SPECTRAL FACTORIZATION

Polynomial root finding and the Leja ordering

Spectral factorization is an important ingredient in the design of minimum-phase filters, and has many other applications. First we need to learn about polynomial root finding, and the problem of forming a polynomial from its roots.

1. THE `roots`, `poly`, AND `leja` COMMANDS
2. TOTAL LOSS OF ACCURACY
3. ZEROS OF HIGH ORDER
4. ZERO REFLECTIONS
5. REAL IMPULSE RESPONSES WITH SAME FREQUENCY RESPONSE MAGNITUDE
6. SPECTRAL FACTORIZATION
7. DIFFERENT SPECTRAL FACTORIZATIONS OF THE SAME TRANSFER FUNCTION
8. A PROGRAM FOR SPECTRAL FACTORIZATION
9. THE `seprts` PROGRAM
10. THE `sfact` PROGRAM
11. OTHER METHODS

THE roots, poly, AND leja COMMANDS

Consider the transfer function of an FIR filter of length N.

$$H(z) = \sum_{n=0}^{N-1} h(n)\, z^{-n} \tag{1}$$

$$= h(0) + h(1)\, z^{-1} + h(2)\, z^{-2} + \cdots + h(N-1)\, z^{-(N-1)}. \tag{2}$$

It can be written as

$$H(z) = \frac{h(0)\, z^{N-1} + h(1)\, z^{N-2} + h(2)\, z^{N-3} + \cdots + h(N-1)}{z^{N-1}}. \tag{3}$$

The zeros of $H(z)$ are the roots of the polynomial

$$h(0)\, z^{N-1} + h(1)\, z^{N-2} + h(2)\, z^{N-3} + \cdots + h(N-1). \tag{4}$$

For example, if the impulse response $h(n)$ is

$$h(n) = \{\underline{5},\ 2,\ 9,\ 6,\ 2\} \tag{5}$$

where the underlined number represents $h(0)$, then the zeros of $H(z)$ are the roots of the polynomial

$$5\, z^4 + 2\, z^3 + 9\, z^2 + 6\, z + 2. \tag{6}$$

The roots of a polynomial can be found with the Matlab command `roots`, as illustrated in the following code fragment.

```
>> h = [5 2 9 6 2];
>> r = roots(h)

r =
   0.1535 + 1.3306i
   0.1535 - 1.3306i
  -0.3535 + 0.3130i
  -0.3535 - 0.3130i
```

The Matlab command `poly` converts the roots of a polynomial $P(z)$ to the vector of polynomial coefficients. The command `poly` produces a monic polynomial, so it may need to be scaled. The following commands provide an example.

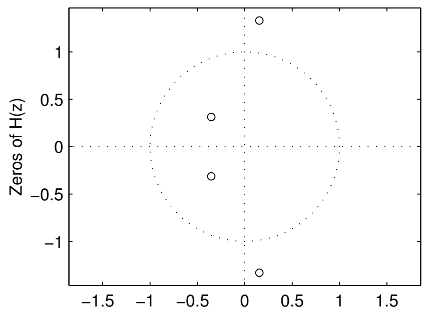
```
>> poly(r)

ans =

    1.0000    0.4000    1.8000    1.2000    0.4000

>> poly(r)*h(1)

ans =

    5.0000    2.0000    9.0000    6.0000    2.0000
```

Note that, because Matlab indexing begins with 1 rather than 0, `h(1)` represents $h(0)$. In some cases, the first coefficient `h(1)` is very small, and it is desirable to do the rescaling in another way, as in the following example.

```
>> p = poly(r)

p =

    1.0000    0.4000    1.8000    1.2000    0.4000

>> C = sqrt((h*h')/(p*p'))

C =
     5

>> C*p

ans =

    5.0000    2.0000    9.0000    6.0000    2.0000
```

A plot of the zeros of $H(z)$ can be obtained using the Matlab command `zplane`. The `zplane` command interprets a column vector to be the roots of a polynomial, and creates a plot of them with the unit circle.

```
zplane(r)
```

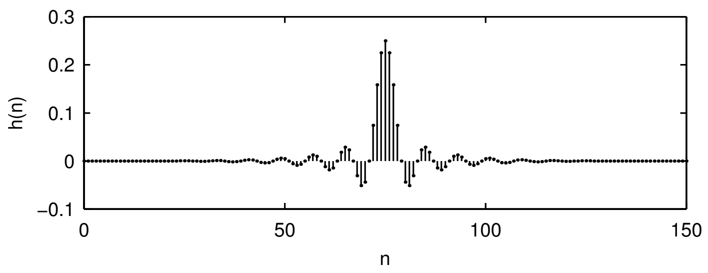


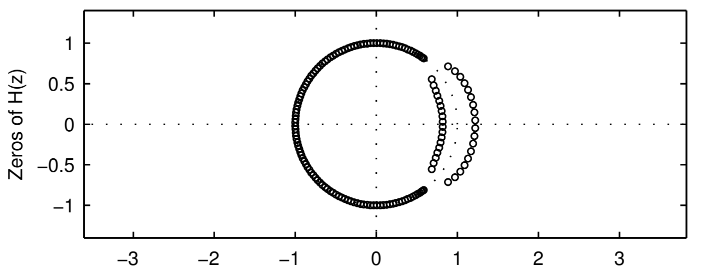
You can also supply `zplane` with `h` directly. The `zplane` command interprets a row vector to be the coefficients of a polynomial.

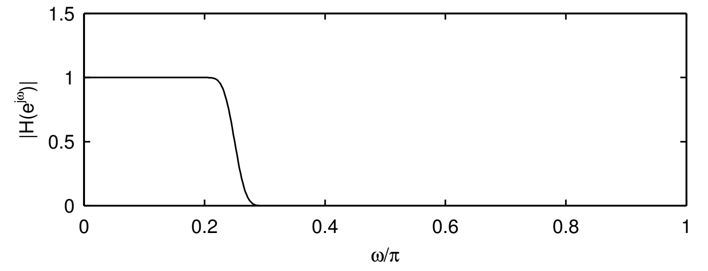
```
zplane(h)
```

In this case, the input argument `h` is a row vector, so the `zplane` command invokes the `roots` command and then produces the plot of the zeros of the polynomial.

