defined in a vector `a`. For example, if the transfer function is $\frac{1+2z^{-1}+3z^{-2}}{1+(1/2)z^{-1}+(1/3)z^{-2}}$, then the following Matlab code generates and displays 10 coefficients of the impulse response:

```
N=10; n=[0:N-1];
b=[1 2 3];
a=[1 1/2 1/3];
h=impz(b,a,N);
stem(n, h)
xlabel('n \rightarrow')
ylabel('Amplitude')
```



Regarding `filter()`: it implements a difference equation according to the transfer function specification of a given discrete-time system. For example, if the transfer function is $\frac{1+2z^{-1}+3z^{-2}}{1+(1/2)z^{-1}+(1/3)z^{-2}}$, then the difference equation is implemented as $y[n] = x[n] + 2x[n-1] + 3x[n-2] - \frac{1}{2}y[n-1] - \frac{1}{3}y[n-2]$. As in the previous Matlab command, `filter()` takes as parameters the transfer function numerator coefficients in vector `b`, and the transfer function denominator coefficients in vector `a`. In addition, it also takes a third parameter that specifies the input sequence (e.g. in a vector `x`). Differently from `conv()`, `filter()` delivers at the output a vector `y` that has the same length as the input vector `x`. For example, if the input is $x[n] = u[n] - u[n-5]$, then the following Matlab code generates and displays 10 coefficients of the output sequence.

```
N=10; n=[0:N-1];
x=[ones(1,5) zeros(1,N-5)];
b=[1 2 3];
a=[1 1/2 1/3];
y=filter(b,a,x);
subplot(2,1,1)
stem(n, x)
ylabel('Amplitude')
subplot(2,1,2)
stem(n, y)
xlabel('n \rightarrow')
```



P2P Exercise 1

In this exercise, we specify the transfer function that describes our causal discrete-time system, obtain its impulse response, and validate it numerically in Matlab.

a) As indicated above, our second-order and causal discrete-time system has two poles, one at $z = \alpha e^{j\beta}$, and another one at $z = \alpha e^{-j\beta}$, where α and β are real-valued, $|\alpha| < 1$, and has two zeros at $z = 0$. Obtain its transfer function and express it in a form containing real-valued coefficients only. Obtain also the corresponding difference equation.

Note: the solution should be: $y[n] = x[n] + 2\alpha\cos(\beta)\, y[n-1] - \alpha^2 y[n-2]$

P2P assessment: **2pt /5** if explanation is clear and results are correct (both transfer function (**1**) and difference equation (**1**))

b) Find a compact (real-valued) expression describing the impulse response of our system (and explain that process to your colleagues). After that, complete and use the following Matlab code to validate numerically your analytical result.

```
ALFA=0.925; BETA=0.275*pi;
b = please complete here
a= please complete here
N=50; n=[0:N-1].';
h=impz(b,a,N);
myh= please complete here
subplot(2,1,1)
stem(n, h)
ylabel('Amplitude')
```

```
subplot(2,1,2)
stem(n, h-myh) % difference should be zero (i.e. < 10^-10)
xlabel('n \rightarrow')
```

Note: the solution should be: $h[n] = \frac{\alpha^n}{\sin(\beta)} \sin\big(\beta(n+1)\big)\, u[n]$

P2P assessment: **3pt /5** if explanation is clear and results are correct (both impulse response (**2**) and Matlab (**1**))

P2P Exercise 2

In this exercise, we analyse the frequency response of the causal discrete-time system as specified above. We obtain compact and real-valued expressions for the magnitude and phase and validate those expression numerically in Matlab.

a) Find a compact (real-valued) expression describing the magnitude frequency response of our second-order system. After that, complete and use the following Matlab code to overlap (perfectly) its graphical representation to that delivered by the `freqz()` Matlab command.

```
grid=2*[0:1/512:1-1/512]; % NOTE: PI is not included here...
ALFA=0.925; BETA=0.275*pi;
b = please complete here
a= please complete here
[H, W]=freqz(b,a,512,'whole'); % 512 is default in Matlab, but is required
      in Octave
figure(1)
plot(W/pi, abs(H)) % NOTE: W/pi is normalized, the range is [0, 2[
title('Frequency Response Magnitude')
pause
myabsH = please complete here
hold on
plot(grid, myabsH, 'm') % NOTE: grid is normalized, the range is [0, 2[
hold off
```

Note: the solution should be: $\left|H\left(e^{j\omega}\right)\right| = \frac{1}{\sqrt{(1-2\alpha\cos(\omega-\beta)+\alpha^2)(1-2\alpha\cos(\omega+\beta)+\alpha^2)}}$

