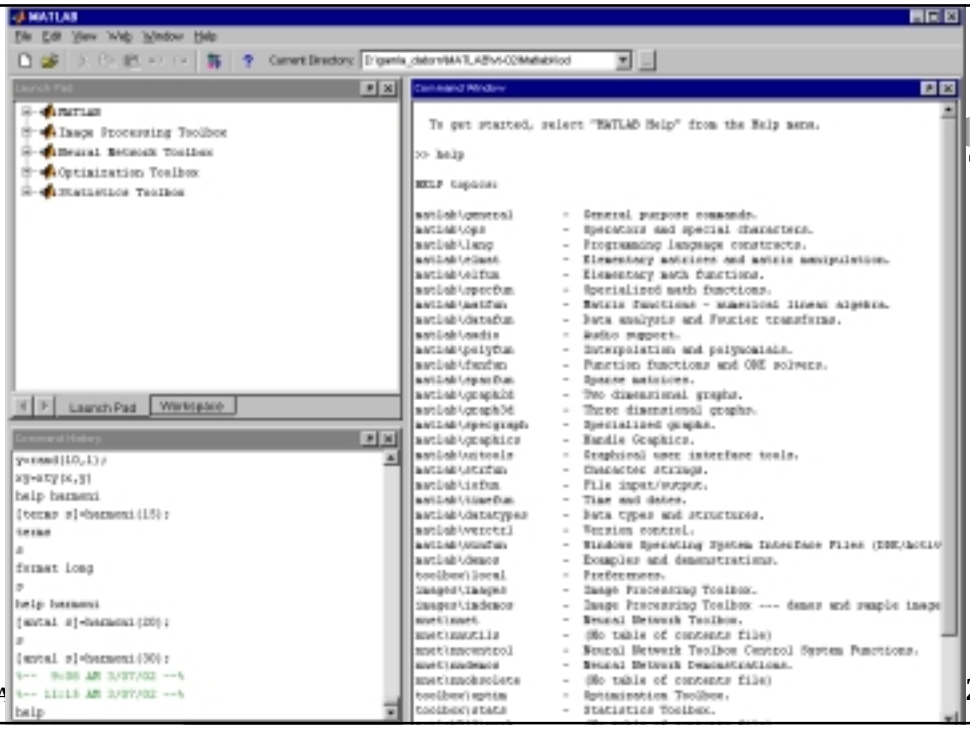
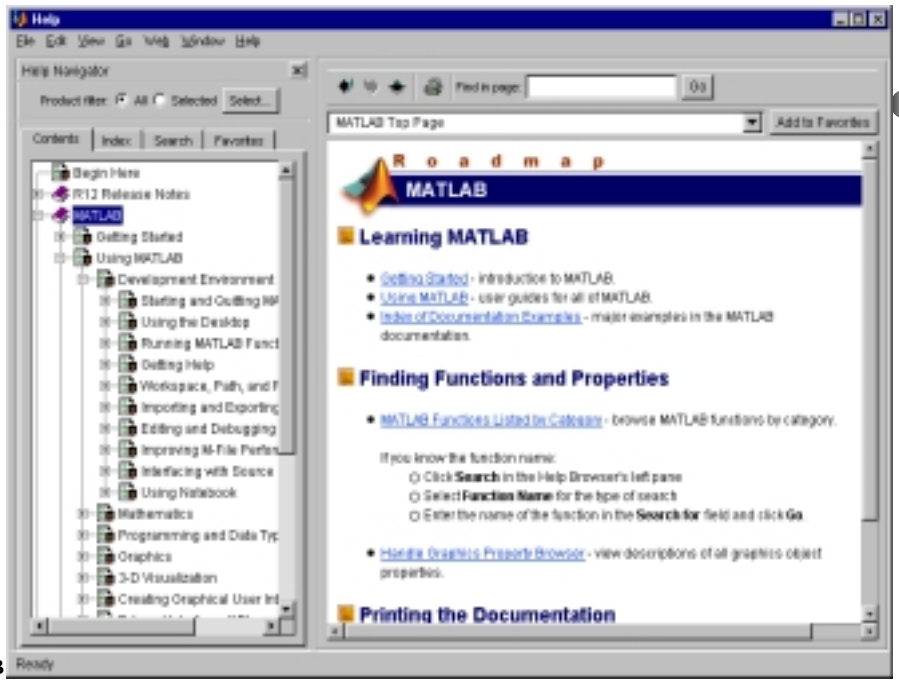
# MATLAB

