Computer Programming: Polynomials, Curve fitting, Interpolation

Asst. Prof. Dr. Yalçın İşler

Izmir Katip Celebi University

Outline

- Polynomials
- Curve-fitting
- Interpolation
- Basic fitting interface
- Examples

Polynomials

- Polynomials are mathematical expressions that are frequently used for problem solving and modeling in science and engineering.
- Polynomials are functions that have the form:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

- The coefficients «a» are real numbers and *n is a nonnegative* integer, is the degree, or order, of the polynomial.

Polynomials (cont'd)

Polynomial	**MATLAB representation**
$8x + 5$	p = [8 5]
$2x^2 - 4x + 10$	d = [2 –4 10]
$6x^2 - 150$, MATLAB form: $6x^2 + 0x - 150$	h = [6 0 –150]
$5x^5 + 6x^2 - 7x$, MATLAB form: $5x^5 + 0x^4 + 0x^3 + 6x^2 - 7x + 0$	c = [5 0 0 6 –7 0]

Value of Polinomial

The value of a polynomial at a point x can be calculated with the function `poly-val` that has the form:

```
polyval(p,x)
```

`p` is a vector with the coefficients of the polynomial.

`x` is a number, or a variable that has an assigned value, or a computable expression.

`x` can also be a vector or a matrix. In such a case the polynomial is calculated for each element (element-by-element), and the answer is a vector, or a matrix, with the corresponding values of the polynomial.

Roots of Poliynomial

Roots of a polynomial are the values of the argument for which the value of the polynomial is equal to zero. For example, the roots of the polynomial $f(x) = x^2 - 2x - 3$ are the values of x for which $x^2 - 2x - 3 = 0$, which are $x = -1$, and $x = 3$.

MATLAB has a function, called `roots`, that determines the root, or roots, of a polynomial. The form of the function is:

```
r = roots(p)
```

`r` is a column vector with the roots of the polynomial.

`p` is a row vector with the coefficients of the polynomial.

Roots (cont'd)

For example, to find the roots of $f(x) = 4x^2 + 10x - 8$ type:

```
>> roots([4 10 -8])

ans =
   -3.1375
    0.6375
```

When the roots of a polynomial are known, the `poly` command can be used for determining the coefficients of the polynomial. The form of the `poly` command is:

```
p = poly(r)
```

`p` is a row vector with the coefficients of the polynomial.

`r` is a vector (row or column) with the roots of the polynomial.

Polynomial Arithmetics

Two polynomials can be added (or subtracted) by adding the vectors of the coefficients. If the polynomials are not of the same order (which means that the vectors of the coefficients are not of the same length), the shorter vector has to be modified to be of the same length as the longer vector by adding zeros (called padding) in front.

$$f_1(x) = 3x^6 + 15x^5 - 10x^3 - 3x^2 + 15x - 40 \text{ and } f_2(x) = 3x^3 - 2x - 6$$

```
>> p1=[3 15 0 -10 -3 15 -40];
>> p2=[3 0 -2 -6];
>> p=p1+[0 0 0 p2]
p =
     3    15     0    -7    -3    13   -46
```

Three 0's are added in front of `p2`, since the order of `p1` is 6 and the order of `p2` is 3.

Polynomial Arithmetics (cont'd)

Two polynomials can be multiplied with the MATLAB built-in function `conv` which has the form:

```
c = conv(a,b)
```

`c` is a vector of the coefficients of the polynomial that is the product of the multiplication.

`a` and `b` are the vectors of the coefficients of the polynomials that are being multiplied.

- The two polynomials do not have to be of the same order.
- Multiplication of three or more polynomials is done by using the `conv` function repeatedly.

```
>> pm=conv(p1,p2)
pm =
     9    45    -6   -78   -99    65   -54   -12   -10   240
```

which means that the answer is:

$$9x^9 + 45x^8 - 6x^7 - 78x^6 - 99x^5 + 65x^4 - 54x^3 - 12x^2 - 10x + 240$$

Polynomial Arithmetics (cont'd)

A polynomial can be divided by another polynomial with the MATLAB built-in function `deconv` which has the form:

```
[q,r] = deconv(u,v)
```

`q` is a vector with the coefficients of the quotient polynomial.

`r` is a vector with the coefficients of the remainder polynomial.

