Signal Processing with MATLAB

Welcome and Goodluck

Syed Khaleel Ahmed

Dept. of Electronics and Communication Engg.,
Universiti Tenaga Nasional

April 15, 2016

Objectives

On the successful completion of this course, you should be able to

- use MATLAB to represent signals and systems,
- use MATLAB to perform time-domain and frequency-domain analysis,
- use MATLAB to design linear filters,
- use the signal processing GUIs available in MATLAB, and
- use the signal processing blockset.

Outline

Outline

Introduction: Digital Signal Processing (DSP)

1. What is DSP?
2. Continuous-Time and Discrete-Time
3. Analog and digital signals
4. Signal processing
5. Development of DSP
6. Digital Signal Processors (DSPs)
7. Applications of DSP
8. MATLAB and the Signal Processing Toolbox

Note

for a summary of functions in the Signal Processing Toolbox type

```
>> help signal
```


Outline

1. *Introduction*
2. *Representation of Signals*
3. Representation of Systems (Filters)
4. Time-Domain Analysis
5. Frequency Domain Analysis
6. Filter Design
7. Graphical User Interface
8. Signal Processing Blockset

Representation of Signals

What is a signal?

- *functions* of one or more independent variables, which
  - contain *information* about the
  - *behavior* or nature of some
  - *process* or phenomenon.

Example

1. *Current in an electrical circuit is a function of time.*
2. *Photograph - brightness function of 2 spatial variables.*
3. *Video - brightness function of 3 variables (2 spatial and 1 time).*

Representation of Signals (contd.)

Signal Classification

Several, two are very critical

1. Continuous-Time and Discrete-Time

   $x(t)$ is a *continuous-time* signal if it has *a value defined at each point in time t*.
   Example: Current through a resistor.

   $x[n]$ is a *discrete-time signal if it has a value defined only at discrete points in time n*.
   Example: The Stock market index.

   The *independent variable* may be *inherently discrete* or may *become discrete due to sampling* of a continuous-time signal.

Signal Classification (contd.)

Several, two are very critical

1. Continuous-Time and Discrete-Time
2. Analog and Digital

$x(t)$ or $x[n]$ is an *analog signal* if it can take any *real or complex value*.

Example: Current through a resistor.

$x(t)$ or $x[n]$ is a *digital signal* if it can take, *values only from a discrete set*.

Example:
Current through a resistor *as measured by a digital ammeter*.

Continuous-Time Use *Vectors* or *Arrays*

EXAMPLE

```
>> t = -5:.05:5;
>> x = sin( pi*t );
>> plot( t, x ), grid
>> title('Sinusoidal Signal x(t)=sin \pi t')
>> axis( [ -5 5 -1.1 1.1 ] )
```

Other MATLAB Defined: cos, tan, exp, sinc, square, sawtooth, chirp

Not MATLAB Defined: Unit-Impulse, Unit-Step, Unit-Ramp, Rectangular Pulse.

READER

Write a MATLAB function to plot a continuous-time unit-step signal.

Discrete-Time Use *Vectors* or *Arrays*

EXAMPLE

```
>> n = -10:10;
>> x = sinc( pi*n/6 );
>> stem( n, x, 'filled' ), grid
>> title( 'Sinc Signal x[n]=sinc \pi n' )
```

Other MATLAB Defined: cos, tan, exp, sinc, square, sawtooth, chirp

Other not MATLAB Defined: Unit-Impulse, Unit-Step, Unit-Ramp, Rectangular Pulse.

READER

Write a MATLAB function to plot a discrete-time unit-step signal.

Discrete-Time (contd.) Use *Vectors* or *Arrays*

EXAMPLE (GENERATING OTHER SIGNALS)

```
>> n = -10:10;
>> d = ( n==0 );
>> subplot( 211 ), stem( n, d, 'filled' )
>> title( 'Unit Impulse \delta[n]' )
>>
>> u = ( n>=0 );
>> subplot( 212 ), stem( n, u, 'filled' )
>> title( 'Unit Step u[n]' )
```

Other not MATLAB Defined: Unit-Impulse, Unit-Step, Unit-Ramp, Rectangular Pulse.