**MATrix LABoratory**

- **Calculator for matrices**
  - **Environment and programming language**
- **Visualization of data**
  - **Advanced graphics**





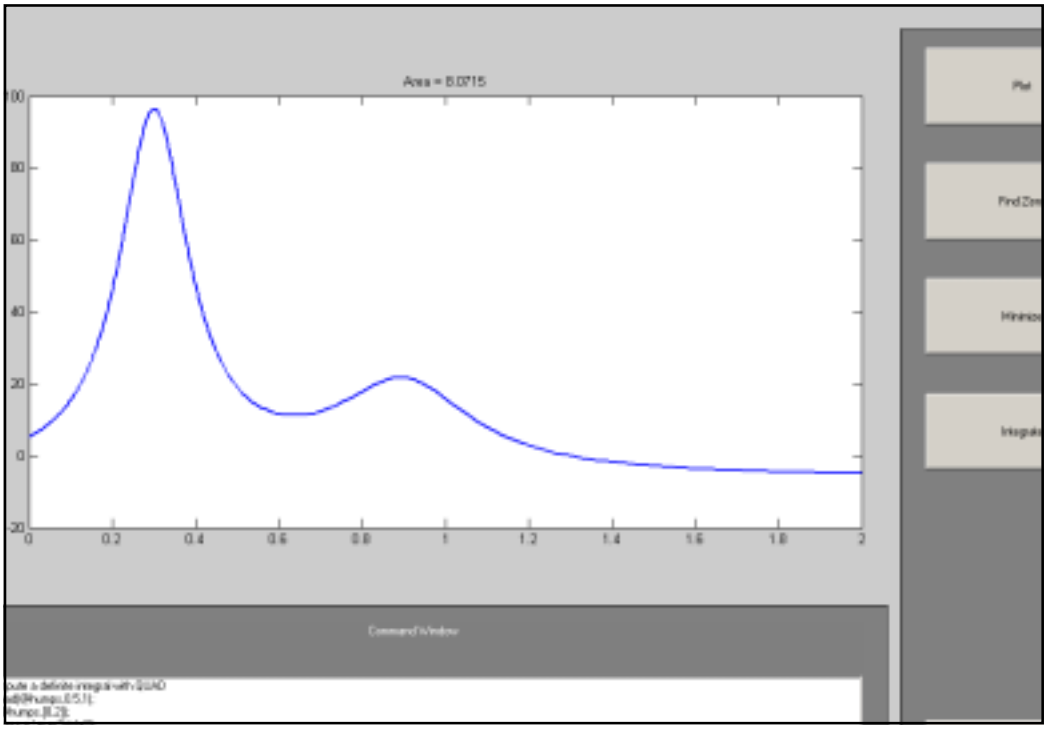





## The language – the command syntax

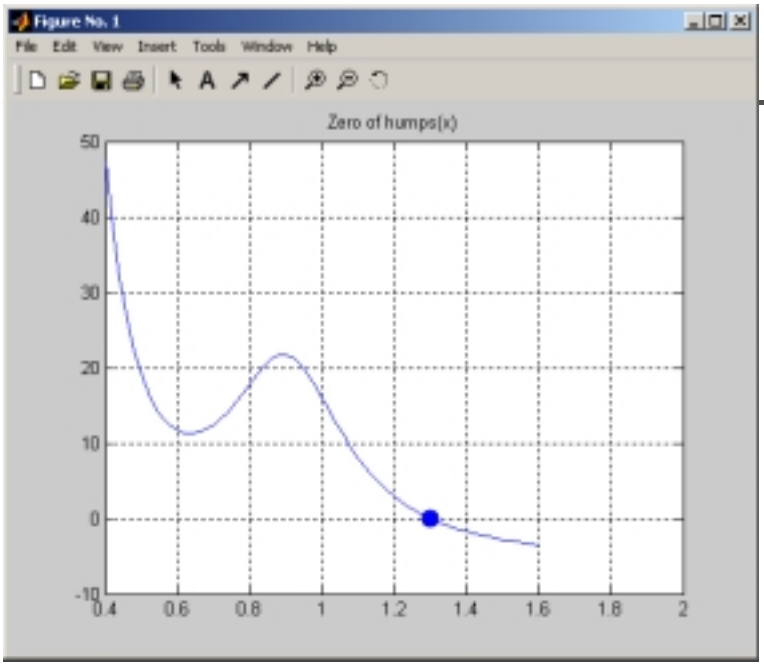
- Resambles mathematics
- A lot of predefined functions and commands
  - Advanced algorithms/methods
  - Initially developed for linear algebra ☺
- Simple to create your own functions and main-programs (M-files)


