Polynomial Functions

- Since polynomials occur frequently in mathematics and engineering, MATLAB has a collection of very useful functions for working with them
- The general form of an *N*th degree polynomial function is

$$f(x) = a_0 x^N + a_1 x^{N-1} + \ldots + a_{N-2} x^2 + a_{N-1} x + a_N$$

  where *degree* denotes the highest power of x present
- The polynomial coefficients are $_0, a_1, \ldots, a_$
- You may be bothered by the fact that the powers of x and coefficient subscripts appear to be opposite each other
  - This is just the notation adopted by MATLAB for expressing polynomials; in fact the sum of the coefficient subscript and variable exponent are always a constant of N
- A more compact notation for writing the above is to use *sum* notation

$$f(x) = \sum_{n=0}^{N} a_n x^{N-n}$$

  Note: $x^0 = 1$

Example: A familiar polynomial function is the quadratic polynomial

$$y(x) = a_0 x^2 + a_1 x + a_2$$

Polynomial Evaluation

- Using techniques studied thus far we can use MATLAB used to evaluate $f(x)$ over a range of x values as follows:

```
% Define the range on x
x = x_min:x_step:x_max;
f = a_0*x.^N + a_1*x.^(N-1) + ...
    a_Nm2*x.^2 + a_Nm1*x + a_N;
```

- Taking the quadratic given above, with defined coefficients, we can write

```
» a0 = 3; a1 = 2; a2 = 10; % Define the coefficients
» x = 0:20/100:20; % Define the range on x
» f = a0*x.^2 + a1*x + a2; % Evaluate f(x)
```

- To polynomial evaluation MATLAB comes with the pre-defined function `polyval`

```
» help polyval
 POLYVAL Evaluate polynomial.
    Y = POLYVAL(P,X), when P is a vector of length N+1
whose elements are the coefficients of a polynomial, Y
is the value of the polynomial evaluated at X.

     Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)

    If X is a matrix or vector, the polynomial is evalu-
ated at all points in X.  See also POLYVALM for evalua-
tion in a matrix sense.
```

- To evaluate the quadratic given above we would write

```
» x = 0:20/100:20;
» y = polyval([3, 2, 10],x);
```

Polynomial Operations

- In loose terms we can think of the polynomial coefficient vector, a, as the polynomial itself
- The sum of two polynomials, say

$$g(x) = a_0x^3 + a_1x^2 + a_2x + a_3$$

$$h(x) = b_0x^2 + b_1x + b_2$$

  is the sum of coefficients corresponding to like powers of x

$$s(x) = g(x) + h(x)$$
$$= a_0x^3 + (a_1 + b_0)x^2 + (a_2 + b_1)x + (a_3 + b_2)$$

- For MATLAB evaluation this amounts to simply adding corresponding coefficient vectors a and b by first prepending the b vector with a leading zero, i.e.,

```
g = [a0, a1, a2, a3];
h = [0,  b0, b1, b2]; % prepending only required
s = g + h;            % for poly add and subtract
```

- If one polynomial is a scalar multiple of another, say

$$g(x) = k \cdot f(x)$$

  the corresponding coefficient vectors are related by k, i.e., if

```
g = polyval(a,x);
f = polyval(b,x);
% Or given g and the fact that b = k*a
f = k*g;
```

- Suppose we wish to multiply or divide two polynomials

- Recall how tedious it is to multiply two polynomials

$$
\begin{aligned}
g(x) &= (a_0x^2 + a_1x + a_2)(b_0x^2 + b_1x + b_2) \\
&= a_0x^2(b_0x^2 + b_1x + b_2) + a_1x(b_0x^2 + b_1x + b_2) \\
&\quad + a_2(b_0x^2 + b_1x + b_2) \\
&= a_0b_0x^4 + (a_0b_1 + a_1b_0)x^3 + (a_0b_2 + a_1b_1 + a_2b_0)x^2 \\
&\quad + (a_1b_2 + a_2b_1)x + a_2b_2
\end{aligned}
$$

- If `c` is the corresponding coefficient vector of `g` it follows that

```
c = [(a0*b0), (a0*b1+a1*b0), (a0*b2+a1*b1+a2*b0),
     (a1*b2+a2*b1), (a2*b2)];
```