Representation of Signals (contd.)

Discrete-Time (contd.) Use *Vectors* or *Arrays*

Example (General signals using Matlab defined functions)

```
>> n = -10:10;
>>
>> x1 = square( pi*n/3 );
>> subplot(311), stem( n, x1, 'filled' ), title('x1')
>>
>> x2 = square( pi*n/3 ) + 1;
>> subplot(312), stem( n, x2, 'filled' ), title('x2')
>>
>> x3 = ( square( pi*n/3 ) + 1 )/2;
>> subplot(313), stem( n, x3, 'filled' ), title('x3')
```

Other not Matlab Defined:

Unit-Impulse, Unit-Step, Unit-Ramp, Rectangular Pulse.

Outline

Representation of Systems (Filters)

1. Transfer Function – Continuous-Time tf

$$G(s) = \frac{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}.$$

Example

$$G(s) = \frac{s+4}{s^3 + 6s^2 + 11s + 6}$$

```
>> num = [1 4];
>> den = [1 6 11 6];
>> sysG = tf(num,den)

Transfer function:
        s + 4
----------------------
s^3 + 6 s^2 + 11 s + 6
```


Representation of Systems (contd.)

2. Zero-Pole-Gain – Continuous-Time zpk

$$G(s) = \frac{K(s+b_1)(s+b_2)\cdots(s+b_m)}{(s+a_1)(s+a_2)\cdots(s+a_n)}.$$

Example

$$G(s) = \frac{10(s+4)}{(s+1)(s+2)(s+3)}$$

```
>> z = [ -4 ];
>> p = [ -1; -2; -3 ];
>> k = 10;
>> sysG = zpk( z, p, k )

Zero/pole/gain:
    10 (s+4)
-----------------
(s+1) (s+2) (s+3)
```

3. Partial Fraction Expansion – Continuous-Time

residue

$$G(s) = \frac{A_1}{s+a_1} + \frac{A_2}{s+a_2} + \cdots + \frac{A_n}{s+a_n}.$$

EXAMPLE

$$G(s) = \frac{s+4}{s^3+6s^2+11s+6}$$

```
>> num = [ 1 4 ];
>> den = [ 1 6 11 6 ];
>> [r p k] = residue(num,den)
r =

    0.5000
   -2.0000
    1.5000

p =

   -3.0000
   -2.0000
   -1.0000

k =
     []
```


1. Transfer Function – Discrete-Time

$$G(z) = \frac{b_0 z^n + b_1 z^{n-1} + \cdots + b_{n-1} z + b_n}{a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n} \quad (1)$$

$$= \frac{b_0 + b_1 z^{-1} + \cdots + b_{n-1} z^{-(n-1)} + b_n z^{-n}}{a_0 + a_1 z^{-1} + \cdots + a_{n-1} z^{-(n-1)} + a_n z^{-n}}. \quad (2)$$

2. Zero-Pole-Gain – Discrete-Time

$$G(z) = \frac{(z-\beta_1)(z-\beta_2)\cdots(z-\beta_n)}{(z-\alpha_1)(z-\alpha_2)\cdots(z-\alpha_n)} \quad (3)$$

$$= \frac{(1-\beta_1 z^{-1})(1-\beta_2 z^{-1})\cdots(1-\beta_n z^{-1})}{(1-\alpha_1 z^{-1})(1-\alpha_2 z^{-1})\cdots(1-\alpha_n z^{-1})}. \quad (4)$$

3. Partial Fraction Expansion – Discrete-Time

$$G(z) = \frac{A_1 z}{z-\alpha_1} + \frac{A_2 z}{z-\alpha_2} + \cdots + \frac{A_n z}{z-\alpha_n} \quad (5)$$

$$= \frac{A_1}{1-\alpha_1 z^{-1}} + \frac{A_2}{1-\alpha_2 z^{-1}} + \cdots + \frac{A_n}{1-\alpha_n z^{-1}}. \quad (6)$$