The `roots` commands works even for quite long FIR filters (N over a hundred). For example, the following figure shows an FIR filter of length $N = 151$ and its zero diagram.







The following Matlab commands were used.

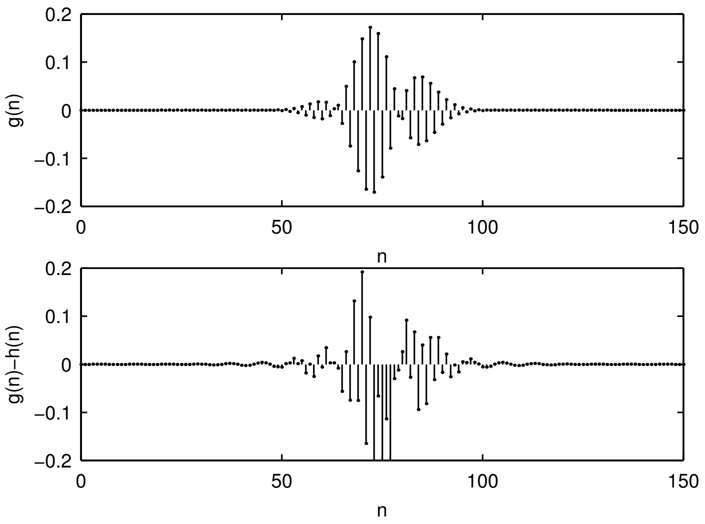
```
wp = 0.2;
ws = 0.3;
N = 150;
h = firls(N,[0 wp ws 1],[1 1 0 0]);
r = roots(h);
[H,w] = freqz(h);

subplot(3,1,1), stem(0:N,h,'.')
subplot(3,1,2), zplane(r);
subplot(3,1,3), plot(w/pi,abs(H))
```

TOTAL LOSS OF ACCURACY

A numerical problem can occur when using the command `poly` to convert the roots of a polynomial into the polynomial coefficients. Specifically, due to finite precision effects, the coefficients of the polynomial reconstructed from the roots may bear little resemblance to the original coefficients. This can happen when the degree of the polynomial is very high. The following example, where we attempt to retrieve `h` from `r`, illustrates this inaccuracy.

```
g = poly(r);                 % form poly coeffs from roots
C = sqrt((h*h')/(g*g'));    % normalization constant
g = C*g;                     % rescale poly coeffs

subplot(3,1,1), stem(0:N,g,'.')
subplot(3,1,2), stem(0:N,g-h,'.')
```

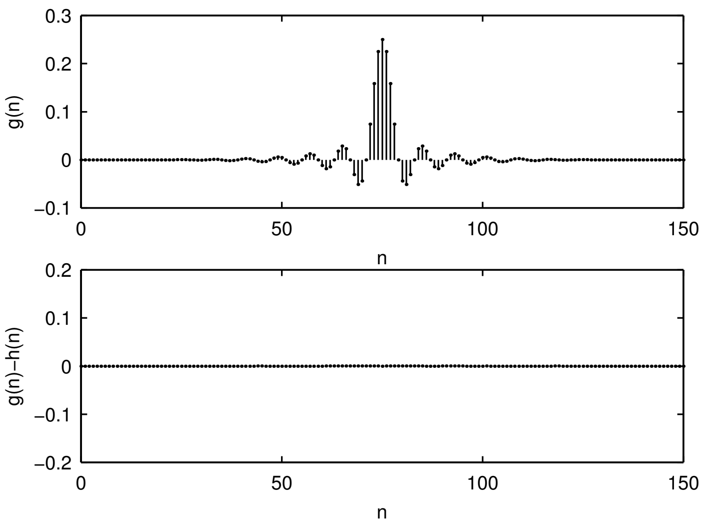
The following diagram illustrates `g` and the error between `h` and `g`. Ideally, we would have `h=g`.



From the figure, it is clear that the reconstruction of the coefficients from the roots can be completely unreliable if their lengths are very long.

A solution to this numerical problem is to order the roots of the polynomial appropriately before using the `poly` command. Specifically, the *Leja* ordering can dramatically improve the accuracy of the polynomial coefficients [2]. The Matlab program `leja`, (not currently included in Matlab, but available on the web), orders the roots so as to improve the numerical properties of the `poly` command. Performing again the previous example, but adding the Leja ordering, improves the accuracy as shown in the following example.

```
r = leja(r);                % order roots according to the Leja ordering
g = poly(r);                % form poly coeffs from roots
C = sqrt((h*h')/(g*g'));    % normalization constant
g = C*g;                    % rescale poly coeffs

subplot(3,1,1), stem(0:N,g,'.')
subplot(3,1,2), stem(0:N,g-h,'.')
```



ZEROS OF HIGH ORDER

We have noted in the previous section the complete loss of accuracy that can occur with the `poly` command when it is used for high order polynomials; and we have illustrated how this numerical problem can be alleviated with the Leja ordering. We should also mention a numerical problem that can occur with the `roots` command. When a polynomial $P(z)$ contains a zero z_o of degree 5 or higher, say, the accuracy of z_o is quite poor. For example, the polynomial $P(z) = \sum_n \binom{K}{n} z^n$ has a zero at $z = -1$ of order K. With $K = 6$ we have the following numerical result.

```
>> p = [1 6 15 20 15 6 1];
>> roots(p)

ans =

  -1.0039
  -1.0019 + 0.0034i
  -1.0019 - 0.0034i
  -0.9980 + 0.0034i
  -0.9980 - 0.0034i
  -0.9961
```

We can see what the error is, with the command

```
>> abs(roots(p)+1)

ans =

    0.0039
    0.0039
    0.0039
    0.0039
    0.0039
    0.0039
```

It can be seen that the accuracy is not good. With $K = 10$ we have the following numerical result.

```
>> p = [1 10 45 120 210 252 210 120 45 10 1];
>> roots(p)

ans =

  -1.0535
  -1.0430 + 0.0317i
  -1.0430 - 0.0317i
  -1.0159 + 0.0507i
  -1.0159 - 0.0507i
  -0.9831 + 0.0500i
  -0.9831 - 0.0500i
  -0.9573 + 0.0306i
  -0.9573 - 0.0306i
  -0.9476
```

We can see what the error is, with the command

```
>> abs(roots(p)+1)

ans =

    0.0535
    0.0534
    0.0534
    0.0532
    0.0532
    0.0528
    0.0528
    0.0525
    0.0525
    0.0524
```

The accuracy gets worse fast as the multiplicity of the zero increases. It is more than 10 times worse for $K = 10$ than it is for $K = 6$!