- A MATLAB function that performs this operation automatically is `conv(a,b)` which stands for convolution (you learn more about this function in *Linear Systems Theory – ECE 3510*)

- In the above example we would type

```
% To find the coefficients of  g(x) = a(x)*b(x) we
% would write
g = conv(a,b)
```

Example: $g(x) = (x^2 + 2x + 3)(2x^2 + x + 2)$

```
» a = [1, 2, 3]; b = [2, 1, 2];
» g = conv(a,b)
g =
     2     5    10     7     6
```

- $g(x)$ is thus

$$g(x) = 2x^4 + 5x^3 + 10x^2 + 7x + 6$$

- Polynomial division can be accomplished in like fashion using the function `deconv(a,b)` which stands for deconvolution

Example:

$$h(x) = \frac{g(x)}{b(x)} = \frac{2x^4 + 5x^3 + 10x^2 + 7x + 6}{x^2 + 2x + 2}$$

- Hand calculation requires long division of the polynomial coefficient vectors, including a remainder vector

$$\begin{array}{r} 2x^2 + x + 4 \\ x^2 + 2x + 2 \overline{\big)\ 2x^4 + 5x^3 + 10x^2 + 7x + 6} \\ \underline{2x^4 + 4x^3 + 4x^2} \\ x^3 + 6x^2 + 7x + 6 \\ \underline{x^3 + 2x^2 + 2x} \\ 4x^2 + 5x + 6 \\ \underline{4x^2 + 8x + 8} \\ \text{Remainder} \longrightarrow\ -3x - 2 \end{array}$$

- To find the quotient and remainder polynomials in MATLAB write

```
» g = [2, 5, 10, 7, 6];
» b = [1, 2, 2];
» [h,r] = deconv(g,b);
```

```
» h
h =
     2     1     4 % --> h(x) = 2x^2+x+4
» r
r =
     0     0     0    -3    -2 % --> r(x) = -3x-2
```

