



INTRODUCTION TO SCIENTIFIC & ENG. COMPUTING

BIL 108E, CRN 44448



Week.7_ Arrays & Nonlinear Equation
Week.8_ Solving of Simple Equations

Dr. Feyzi HAZNEDAROĞLU

İstanbul Teknik Üniversitesi - İnşaat Fakültesi
Room : 250A, e-mail : haznedar@itu.edu.tr

01 - 04 - 2011



TENTATIVE SCHEDULE

Wk	Date	Topics
1	Feb. 11	Introduction to Scientific and Engineering Computing
2	Feb. 18	Introduction to Program Computing Environment
3	Feb. 25	Variables, Operations and Simple Plot
4	Mar. 04	Algorithms and Logic Operators
5	Mar. 11	Flow Control, Errors and Source of Errors
6	Mar. 18	Functions & Linear Algebra
7	Mar. 25	Arrays
8	Apr. 01	Solving of Simple Equations
9	Apr. 08	Polynomials Examples Exam 1
10	Apr. 15	Applications of Curve Fitting
11	Apr. 22	Applications of Interpolation
12	Apr. 29	Applications of Numerical Integration
13	May.06	Symbolic Mathematics
14	May.13	Ord Dif.Equations (ODE) solution with Built-in Functions

Feyzi Haznedaroglu

weeks # 8

2



Lecture # 8 - Solving of Simple Equations

1- POLYNOMIALS

2- APPROXIMATION OF DATA



POLYNOMIALS

- Definition: n^{th} degree polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$
- Coefficients of the polynomial
 $a_n, a_{n-1}, \dots, a_2, a_1, a_0$
- n : degree of the polynomial



POLYNOMIALS

MATLAB toolbox: **polyfun**

- **polyval**: returns the value of the polynomial at a point **x**. Input arguments are vector **p** and vector **x**.

$$p \rightarrow a_n, a_{n-1}, \dots, a_1, a_0$$

x is the **abscissae**, where the polynomials is evaluated.

absis

- $y = \text{polyval}(p, x)$



POLYNOMIALS

MATLAB toolbox: **polyfun**

- **polyval**: returns the value of the polynomial at a point **x**. Input arguments are vector **p** and vector **x**.

$$p \rightarrow a_n, a_{n-1}, \dots, a_1, a_0$$

x is the **abscissae**, where the polynomials is evaluated.

absis

- $y = \text{polyval}(p, x)$

EXAMPLE:

Given: $p(x) = x^7 + 3x^2 - 1,$

$$x_k = -1 + k/4 \quad \text{for } k = 0, \dots, 8$$

Find: Plot the given function.



POLYNOMIALS

Command Window

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

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 evaluated at all
points in X. See POLYVALM for evaluation in a matrix sense.

[Y,DELTA] = POLYVAL(P,X,S) uses the optional output structure S created
by POLYFIT to generate prediction error estimates DELTA. DELTA is an
estimate of the standard deviation of the error in predicting a future
observation at X by P(X).

If the coefficients in P are least squares estimates computed by
POLYFIT, and the errors in the data input to POLYFIT are independent,
normal, with constant variance, then Y +/- DELTA will contain at least
```



POLYNOMIALS

Command Window

If the coefficients in P are least squares estimates computed by POLYFIT, and the errors in the data input to POLYFIT are independent, normal, with constant variance, then Y +/- DELTA will contain at least 50% of future observations at X.

$$\hat{x} = (x - \mu_1) / \mu_2$$

$Y = \text{POLYVAL}(P,X,[],MU)$ or $[Y,DELTA] = \text{POLYVAL}(P,X,S,MU)$ uses $XHAT = (X - MU(1)) / MU(2)$ in place of X. The centering and scaling parameters MU are optional output computed by POLYFIT.

Class support for inputs P,X,S,MU:
float: double, single

See also [polyfit](#), [polyvalm](#).

Overloaded methods:
[gf/polyval](#)

Reference page in Help browser
[doc polyval](#)



POLYNOMIALS

- $y = \text{polyval}(p,x,[],\mu)$ or $[y,\text{delta}] = \text{polyval}(p,x,S,\mu)$ use $\hat{x} = (x - \mu_1) / \mu_2$ in place of x . In this equation $\mu_1 = \text{mean}(x)$, and $\mu_2 = \text{std}(x)$. The centering and scaling parameters $\mu = [\mu_1, \mu_2]$ are optional output computed by `polyfit`.

Remarks

- The `polyvalm` (p,x) function, with x a matrix, evaluates the polynomial in a matrix sense. See `polyvalm` for more information.

EXAMPLE

- The polynomial $p(x) = 3x^2 + 2x + 1$ is evaluated at $x = 5, 7,$ and 9 with;
`p = [3 2 1];`
`polyval (p,[5 7 9])` which results in
`ans =`
`86 162 262`



POLYNOMIALS

EXAMPLE cont'd.:

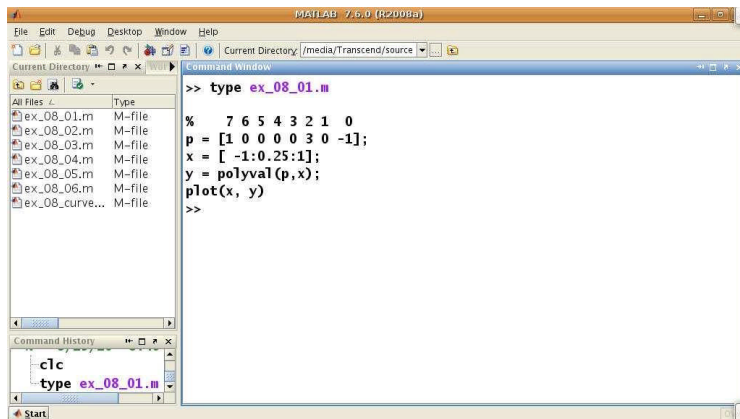
```
% 7 6 5 4 3 2 1 0
p = [1 0 0 0 0 3 0 -1];
x = [-1:0.25:1];
y = polyval (p,x);
plot (x, y)
```



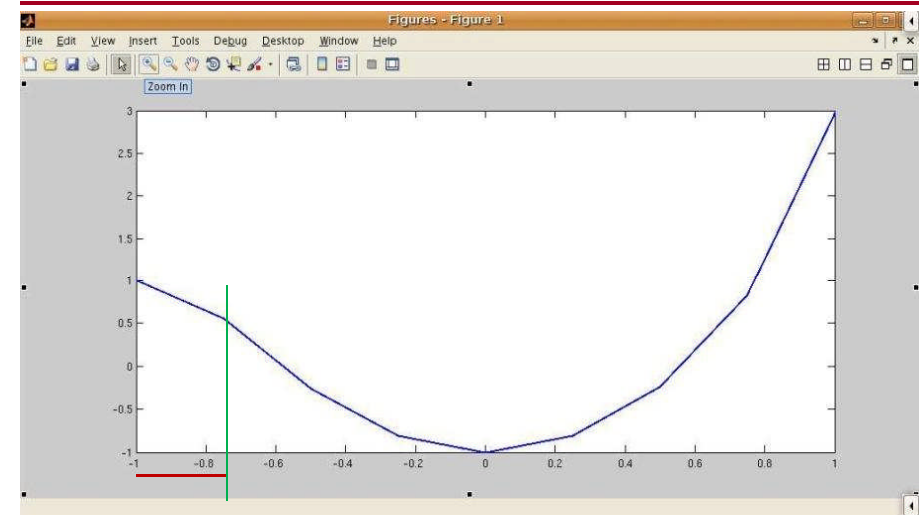
POLYNOMIALS

EXAMPLE cont'd.:

```
% 7 6 5 4 3 2 1 0
p = [1 0 0 0 0 3 0 -1];
x = [-1:0.25:1];
y = polyval (p,x);
plot (x, y)
```



POLYNOMIALS





POLYNOMIALS

POLYNOMIALS

- **roots**: provides an approximation of the zeros of a polynomial.
Usage: `r = roots(p)`
- **poly**: returns the coefficients of the polynomial, whose zeros are given.
Usage: `p = poly(r)`
- <http://www.mathworks.com/help/techdoc/ref/roots.html>

EXAMPLE:

- Given: $p(x) = x^3 - 6x^2 + 11x - 6$
- Find: Compute the zeros of the polynomial.



POLYNOMIALS

POLYNOMIALS

- **roots**: provides an approximation of the zeros of a polynomial.
Usage: `r = roots(p)`
- **poly**: returns the coefficients of the polynomial, whose zeros are given.
Usage: `p = poly(r)`
- <http://www.mathworks.com/help/techdoc/ref/roots.html>

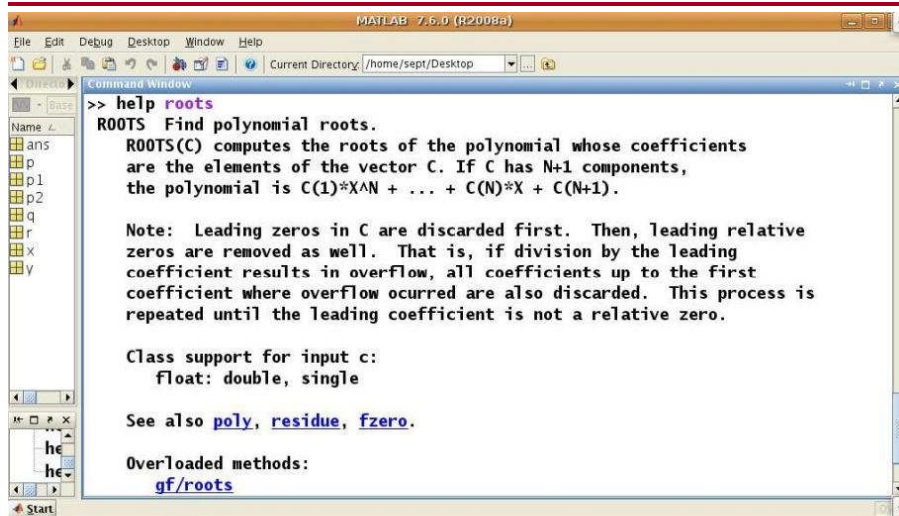
EXAMPLE:

- Given: $p(x) = x^3 - 6x^2 + 11x - 6$
- Find: Compute the zeros of the polynomial.

```
p = [1 -6 11 -6]; format long;
roots(p)
```



POLYNOMIALS



POLYNOMIALS

roots - Polynomial roots

Description: $r = \text{roots}(c)$ returns a column vector whose elements are the roots of the polynomial c . Row vector c contains the coefficients of a polynomial, ordered in descending powers. If c has $n+1$ components, the polynomial it represents is $c_1s^n + \dots + c_n s + c_{n+1}$

Remarks

- Note the relationship of this function to $p = \text{poly}(r)$, which returns a row vector whose elements are the coefficients of the polynomial. For vectors, **roots** and **poly** are inverse functions of each other, up to ordering, scaling, and roundoff error.

Examples

$s^3 - 6s^2 - 72s - 27$ is represented in MATLAB as $p=[1 \ -6 \ -72 \ -27]$
The roots of polynomial are returned in a column vector by $r = \text{roots}(p)$

```
r =
    12.1229
    -5.7345
    -0.3884
```



POLYNOMIALS

EXAMPLE:

- Given: $p(x) = x^3 - 6x^2 + 11x - 6$
- Find: Compute the zeros of the polynomial.

`p = [1 -6 11 -6]; format long;`
`roots(p)`

```

>> type ex_08_02.m
p = [1 -6 11 -6]; format long;
roots(p)
>>
ans =
    3.000000000000002
    1.999999999999998
    1.000000000000000
  
```



POLYNOMIALS

```

>> type ex_08_02.m
p = [1 -6 11 -6]; format long;
roots(p)
>> ex_08_02

ans =

    3.000000000000002
    1.999999999999998
    1.000000000000000

>>
  
```



POLYNOMIALS

EXAMPLE: poly

```

>> help poly
POLY Convert roots to polynomial.
POLY(A), when A is an N by N matrix, is a row vector with
N+1 elements which are the coefficients of the
characteristic polynomial, DET(lambda*EYE(SIZE(A)) - A) .

POLY(V), when V is a vector, is a vector whose elements are
the coefficients of the polynomial whose roots are the
elements of V . For vectors, ROOTS and POLY are inverse
functions of each other, up to ordering, scaling, and
roundoff error.

ROOTS(POLY(1:20)) generates Wilkinson's famous example.

Class support for inputs A,V:
float: double, single

See also roots, conv, residue, polyval.
  
```



POLYNOMIALS

poly - Polynomial with specified roots

Syntax `p = poly(A)`
`p = poly(r)`

Description

- `p = poly(A)` where `A` is an `n`-by-`n` matrix returns an `n+1` element row vector whose elements are the coefficients of the characteristic polyn., $\det(sI - A)$. The coefficients are ordered in descending powers: if a vector `c` has `n+1` components, the polynomial it represents is $c_1s^n + \dots + c_n s + c_{n+1}$
- `p = poly(r)` where `r` is a vector returns a row vector whose elements are the coefficients of the polynomial whose roots are the elements of `r`.

Remarks

Note the relationship of this command to `r = roots(p)` which returns a column vector whose elements are the roots of the polynomial specified by the coefficients row vector `p`. For vectors, `roots` and `poly` are inverse functions of each other, up to ordering, scaling, and roundoff error.



POLYNOMIALS

Example_1

A =

```
1 2 3
4 5 6
7 8 0
```

is returned in a row vector by poly: `p = poly(A)`

p =