Minimize
Integrate
Info
Close
Launch Pad
Workspace
Command Window
Command History
chainmat - Matrix chain multiplication optimization.
General Demo/Helper functions.
cmdlnwin - Demo gateway routine for playing command line demos.
cmdlnopn - Set up for command line demos.
cmdlnend - Clean up after command line demos.
fundemos - Find demos available for individual toolboxes.
helpfun - Utility function for displaying help text conveniently.
pltmat - Display a matrix in a figure window.
MATLAB/Helper functions.
barbrp - The graph of the Buckminster Fuller geodesic dome.
peaks - A sample function of two variables.
membrane - Generates MathWorks' logo.
See also HIMDEMOS
demos is both a directory and a function.
DEMOS Demo list for the Image Processing Toolbox.


Area = 8.0715
Plot
Minimize
Integrate
Command Window

```
>>  type humps
function [out1,out2] = humps(x)
%HUMPS  A function used by QUADDEMO, ZERODEMO and FPLOTDEMO.
%   Y = HUMPS(X) is a function with strong maxima near x = .3
%   and x = .9.
%
%   [X,Y] = HUMPS(X) also returns X.  With no input arguments,
%   HUMPS uses X = 0:.05:1.
%
%   Example:
%      plot(humps)
%
%   See QUADDEMO, ZERODEMO and FPLOTDEMO.
%   Copyright 1984-2001 The MathWorks, Inc.
%   $Revision: 5.7 $  $Date: 2001/04/15 12:03:04 $
if nargin==0, x = 0:.05:1; end
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
if nargout==2,
  out1 = x; out2 = y;
else
  out1 = y;
end
>> zerodemo
```





## Variables in Matlab

- **no declaration of variables (matrix is the key-word)**
  - **Variables are defined by assignment**
- **Assignments are done by = ended by Enter**

```
>>A = 7.5*2
>>7.5*2
```

**(result in** ans**),** ans is a variable

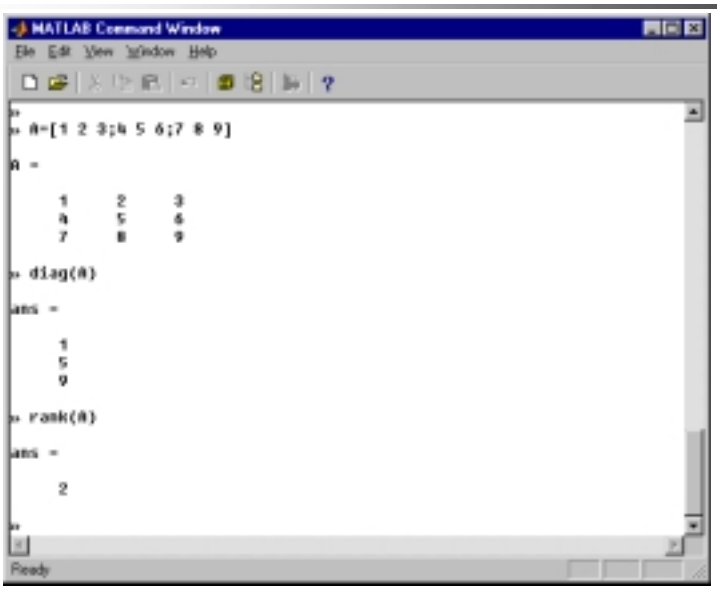
- **1×1-matrices are called scalars**
- **Semicolon at the end of a statement inhibits print-out**
- **Case-sensitive**



## ...more

- **Comments start with % and long for the rest of the line**
- **The basic data type is matrix (normally with 16 decimals precision)**
- **Possible to handle complex numbers**

## Create a matrix






## Matrices

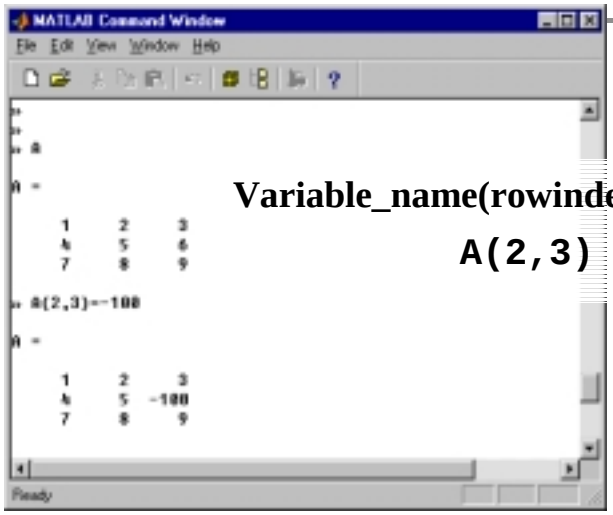
**assignment**

- **surrounded by** `[]`
- **Row wise**
- **space or comma between elements**
- **rows in a matrix are separated by semicolon**