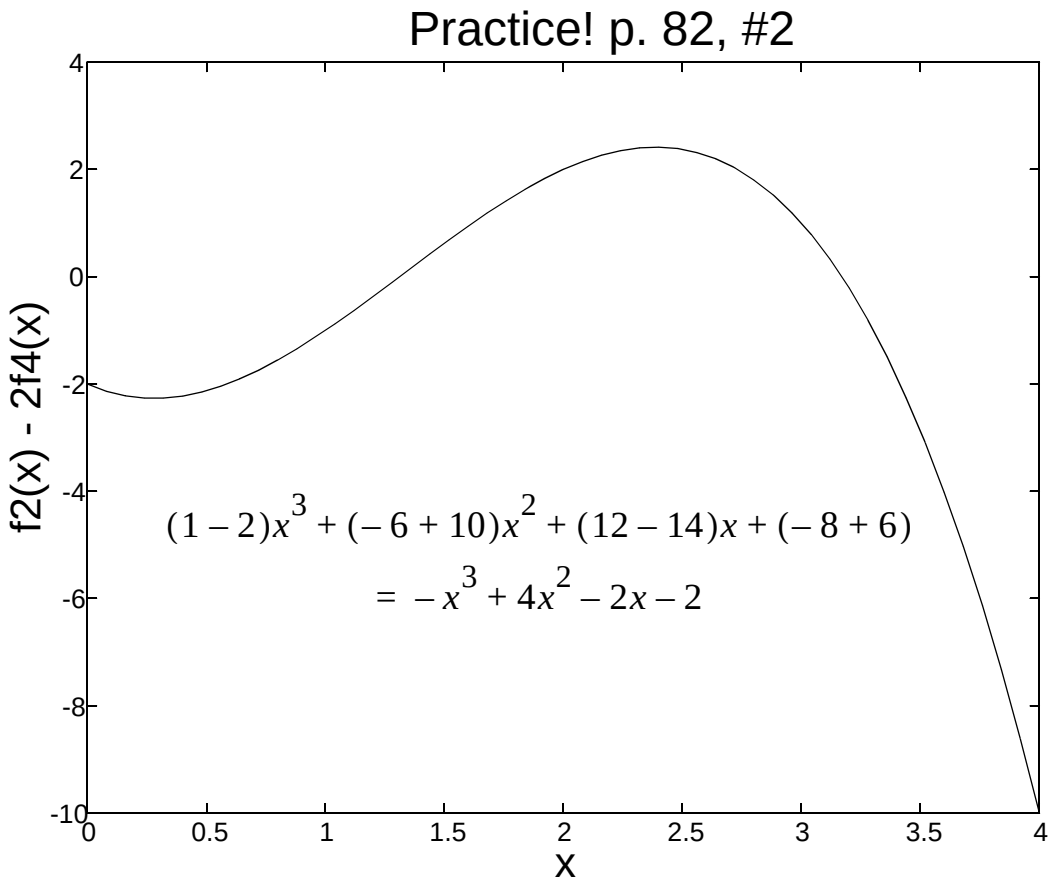
Example: Practice! p. 82, #2 & #5

$$f_2(x) = x^3 - 6x^2 + 12x - 8$$

$$f_4(x) = x^3 - 5x^2 + 7x - 3$$

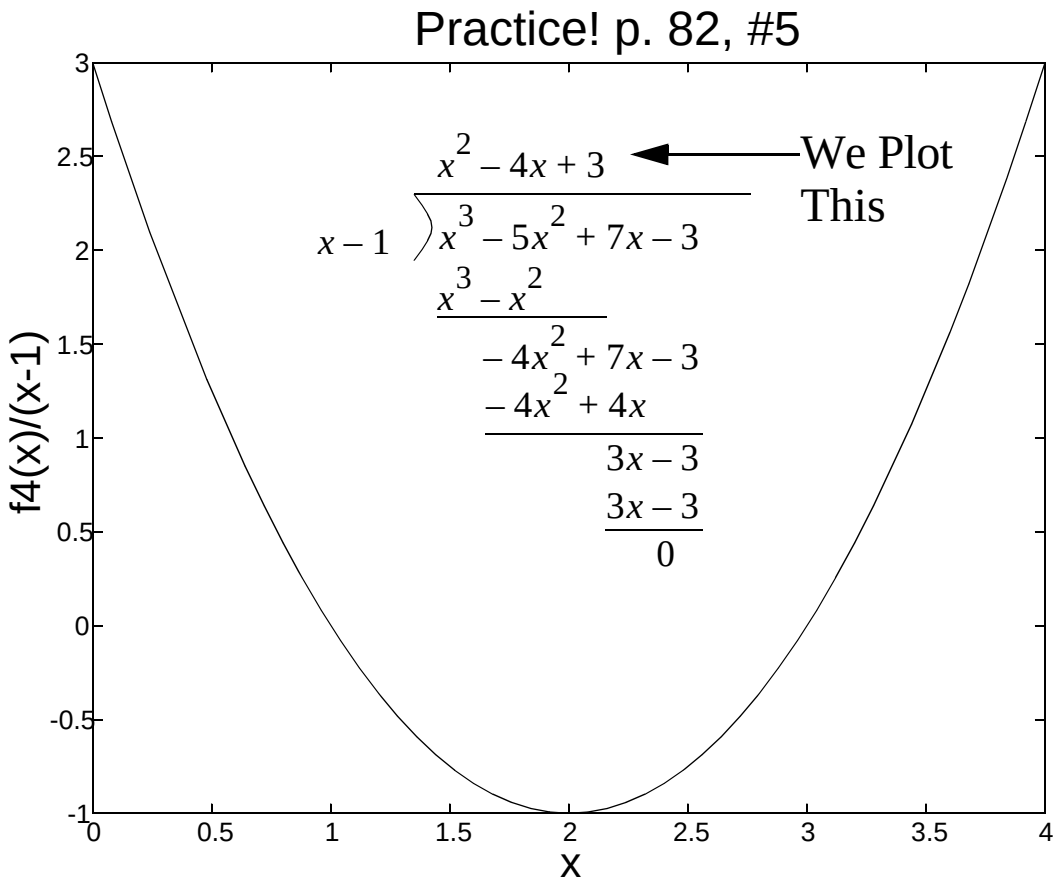
- For #2 we must evaluate $f_2(x) - 2f_4(x)$ on [0, 4]

```
» x = 0:4/50:4; % Use 51 points
» a_f2 = [1 -6 12 -8];
» a_f4 = [1 -5 7 -3];
» a_p2 = a_f2 - 2*a_f4;
» f_p2 = polyval(a_p2,x);
» plot(x,f_p2)
» title('Practice! p. 82, #2','fontsize',18)
» ylabel('f2(x) - 2f4(x)','fontsize',16)
» xlabel('x','fontsize',16)
```