```
1 -6 -72 -27
```

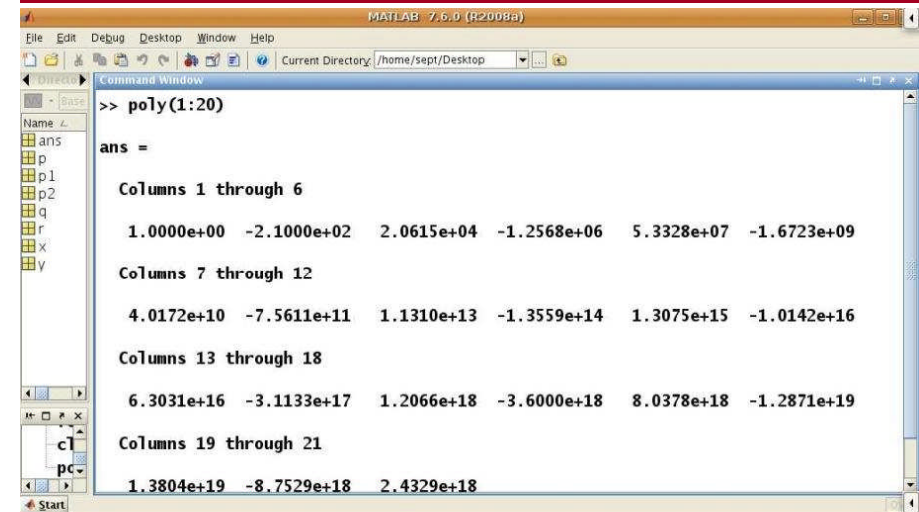
- The roots of this polynomial (eigenvalues of matrix A) are returned in a column vector by roots: `r = roots(p)`

r =

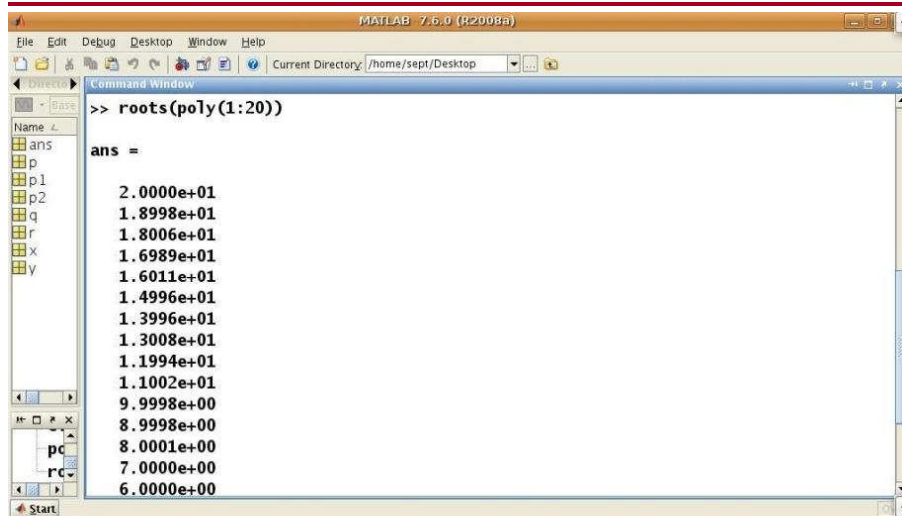
```
12.1229
-5.7345
-0.3884
```



POLYNOMIALS



POLYNOMIALS



POLYNOMIALS

EXAMPLE:

The result is not always accurate.

Given: $p(x) = (x + 1)^7$

Find: Compute the zeros of the polynomial.

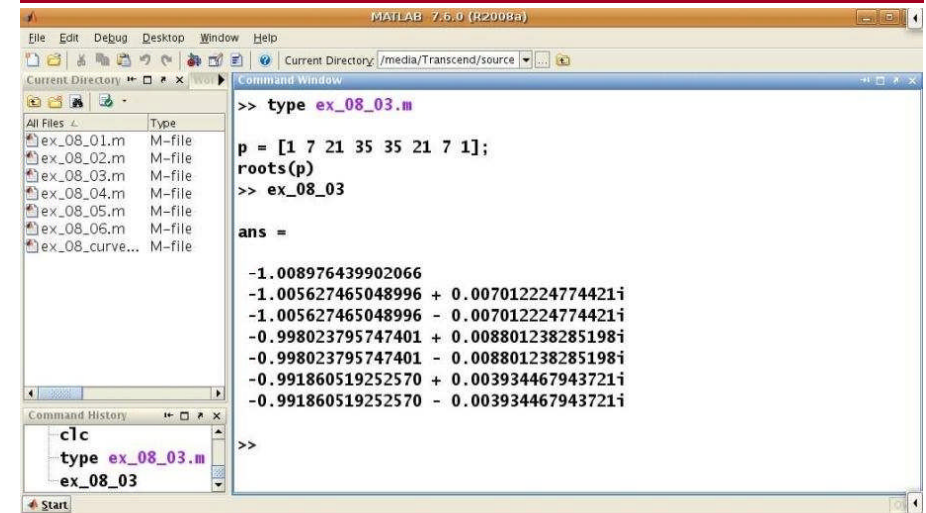
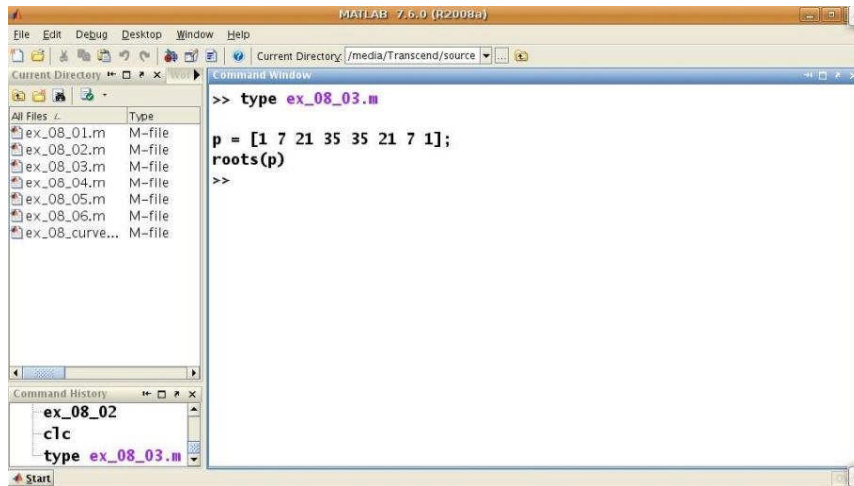
Answer: $\alpha = -1$

- In fact, numerical methods for the computation of the polynomial roots with multiplicity larger than one are particularly subject to roundoff errors.



EXAMPLE cont'd.:

`p = [1 7 21 35 35 21 7 1]; roots(p)`



ADDITION AND SUBTRUCTION OF POLYNOMIALS

- Polynomial addition and subtraction is the same as vector addition/subtraction operators.
- If the order of two polynomials (size of two vectors) does not match, zero should be added in order to match size of vectors.



ADDITION AND SUBTRUCTION OF POLYNOMIALS

- Polynomial addition and subtraction is the same as vector addition/subtraction operators.
- If the order of two polynomials (size of two vectors) does not match, zero should be added in order to match size of vectors.

EXAMPLE:

- Given: $p_1(x) = x^4 - 1$,
 $p_2(x) = x^3 - 1$

- Find: Compute the sum of two polynomials.
- Answer: $p = x^4 + x^3 - 2$



POLYNOMIALS

EXAMPLE cont'd.:

```
p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
disp(p1);
disp(p2);
p = p1 + [0 p2]
```

```

MATLAB 7.6.0 (R2008a)
Current Directory: /media/Transcend/source
Command Window
>> type ex_08_06.m
p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
disp(p1);
disp(p2);
p = p1 + [0 p2]
>>

Command History
conv(p2,q)+r
clc
type ex_08_06.m
ex_08_06
Start

```



POLYNOMIALS

```

MATLAB 7.6.0 (R2008a)
Current Directory: /media/Transcend/source
Command Window
>> type ex_08_06.m
p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
disp(p1);
disp(p2);
p = p1 + [0 p2]
>> ex_08_06
    1    0    0    0   -1
    1    0    0   -1
p =
    1    1    0    0   -2
>>