Finding a zero of even moderate degree is well known to be an inherently ill-conditioned problem. However, it does not usually occur except by design, in which case one knows the zero z_o and its degree. Then one can extract it from the polynomial with the `deconv` command.

ZERO REFLECTIONS

In this section, it will be shown that an FIR filter $h(n)$ is not uniquely determined by the magnitude of its frequency response $|H(e^{j\omega})|$. For a given FIR filter $h(n)$, we will see how to obtain other FIR filters that have exactly the same frequency response magnitude $|H(e^{j\omega})|$.

Consider the transfer function $H(z)$ of a length N FIR filter,

$$H(z) = z^{-(N-1)} \left(h(0)\, z^{N-1} + h(1)\, z^{N-2} + h(2)\, z^{N-3} + \cdots + h(N-1)\right). \tag{7}$$

We can factor the transfer function $H(z)$ as

$$H(z) = z^{-(N-1)} \cdot h(0) \cdot (z - c_1) \cdot (z - c_2) \cdot \cdots \cdot (z - c_{N-1}) \tag{8}$$

where the roots c_i are complex in general. $c_1, \ldots, c_{N-1}$ are called the zeros of $H(z)$. The frequency response $H^f(\omega) := H(e^{j\omega})$ is then given by

$$H^f(\omega) = e^{-j\,(N-1)\,\omega} \cdot h(0) \cdot (e^{j\omega} - c_1) \cdot (e^{j\omega} - c_2) \cdot \cdots \cdot (e^{j\omega} - c_{N-1}) \tag{9}$$

and the magnitude of the frequency response $|H^f(\omega)|$ is given by

$$|H^f(\omega)| = |h(0)| \cdot |e^{j\omega} - c_1| \cdot |e^{j\omega} - c_2| \cdot \cdots \cdot |e^{j\omega} - c_{N-1}|. \tag{10}$$

For convenience, let us define

$$\boxed{G_i(z) := z - c_i.} \tag{11}$$

Then we can write

$$H(z) = z^{-(N-1)} \cdot h(0) \cdot G_1(z) \cdot G_2(z) \cdots G_{N-1}(z) \tag{12}$$

and

$$\boxed{|H^f(\omega)| = |h(0)| \cdot |G_1^f(\omega)| \cdot |G_2^f(\omega)| \cdot \dots \cdot |G_{N-1}^f(\omega)|.} \tag{13}$$

Let us look at the magnitude of the frequency response of the function

$$G(z) = z - c. \tag{14}$$

Noting that $|a + j\,b|^2 = (a + j\,b) \cdot \overline{(a + j\,b)}$, we can write

$$\begin{aligned} |G^f(\omega)|^2 &= G^f(\omega) \cdot \overline{G^f(\omega)} && (15)\\ &= (e^{j\omega} - c) \cdot (e^{-j\omega} - \overline{c}) && (16)\\ &= 1 - c \cdot e^{-j\omega} - \overline{c} \cdot e^{j\omega} + c \cdot \overline{c} && (17)\\ &= 1 - 2\,\mathsf{Re}\{c \cdot e^{-j\omega}\} + |c|^2 && (18) \end{aligned}$$

so

$$|G^f(\omega)| = \sqrt{1 - 2\,\mathsf{Re}\{c \cdot e^{-j\omega}\} + |c|^2}. \tag{19}$$

Now consider the function $F(z)$ obtained from $G(z)$ by moving the zero c to $1/\overline{c}$ and scaling by $\overline{c}$,

$$F(z) = \overline{c} \cdot (z - 1/\overline{c}) = \overline{c}\,z - 1. \tag{20}$$

Then the zero of $F(z)$ is $z = 1/\overline{c}$, and $|F^f(\omega)|$ can be obtained as

$$\begin{aligned} |F^f(\omega)|^2 &= F^f(\omega) \cdot \overline{F^f(\omega)} && (21)\\ &= (\overline{c}\,e^{j\omega} - 1) \cdot (c\,e^{-j\omega} - 1) && (22)\\ &= \overline{c} \cdot c - c \cdot e^{-j\omega} - \overline{c} \cdot e^{j\omega} + 1 && (23)\\ &= |c|^2 - 2\,\mathsf{Re}\{c \cdot e^{-j\omega}\} + 1 && (24) \end{aligned}$$

so

$$|F^f(\omega)| = \sqrt{1 - 2\,\mathsf{Re}\{c \cdot e^{-j\omega}\} + |c|^2} = |G^f(\omega)|. \tag{25}$$

If we define

$$\boxed{F_i(z) := \overline{c_i} \cdot (z - 1/\overline{c_i}) = \overline{c_i}\, z - 1} \tag{26}$$

then we have

$$\boxed{|F_i^f(\omega)| = |G_i^f(\omega)|.} \tag{27}$$

That is, *moving the zero c to $1/\overline{c}$ and scaling by $\overline{c}$ does not affect the magnitude of the frequency response $G^f(\omega)$.*

Note that $1/\overline{c}$ can be seen to be a 'reflection' of c about the unit circle $|z| = 1$.