1. Transfer Function – Discrete-Time

filt

EXAMPLE

$$G(z) = \frac{1+z^{-1}}{1-0.75z^{-1}+0.125z^{-2}}$$

```
>> numTF = [ 1 1 ];
>> denTF = [ 1 -0.75 0.125 ];
>> sysGTF = filt( numTF, denTF )

Transfer function:
         1 + z^-1
--------------------------
1 - 0.75 z^-1 + 0.125 z^-2

Sampling time: unspecified
```


1. Transfer Function – Discrete-Time (contd.)

filt

EXAMPLE

$$G(z) = \frac{1+z^{-1}}{1-0.756z^{-1}+0.125z^{-2}}; \quad T_s = 0.001$$

```
>> numTF = [ 1 1 ];
>> denTF = [ 1 -0.75 0.125 ];
>> sysGTF = filt( numTF, denTF, 1e-3 )

Transfer function:
         1 + z^-1
--------------------------
1 - 0.75 z^-1 + 0.125 z^-2

Sampling time: 0.001
```

2. Zero-Pole-Gain – Discrete-Time   zpk

EXAMPLE

$$G(z) = \frac{10\,(1+z^{-1})}{(1-0.5z^{-1})\,(1-0.25z^{-1})} = \frac{10z(z+1)}{(z-0.5)\,(z-0.25)}$$

```
>> zeroG = [ 0; -1 ];
>> poleG = [ 0.5; 0.25 ];
>> gainG = 10;
>> sysGZPK = zpk( zeroG, poleG, gainG, -1 )
Zero/pole/gain:
    z (z+1)
----------------
(z-0.5) (z-0.25)

Sampling time: unspecified
```


2. Zero-Pole-Gain – Discrete-Time (contd.)   zpk (contd.)

EXAMPLE

$$G(z) = \frac{10\,(1+z^{-1})}{(1-0.5z^{-1})\,(1-0.25z^{-1})} = \frac{10z(z+1)}{(z-0.5)\,(z-0.25)}; \quad T_s = 10^{-3}$$

```
>> zeroG = [ 0; -1 ];
>> poleG = [ 0.5; 0.25 ];
>> gainG = 10;
>> sysGZPK = zpk( zeroG, poleG, gainG, 1e-3 )
Zero/pole/gain:
    z (z+1)
----------------
(z-0.5) (z-0.25)

Sampling time: 0.001
```


3. Partial Fraction – Discrete-Time   residuez

EXAMPLE

$$G(z) = \frac{1+z^{-1}}{(1-0.5z^{-1})\,(1-0.25z^{-1})} = \frac{6}{1-0.5z^{-1}} - \frac{5}{1-0.25z^{-1}}$$

```
>> resG = [ 6; -5 ];
>> poleG = [ 0.5; 0.25 ];
>> kG = [];
>> [ numPF denPF ] = residuez( resG, poleG, kG )

numPF =
     1     1

denPF =
    1.0000   -0.7500    0.1250
```


4. State-Space – Discrete-Time   ss

EXAMPLE

$$x[n+1] = A\,x[n] + B\,u[n]; \quad y[n] = C\,x[n] + D\,u[n]$$

```
>> A = [ 1 6; 2 -5 ];
>> B = [ 0.5; 0.25 ];
>> C = [ 1 2 ];
>> D = 0;
>>
>> sysG = ss( A, B, C, D );
>>
>> sysG = ss( A, B, C, D, -1 );
>>
>> sysG = ss( A, B, C, D, 0.001 );
```

Representation of Systems (contd.)

5. Second-Order Section – Discrete-time only

Example

$$G(z) = \frac{1+z^{-1}+z^{-2}}{1-z^{-2}} \times \frac{-2+3z^{-1}+z^{-2}}{1+10z^{-1}+z^{-2}}$$

```
>> sos = [  1 1 1    1  0 -1
           -2 3 1    1 10  1  ]

sos =

     1     1     1     1     0    -1
    -2     3     1     1    10     1
```


Representation of Systems (contd.)