Command History
clc
type ex_08_06.m
ex_08_06
Start

```



POLYNOMIALS

MULTIPLICATION AND DIVISION OF POLYNOMIALS

- **conv**: returns the coefficients of the polynomial given by the product of two polynomials.
Usage: **conv(p1, p2)**
- **deconv**: provides the coefficients of the polynomials obtained on dividing p1 by p2.
Usage: **[q, r]= deconv(p1, p2)**
q: quotient of the division,
r: remainder of the division
p₁(x) = q(x) p₂(x) + r(x)



POLYNOMIALS

MULTIPLICATION AND DIVISION OF POLYNOMIALS

- **conv**: returns the coefficients of the polynomial given by the product of two polynomials.
Usage: **conv(p1, p2)**
- **deconv**: provides the coefficients of the polynomials obtained on dividing p1 by p2.
Usage: **[q, r]= deconv(p1, p2)**
q: quotient of the division, (bölüm,oran)
r: remainder of the division (kalanı)
p₁(x) = q(x) p₂(x) + r(x)

EXAMPLE:

Given: p₁(x) = x⁴ - 1, p₂(x) = x³ - 1

Find: Compute the product of two polynomials.

Answer: p = x⁷ - x⁴ - x³ + 1



POLYNOMIALS

```

MATLAB 7.6.0 (R2008a)
File Edit View Graphics Debug Desktop Window Help
Current Directory: /home/sept/Desktop
Workspace: ans, p, p1, p2, q, r, x, y
Command Window:
>> help conv
Convolution and polynomial multiplication.
C = CONV(A, B) convolves vectors A and B. The resulting
vector is length LENGTH(A)+LENGTH(B)-1.
If A and B are vectors of polynomial coefficients, convolving
them is equivalent to multiplying the two polynomials.

Class support for inputs A,B:
Float: double, single

See also deconv, conv2, convn, filter and, in the signal
Processing Toolbox, xcorr, convmtx.

Overloaded methods:
gf/conv

Reference page in Help browser
doc conv

Command History:
help conv
clc
help conv

```



POLYNOMIALS

EXAMPLE cont'd.:

```

p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
p=conv(p1 ,p2)

```

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Command Window:
>> type ex_08_04.m
p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
p=conv(p1 ,p2)
>> |

Command History:
ex_08_03
clc
type ex_08_04.m

```



POLYNOMIALS

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Command Window:
>> type ex_08_04.m
p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
p=conv(p1 ,p2)

>> ex_08_04

p =

    1     0     0    -1    -1     0     0     1

>>

Command History:
clc
type ex_08_04.m
ex_08_04

```



POLYNOMIALS

EXAMPLE:

Given: $p_1(x) = x^4 - 1$,
 $p_2(x) = x^3 - 1$

Find: Compute the division of two polynomials.

Answer: $q(x) = x$, $r(x) = x - 1$

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /home/sept/Desktop
Workspace: ans, p, p1, p2, q, r, x, y
Command Window:
>> help deconv
DECONV Deconvolution and polynomial division.
[Q,R] = DECONV(B,A) deconvolves vector A out of vector B. The re
is returned in vector Q and the remainder in vector R such that
B = conv(A,Q) + R.

If A and B are vectors of polynomial coefficients, deconvolution
is equivalent to polynomial division. The result of dividing B b
A is quotient Q and remainder R.

Class support for inputs B,A:
Float: double, single

See also conv, residue.

Overloaded methods:
gf/deconv

Command History:
clc
help conv
help deconv

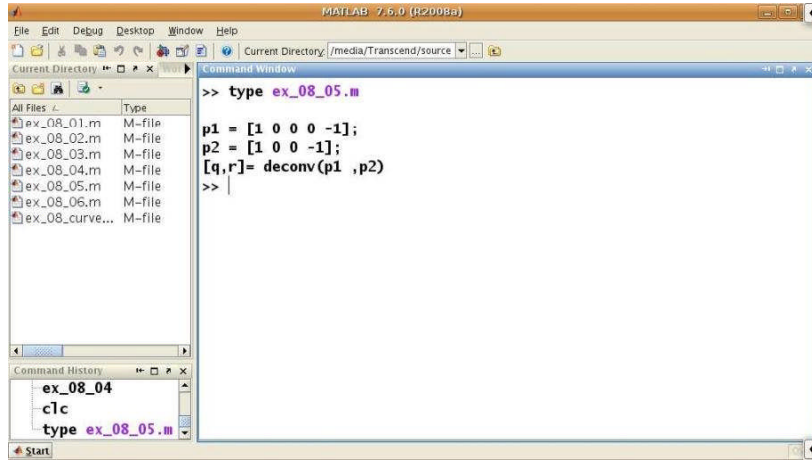
```



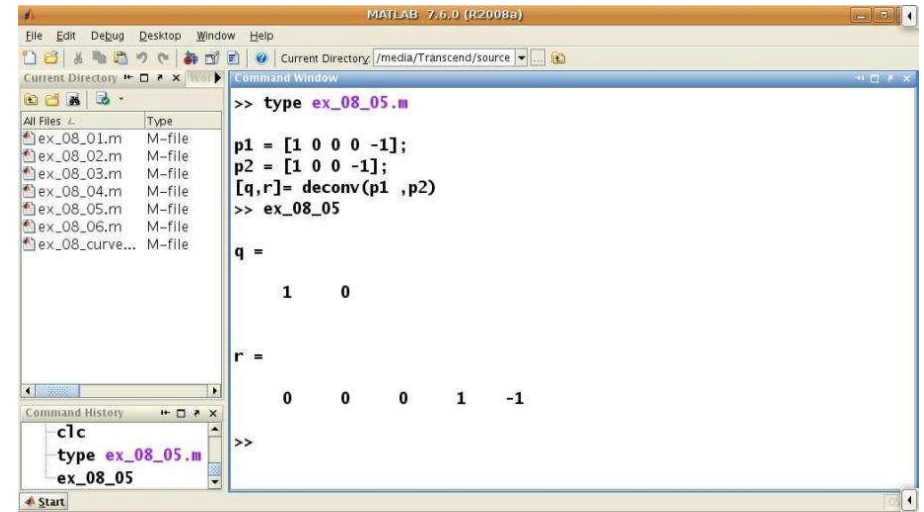
POLYNOMIALS

EXAMPLE cont'd.:

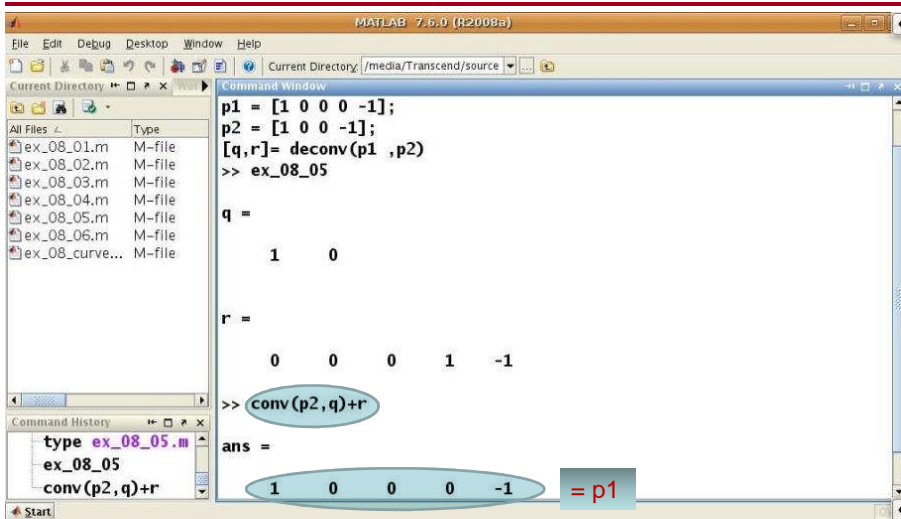
```
p1 = [1 0 0 0 -1];
p2 = [1 0 0 -1];
[q,r]= deconv(p1 ,p2)
```



