SIGNALS AND SYSTEMS LABORATORY 9:
The Z Transform, the DTFT, and Digital Filters

INTRODUCTION

The Z transform pairs that one encounters when solving difference equations involve discrete-time signals, which are geometric (or exponential) in the time domain and *rational* in the frequency domain. MATLAB provides tools for dealing with this class of signals. Our goals in this lab are to

i. gain experience with the MATLAB tools

ii. experiment with the properties of the Z transform and the Discrete Time Fourier Transform

iii. develop some familiarity with filters, including the classical Butterworth and Chebychev lowpass and bandpass filters, all-pass filters, and comb filters.

THE Z TRANSFORM AND THE DTFT

The *Z transform* provides a *frequency domain* version of a discrete-time signal. Discrete time signals are sequences, and the Z transform is defined by

$$(1) \qquad h[k] \quad \xleftrightarrow{\text{Z transform}} \quad H(z) = \sum_{k=-\infty}^{\infty} h[k] z^{-k} \ .$$

Consider, for example, the elementary Z transform pair

$$(2) \qquad h[k] = a^k u[k] \quad \xleftrightarrow{Z} \quad H(z) = \frac{1}{1 - az^{-1}}$$

where $u[k]$ is the unit step function. The time domain sequence $h[k]$ and the frequency function $H(z)$ are alternate ways of describing the same signal. In the time domain, $h[k]$ is *exponential*. In the frequency domain, is *rational* or, by definition, the ratio of two *polynomials.* For discrete-time applications, we will use the representation

$$(3) \qquad H(z) = z^{\upsilon} \frac{B(z)}{A(z)} ,$$

where $B(z) = b[0] + b[1]z^{-1} + \ldots + b[m]z^{-m}$, $A(z) = 1 + a[1]z^{-1} + a[2]z^{-2} + \ldots + a[n]z^{-n}$, and υ is an integer. This numbering of the coefficients is not standard for MATLAB, if we are talking about polynomials, but it is consistent with the way the row vectors *a* and *b* are used in the filter function. The indices will be off by one, of course. The representation is unique if we demand that the end coefficients $b[0], b[m], \text{and } a[n]$ are not zero.

The *poles* of $H(z)$ are the roots of the denominator polynomial $A(z)$. At a pole, $H(z)$ becomes infinite. The *zeros* of $H(z)$ are the roots of the numerator polynomial $B(z)$. At a zero, $H(z)$ is zero. A pole-zero plot of $H(z)$ simply places the poles (using the symbol 'x') and the zeros (using the symbol 'o') on the complex plane. For stability, it is necessary that the poles of $H(z)$ be inside the unit disk, or in other words have absolute value less than one. The complex frequency response is computed by evaluating $H(z)$ on the unit circle $z = e^{j\theta}$, $0 \le \theta < 2\pi$.

This class of signals is most appropriate for discrete time linear filters. All filters which can be updated with a finite number of multiplications and additions per output sample will be in this class. In other words, these are the only filters that can be realized by DSP processors. The choice of the letter '*h*' for the above signal is commonly used for filters. In the time domain, $h[k]$ is the *unit pulse response sequence* of the filter. In the frequency domain, $H(z)$ is the *transfer function* of the filter. If we set $z = e^{j\theta}$, then we get the *complex frequency response function* $H(e^{j\theta})$. In fact, with $z = e^{j\theta}$, the Z transform becomes the *DTFT*, or *Discrete Time Fourier Transform*:

(4) $$h[k] \xleftrightarrow{\text{DTFT}} H(e^{j\theta}) = \sum_{-\infty}^{\infty} h[k]e^{-jk\theta} .$$

This transform has the inversion rule

(5) $$h[k] = \int_{-\pi}^{\pi} H(e^{j\theta})e^{jk\theta} \frac{d\theta}{2\pi} .$$