`u` is a vector with the coefficients of the numerator polynomial.

`v` is a vector with the coefficients of the denominator polynomial.

For example, dividing $2x^3 + 9x^2 + 7x - 6$ by $x + 3$ is done by:

```
>> u=[2 9 7 -6];
>> v=[1 3];
```

```
>> [a b]=deconv(u,v)
a =
     2     3    -2
b =
     0     0     0     0
```

The answer is: $2x^2 + 3x - 2$.

Remainder is zero.

Polynomial Derivatives

`k = polyder(p)`	Derivative of a single polynomial. `p` is a vector with the coefficients of the polynomial. `k` is a vector with the coefficients of the polynomial that is the derivative.
`k = polyder(a,b)`	Derivative of a product of two polynomials. `a` and `b` are vectors with the coefficients of the polynomials that are multiplied. `k` is a vector with the coefficients of the polynomial that is the derivative of the product.
`[n d] = polyder(u,v)`	Derivative of a quotient of two polynomials. `u` and `v` are vectors with the coefficients of the numerator and denominator polynomials. `n` and `d` are vectors with the coefficients of the numerator and denominator polynomials in the quotient that is the derivative.

Polynomial Derivatives (cont'd)

For example, if $f_1(x) = 3x^2 - 2x + 4$, and $f_2(x) = x^2 + 5$, the derivatives of $3x^2 - 2x + 4$, $(3x^2 - 2x + 4)(x^2 + 5)$, and $\dfrac{3x^2 - 2x + 4}{x^2 + 5}$ can be determined by:

```
>> f1= 3 -2 4];
>> f2=[1 0 5];
```

Creating the vectors coefficients of f_1 and f_2.

```
>> k=polyder(f1)
k =
     6    -2
```

The derivative of f_1 is: $6x - 2$.

```
>> d=polyder(f1,f2)
d =
    12    -6    38   -10
```

The derivative of f_1*f_2 is: $12x^3 - 6x^2 + 38x - 10$.

```
>> [n d]=polyder(f1,f2)
n =
     2    22   -10
d =
     1     0    10     0    25
```

The derivative of $\dfrac{3x^2 - 2x + 4}{x^2 + 5}$ is: $\dfrac{2x^2 + 22x - 10}{x^4 + 10x^2 + 25}$.

Curve-fitting

- Curve fitting, also called regression analysis, is a process of fitting a function to a set of data points. The function can then be used as a mathematical model of the data. Since there are many types of functions (linear, polynomial, power, exponential, etc.) curve fitting can be a complicated process. Many times there is some idea of the type of function that might fit the given data and the need is only to determine the coefficients of the function.
- In other situations, where nothing is known about the data, it is possible to make different types of plots which provide information about possible forms of functions that might give a good fit to the data.

$$y = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$$

$$y = bx^m \qquad \text{(power function)}$$

$$y = be^{mx} \quad \text{or} \quad y = b10^{mx} \qquad \text{(exponential function)}$$

$$y = m\ln(x) + b \quad \text{or} \quad y = m\log(x) + b \qquad \text{(logarithmic function)}$$

$$y = \frac{1}{mx + b} \qquad \text{(reciprocal function)}$$

polyfit

Curve fitting with polynomials is done in MATLAB with the `polyfit` function, which uses the least squares method. The basic form of the `polyfit` function is:

```
p = polyfit(x,y,n)
```

`p` is the vector of the coefficients of the polynomial that fits the data.

`x` is a vector with the horizontal coordinate of the data points (independent variable).
`y` is a vector with the vertical coordinate of the data points (dependent variable).
`n` is the degree of the polynomial.

$$y = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$$

polyfit for other function types

Function		polyfit function form
power	$y = bx^m$	`p=polyfit(log(x),log(y),1)`
exponential	$y = be^{mx}$ or	`p=polyfit(x,log(y),1)` or
	$y = b10^{mx}$	`p=polyfit(x,log10(y),1)`
logarithmic	$y = m\ln(x) + b$ or	`p=polyfit(log(x),y,1)` or
	$y = m\log(x) + b$	`p=polyfit(log10(x),y,1)`
reciprocal	$y = \frac{1}{mx + b}$	`p=polyfit(x,1./y,1)`

After plotting the sample values, you can decide the suitable form of the polyfit function.