Other Representations

6. Lattice Structure – Discrete-time only

7. Convolution Matrices – Discrete-time only

Representation of Systems (contd.)

Transformations between Representations

- zpk
- tf
- ss
- residuez for DT,

  residue for CT

- tf2sos Transfer Function to Second Order Section conversion.
- sos2tf,
- zp2sos,
- sos2zp,
- sos2ss,
- ss2sos

Representation of Systems (contd.)

Transformations between representations (contd.)

Example

$$G(z) = \frac{1+z^{-1}+z^{-2}}{1-z^{-2}} \times \frac{-2+3z^{-1}+z^{-2}}{1+10z^{-1}+z^{-2}}$$

```
>> sos = [ 1 1 1   1  0 -1;   -2 3 1   1 10  1  ];
>>
>> [b, a ] = sos2tf( sos )

b =
    -2     1     2     4     1

a =
     1    10     0   -10    -1
```

Representation of Systems (contd.)

Transformations between representations (contd.)

Example

Create a 5th order filter and convert to second order sections

```
>> b = rand( 1, 6 );
>> a = rand( 1, 6 );
>> [ sos, g ] = tf2sos( b, a )

sos =

   1.0000    0.4396         0    1.0000    1.3759         0
   1.0000    1.3189    0.8249    1.0000    0.8638    1.3496
   1.0000   -1.5103    1.4328    1.0000   -0.7892    0.8215

g =

    1.4702
```


Outline

Time-Domain Analysis

Time Response of LTI Systems

Response of an LTI system y to an input x is given by

1. **Continuous-Time:** The Convolution integral

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \doteq x(t) * h(t)$$

2. **Discrete-Time:** The Convolution Sum

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \doteq x[n] * h[n]$$

where h is the impulse response of the LTI system.

Time-Domain Analysis (contd.)

Convolution **conv**

Example

Suppose

$$h[n] = \begin{cases} \alpha^n & 0 \le n \le 6 \\ 0 & otherwise \end{cases} \qquad \alpha = \frac{1}{2}$$

$$x[n] = \begin{cases} 1 & 0 \le n \le 4 \\ 0 & otherwise \end{cases}$$

```
>> n = 0:12;
>> x = double( ( 0<=n ) & ( n<=4 ) );
>> subplot(311), stem( n, x, 'filled' )
>>
>> h = ( ( 1/2 ).^n ).*( ( 0<=n ) & ( n<=6 ) );
>> subplot(312), stem( n, h, 'filled' )
>>
>> y = conv( x, h );
>> subplot(313), stem( n, y(1:13), 'filled' )
```

Convolution **conv (contd.)**

READER

$$h[n] = \begin{cases} \alpha^n & 0 \leq n \leq 20 \\ 0 & \text{otherwise} \end{cases} \qquad \alpha = \frac{1}{2}$$

$$x[n] = \begin{cases} 1 & 0 \leq n \leq 15 \\ 0 & \text{otherwise} \end{cases}$$

READER

$$h[n] = \begin{cases} 1 & 0 \leq n \leq 4 \\ 0 & \text{otherwise} \end{cases} \qquad \alpha = \frac{1}{2}$$

$$x[n] = \begin{cases} 1 & 0 \leq n \leq 2 \\ \alpha^n & 3 \leq n \leq 6 \\ 0 & \text{otherwise} \end{cases}$$