But there are conditions. The example in equation (2) will be consistent with equation (5) if and only if $|a| < 1$. Just as there was in the Laplace Transform, there will be a *Region of Convergence*, or ROC, and the choice of inversion must be consistent with the ROC. There are two important cases:

Application	**Condition on h**	**Region of Convergence**
Causal sequences	$h[k] = 0$, for $k < 0$	$\lvert z\rvert >$ max of the set of pole radii
Anti-Causal sequences	$h[k] = 0$, for $k < 0$	$\lvert z\rvert <$ min of the set of pole radii
Finite Energy sequences	$\sum_{-\infty}^{\infty} \lvert h[k]\rvert^2 < \infty$	$1-\varepsilon < \lvert z\rvert < 1+\varepsilon$

The first case is the one to respect when you are solving initial value problems, or difference equations for causal systems, like sampled data control systems and real-time digital filters. The second case is used when designing matched filters $H(-z)$ or factoring spectral polynomials as $S(z) = H(z)H(-z)$. The third case is the one to use when the application need not be causal, but must involve a bounded sequence. Such problems are common in communication systems. This is the case that one must assume when the discrete time Fourier transform is computed. In the causal case, poles outside the unit circle, i.e. with absolute values greater than one, mean that the system is unstable. In this case the sequence $h[k]$ will be unbounded, and the response to a unit pulse will explode. In the finite energy case, poles outside the unit circle mean only that the inverse transform $h[k]$ will not be causal, but it will be bounded. There will be no difference between these two cases if all the poles of $H(z)$ are inside the unit circle. Then $h[k]$ will be both bounded and causal.

RELATING THE LAPLACE AND Z TRANSFORMS AND THE FOURIER AND DISCRETE-TIME FOURIER TRANSFORMS

Laplace transforms use the *s-plane*, and frequency response is computed on the imaginary axis $s = j\omega$. The Z transform uses the *z-plane*, and the frequency response is computed on the unit circle $z = e^{j\theta}$. How do these differences come about? The Laplace transform is appropriate for *continuous-time* systems, while the Z transform is appropriate for *discrete-time* systems. In control system applications, discrete time systems are called *sampled-data systems*. In signal processing applications, they are called *digital filters* or *digital signal processors*. Suppose that we start with continuous time signals, and sample them to get discrete time signals. Let the sampling frequency be f_s, and the sampling period be $t_s = 1/f_s = 2\pi/\omega_s$. Consider a signal $x(t)$ with Laplace transform

(6) $$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt ,$$

and let $y[k] = x(kt_s)$ be the discrete time signal obtained by sampling $x(t)$. The Z Transform of the sequence of samples is

(7) $$Y(z) = \sum_{k=-\infty}^{\infty} y[k] z^{-k} \ .$$

This sum approximates the integral in equation (6), if we set

(8) $$z = e^{st_s} \ .$$

This relation maps the s plane into the z plane. It is important to understand this mapping if we want to think about using digital signal processing to approximate continuous time signal processing. The following table summarizes the key observations.

s plane		$z = e^{st_s}$	**z plane**
DC, or zero frequency	$s = 0$	$z = 1$	DC, in the z plane
Imaginary axis	$s = j\omega$	$z = e^{j\omega t_s}$	Unit circle
half sampling frequency	$s = j\omega_s / 2$	$z = -1$	highest usable frequency
right half plane	$\mathrm{Re}(s) > 0$	$\lvert z \rvert > 1$	outside the circle
left half plane	$\mathrm{Re}(s) < 0$	$\lvert z \rvert < 1$	inside the circle

Using equation (8) in equation (7), we get

(9) $$t_s Y(e^{st_s}) = t_s \sum_{k=-\infty}^{\infty} y[k] z^{-k} = \sum_{k=-\infty}^{\infty} x(kt_s) e^{-skt_s} t_s \approx \int_{-\infty}^{\infty} x(t) e^{-st} dt = X(s) \ .$$

Therefore the Z transform of *y*[*k*] can approximate the Laplace Transform of *x*(*t*), and the DTFT of *y*[*k*] can approximate the Fourier Transform of *x*(*t*), as in the following table:

Laplace Transform / Z Transform	$X(s) \approx t_s Y(z)$	provided $z = e^{st_s}$, and $\lvert \mathrm{imag}(s) \rvert < \omega_s / 2$
Fourier Transform / DTFT	$X(j\omega) \approx t_s Y(e^{j\theta})$	provided $\theta = \omega\, t_s = 2\pi\omega / \omega_s$ and $\lvert \omega \rvert < \omega_s / 2$, or $\lvert \theta \rvert < \pi$

The approximation has its limits, because going around the unit circle is a periodic motion. The DTFT $Y(e^{j\theta})$ is 2π periodic in $\theta = \omega t_s = 2\pi\omega / \omega_s$ and the approximation to $X(j\omega)$ is therefore periodic with period ω_s. Because of symmetry about zero, this means that the approximation is good only to the half sampling frequency $\omega_s / 2$. In digital signal processing, bandwidth is limited. For greater bandwidth, one must sample faster. One should always be aware of this. It is one of the two fundamental limitations, the other being finite word length effects.

There is a precise formula which relates $Y(e^{j\theta})$ and $X(j\omega)$, when $y[k] = x(kt_s)$, as opposed to the approximation that we have already mentioned. Whenever one *samples* in the time domain, then there will be *aliasing* in the frequency domain. The formula is this:

(10) $$y[k] = x(kt_s) \xleftrightarrow{DTFT} Y(e^{j\omega t_s}) = \frac{1}{t_s}\sum_{n=-\infty}^{\infty} X\big(j(\omega - n\omega_s)\big).$$

(sampling in time) $\longleftrightarrow$ (aliasing in frequency)

When $X(j\omega)$ is bandlimited to $|\omega| < \omega_s/2$, the formula $X(j\omega) = t_s \cdot Y(e^{j\omega \cdot t_s})$ will hold exactly for $|\omega| < \omega_s/2$. (This is the *sampling theorem*. We will study this in more depth in the next lab.)

MATLAB TOOLS FOR DISCRETE TIME SYSTEMS

Using the 'help' command, investigate the MATLAB standard functions: 'residue', 'conv', 'filter', 'impz', 'freqz', and 'invztrans'. Then look at the homebrew functions 'plotZTP.m', 'z_rev.m', and 'z_mult.m' located on the web page under 'Functions for Lab 9'. The MATLAB tool 'freqz' is very handy, but we will use 'plotZTP.m' to gain the same information, and more! (NB: The 'plotZTP.m' requires two other homebrew functions, 'gpfx.m', 'pzd.m')

Partial Fraction Expansions

If $B(z)$ has degree equal to that of $A(z)$, and if $A(z)$ does not have repeated roots, then the Z transform pair for $H(z)=B(z)/A(z)$ is

(11) $$h[k] = b[0]\delta[k] + \left[\sum_{m=1}^{n} \gamma[m]\alpha[m]^{k-1}\right] u[k-1] \xleftrightarrow{Z} H(z) = b[0] + \sum_{m=1}^{n} \frac{\gamma[m]}{z-\alpha[m]} = b[0] + z^{-1}\sum_{m=1}^{n} \frac{\gamma[m]}{1-\alpha[m]z^{-1}}.$$

The right hand side of the above is a *partial fraction expansion*, of $H(z)$. Under the conditions specified, the parameters $b[0], \gamma[1], \gamma[2]..., \gamma[m], \alpha[1], \alpha[2], ..., \alpha[m]$ can be computed using the MATLAB tool 'residue'. For example

```
»[b,a]=butter(3,.1) % construct an example H(z)

b = 0.0029 0.0087 0.0087 0.0029
        a =1.0000 -2.3741 1.9294 -0.5321

»[gamma,alpha,bzero]=residue(b,a) % do a partial fraction expansion
        gamma =
         -0.1102 - 0.1083i
         -0.1102 + 0.1083i
         0.2361
        alpha =
         0.8238 + 0.2318i
         0.8238 - 0.2318i
         0.7265
        bzero =0.0029
```

From this you can write down the partial fraction expansion for $H(z)$. Do It.