Interpolation

- Interpolation is estimation of values between data points.
- In onedimensional interpolation each point has one independent variable (*x)* and one dependent variable (*y).*
- In two-dimensional interpolation each point has two independent variables (*x* and *y)* and one dependent variable (*z).*

1D Interpolation

One-dimensional interpolation in MATLAB is done with the `interp1` (last character is the number one) function, which has the form:

```
yi = interp1(x,y,xi,'method')
```

`yi` is the interpolated value.

`x` is a vector with the horizontal coordinate of the input data points (independent variable).
`y` is a vector with the vertical coordinate of the input data points (dependent variable).
`xi` is the horizontal coordinate of the interpolation point (independent variable).

Method of interpolation, typed as a string (optional).

1D Interpolation (cont'd)

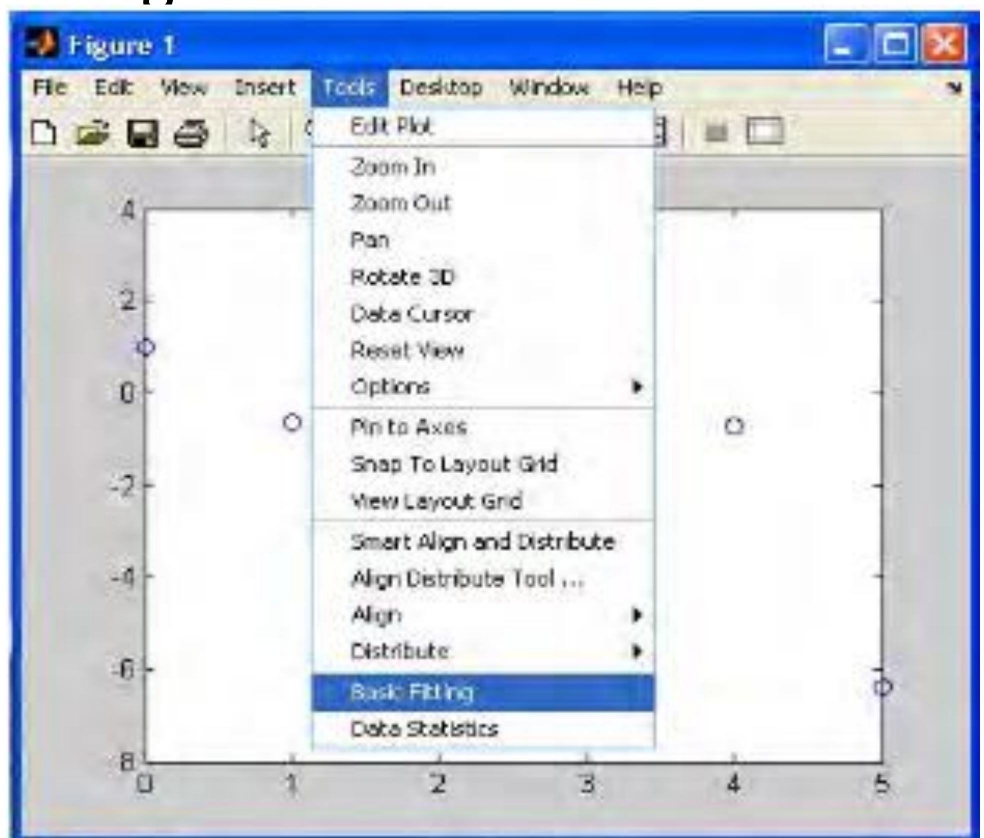
- The vector `x` must be monotonic (the elements in ascending or descending order).
- `xi` can be a scalar (interpolation of one point) or a vector (interpolation of many points). Respectively, `yi` is a scalar or a vector with the corresponding interpolated values.
- MATLAB can do the interpolation using one of several methods that can be specified. These methods include:

`'nearest'`	returns the value of the data point that is nearest to the interpolated point.
`'linear'`	uses linear spline interpolation.
`'spline'`	uses cubic spline interpolation.
`'pchip'`	uses piecewise cubic Hermite interpolation, also called `'cubic'`