Solution: *Refer myfilt.m*

Convolution **conv (contd.)**

EXAMPLE

Suppose

$$h[n] = \begin{cases} \alpha^n & -2 \leq n \leq 4 \\ 0 & \text{otherwise} \end{cases} \quad \alpha = \frac{1}{2} \text{ and } x[n] = \begin{cases} 1 & -1 \leq n \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

```
>> nx = -5:10;
>> x = double( ( -1<=nx ) & ( nx<=3 ) );
>> nh = nx;
>> h = ( ( 1/2 ).^nh ).*( ( -2<=nh ) & ( nh<=4 ) );
>> y = conv( x, h );
>> ny = min( nx ) + min( nh ):max( nx ) + max( nh );
>> subplot(311), stem( nx, x, 'filled' )
>> subplot(312), stem( nh, h, 'filled' )
>> subplot(313), stem( ny, y, 'filled' )
```


Transfer Function **filter**

EXAMPLE (GENERAL RESPONSE)

$$y[n] - 0.75y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

$$G(z) = \frac{1 + z^{-1}}{1 - 0.75z^{-1} + 0.125z^{-2}}$$

```
>> numz = [ 1 1 ];
>> denz = [ 1 -0.75 0.125 ];
>> n = -10:10;
>> x = ( n>=0 );
>> y = filter( numz, denz, x);
>> subplot(211), stem( n, x, 'filled')
>> subplot(212), stem( n, y, 'filled')
```


Transfer Function **filter, impz**

EXAMPLE (IMPULSE RESPONSE)

$$y[n] - 0.75y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

$$G(z) = \frac{1 + z^{-1}}{1 - 0.75z^{-1} + 0.125z^{-2}}$$

```
>> numz = [ 1 1 ];
>> denz = [ 1 -0.75 0.125 ];
>> n = 0:10;
>> d = ( n==0 );
>> h = filter( numz, denz, d );
>> subplot(311), stem( n, d, 'filled' )
>> subplot(312), stem( n, h, 'filled' )
>>
>> h2 = impz( numz, denz, 11 );
>> subplot(313), stem( n, h2, 'filled' )
```

Time-Domain Analysis (contd.)

Transfer Function **filter, impz (contd.)**

Reader (General Response)

Suppose

$$y[n] - 0.75y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

Determine the response to the input

$$x[n] = \begin{cases} 1 & 2 \leq n \\ 0 & otherwise \end{cases}$$

Solution: *Refer myfilt.m*

Time-Domain Analysis (contd.)

Transfer Function **filter, impz (contd.)**

Reader (General Response)

Suppose

$$G(z) = \frac{1 + z^{-1}}{1 - 0.75z^{-1} + 0.125z^{-2}}.$$

Determine the response to the input

$$x[n] = \begin{cases} 1 & 0 \leq n \leq 5 \\ 0 & otherwise \end{cases}$$

Solution: *Refer myfilt2.m*

Outline

Frequency Domain Analysis

Frequency Response

Recall that for a Linear Time-Invariant (LTI) system, an input

$$x[n] = e^{\jmath\omega n}$$

produces an output

$$y[n] = H(e^{\jmath\omega})e^{\jmath\omega n}$$

where

$$H(e^{\jmath\omega}) = \sum_{n=-\infty}^{+\infty} h[n]e^{-\jmath\omega n}$$

is called the *frequency response* of the system, or the *Fourier transform* of the impulse response $h[n]$.

Also, the Fourier transform of the output is given by

$$Y(e^{\jmath\omega}) = H(e^{\jmath\omega})X(e^{\jmath\omega}).$$

Frequency response (freqz) or Discrete-time Fourier transform

EXAMPLE

$$y[n] - 0.75y[n-1] + 0.125y[n-2] = x[n] + x[n-1]$$

$$H\left(e^{\jmath\omega}\right) = \frac{1 + e^{-\jmath\omega}}{1 - 0.75e^{-\jmath\omega} + 0.125e^{-2\jmath\omega}}$$

```
>> numz = [ 1 1 ];
>> denz = [ 1 -0.75 0.125 ];
>> w = linspace( -pi, pi, 101 );
>> Hejw = freqz( numz, denz, w );
>> subplot(211), plot( w, abs( Hejw ) ), grid
>> subplot(212), plot( w, angle( Hejw )*180/pi ), grid
```