P2P assessment: **3pt /5** if explanation is clear and results are correct (both frequency response (**2**) and Matlab (**1**))

b) Find a compact (real-valued) expression describing the phase frequency response of our second-order system. In addition to the above Matlab code, complete and use the following code to overlap (perfectly) the graphical representation to that delivered by the `freqz()` Matlab command.:

```
figure(2)
plot(W/pi, angle(H))
title('Frequency Response Phase')
pause
myPHASE = please complete here
hold on
plot(grid, myPHASE, 'm')
hold off
```

Note: the solution should be: $\angle H\left(e^{j\omega}\right) = -\tan^{-1}\frac{\alpha\sin(\omega-\beta)}{1-\alpha\cos(\omega-\beta)} - \tan^{-1}\frac{\alpha\sin(\omega+\beta)}{1-\alpha\cos(\omega+\beta)}$

P2P assessment: **2pt /5** if explanation is clear and results are correct (both phase response (**1**) and Matlab (**1**))

Extra P2P Exercise 3

In this exercise we evaluate how the frequency response of a discrete-time system affects (from system input to system output) an infinite-length real-valued sine (or co-sine) function, and validate numerically that evaluation in Matlab.

In a recent lecture, we have shown that complex exponentials are the eigenfunctions of linear and shift-invariant (LSI) discrete-time systems. That is, if an LSI discrete-time system is characterized by the impulse response $h[n]$, and is excited by the input sequence $x[n] = e^{jn\omega_0}$, then the output sequence is $y[n] = H(e^{j\omega_0})e^{jn\omega_0}$, where $H(e^{j\omega_0}) = \sum_{n=-\infty}^{+\infty} h[n]\, e^{-jn\omega}\Big|_{\omega=\omega_0}$ is the frequency response of the LSI system when it is evaluated for $\omega = \omega_0$.

a) Show that, in general terms, if $x[n] = \sin(n\omega_0)$, or if $x[n] = \cos(n\omega_0)$, and if $h[n]$ is real-valued (i.e. if $h[n]$ is complex-valued then the following is ***not*** true), then $y[n] = |H(e^{j\omega_0})|\sin\left(n\omega_0 + \angle H(e^{j\omega_0})\right)$, or $y[n] = |H(e^{j\omega_0})|\cos\left(n\omega_0 + \angle H(e^{j\omega_0})\right)$. These results presume that we write $H(e^{j\omega_0}) = |H(e^{j\omega_0})|e^{j\angle H(e^{j\omega_0})}$, where $|H(e^{j\omega_0})|$ represents the magnitude part of the frequency response of the system when it is evaluated for $\omega = \omega_0$, and $\angle H(e^{j\omega_0})$ represents the phase part of the frequency response of the system when it is evaluated for $\omega = \omega_0$.

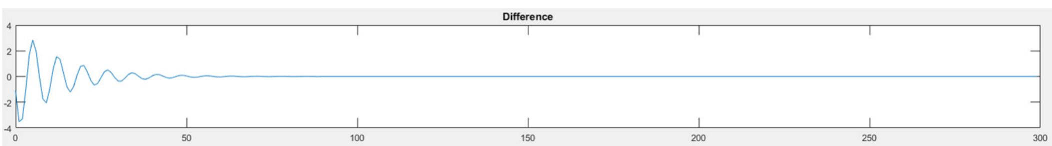
b) Using as a baseline the Matlab code that is used in P2P Exercise 2, and assuming the LSI discrete-time system specification as above, assume further that ω_0 = `OMEGA0= 0.7404`, and use the following Matlab code to show that `MAG=`$|H(e^{j\omega_0})|$, and `PHI=`$\angle H(e^{j\omega_0})$, are as follows:

```
OMEGA0= 0.7404;
[H, W]=freqz(b,a,[0 OMEGA0]);
MAG=abs(H(2));
PHI=angle(H(2));
% do your confirm that MAG = 5.1418 ?
% do you confirm that PHI 0.2159 ?
```

c) Now, create a long sinusoidal sequence with 10^5 samples, obtain the (numerical) system output using the Matlab command `filter()`, program your analytical solution, and check the difference between numerical and analytical results using the following Matlab code:

```
N=1E5; n=[0:N-1];
xsine=sin(n*OMEGA0).'; % you may check that cos() also works
ysine=filter(b,a,xsine);
myysine= please complete here .'; % don't forget transposition .'
plotrange=[1:300];
figure(3)
subplot(4,1,1)
plot(n(plotrange), xsine(plotrange))
title('INPUT')
subplot(4,1,2)
plot(n(plotrange), ysine(plotrange))
title('Output (numerical)')
subplot(4,1,3)
plot(n(plotrange), myysine(plotrange))
title('Output (analytical)')
subplot(4,1,4)
plot(n(plotrange), ysine(plotrange)-myysine(plotrange))
     title('difference signal')
```

Note: you should obtain the following difference signal:



How do you explain to your Colleagues that the first few samples are not zero ?