The result that $|F^f(\omega)| = |G^f(\omega)|$ can be used to see how $|H^f(\omega)|$ changes when one of the zeros of $H(z)$ is moved from $z = c_i$ to $z = 1/\overline{c_i}$. For example, suppose $h(n)$ is an FIR filter of length 5.

$$h(n) = \{\underline{5},\ 2,\ 9,\ 6,\ 2\} \tag{28}$$

Then

$$\begin{aligned} H(z) &= z^{-4}\,(5\,z^4 + 2\,z^3 + 9\,z^2 + 6\,z + 2) && (29)\\ &= z^{-4} \cdot h(0) \cdot (z - c_1) \cdot (z - c_2) \cdot (z - c_3) \cdot (z - c_4) && (30)\\ &= z^{-4} \cdot h(0) \cdot G_1(z) \cdot G_2(z) \cdot G_3(z) \cdot G_4(z) && (31) \end{aligned}$$

where $h(0) = 5$ and

$$c_1 = 0.1535 + j\,1.3306, \qquad c_2 = 0.1535 - j\,1.3306, \tag{32}$$

$$c_3 = -0.3535 + j\,0.3130, \qquad c_4 = -0.3535 - j\,0.3130. \tag{33}$$

We can write:

$$|H^f(\omega)| = |h(0)| \cdot |G_1^f(\omega)| \cdot |G_2^f(\omega)| \cdot |G_3^f(\omega)| \cdot |G_4^f(\omega)|. \tag{34}$$

Now let us define a new filter, for example,

$$H_1(z) := z^{-4} \cdot h(0) \cdot (z - c_1) \cdot (z - c_2) \cdot (z - c_3) \cdot \overline{c_4} \cdot (z - 1/\overline{c_4}) \tag{35}$$

$$= z^{-4} \cdot h(0) \cdot G_1(z) \cdot G_2(z) \cdot G_3(z) \cdot F_4(z); \tag{36}$$

then

$$|H_1^f(\omega)| = |h(0)| \cdot |G_1^f(\omega)| \cdot |G_2^f(\omega)| \cdot |G_3^f(\omega)| \cdot |F_4^f(\omega)| \tag{37}$$

$$= |h(0)| \cdot |G_1^f(\omega)| \cdot |G_2^f(\omega)| \cdot |G_3^f(\omega)| \cdot |G_4^f(\omega)| \tag{38}$$

$$= |H^f(\omega)| \tag{39}$$

because $|F_4^f(\omega)| = |G_4^f(\omega)|$.

The following Matlab commands can be used to calculate the coefficients of the newly obtained filter $h_1(n)$.

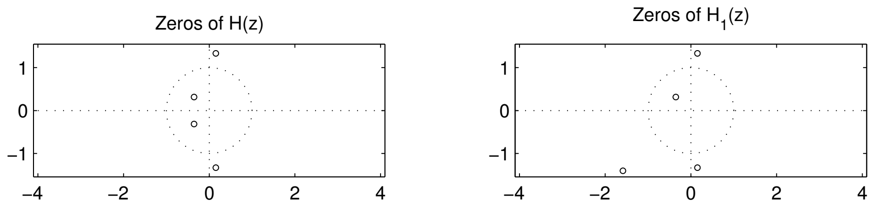
```
h = [5 2 9 6 2];         % poly coeffs
r = roots(h);            % obtain roots
ra = r([1 2 3]);         % roots 1,2,3
rb = conj(r([4]));       % root 4, conjugate
r1 = [ra; 1./rb];        % new set of roots
C = h(1)*rb;             % normalization const
h1 = C*poly(r1);         % obtain new poly coeffs
```

The polynomial coefficients `h1` are complex:

```
>> h1.'

  -1.7675 + 1.5651i
  -4.5922 + 0.6261i
  -3.3622 + 4.0334i
  -8.6699 + 1.5048i
  -3.1711 + 2.8080i
```

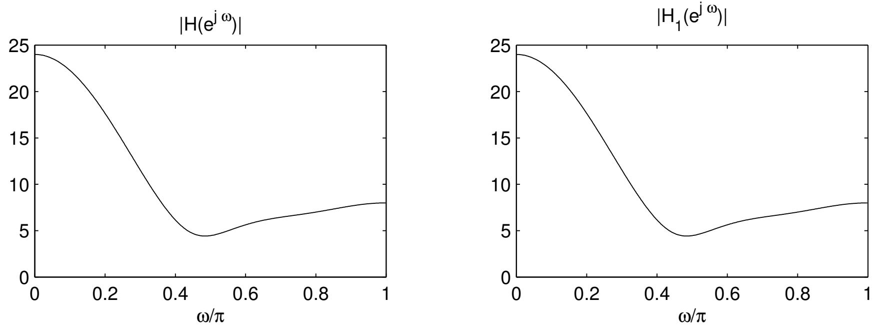
We can compare the zero-diagrams of $H(z)$ and $H_1(z)$ with the commands,

```
subplot(2,1,1), zplane(h);
subplot(2,1,2), zplane(h1);
```



The polynomial coefficients `h1` can not be real because the zeros of $H_1(z)$ do not occur in complex conjugate pairs.
It can be numerically verified that $|H_1^f(\omega)| = |H^f(\omega)|$ with the following commands.

```
[H,w] = freqz(h);
[H1,w] = freqz(h1);
subplot(1,2,1), plot(w/pi,abs(H))
subplot(1,2,2), plot(w/pi,abs(H1))
```



The fact that $|H_1^f(\omega)| = |H^f(\omega)|$ can also be verified using directly $h(n)$ and $h_1(n)$. Note that

$$|H^f(\omega)|^2 \quad \overset{DTFT}{\Longleftrightarrow} \quad h(n) * \overline{h(-n)} \tag{40}$$

$$|H_1^f(\omega)|^2 \quad \overset{DTFT}{\Longleftrightarrow} \quad h_1(n) * \overline{h_1(-n)} \tag{41}$$

If we define the autocorrelations

$$p(n) := h(n) * \overline{h(-n)} \tag{42}$$

and

$$p_1(n) := h_1(n) * \overline{h_1(-n)} \tag{43}$$

then $|H_1^f(\omega)|^2 = |H^f(\omega)|^2$ if and only if $p(n) = p_1(n)$. Accordingly, we can check that $|H_1^f(\omega)| = |H^f(\omega)|$ by evaluating the two autocorrelation signals with the following commands.

```
>> p = conv(h,conj(h(end:-1:1)))

p =

    10    34    75    94   150    94    75    34    10

>> p1 = conv(h1,conj(h1(end:-1:1)))

p1 =

   1.0e+02 *

  Columns 1 through 4

   0.1000             0.3400 - 0.0000i   0.7500 + 0.0000i   0.9400 - 0.0000i

  Columns 5 through 8

   1.5000             0.9400 + 0.0000i   0.7500 - 0.0000i   0.3400 + 0.0000i

  Column 9

   0.1000

>> real(p1)

ans =

  Columns 1 through 7

   10.0000   34.0000   75.0000   94.0000  150.0000   94.0000   75.0000

  Columns 8 through 9
```

```
   34.0000   10.0000
```