Frequency Response

The *discrete Fourier transform (DFT)* of the length-N sequence $x[n]$ is

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-\jmath\frac{2\pi k}{N}n}, \quad 0 \le k \le N-1.$$

Similarly the *inverse discrete Fourier transform (IDFT)* is

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{\jmath\frac{2\pi n}{N}k}, \quad 0 \le n \le N-1.$$

Discrete Fourier transform (fft)

EXAMPLE (N-POINT DFT)

$$x_1[n] = \begin{cases} 1, & 0 \le n \le M-1 \\ 0, & \text{otherwise} \end{cases}$$

```
>> M = 8; N = 16; n = 0:N-1;
>> x1 = [ ones(1,M) zeros(1,N-M) ];
>> X1 = fft( x1, N );  % N-point DFT
>> subplot(221), stem( n, x1, 'filled'), title('x_1[n]')
>> subplot(223), stem( n, abs(X1) ), title('|X_1[k]|')
>> subplot(224), stem( n, angle(X1)*180/pi ),
>> title('<X_1[k]')
>>
>> xx1 = ifft( X1, N );
>> subplot(222), stem( n, xx1, 'filled' ), title('xx_1[n]')
```


Discrete Fourier transform (fft (contd.))

READER (N-POINT DFT)

Consider the signal

$$x[n] = \begin{cases} 1 & 0 \le n \le 3 \\ 0 & \text{otherwise} \end{cases}$$

Determine its N-point DFT

1. $N = 4$
2. $N = 8$
3. $N = 16$

Compare with its frequency response $X\left(e^{\jmath\omega}\right)$

Outline

1. *Introduction*
2. *Representation of Signals*
3. *Representation of Systems (Filters)*
4. *Time-Domain Analysis*
5. *Frequency Domain Analysis*
6. *Filter Design*
7. Graphical User Interface
8. Signal Processing Blockset

Filters

Ideal Filters – Common Types – Frequency Response

- Ideal Lowpass Filters (LPF).
- Ideal Highpass Filters (HPF).
- Ideal Bandpass Filters (BPF).
- Ideal Bandstop Filters (BSF).
- Ideal Comb Filters.
- Notch Filters.

Filters (contd.)

Ideal Filters – Frequency/Impulse Response

- $H_a(\jmath\Omega)$ is symmetric (even)
- Sufficient to consider $\Omega > 0$
- $h_a(t)$ real-valued.
- Practical Considerations
  - $|H_a(\jmath\Omega)| = 1$ in pass band not possible
  - $|H_a(\jmath\Omega)| = 0$ in stop band not possible
  - Abrupt transition from pass band to stop band not possible

Filters (contd.)

Ideal Filters – Practical Considerations

- Ideal Filters not possible to build. Therefore, need to approximate.
- Will consider LPFs. Same concept applies to all filters.
- Relax requirements as follows

  $1 - \delta_p \leq |H(\jmath\Omega)| \leq 1 + \delta_p$ in pass band ($0 \leq \Omega \leq \Omega_p$)

  $|H(\jmath\Omega)| \leq \delta_s$ in stop band ($\Omega_s \leq \Omega \leq \infty$)

  Transition from pass band to stop band gradual

  . $\Longrightarrow$ transition band ($\Omega_p \leq \Omega \leq \Omega_s$)

Filters (contd.)

Practical Filters – Specifications

- δ_p – passband ripple $\qquad$ δ_s – stopband ripple
- $\alpha_p = -20\log_{10}(1-\delta_p)$ – peak passband ripple
- $\alpha_s = -20\log_{10}(\delta_s)$ – minimum stopband attenuation
- Ω_p – passband edge frequency $\qquad$ Ω_s – stopband edge frequency
- $0 \le \Omega \le \Omega_p$ – passband $\qquad$ $\Omega_s \le \Omega \le \infty$ – stopband
- $\Omega_p \le \Omega \le \Omega_s$ – transition band
- $\mathcal{G}(\Omega) = 20\log_{10}|H_a(\jmath\Omega)|$ – gain function
- $a(\Omega) = -20\log_{10}|H_a(\jmath\Omega)|$ – attenuation (or loss) function.