Simulating Digital Filters

The function 'conv' does sequence convolution. The MATLAB function 'filter' will approximate the convolution action of the discrete-time filter $H(z)$. The input parameters are the row vectors 'b' and 'a' which parameterize $H(z)$, and the long row vector x which represents the input. The output is a row vector 'y' whose size is the same as that of 'x'. Suppose that $h[k]$ is causal, and has a Z transform $H(z)$ given by equation (3), with υ equal to zero. Suppose that the input sequence $x[k]$ is also causal. Then the output sequence $y[k]$ will also be causal and will satisfy the difference equation

$$(12) \qquad a*y=b*x \text{, or } \sum_{i=0}^{n} a[i]y[k-i]=\sum_{i=0}^{m} b[i]x[k-i] \text{, or } y[k]=-\sum_{i=1}^{n} a[i]y[k-i]+\sum_{i=0}^{m} b[i]x[k-i]$$

The solution to this difference equation is the convolution of the input sequence and the pulse response sequence:

$$(13) \qquad y=h*x \text{, or } y[k]=\sum_{i=0}^{k} h[i]x[k-i] \text{, where } \sum_{i=0}^{n} a[i]h[k-i]=\sum_{i=0}^{m} b[i]\delta[k-i]$$

The MATLAB function 'filter' follows equation (12), while the MATLAB function 'conv' follows equation (13). There is a difference in the *tails* however. The length of the sequence filter(b,a,x) will be the same as the sequence 'x'. The length of the sequence conv(h,x) will be the sum of the lengths of 'h' and 'x' minus one. Try this:

```
»[b,a]=butter(3,.1); % construct a butterworth lowpass filter
        »x=randn(1,51); % construct a white noise sequence for k=0 to 50
        »y=filter(b,a,x); % construct 51 samples of the output
        »subplot(2,1,1),plot(y),axis([0 100 -1 1])
        »h=filter(b,a,[1,zeros(1,50)]); % construct pulse response, 0 to 50
        »y=conv(h,x); % now use the convolution sum
        »subplot(2,1,2),plot(y),axis([0 100 -1 1]) % compare with the previous y
```

The two graphs should agree for k=1 to 51, but the lower one will have more values. These extra values will not be correct, however, because the pulse response is good only out to k=50.

'*PlotZTP.m*': *Displaying Z Transform Pairs*

The following function will make a plot with four parts, a pole-zero diagram, a graph of $h[k]$ from -tmax to tmax, and graphs of the magnitude and phase of the complex frequency response function $H(e^{j\theta})$. It uses the representation of equation (5), and will assume that the sequence $h[k]$ is bounded. It will therefore plot non-causal sequences when necessary. An example of the output is given in Figure One for the third order Butterworth digital filter constructed previously. The frequency response plots use a normalized frequency, so that the maximum of 1 corresponds to the half sampling frequency $f_s/2$, or $\theta=\pi$. Notice that the time domain signal is plotted using the MATLAB function 'stem' to emphasize its discrete nature. The command used is »plotZTP(b,a,0,30). Type

```
»help plotZTP
 plotZTP(b,a,nu,tmax)
 Display Z transform pairs: plot
 (1) pole zero diagram,
 (2) frequency response
 (3) pulse response on [-tmax,tmax]
 h(k) <--> H(z)=(z^nu)*B(z)/A(z),
 B(z)=b0+b1*z^(-1)+...+bm*z^(-m), etc.
 b=[b0,b1,...,bm],a=[1,a1,a2,...,an]
 b0,bm,an should all be nonzero
 This function requires two other
 homebrew functions, 'gpfx.m' and 'pzd.m'
```

This is what you see:

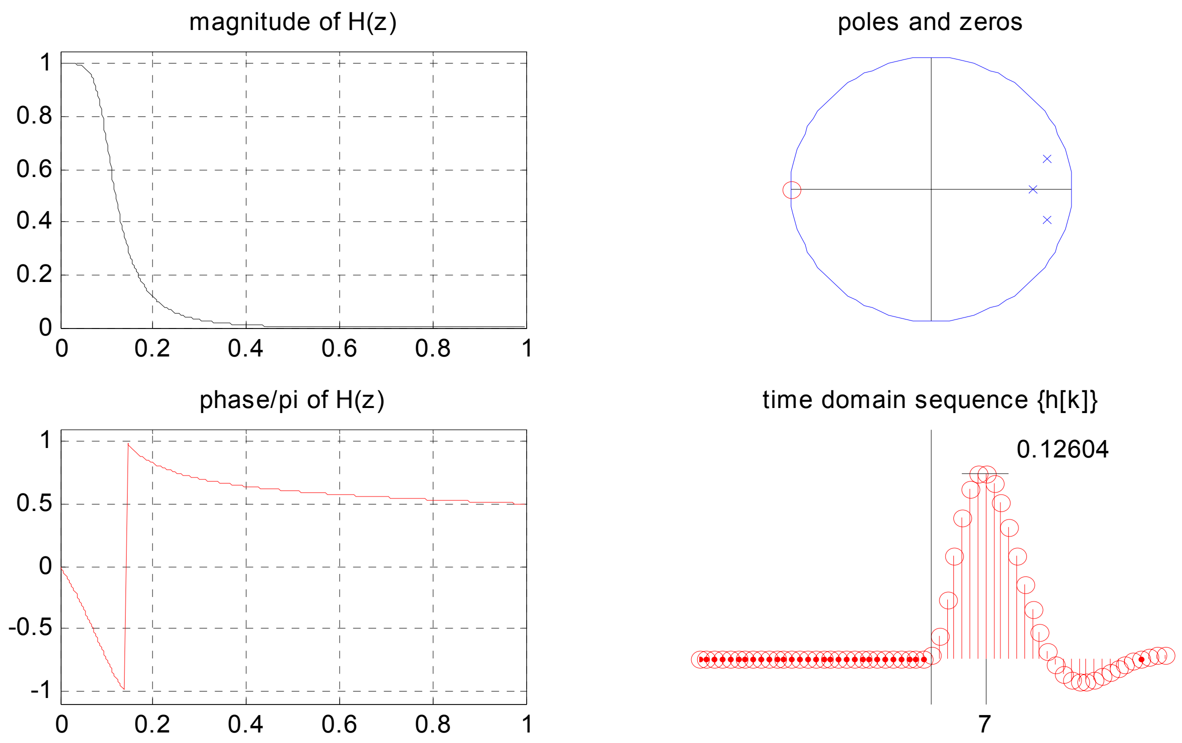

Figure One. A Butterworth filter Z-Transform pair.

PROPERTIES OF THE Z TRANSFORM

The following table contains Z transform properties that can be exploited with profit.

Property	**Time Domain**	**Z Domain**
convolution-multiplication	$y[k]=\sum_{i=-\infty}^{\infty} h[i]x[k-i]$	$Y(z)=H(z)X(z)$
time shift	$y[k]=x[k-m]$	$Y(z)=z^{-m}X(z)$
Modulation	$y[k]=\cos(k\theta_0)x[k]$	$Y(z)=\frac{1}{2}X(e^{j\theta_0}z)+\frac{1}{2}X(e^{-j\theta_0}z)$
time reversal	$y[k]=x[-k]$	$Y(z)=X(z^{-1})$
decimation by two	$y[k]=\begin{cases} x[k], \text{ if } k \text{ is even} \\ 0, \text{ if } k \text{ is odd}\end{cases}$	$Y(z)=\frac{1}{2}[X(z)+X(-z)]$

Assignment:

For this assignment, you will need the following homebrew functions: 'plotZTP.m', 'gpfx.m', 'pzd.m', 'zmodu.m', 'zmult.m', and 'z_rev.m'. These are located on the web page under 'Functions for Lab 9'. The rest of the functions will either be built-in MATLAB functions, or functions that you create.