- For #5 we must evaluate $(f_4(x))/(x-1)$ on $[0, 4]$

```
» [q_f5,r_f5] = deconv(a_f4,[1 -1]); % Divide the polys
» q_f5
q_f5 =
     1    -4     3 % The quotient is a quadratic
» r_f5
r_f5 =
     0     0     0     0 % Note there is no remainder!
» f_p5 = polyval(q_f5,x);
» plot(x,f_p5)
» title('Practice! p. 82, #5','fontsize',18)
» ylabel('f4(x)/(x-1)','fontsize',16)
» xlabel('x','fontsize',16)
```



Roots of Polynomials

- A problem related to evaluating polynomials is solving for the roots of a polynomial
- Recall that the roots of $y = f(x)$ are those values of x where $f(x) = 0$
- For polynomials with real number coefficients, the roots may be real numbers and/or pairs of complex conjugate numbers
  - For a third-order polynomial the roots may be either three real roots or one real root and a complex pair
  - For a second-order polynomial the roots may be either two real roots or a complex conjugate pair

- If we plot $f(x)$ we find that the real roots correspond to those locations where the polynomial crosses the x-axis
- Recall that for a quadratic $y(x) = a_0x^2 + a_1x + a_2$ the roots are (quadratic formula)

$$r_1, r_2 = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0a_2}}{2a_0}$$

  Note: The roots are complex if $4a_0a_2 > a_1^2$

- Hand calculation of polynomial roots becomes impractical as the order increases, hence we often turn to a numerical solution
- The MATLAB function `roots(a)` finds the roots of a polynomial with coefficient vector `a`

Example: $f(x) = 2x^4 + 5x^3 + 10x^2 + 7x + 6$

```
» p = [2 5 10 7 6];
» r = roots(p)
r =
  -1.0000 + 1.4142i
  -1.0000 - 1.4142i
  -0.2500 + 0.9682i
  -0.2500 - 0.9682i
» polyval(p,r)% Check by evaluating p
              % at the root locations
ans =  1.0e-013 *
  -0.1243 + 0.0089i
  -0.1243 - 0.0089i
  -0.0533 + 0.0355i
  -0.0533 - 0.0355i % Functional values are small
```

- Given the roots of a polynomial $r_1, r_2, \ldots, r_N$ we know that

$$f(x) = a_0 x^N + a_2 x^{N-1} + \ldots + a_{N-1} x + a_N$$
$$= (x - r_1)(x - r_2)\ldots(x - r_N)$$

- The MATLAB function `poly(r)` effectively reconstructs the polynomial coefficient vector, `a`, given the vector of roots by repeated polynomial multiplication

Example: Continue the previous example

```
» poly(r)
ans =
    1.0000    2.5000    5.0000    3.5000    3.0000
```

- What happened?
  - The `poly()` function must assume that a_0 is normalized to unity
  - In the earlier example $a_0 = 2$, so to get the correct answer we multiply by two following the `poly()` operation, e.g.,

```
» 2*poly(r)
ans =
    2.0000    5.0000   10.0000    7.0000    6.0000
```

Examples: Practice! p. 86, #4

Given

$$g_4(x) = x^5 - 3x^4 - 10x^3 + 27x^2 + 10x - 24$$

Determine the real roots of $g_4(x)$ and then plot the polynomial over the appropriate interval to verify that the polynomial crosses the x-axis at the real root locations

```
» a_g4 = [1 -3 -11 27 10 -24];
» roots(a_g4)' % Use transpose to make output compact
ans =
    4.0000   -3.0000    2.0000    1.0000   -1.0000» x =
-4:(5+4)/100:5; % Plot 101 points
» g4 = polyval(a_g4,x);
» plot(x,g4);grid
» title('Practice! p. 86, #4','fontsize',18)
» ylabel('g4(x)','fontsize',16)
» xlabel('x','fontsize',16)
```