**Printing values**

- **The name of the variable and Enter**
- **>>A**
- **The value returned by a function**
- **>>eig(A) %prints eigenvalues of A**
- **Make** `help eig`

## Access to individual elements in matrix



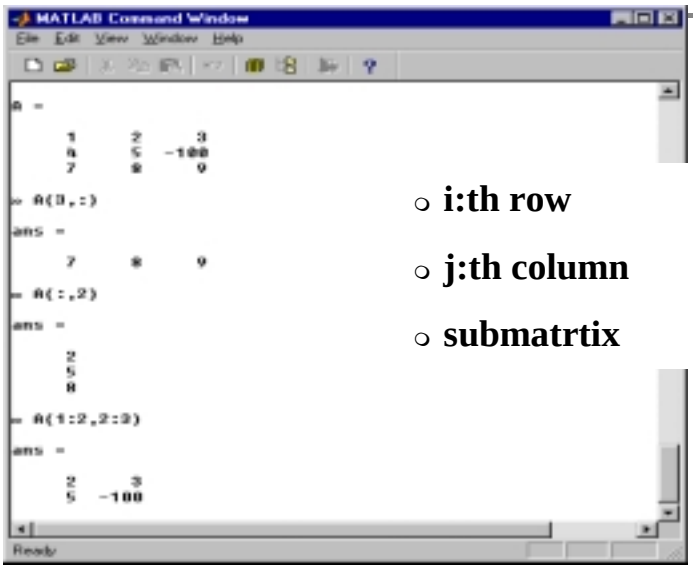

**Variable_name(rowindex,columnindex)**

**`A(2,3)`**

Note! Difference between assignment and accessing



## Part of a matrix




all

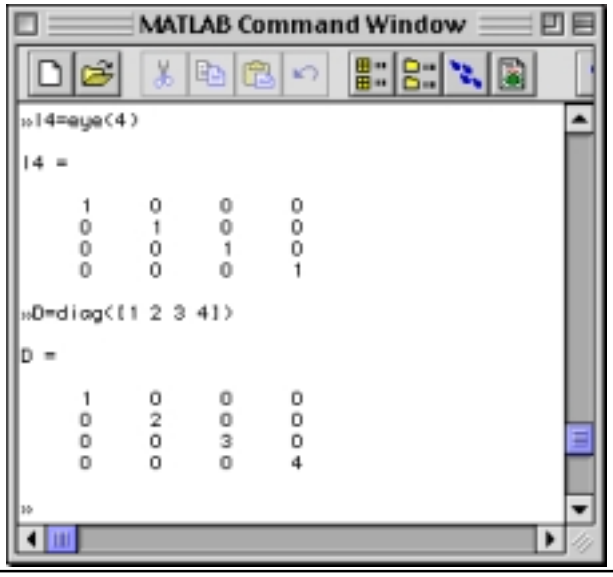
- **i:th row** `A(i,:)`
- **j:th column** `A(:,j)`
- **submatrtix** `A(i:j,k:l)`

## Special matrices

- `ones(n)` — n × n-matrix with ones
- `ones(m,n)` — m × n-matrix
- `zeros(n)` — n × n-matrix with zeros
- `eye(n)` — identity matrix of order n × n
- `diag()` — gives the main diagonal/creates a diagonal matrix
- `triu(),tril()` — gives upper or lower triangular matrix



## eye & diag



## : operator

- `start : step : stop`
  - gives a sequences of values (start < stop)
    - `start,start+step,start+2*step,.,stop`

- `x = -pi/2:2*pi/60:pi/2;`
  gives a row vector
- `[m n]=size(A); %gives dimension of A`



## Matrix operations

- `A'` (conjugate transpose) transpose
- `det(A)` determinant
- `inv(A)` inverse
- `eig(A)` eigenvalues
- `norm(A)` 2-norm
- `A*B,A+B`

¡ matrisoperatorer !