It can be seen that $p(n) = p_1(n)$ as expected. In this code fragment, notice that `h(end:-1:1)` produces a reversed version of `h`.

```
>> h

h =

     5     2     9     6     2

>> h(end:-1:1)

ans =

     2     6     9     2     5
```

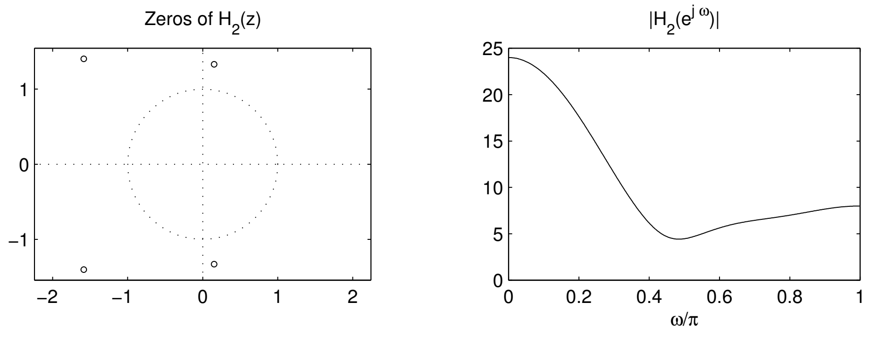
As a second example, let us define a new filter $h_2(n)$ by

$$H_2(z) := z^{-4} \cdot h(0) \cdot (z - c_1) \cdot (z - c_2) \cdot \overline{c_3} \cdot (z - 1/\overline{c_3}) \cdot \overline{c_4} \cdot (z - 1/\overline{c_4}). \tag{44}$$

Then we have $|H_2^f(\omega)| = |H^f(\omega)|$ as above. The coefficients of the filter $h_2(n)$ can be obtained with the following commands.

```
>> h = [5 2 9 6 2];          % poly coeffs
>> r = roots(h);             % obtain roots
>> ra = r([1 2]);            % roots 1,2
>> rb = conj(r([3 4]));      % roots 3, 4, conjugate
>> r2 = [ra; 1./rb];         % new set of roots
>> C = h(1)*prod(rb);        % normalization const
>> h2 = C*poly(r2)           % obtain new poly coeffs

h2 =

    1.1148    3.1928    5.9147    4.8072    8.9705
```

In this case the coefficients $h_2(n)$ are real, as the zeros of $H_2(z)$ come in complex conjugate pairs. The zero diagram of $H_2(z)$ and the frequency response magnitude $|H_2^f(\omega)|$ are shown in the following figure.



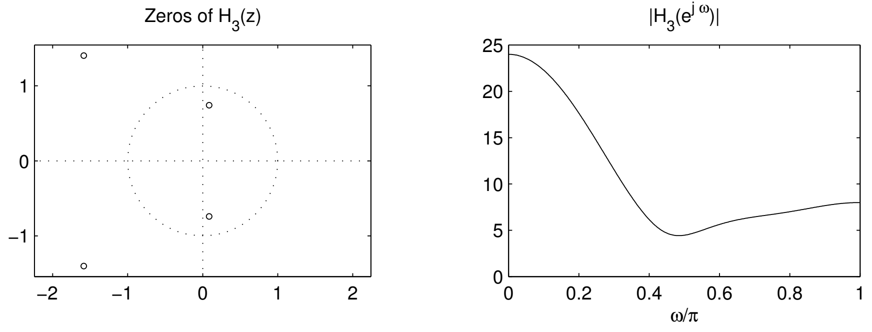
As a third example, let us define a new filter $h_3(n)$ by reflecting *all* of the zeros about the unit circle.

$$H_3(z) := z^{-4} \cdot h(0) \cdot \overline{c_1} \cdot (z - 1/\overline{c_1}) \cdot \overline{c_2} \cdot (z - 1/\overline{c_2}) \cdot \overline{c_3} \cdot (z - 1/\overline{c_3}) \cdot \overline{c_4} \cdot (z - 1/\overline{c_4}). \tag{45}$$

Then we have again $|H_3^f(\omega)| = |H^f(\omega)|$, and the coefficients $h_3(n)$ can be obtained as follows.