Extra P2P Exercise 4

In this exercise, we show that if the input sine wave that is specified in P2P Exercise 3 is contamined by noise, at a certain SNR, it comes out of the system with an improved SNR. We find the theoretical gain in SNR, and validate that result numerically in Matlab.

a) The following Matlab code (which is a continuation of the Matlab code in P2P Exercise 3) generates noise with a uniform Probability Density Function (PDF) of the samples' amplitudes, whose auto-correlation function that consists of an impulse and such that when the noise is combined with the sine wave the resulting SNR is 10 dB.

```
SNR=10; MAXLAG=30;
a=sqrt(3)/(sqrt(2)*10.^(SNR/20));
xnoise=2*a*(rand(N,1)-0.5);
Ps=mean((abs(xsine)).^2);
Pn=mean((abs(xnoise)).^2);
10*log10(Ps/Pn)
% to confirm that rx[ell]=K*DELTA[ell]
[rx, lag]=xcorr(xnoise, MAXLAG);
rx=rx/length(xnoise);
figure(4)
stem(lag, rx)
xlabel('$$\ell$$ (samples)','Interpreter', 'Latex');
ylabel('$$r_{X}[\ell]$$','Interpreter', 'Latex'); pause
```

Explain to your Colleagues:

- if the auto-correlation function is such that it consists of an impulse, how do we call this type of noise ? why ?
- the rationale that justifies that the above parameter "a" and that leads to a 10 dB SNR
- why running this code multiple times generates a practical SNR that is not exactly equal to, but is very close to, 10.0 dB.

b) The following Matlab code (which is a continuation of the above Matlab code) allows to evaluate numerically the SNR of the signal after filtering:

```
ynoise=filter(b,a,xnoise);
Ps=mean((abs(ysine)).^2);
Pn=mean((abs(ynoise)).^2);
10*log10(Ps/Pn)
```

Running this piece of Matlab (plus the code in **a)**) reveals that the output SNR is an improvement of the input SNR by about 6 dB. How do you explain that to your peers ?

c) [NOTE: this last question is more atypical than the previous ones, therefore, Student Ey should receive contributions/help/suggestions from the remaining group Students]

We state here, without proof, that if the impulse response of our LSI system is $h[n]$, and if the input auto-correlation is $r_x[\ell]$, then the output auto-correlation is given by $r_y[\ell] = h[\ell] * h^*[-\ell] * r_x[\ell]$. If we just consider the noise part of the input signal, it was shown in **a)** that $r_x[\ell] = K\delta[\ell]$, where K is the average power of the input noise.

(1) Show that $K = P_{n\_inp} = \frac{a^2}{3}$, where a is the theoretical value that is specified in the Matlab code of P2P Exercise 4 **a)** .

(2) Find the theoretical value of the average power of the output noise $P_{n\_out} = r_y[0]$ where $r_y[0] = \sum_{n=-\infty}^{+\infty} |h[n]|^2 \ P_{n\_inp}$. This value may be conveniently computed using the Parseval theorem in the Z-domain. Remember that, in this case, in the Z-domain, the region of convergence consists of a ring, which means that when you apply the contour line integral and, therefore, the residue theorem, only two poles matter.

Note: the solution should be: $P_{n\_out} = \frac{1+\alpha^2}{1-\alpha^2} \cdot \frac{1}{1-2\alpha^2\cos(2\beta)+\alpha^4} P_{n\_inp}$

(3) Confirm that your theoretical output SNR is easily computed as:

```
Ps=(MAG^2)/2;
Pn=(a^2)/3;
Pn=(1+ALFA^2)/(1-ALFA^2)*1/(1-
     2*(ALFA^2)*cos(2*BETA)+ALFA^4)*Pn;
10*log10(Ps/Pn)
% = 16.1422   (and the input SNR is 10 dB)
```

Extra P2P Exercise 5

In this exercise, we look at the PDF of all signals of interest in these set of P2P exercises. Use the following Matlab code to check the PDF of several signals:

```
 [H X]=hist(x,50); equalize=50/(max(x)-min(x));
bar(X, H/sum(H)*equalize, 0.5);
ylabel('PDF'); xlabel('x[n] amplitude'); pause
```

run this piece of code after setting x to different alternatives :

```
x=xnoise;
x=ynoise;
x=xsine;
x=ysine;
x=xnoise+xsine;
x=ynoise+ysine;
```

Can you anticipate what cases correspond to the following two plots ? And how do you explain their different shapes ?