POLYNOMIALS



POLYNOMIALS



POLYNOMIALS

- **polyint**: returns the coefficients of the primitive of the polynomial.
Usage: $y = \text{polyint}(p)$,
y: coefficients of $\int_0^x p(t) dt$. Türetilmemiş
- **polyder**: returns the derivative of the polynomial, whose coefficients are given by the components of the vector p.
Usage: $y = \text{polyder}(p)$,
y: coefficients of $p'(x)$



CURVE FITTING

- Approximating a function f consists of replacing it by another func. f .
- A function f can be replaced in a given interval by its Taylor polynomial.
- It requires the knowledge of f and its derivatives up to the order n at a given point x_0 .



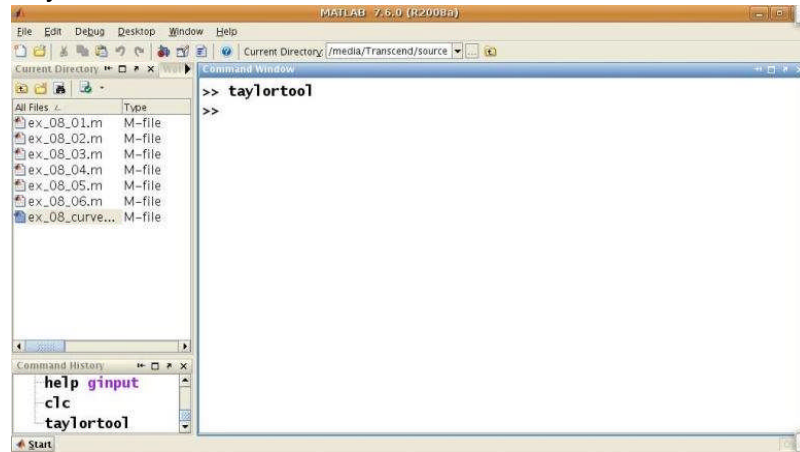
CURVE FITTING

- Approximating a function f consists of replacing it by another func. f .
- A function f can be replaced in a given interval by its Taylor polynomial.
- It requires the knowledge of f and its derivatives up to the order n at a given point x_0 .
- Taylor polynomial may fail to accurately represent f far enough from the point x_0 .
- Use `taylortool` for the computation of Taylor's polynomial of arbitrary degree for any given function f .
- The agreement between the function and its Taylor polynomial is very good in a small neighborhood of x_0 .



CURVE FITTING

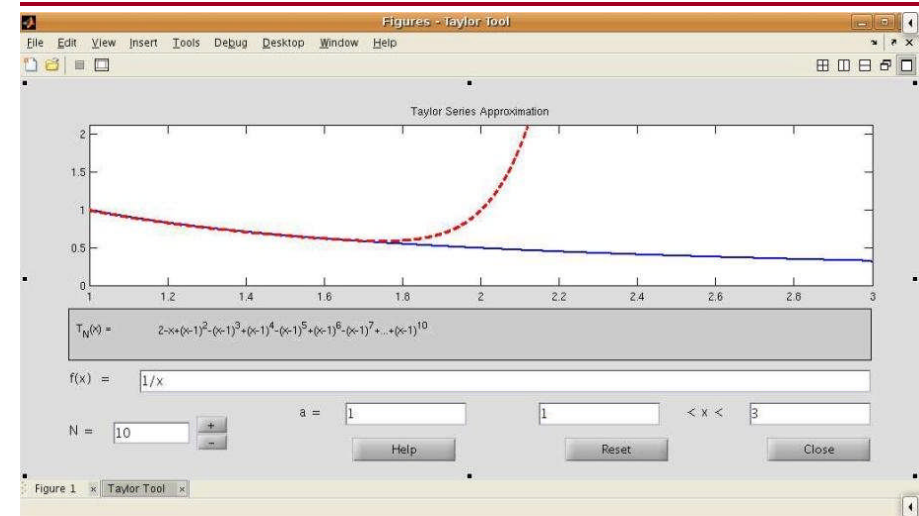
Taylortool



Comparison between the function $f(x) = 1/x$ and its Taylor polynomial of degree 10 for the point $x_0 = 1$.

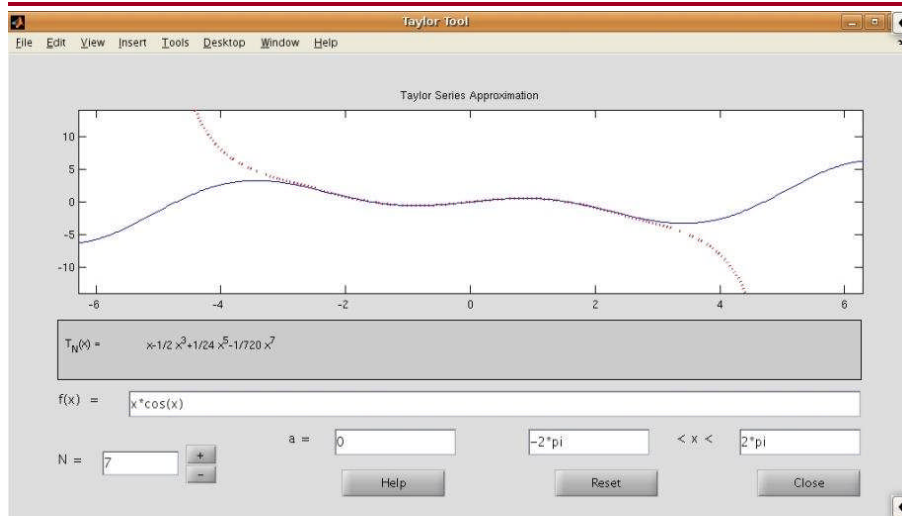


CURVE FITTING





CURVE FITTING



CURVE FITTING

CURVE FITTING

- A function is known only through its values at some given points.
- $n+1$ couples (x_i, y_i) are given, $i = 0, 1, 2, \dots, n+1$.