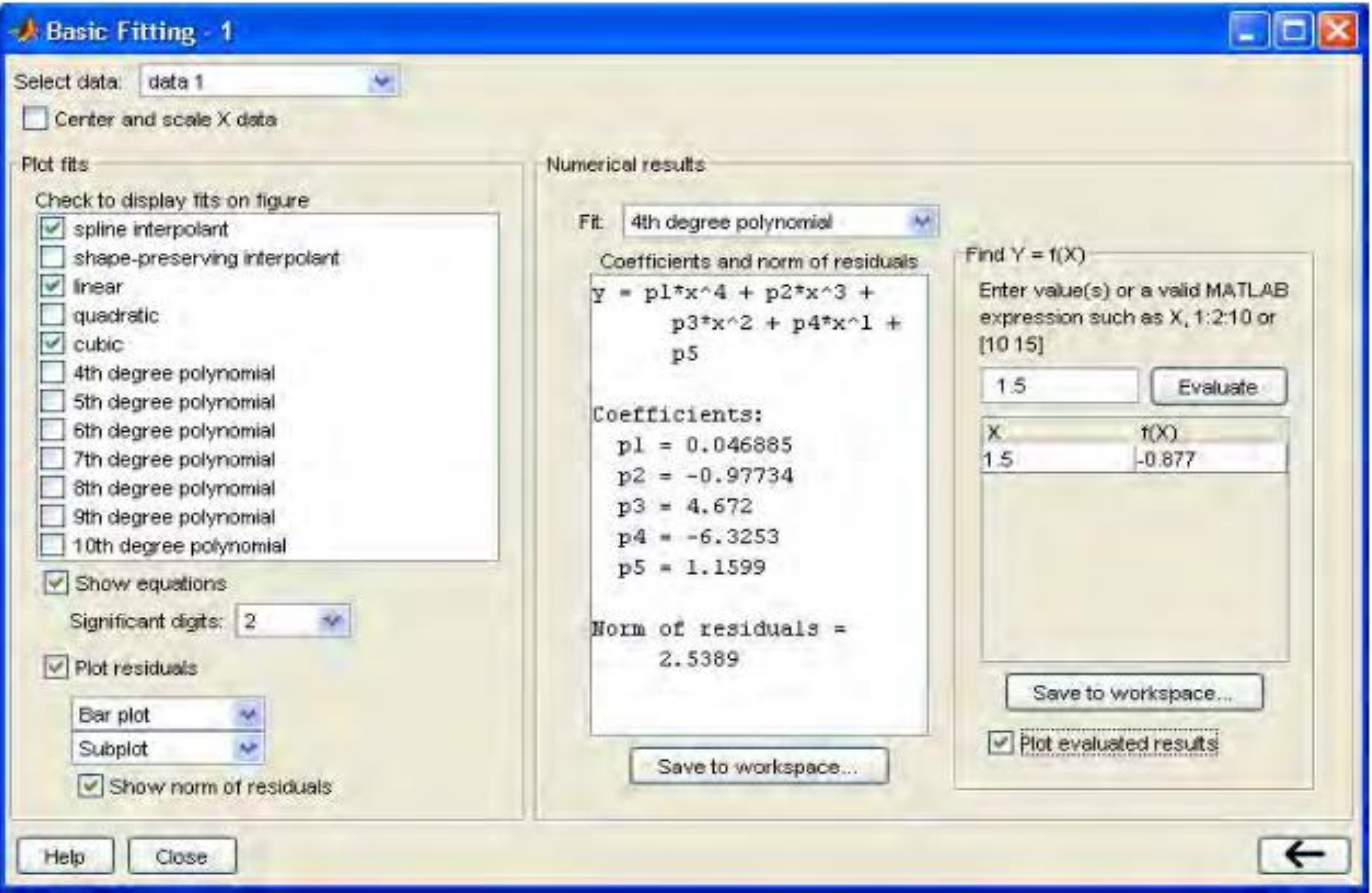
Basic fitting interface

- To activate the basic fitting interface, the user first has to make a plot of the data points. Then, the interface is activated by selecting **Basic Fitting** in the **Tools menu**. This opens the Basic Fitting Window.

Several fits can be selected and displayed simultaneously.



Basic fitting interface (cont'd)



Laboratory Session

Do both examples and sample applications in Chapter 8 of the textbook.

Homework #10

Not later than the next week:

Solve problems 2, 10, 15, and 18 from the Chapter 8 of the textbook using Matlab.