## Element wise operations

```
x=linspace(0,1); %!Row vector
y=x.^n.*exp(x);% gives row vector
plot(x,y,'k')

plot(x, x.^n.*exp(x))
```



## Control structures...

```
for variable = expression
   statements
end
for i = 1:2:n
   statements
end
for i= 1:n
  statements
end
```

**Compare C**

```
for (i=1; i<=n; i=i+2)
{
   statements
}
```

## ...more on control

```
if logical expression
   statements
end
```

```
if logical expression
   statements
else
   statements
end
```

```
if logical expression
   statements
elseif logical expression
   statements
end
```



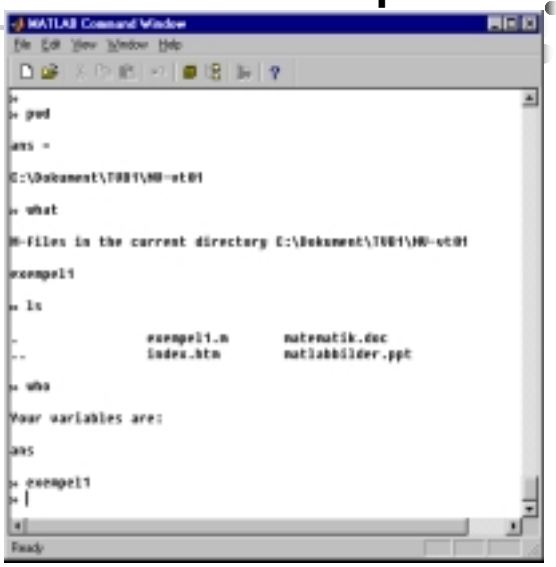
## ...more...

```
while logical expression
   statements
end
```

## exempel1.m

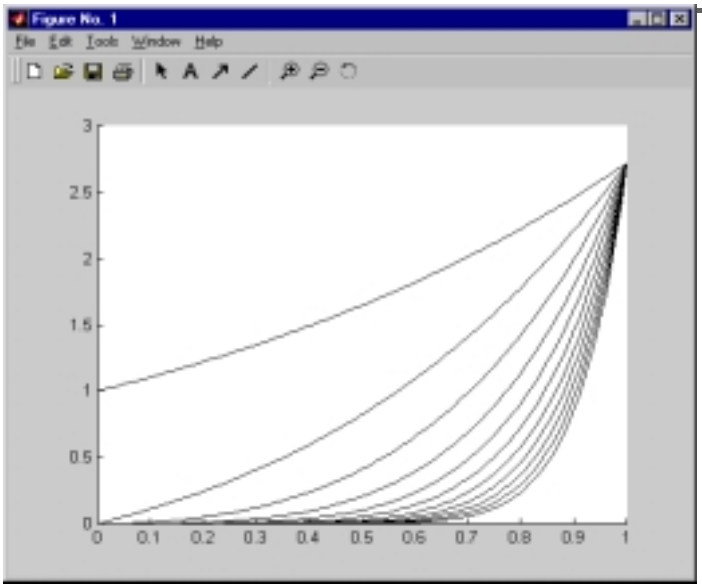
**Your own file** (exempel1.m)

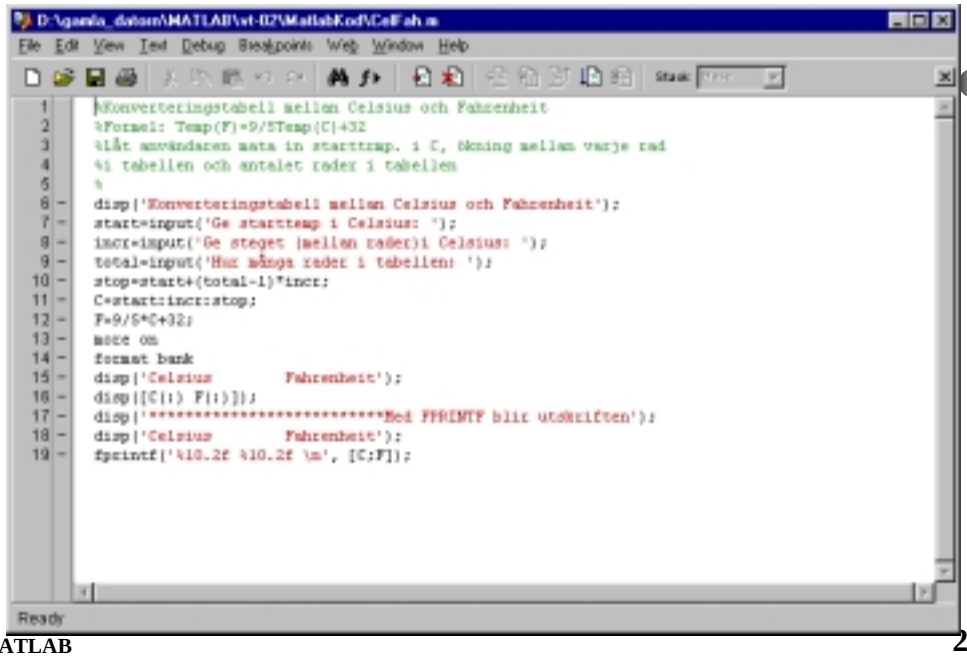
```
x=linspace(0,1);
hold on
for  n=0:10
  y=x.^n.*exp(x);
  plot(x,y,'k')
end
```