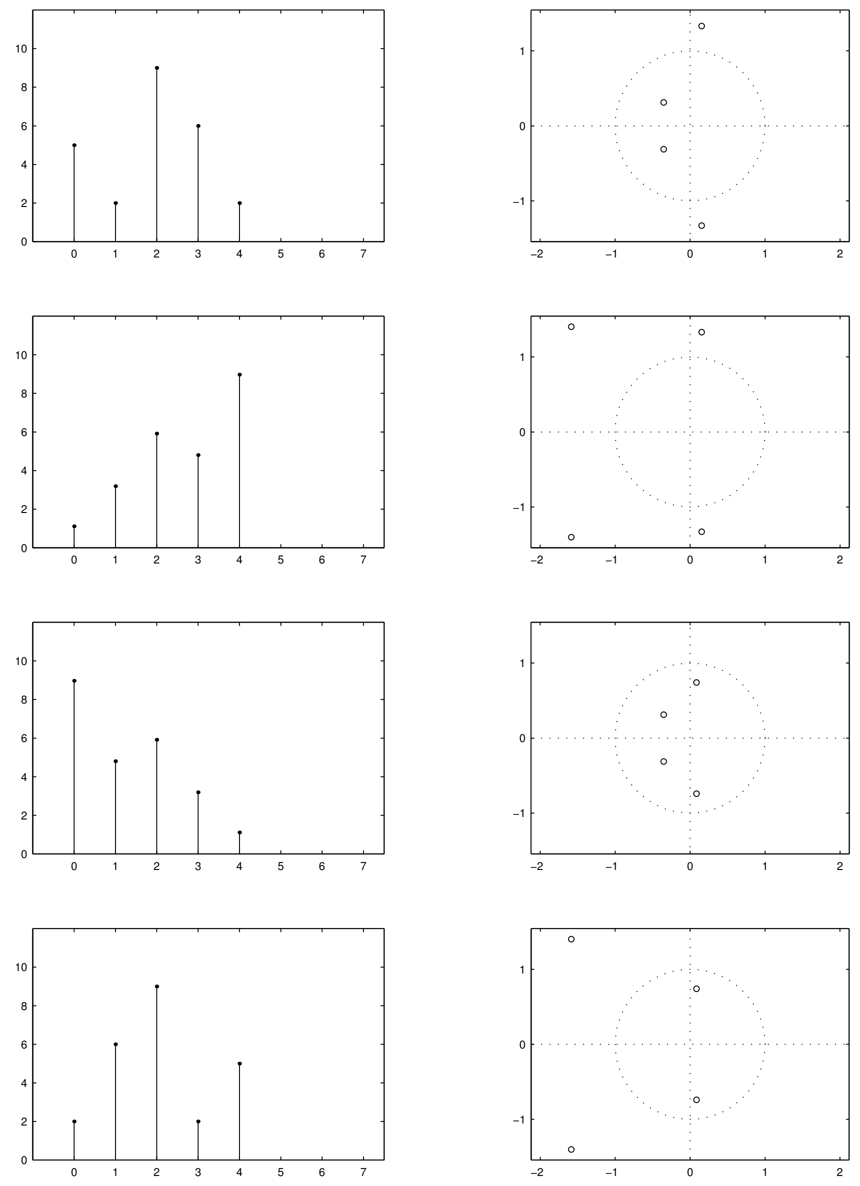
```
>> h = [5 2 9 6 2];              % poly coeffs
>> r = roots(h);                 % obtain roots
>> ra = r([]);                   % (no roots)
>> rb = conj(r([1 2 3 4]));      % roots 1,2,3,4, conjugate
>> r3 = [ra; 1./rb];             % new set of roots
>> C = h(1)*prod(rb);            % normalization const
>> h3 = C*poly(r3)               % obtain new poly coeffs

h3 =

    2.0000    6.0000    9.0000    2.0000    5.0000
```

In this case the coefficients $h_3(n)$ are exactly the same as $h(n)$, but in reverse order, $h_3(n) = h(4 - n)$. This will always be the case when all of the zeros are reflected. The zero diagram of $H_3(z)$ and the frequency response magnitude $|H_3^f(\omega)|$ are shown in the following figure.



For the filter $h(n)$ in this and the previous examples, it can be seen that there are 4 different real-valued FIR filters $h_i(n)$ with the same frequency response magnitude. They are shown together in the following figure.

REAL IMPULSE RESPONSES WITH SAME FREQUENCY RESPONSE MAGNITUDE



SPECTRAL FACTORIZATION

The (real-valued) spectral factorization problem we deal with here can be stated as follows.
Given $p(n)$, find $h(n)$ (real-valued) such that

$$p(n) = h(n) * h(-n), \tag{46}$$

or equivalently,

$$P(z) = H(z)\, H(1/z) \tag{47}$$

or

$$P^f(\omega) = |H^f(\omega)|^2. \tag{48}$$

The sequence $p(n)$ is called the autocorrelation of $h(n)$. We will focus here on the case where both $p(n)$ and $h(n)$ are FIR. There is also a complex-valued spectral factorization problem, where one has $p(n) = h(n) * \overline{h(-n)}$. But for convenience, we deal here with only the real case.
Note that not just any $p(n)$ can be factored as in (46). The sequence $p(n)$ must satisfy certain conditions.

1. $p(n)$ must be real and symmetric,

$$p(n) = p(-n).$$

   The sequence $h(n)*h(-n)$ is always a real symmetric sequence when $h(n)$ is real-valued, so if $p(n)$ is not real symmetric, then $p(n)$ can not be written in the form of (46).

2. If z_o is a zero of $P(z)$, then $1/z_o$ is also a zero of $P(z)$. This comes from (47). The zero z_o must be a zero of either $H(z)$ or $H(1/z)$ (or both). In either case, $1/z_o$ is a zero of the other and is therefore a zero of $P(z)$.

3. $P^f(\omega)$ must be real and nonnegative,

$$P^f(\omega) \geq 0$$

   This comes from (48). $|H^f(\omega)|$ can not be negative because it is the absolute value of a function, so if $P^f(\omega)$ is negative for any $-\pi \leq \omega \leq \pi$, then $p(n)$ can not be written in the form of (46). Note that $P^f(\omega)$ is always real when $p(n)$ is a real symmetric sequence.

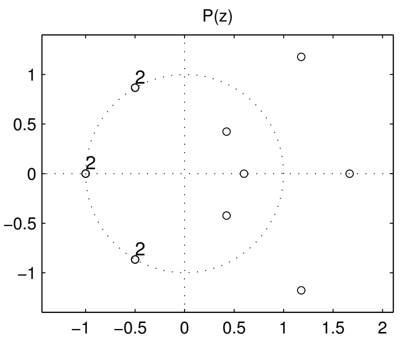
4. Any zero of $P(z)$ on the unit circle must be of *even* multiplicity.

   If z_o is a zero of $H(z)$, and if $h(n)$ is real as we have assumed, then $\overline{z}_o$ is also a zero of $H(z)$. Therefore, $1/\overline{z}_o$ is a zero of $H(1/z)$. Suppose z_o is on the unit circle ($z_o = e^{j\,\omega_o}$). If $e^{j\omega_o}$ is a zero of $H(z)$, then $1/\overline{z}_o = 1/e^{-j\omega_o} = e^{j\omega_o}$ is also a zero of $H(1/z)$ Similarly, if $e^{j\omega_o}$ is a zero of $H(z)$ of multiplicity d, then it is also a zero of $H(1/z)$ of multiplicity d. Clearly each zero of $P(z)$ on the unit circle must have even multiplicity, otherwise it will be impossible to construct $h(n)$ so that $P(z) = H(z)\,H(1/z)$.