Filters (contd.)

Practical Filters – Normalized Specifications

- Passband: $\frac{1}{\sqrt{1+\epsilon^2}} \le |H_a(\jmath\Omega)| \le 1. \implies$ Maximum passband gain = 0 dB
- Stopband: $|H_a(\jmath\Omega)| \le \frac{1}{A} =$ Maximum stopband ripple .
  $\implies$ minimum stopband attenuation = $-20\log_{10}\left(\frac{1}{A}\right)$.
- Transition Ratio (or selectivity parameter)

$$k = \frac{\Omega_p}{\Omega_s} < 1 \text{ for an LPF.}$$

- Discrimination parameter

$$k_1 = \frac{\epsilon}{\sqrt{A^2-1}} << 1 \text{ usually.}$$

Filter Design

Digital Filters

Recall that the general form of the transfer function of a digital filter is

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_{m-1}z^{-(m-1)} + b_m z^{-m}}{a_0 + a_1 z^{-1} + \cdots + a_{n-1}z^{-(n-1)} + a_n z^{-n}}.$$

- When $n = 0$, the denominator is a constant. Such a filter is an *FIR, all-zero, non-recursive, or moving average(MA)* filter.
- When $m = 0$, the numerator is a constant. Such a filter is an *IIR, all-pole, recursive, or autoregressive(AR)* filter.
- When $n > 0$ and $m > 0$, the filter is an *IIR, pole-zero, recursive, or autoregressive moving average(ARMA)* filter.

Filter Design (contd.)

Classical IIR Filters

1. Butterworth - butter

   [b, a] = butter(n, Wn, options) $\qquad$ returns the transfer function

   [z, p, k] = butter(n, Wn, options) $\qquad$ returns the zero-pole-gain

   [A, B, C, D] = butter(n, Wn, options) $\qquad$ state-space representation

2. Chebyshev Type I - cheby1
3. Chebyshev Type II - cheby2
4. Elliptic - ellip
5. Bessel (analog only) - besself

Filter Design (contd.)

Classical IIR Filters (contd.)

Example (Butterworth IIR filter)

For data sampled at 1000 *Hz, design a* 9^{th} *order highpass Butterworth IIR filter with cutoff frequency of* 300 *Hz,*

```
>> [ b, a ] = butter( 9, 300/500, 'high' );
>> freqz( b, a, 128, 1000 )
>> impz( b, a )
```

Example (Chebyshev Type I filter)

For data sampled at 1000 *Hz, design a* 9^{th} *order lowpass Chebyshev Type I filter with* 0.5 *dB of ripple in the passband and a cutoff frequency of* 300 *Hz,*

```
>> [ b, a ] = cheby1( 9, 0.5, 300/500 );
>> freqz( b, a, 512, 1000 )
>> impz( b, a )
```


Filter Design (contd.)

FIR Filters

1. Windowing — fir1, fir2, kaiserord, bartlett, barthannwin, blackman, blackmanharris, bohmanwin, chebwin, gausswin, hamming, hann, kaiser, nuttallwin, parzenwin, rectwin, triang, tukeywin.
   Apply window to truncated inverse Fourier transform of desired "brick wall" filter
2. Multiband with Transition Bands — firls, firpm, firpmord
   Equiripple or least squares approach over sub-bands of the frequency range
3. Constrained Least Squares — fircls, fircls1
   Minimize squared integral error over entire frequency range subject to maximum error constraints
4. Arbitrary Response — cfirpm
   Arbitrary responses, including nonlinear phase and complex filters
5. Raised Cosine — firrcos
   Lowpass response with smooth, sinusoidal transition

Filter Design (contd.)