- Approximate function \tilde{f} :

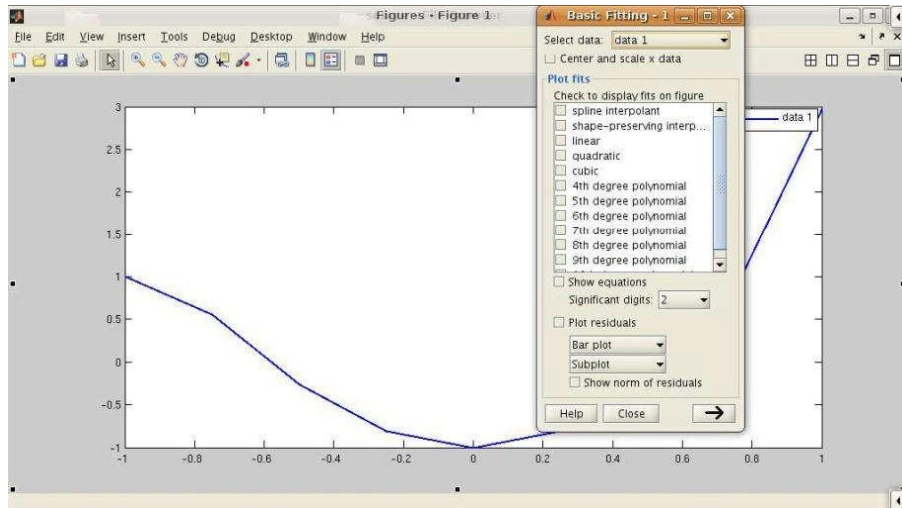
$$\tilde{f}(x_i) = y_i, \text{ where } i = 0, 1, 2, \dots, n$$

- \tilde{f} is called *interpolant* of the set of data
- There exist different types of interpolant:
 - Polynomial interpolant
 - Trigonometric interpolant
 - Rational interpolant



CURVE FITTING

BASIC FITTING:

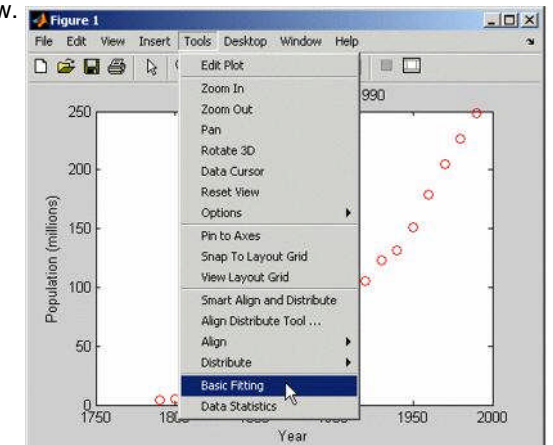


CURVE FITTING

Opening the Basic Fitting GUI

To use the Basic Fitting GUI, you must first plot your data in a figure window, using any MATLAB plot commands that produces (only) x & y.

- To open the Basic Fitting GUI, select **Tools > Basic Fitting** from the menus at the top of the figure window.



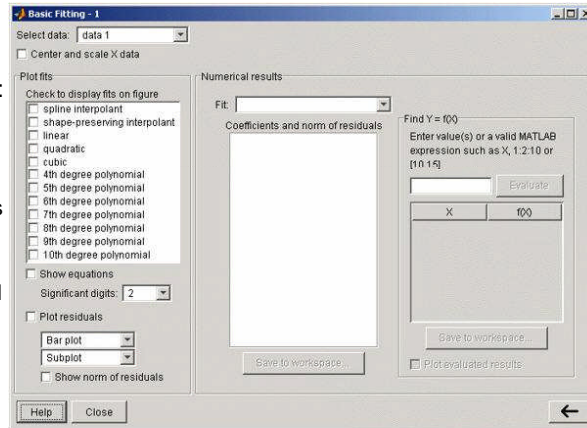


CURVE FITTING

To expand or collapse the panels, use the arrow button in the lower right corner of the interface.

The GUI consists of 3 panels:

1. For selecting a model and plotting options
2. For examining and exporting model coefficients and norms of residuals
3. For examining and exporting interpolated and extrapolated values.



CURVE FITTING

- The most common method of finding the best fit to data point is the *least squares method*.

- **polyfit** function uses least squares method.
- **polyfit** function returns the coefficients of a polynomial for a given data set (x_i, y_i) .
- Usage: **p = polyfit(x, y, n)**
 n : degree of the polynomial.
 x and y : data set (x_i, y_i) .



CURVE FITTING

EXAMPLE

- In the table below we report the values of the *sea water density ρ (in kg/m³)* corresponding to *different values of the temperature T (in degrees Celsius)*:

T	4°	8°	12°	16°	20°
ρ	1000.7794	1000.6427	1000.2805	999.716	998.9700



CURVE FITTING

EXAMPLE

- In the table below we report the values of the *sea water density ρ (in kg/m³)* corresponding to *different values of the temperature T (in degrees Celsius)*:

T	4°	8°	12°	16°	20°
ρ	1000.7794	1000.6427	1000.2805	999.716	998.9700

```
T = [4 8 12 16 20];
rho = [1000.7794 1000.6427 1000.2805 999.7165 998.9700];
px = polyfit(T, rho, 3) % [ x, y, degree ]
Tx = linspace(4,20,100); % [4, 4.1616 ,..., 20]
rhox = polyval(px, Tx) % px func. At points Tx
plot(T, rho, 'o') % plot T versus rho with 'o' sign.
hold('on')
plot(Tx, rhox, '-')
grid('on')
xlabel('T')
ylabel('rho')
legend('data', 'curve fit')
```



CURVE FITTING

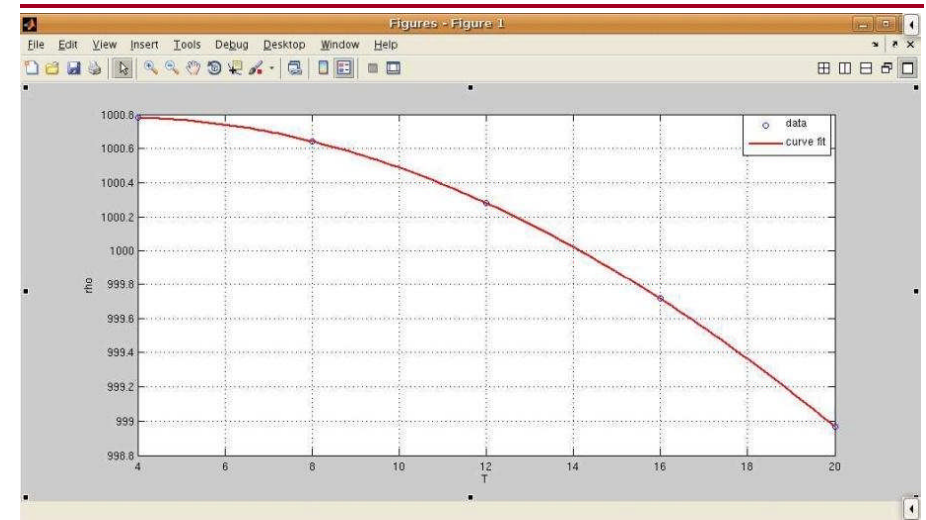
```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace
Name Value
T [4,8,12,16,20]
ans [1.3967,3]
p [1,1,0,0,-]
p1 [1,0,0,0,-]
p2 [1,0,0,-1]
q [1,0]
r [0,0,0,1,-]
rho [1.0008e]
x [-1,-0.75]
y [1,0.5540]
Command Window
>> type ex_08_36.m
T = [4 8 12 16 20];
rho = [1000.7794 1000.6427 1000.2805 999.7165 998.9700];
px = polyfit(T, rho, 3)
Tx = linspace(4,20,100);
rhox=polyval(px, Tx)
plot(T, rho, 'o')
hold('on')
plot(Tx, rhox, '-')
grid('on')
xlabel('T')
ylabel('rho')
legend('data', 'curve fit')
>>
Command History
clc
type ex_08_36
Start

```



CURVE FITTING



CURVE FITTING

LEAST SQUARES METHOD

- If the degree of the polynomial increases, interpolation does not guarantee a better approximation of a given function.
- In least - squares approximation we look for an approximant f which is a polynomial of degree m (typically, $m \ll n$) that minimizes the mean-square error $1/n \sum_{i=0}^n [y_i - \tilde{f}(x_i)]^2$. The same minimization criterion can be applied for a class of functions that are not polynomials.