Note that as we saw in the previous section, reflecting a zero across the unit circle, $|z| = 1$, does not affect $|H^f(\omega)|$. Therefore, from

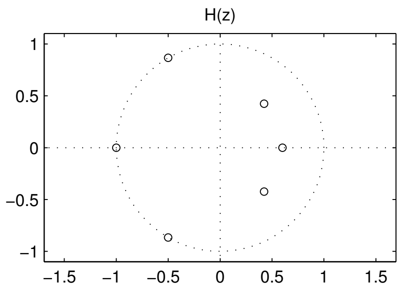
(48), once one solution $H^f(\omega)$ is found, other solutions can be obtained by reflecting zeros across the unit circle.

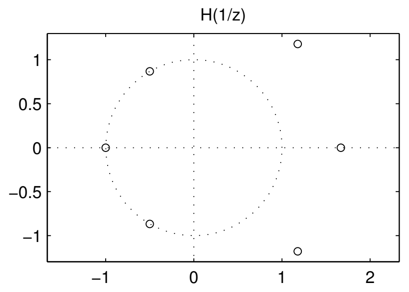
The solution to the spectral factorization problem is most conveniently explained using the Z-transform. Specifically, the diagram of the zeros of $P(z)$ indicate how $H(z)$ can be chosen. For example, suppose $P(z)$ has the zeros shown in the following diagram.

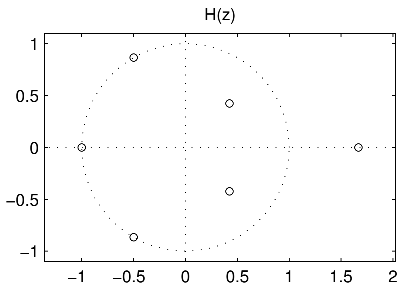


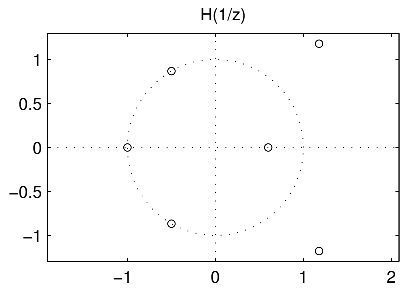
Then $H(z)$ can have any one of the four following zero diagrams shown in the left panels. The right panels show the zeros of $H(1/z)$.

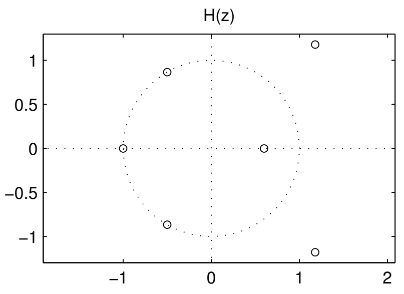
DIFFERENT SPECTRAL FACTORIZATIONS OF THE SAME TRANSFER FUNCTION

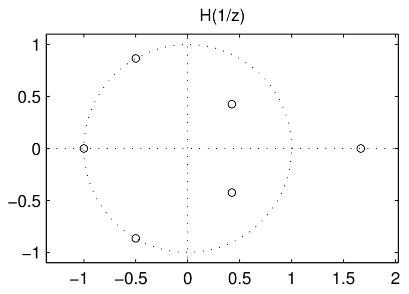


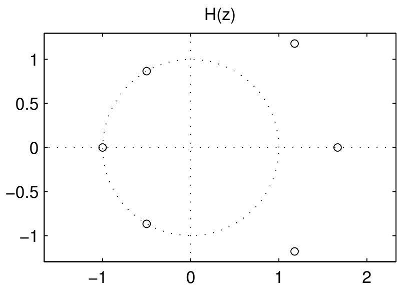


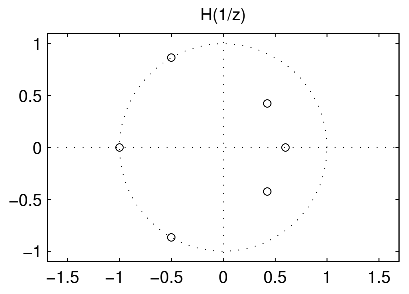












A PROGRAM FOR SPECTRAL FACTORIZATION

To write a program to perform spectral factorization, one can

1. first find the roots of $P(z)$,
2. identify an appropriate subset of those roots,
3. construct $h(n)$ from that subset of roots.

A simple way to do this is to take *all* the roots inside the unit circle, $|z| < 1$, and *half* of those roots on the unit circle, $|z| = 1$, and *none* of those roots outside the unit circle, $|z| > 1$. (This will give the minimum-phase solution.)
First we get the zeros using the Matlab command

```
rts = roots(p);
```

Numerically, we must allow a small margin about the unit circle, because we will not have $|z_o| = 1$ exactly with finite precision. To select the roots inside the unit circle, we can use the commands

```
SN = 0.0001;
irts = rts(abs(rts)<(1-SN));
```

The variable `SN` is a 'small number'. This command selects all roots z for which $|z| < (1 - \mathtt{SN})$.
To select the roots on the unit circle (within some margin), we can use the command

```
orts = rts((abs(rts)>=(1-SN)) & (abs(rts)<=(1+SN)));
```

This command selects all roots z for which $(1 - \texttt{SN}) < |z| < (1 + \texttt{SN})$. As each zero of $P(z)$ on the unit circle should be of even multiplicity, `orts` must be a vector of even length. We can check for this with the command,

```
N = length(orts);
if rem(N,2) == 1
   disp('Sorry, but there is a problem here')
end
```

The next step is to select an appropriate subset of `orts`. We can do this by sorting the zeros by their angles and then taking every second element of the sorted list. This can be done with the commands,

```
[a,k] = sort(angle(orts));
orts = orts(k(1:2:end));
```

Finally, we form the set of zeros of $H(z)$ by concatenating `irts` and `orts`,

```
r = [irts; orts];
```

Putting these commands together, we have the following program called `seprts`.