1. *Multiplication-Convolution*

Construct two Z transform parameterizations, corresponding to Butterworth lowpass and bandpass filters as follows:

```
»[bh,ah]=butter(4,.6);
»[bx,ax]=butter(4,.4,'high');
```

Examine each of these signals using 'plotZTP.m' and print the results. (Use zero for the 'nu' parameter.) Then construct pulse response sequences for each and the convolution of the two and display them as follows:

```
»x=filter(bx,ax,[1,zeros(1,20)]);
»h=filter(bh,ah,[1,zeros(1,20)]);
»figure(1),subplot(3,1,1),stem(x)
»subplot(3,1,2),stem(h)
»y=conv(h,x);
»subplot(3,1,3),stem(y)
```

Now, using the tool for multiplying Z transforms, construct the Z transform representation of y via

```
»[by,ay,ny]=z_mult(bx,ax,0,bh,ah,0);
»figure(2),plotZTP(by,ay,ny,40)
```

The time domain panel should be the same as the previous graph of y.

2. *Time Shifts*

Produce two plots using »plotZTP(bh,ah,nu,30) with the shift parameter 'nu' equal to 5 and then -5. Comment on the resulting graphs. Are there changes in the magnitude plot? The phase plot? The time plot? Explain any changes.

3. *Modulation*

Use the homebrew function 'z_modu' to construct $G(z)=\frac{1}{2}\left[H(e^{j\theta_0}z)+H(e^{-j\theta_0}z)\right]$. If $H(z)$ is a lowpass filter, then $G(z)$ should look like a bandpass filter. (In this example $H(z)$ is a *finite impulse response* or *FIR* filter. It has no poles, only zeros.)

```
»hb=ones(1,20);ha=1;figure(1),plotZTP(hb,ha,0,20)
»[gb,ga,gn]=z_modu(hb,ha,0,.2);figure(2),plotZTP(gb,ga,gn,20)
```

Comment on the resulting graphs.

Assignment:

4. *Time reversal*

Construct $H(z)$ and make plots of $H(z)$, $H(z^{-1})$ and $H(z)\cdot H(z^{-1})$, as follows:

```
»hb=ones(1,10);ha=[1,zeros(1,9),0.5];figure(1),plotZTP(hb,ha,0,50)
»[hb1,ha1,hn1]=z_rev(hb,ha,0);
»figure(2),plotZTP(hb1,ha1,hn1,50)
»[hha,hhb,hhn]=z_mult(hb,ha,0,hb1,ha1,hn1);
»figure(3),plotZTP(hha,hhb,hhn,50)
```

Compare all plots for $H(z^{-1})$ with those of $H(z)$, Explain the differences. Then comment on all four parts of the plots for $H(z)\cdot H(z^{-1})$, relating them to the same graphs for $H(z)$. Why does the phase trivialize? Note that the magnitude of $H(z)\cdot H(z^{-1})$ is the square of the magnitude of $H(z)$.

5. *Decimation by Two*

Study the code for the tools 'z_rev.m' and 'z_modu.m'. Then write a tool to do decimation and produce $G(z)=\frac{1}{2}\left[H(z)+H(-z)\right]$ from $H(z)$. Your specifications should read

```
% [gb,ga,gn]=z_down(hb,ha,hn)
% G(z)=.5*[H(z)+H(-z)]
```

(The job can be done with five lines, having one statement per line.) Test your tool as follows:

```
»[gb,ga,gn]=z_down(hb,ha,0);figure(2),plotZTP(gb,ga,gn,40)
```

The sequence $g[k]$ should agree with the sequence $h[k]$ for even indices, but be zero for odd indices. The poles of $G(z)$ should consist of the poles of $H(z)$, and the negatives of the poles of $H(z)$. But don't expect the zeros to behave the same way. Provide printouts of the plot of $G(z)$ and the code for 'z_down.m'.
(Note: if you use $H(z)$ from part (4), there will be some pole-zero cancellation.)