FIR Filter Design Examples

Example

Design a 48^{th} *order FIR bandpass filter with passband* $0.35 \leq \omega \leq 0.65$

```
>> b = fir1( 48, [ 0.35 0.65 ] );
>> freqz( b, 1, 512 )
>> impz( b, a )
```

Example

Design an LPF using the optimal design method with specifications:
Passband ripple $r_p = 0.01$; *Stopband ripple* $r_s = 0.1$;
Sampling frequency $f_s = 8000$; *Cutoff frequencies* $f = [1500 \quad 2000]$;
Desired amplitudes $a = [1 \quad 0]$;

```
>> [ n, fo, mo, w ] = firpmord( [ 1500 2000 ],...
 [ 1 0 ], [ 0.01 0.1 ], 8000 );
>> b = firpm( n, fo, mo, w );
>> freqz( b, 1, 1024, 8000 )
>> impz( b, a )
```


Outline

Graphical User Interface

Signal Processing Graphical User Interfaces (GUIs)

1. Filter Visualization Tool fvtool(b, a).
   $\Longrightarrow$ *to analyze digital filters.*
2. Filter Design & Analysis Tool fdatool.
   $\Longrightarrow$ *to design or import, and analyze digital FIR and IIR filters.*
3. Signal Processing Tool sptool
   $\Longrightarrow$ *to import, analyze, and manipulate signals, filters, and spectra.*
4. Window Design & Analysis Tool wintool
   $\Longrightarrow$ *to design and analyze windows.*
5. Window Visualization Tool wvtool(w).
   $\Longrightarrow$ *to analyze windows.*

Graphical User Interface (contd.)

Signal Processing Graphical User Interfaces (GUIs) (contd.)

Example

$$y[n] - 0.75y[n-1] + 0.125y[n-2] = x[n] + 0.5x[n-1]$$

$$G(e^{\jmath\omega}) = \frac{1 + e^{-\jmath\omega}}{1 - 0.75e^{-\jmath\omega} + 0.125e^{-2\jmath\omega}}$$

```
>> numz = [ 1 1 ];
>> denz = [ 1 -0.75 0.125 ];
>> fvtool( numz, denz )
>> fdatool
>> sptool
>> wintool
>> w1 = bartlett( 64 );
>> w2 = hamming( 64 );
>> wvtool( w1, w2 );
```


Outline

1. *Introduction*
2. *Representation of Signals*
3. *Representation of Systems (Filters)*
4. *Time-Domain Analysis*
5. *Frequency Domain Analysis*
6. *Filter Design*
7. *Graphical User Interface*
8. *Signal Processing Blockset*

DSP System Toolbox

1. What is it?
2. What is in it?
3. How to access it?
4. How to get help
5. Examples

What is the *DSP System Toolbox*?

- *a tool for DSP algorithm simulation and code generation*
- *contains block libraries for signal processing, linear algebra, & matrix math*
- *works in the Simulink environment*
- *systems defined by interconnecting blocks*
- *blocks can be interconnected to create sophisticated models for simulating such operations as speech and audio processing, wireless digital communications, radar/sonar, medical electronics,*
- *can be used in conjunction with Real-Time Workshop to automatically generate code for real-time execution on DSP hardware*

What is *in* the *DSP System Toolbox*?

Let's take a look...

Accessing the *Signal Processing Blockset*?

1. *At the command prompt, type*

   ```
   >> dsplib
   ```

2. *At the command prompt, type (or click the toolbar)*

   ```
   >> simulink
   ```

   and expand the DSP System Toolbox by clicking the '+' symbol next to it

Getting help

1. **Online:** Place the block in a model, Double-click on the block to open a dialog box, and click Help button
2. **Simulink library browser:** Right-click block & choose from menu.
3. **Help browser:** At the command prompt type

   ```
   >> doc
   ```

   or press F1 on the keyboard
   or select from the menu 'Help→Product Help'

   Click '+' next to the Signal Processing Blockset in Contents tab.
4. **Command line:** At the command prompt type doc('BlockName')

   ```
   >> doc('dspblks/Constant')
   ```

5. **Remote:** go to www.mathworks.com
6. **Release information:** Type whatsnew at the command prompt.

Outline