CURVE FITTING

CURVE FITTING WITH FUNCTIONS OTHER THAN POLYNOMIALS

- Rewrite the function in a first degree polynomial form.
- Power function:
 $y = b x^m \rightarrow \ln y = m \ln x + \ln b$
- Exponential function:
 $y = b e^{mx} \rightarrow \ln y = m x + \ln b$
- Logarithmic function:
 $y = m \ln x + b \rightarrow y = m \ln x + b$
- Reciprocal function:
 $y = 1/(m x + b) \rightarrow 1/y = m x + b$



CURVE FITTING

FUNCTION SELECTION

- For a given data it is possible to foresee which of the functions have the potential for providing a good fit.
- This is done by plotting the data using different combinations of linear and logarithmic axes.

x-axis	y-axis	function
linear	linear	$y = m x + b$
logarithmic	logarithmic	$y = b x^m$
linear	logarithmic	$y = b e^{mx}$
logarithmic	linear	$y = m \ln x + b$
linear	linear	$y = 1 / (m x + b)$



CURVE FITTING

FUNCTION SELECTION

- Exponential functions can not pass through the origin.
- Exponential functions can only fit data with all positive y 's or all negative y 's.
- Logarithmic functions cannot model $x = 0$, or negative values of x .
- For the power function $y = 0$ when $x = 0$.
- The reciprocal equation cannot model $y = 0$.



CURVE FITTING

EXAMPLE:

W	0.64	0.73	0.96	1.21	1.49	1.83	2.41	3.15	3.70	4.83	6.00
t	5.0	4.5	4.0	3.5	3.0	2.5	2.0	1.5	1.0	0.5	0.0

Choose the function for the given data.



CURVE FITTING

EXAMPLE:

W	0.64	0.73	0.96	1.21	1.49	1.83	2.41	3.15	3.70	4.83	6.00
t	5.0	4.5	4.0	3.5	3.0	2.5	2.0	1.5	1.0	0.5	0.0

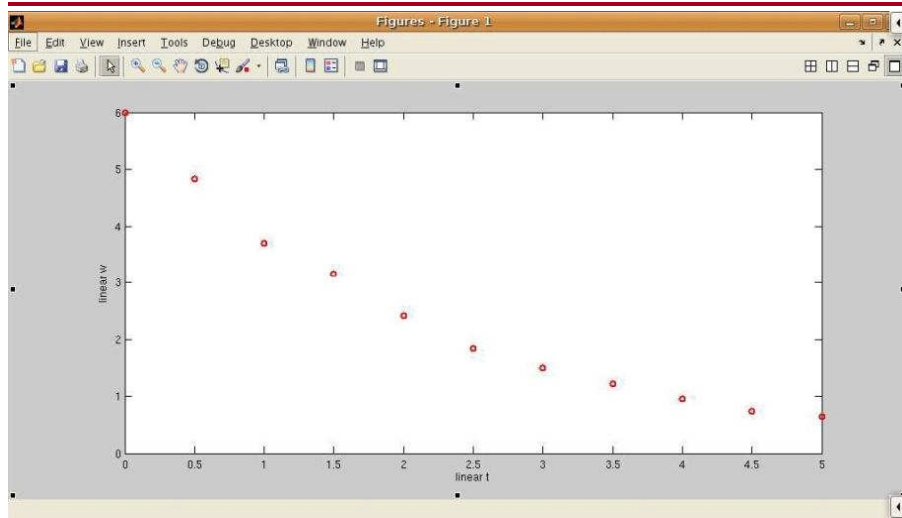
Choose the function for the given data. (in revers order)

```

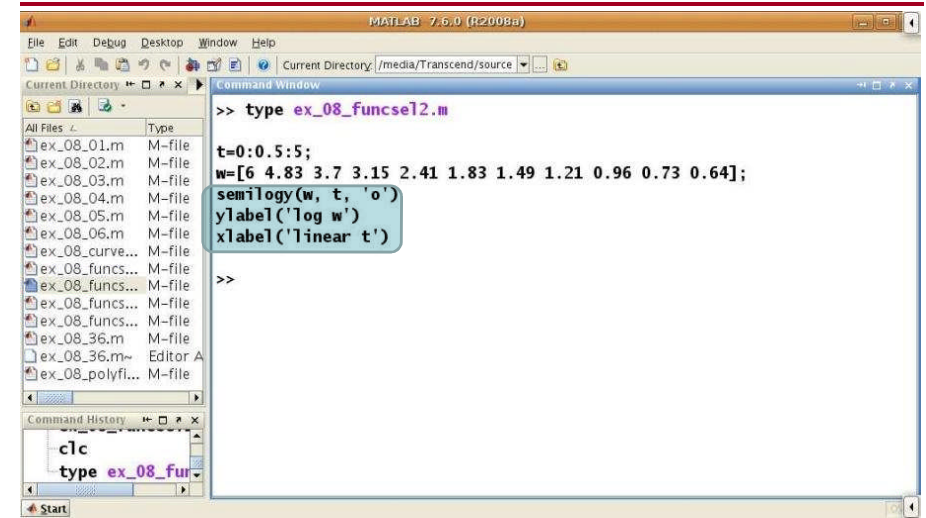
>> type ex_08_funcsel1.m
t=0:0.5:5;
w=[6 4.83 3.7 3.15 2.41 1.83 1.49 1.21 0.96 0.73 0.64];
plot(t, w, 'o')
xlabel('linear t')
ylabel('linear w')
>>
  