THE seprts PROGRAM

```
function r = seprts(p)
% r = seprts(p)
% This program is for spectral factorization.
% The roots on the unit circle must have even degree.
% Roots with high multiplicity will cause problems,
% they should be handled by extracting them prior to
% using this program.

SN = 0.0001;    % Small Number (criterion for deciding if a
                % root is on the unit circle).
rts = roots(p);

% The roots INSIDE the unit circle
irts = rts(abs(rts)<(1-SN));

% The roots ON the unit circle
orts = rts((abs(rts)>=(1-SN)) & (abs(rts)<=(1+SN)));
N = length(orts);
if rem(N,2) == 1
        disp('Sorry, but there is a problem (1) in seprts.m')
        r = [];
return
end

% Sort roots on the unit circle by angle
[a,k] = sort(angle(orts));
orts = orts(k(1:2:end));

% Make final list of roots
r = [irts; orts];
```

It remains only to form $h(n)$ from the subset of the zeros of $P(z)$ that we selected. We can use the command poly to do this — however as we saw earlier, the numerical errors that occur in the poly command can be greatly reduced by using the Leja ordering,

```
r = leja(r);
h = poly(r);
```

Note that poly gives a monic polynomial. The monic polynomial must be scaled correctly, otherwise we will not have (46). Suppose $\tilde{h}(n)$ is the monic polynomial which we need to normalize,

$$h(n) = C\,\tilde{h}(n).$$

To find the normalization constant, we can set $n = 0$ in equation (46) to get,

$$p(0) = |h(0)|^2 + |h(1)|^2 + |h(2)|^2 + \cdots + |h(N-1)|^2 \tag{49}$$

$$= C^2 \left[|\tilde{h}(0)|^2 + |\tilde{h}(1)|^2 + |\tilde{h}(2)|^2 + \cdots + |\tilde{h}(N-1)|^2\right] \tag{50}$$

Solving for C we get

$$C = \sqrt{\frac{p(0)}{|\tilde{h}(0)|^2 + |\tilde{h}(1)|^2 + |\tilde{h}(2)|^2 + \cdots + |\tilde{h}(N-1)|^2}}.$$

Note that $p(0)$ will always be the maximum value among all of $p(n)$, so we can normalize h with the command

```
h = h*sqrt(max(p)/sum(abs(h).^2));
```

The following example illustrates the procedure.

```
>> p = [-2 1 2 7 2 1 -2];
>> r = seprts(p)

r =
   0.5000
  -0.5000 - 0.8660i
  -0.5000 + 0.8660i

>> h = poly(r)

h =

   1.0000             0.5000 - 0.0000i   0.5000 - 0.0000i  -0.5000 + 0.0000i

>> h = real(h)       % (discard zero imag part)

h =
    1.0000    0.5000    0.5000   -0.5000

>> h = h*sqrt(max(p)/sum(abs(h).^2))

h =
    2.0000    1.0000    1.0000   -1.0000
```

We can alway test $h(n)$ to ensure that it really is a spectral factor of $p(n)$. We just need to check (46).

```
>> conv(h,h(end:-1:1))

ans =

   -2.0000    1.0000    2.0000    7.0000    2.0000    1.0000   -2.0000
```

The check shows that $h(n)$ is correct. Remember, though, that there are other $h(n)$ that also work, corresponding to non-minimum phase solutions.

THE `sfact` PROGRAM

Collecting these commands together, we have the program called `sfact`.

```
function [h,r] = sfact(p)
% [h,r] = sfact(p)
% spectral factorization of a polynomial p.
% h: new polynomial
% r: roots of new polynomial
%
% % example:
%    g = rand(1,10);
%    p = conv(g,g(10:-1:1));
%    h = sfact(p);
%    p - conv(h,h(10:-1:1)) % should be 0

% Required subprograms: seprts.m, leja.m
% leja.m is by Markus Lang, and is available from the
% Rice University DSP webpage: http://www.dsp.rice.edu/

if length(p) == 1
   h = p;
   r = [];
   return
end

% Get the appropriate roots.
r = seprts(p);

% Form the polynomial from the roots
r = leja(r);
h = poly(r);

if isreal(p)
   h = real(h);
end

% normalize
h = h*sqrt(max(p)/sum(abs(h).^2));
```

COMPLEX SPECTRAL FACTORIZATION

In fact, `sfact` also works for complex spectral factorization, as the following example illustrates.

```
>> i = sqrt(-1);
>> p = [1-8*i 5+4*i 2-1*i 25 2+1*i 5-4*i 1+8*i];
>> r = seprts(p)

r =

   0.0935 + 0.9599i
  -0.6975 - 0.5233i
   0.4392 - 0.3486i

>> h = poly(r)

h =

   1.0000             0.1648 - 0.0880i   0.3240 - 0.3162i   0.0585 + 0.4679i

>> h = h*sqrt(max(p)/sum(abs(h).^2))

h =

   4.1349             0.6813 - 0.3640i   1.3397 - 1.3074i   0.2418 + 1.9347i
```

Again, we can test $h(n)$ to ensure that it really is a spectral factor of $p(n)$. We just need to check $p(n) = h(n) * \overline{h(-n)}$.

```
> conv(h,conj(h(end:-1:1)))
```

```
ans =

  Columns 1 through 4

   1.0000 - 8.0000i   5.0000 + 4.0000i   2.0000 - 1.0000i  25.0000

  Columns 5 through 7

   2.0000 + 1.0000i   5.0000 - 4.0000i   1.0000 + 8.0000i
```

The check shows that $h(n)$ is correct.

OTHER METHODS

There are other ways to do spectral factorization, for example, the cepstral method [3] or matrix factorization [4]. More references about spectral factorization are given in [5]. We also note that software for polynomial root finding and spectral factorization is described in [1].

References

[1] M. Lang and B.-C. Frenzel. Polynomial root finding. *IEEE Signal Processing Letters*, 1(10):141–143, October 1994.

[2] Nachtigal, Reichel, and Trefethen. A hybrid GMRES algorithm for nonsymmetric linear systesm. *SIAM J. Matr. Anal. and Appp.*, 13:796–825, July 1992.

[3] H. W. Schüssler and P. Steffen. Some advanced topics in filter design. In J. S. Lim and A. V. Oppenheim, editors, *Advanced Topics in Signal Processing*, chapter 8, pages 416–491. Prentice Hall, 1988.

[4] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[5] S.-P. Wu, S. Boyd, and L. Vandenberghe. FIR filter design via spectral factorization and convex optimization. In B. Datta, editor, *Applied and Computational Control, Signals and Circuits*, chapter 2, pages 51–81. Birkhauser, 1997.