6. *Butterworth and Chebychev Filters*

There are ways of choosing the poles and zeros of a filter $H(z)$ so that it acts as a lowpass or bandpass filter. A few classical filter designs are used extensively, including Butterworth and Chebychev filters. Butterworth was an English electrical engineer and Pafnuti L. Tchebycheff (1821-1894) was a professor of Mathematics at the University of Petrograd. Use the MATLAB 'help' command to investigate the functions 'butter', 'cheby1', and 'cheby2'. Then do several repetitions of the commands

```
»n=3;beta=.5;[b,a]=butter(n,beta);plotZTP(b,a,0,30)
```

except vary the filter order 'n' and normalized cutoff frequency 'beta' ($\beta = 2f_c/f_s$). Then experimentally determine an approximate formula for

i. the time position of the peak of the sequence $h[k]$,
ii. the height of the peak of the sequence $h[k]$,
iii. the number of samples between zero crossings in $h[k]$.

Report your results.

Assignment:

7. *All-pass filters*

Let $A(z) = 1 + a[1]z^{-1} + a[2]z^{-2} + \ldots + a[n]z^{-n}$ be any polynomial whose roots are all inside the unit circle. Let $B(z) = z^{-n}A(z^{-1}) = a[n] + a[n-1]z^{-1} + a[n-2]z^{-2} + \ldots + z^{-n}$. The coefficients of $B(z)$ are the same as those of $A(z)$, except in the reverse order. Now let $H(z) = B(z)/A(z)$. This filter will be an *all-pass* filter. Do this and print the resulting graphs:

```
»[b,a]=butter(5,.2);b=flipud(a')';plotZTP(b,a,0,40)
```

Show mathematically why the following are true:

i. The magnitude of the all-pass is one: $\left|H(e^{j\theta})\right| = 1$.
ii. The zeros of $B(z)$ are the reciprocals of the zeros of $A(z)$.
iii. The coefficients of $A(z^{-1})$ are time reversals of those for $A(z)$ and the coefficients of $B(z)$ are those of $A(z)$ made causal.

All-pass filters have nontrivial phase but trivial magnitude. Although it might seem that they are not particularly useful, they have a great many uses. For example, all Butterworth filters can be expressed as one half the sum of two all-pass filters.

8. *Comb filters*

Let $A(z) = 1 + a[n]z^{-n}$ be a polynomial whose interior coefficients are all zero. The filter $H(z) = b[0]/A(z)$ will have special properties. Do this:

```
»a=[1 0 0 0 0 0 0 0 0 0 .7];
»plotZTP(a,1,0,100)
»plotZTP(1,a,0,100)
```

You will see why one of these filters is called a *notch* filter and the other a *comb* filter when you see the magnitude plots. Print the resulting graphs. Show mathematically why the following are true:

i. The roots of $A(z)$ will be equally spaced around a circle of radius $\left|a[n]\right|^{1/n}$.
ii. The pulse response sequence $h[k]$ will be zero unless k is a multiple of n.
iii. The magnitude $\left|H(e^{j\theta})\right|$ is a periodic function in θ, with period $2\pi/n$.

(The same is true for the phase.)

9. We have seen in Lab 4 in EE311 that MATLAB treats the vector $a = \left[a[1] + a[2] + \ldots a[n+1]\right]$ as code for the polynomial $A(s) = a[1]s^{n} + a[2]s^{n-1} + \ldots + a[n+1]$. But in digital filtering we want this to code $A(z) = a[1] + a[2]z^{-1} + \ldots a[n+1]z^{-n}$. Why does this make no difference when computing poles and zeros, complex frequency responses, and so on?