```



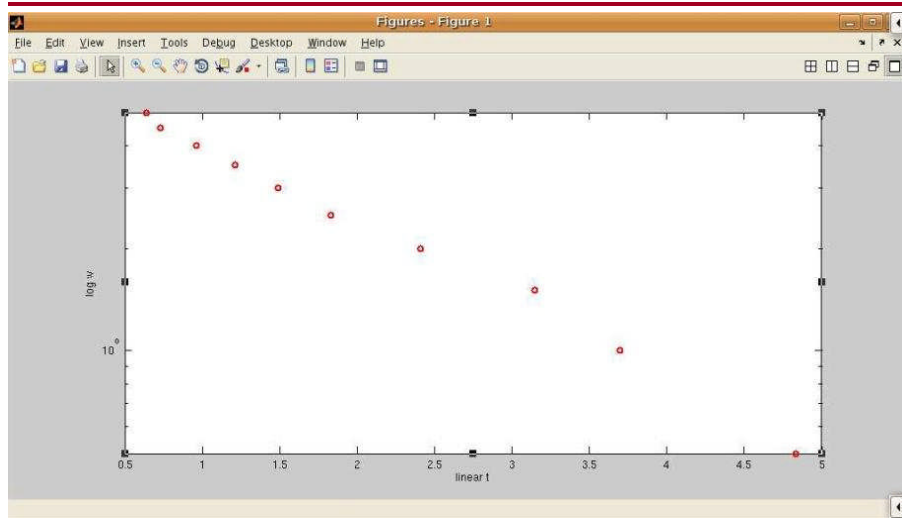
CURVE FITTING



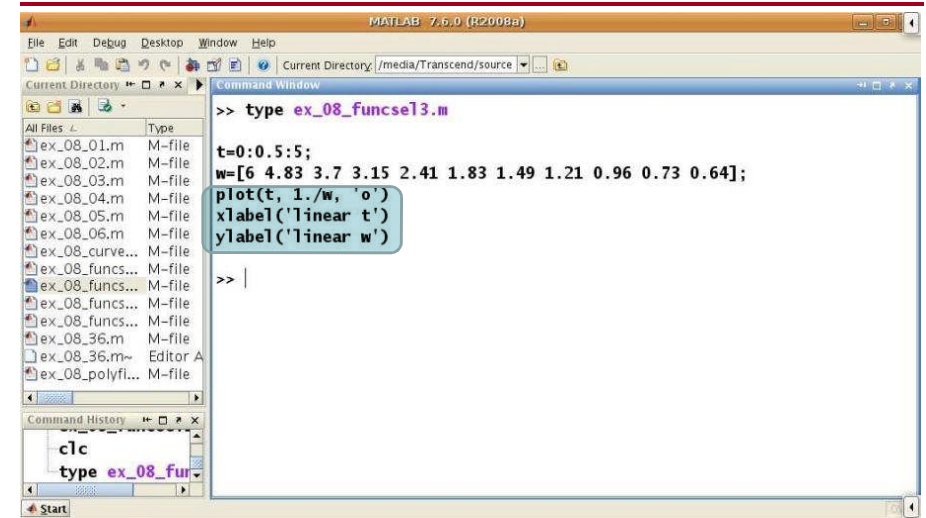
CURVE FITTING



CURVE FITTING

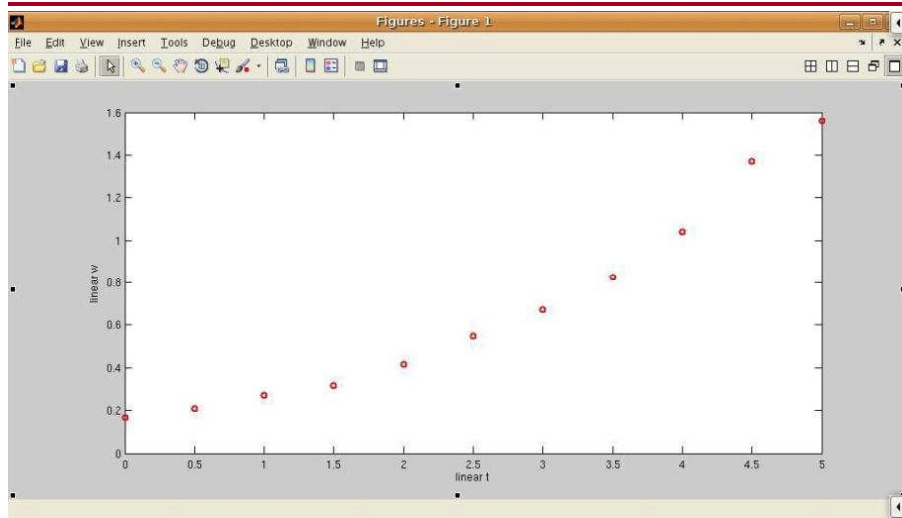


CURVE FITTING





CURVE FITTING



CURVE FITTING

```

1  t = 0:0.5:5;
2  w = [6.00 4.83 3.70 3.15 2.41 1.83 1.49 1.21 0.96 0.73 0.64];
3  p = polyfit(t,log(w),1);
4  m = p(1);
5  b = exp(p(2))           %determine the coefficient b
6  ti = 0:0.1:5;
7  wi = b * exp(m * ti);   %calculate the fnc. value at each element of tm
8  plot(t,w,'o', ti, wi)  %plot the daya points and the function
9  grid
10 xlabel('t, ti')
11 ylabel('w, wi')
12 legend('data', 'polyfit')

```



CURVE FITTING

```

>> clf
>> ex_08_funcsel4

m =

-4.5801e-01

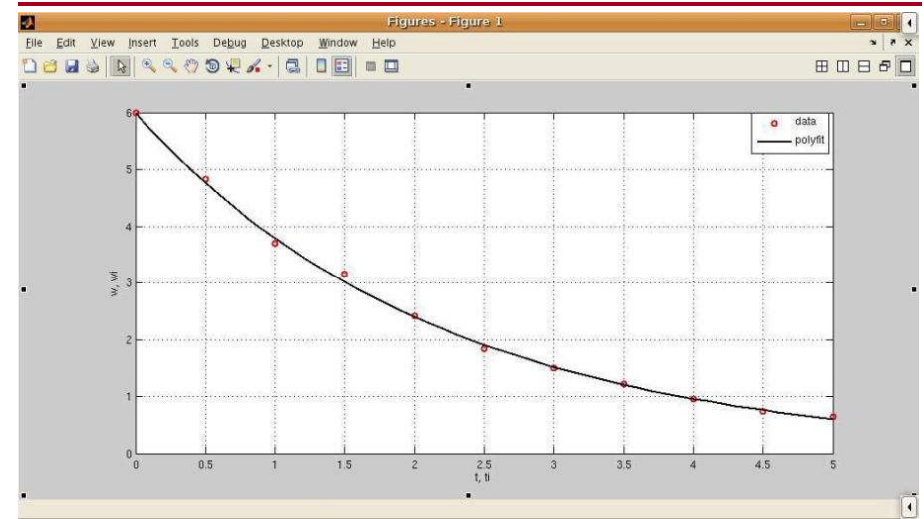
b =

5.9889e+00

```



CURVE FITTING





CURVE FITTING

EXAMPLE:

- The price (in euros) of a magazine has changed as follows:

Nov.87	Dec.88	Nov.90	Jan.93	Jan.95	Jan.96	Nov.96	Nov.00
4.5	5.0	6.0	6.5	7.0	7.5	8.0	8.0

Estimate the price in November 2002 by extrapolating these data.



CURVE FITTING

EXAMPLE:

Latitude	δ_K			
	$K = 0.07$	$K = 1.5$	$K = 2.0$	$K = 3.0$
65	-3.1	3.52	6.05	9.3
55	-3.22	3.62	6.02	9.3
45	-3.3	3.65	5.92	9.17
35	-3.32	3.52	5.7	8.82
25	-3.17	3.47	5.3	8.1
15	-3.07	3.25	5.02	7.52
5	-3.02	3.15	4.95	7.3
-5	-3.02	3.15	4.97	7.35
-15	-3.12	3.2	5.07	7.62
-25	-3.2	3.27	5.35	8.22
-35	-3.35	3.52	5.62	8.8
-45	-3.37	3.7	5.95	9.25
-55	-3.25	3.7	6.1	9.5

- Variation of the average yearly temperature on the Earth for four different values of the concentration K of carbon acid at different latitudes.
 - Compute the least-squares polynomial of degree 4 that approximates the values of K reported in the Table given above.



References for Week 8

- 1 Alfio Quarteroni, Fausto Saleri, Scientific Computing with Matlab and Octave, Springer, 2006.
- 2 Brian Hahn, Daniel T.Valentine, Essential Matlab for Engineers and Scientists, Elsevier, 2010.
- 3 <http://www.mathworks.com/help/techdoc/ref/f16-5872.html#f16-6325>

Have a nice Week End