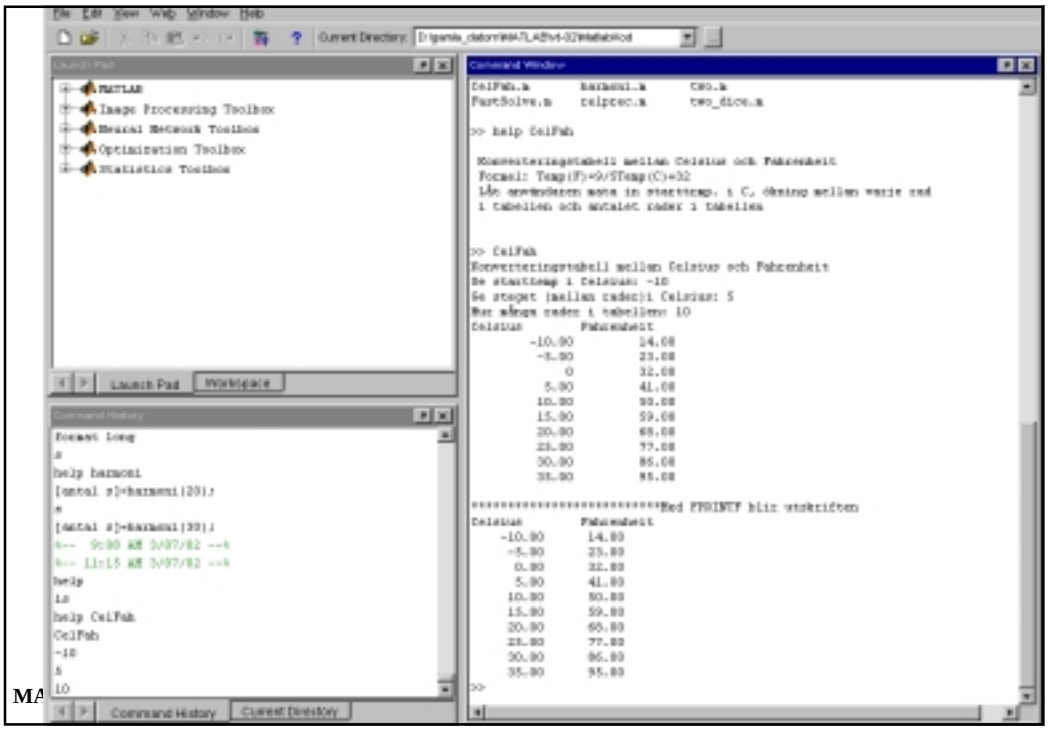

## Figure window








```
%Konverteringstabell mellan Celsius och Fahrenheit
%Formel: Temp(F)=9/5Temp(C)+32
%Låt användaren mata in starttemp. i C, ökning mellan varje rad
%i tabellen och antalet rader i tabellen
%
disp('Konverteringstabell mellan Celsius och Fahrenheit');
start=input('Ge starttemp i Celsius: ');
incr=input('Ge steget (mellan rader)i Celsius: ');
total=input('Hur många rader i tabellen: ');
stop=start+(total-1)*incr;
C=start:incr:stop;
F=9/5*C+32;
more on
format bank
disp('Celsius        Fahrenheit');
disp([C(:) F(:)]);
disp('*************************Med FPRINTF blir utskriften');
disp('Celsius        Fahrenheit');
fprintf('%10.2f %10.2f \n', [C;F]);
```




```
>>help CelFah

 Konverteringstabell mellan Celsius och Fahrenheit
 Formel: Temp(F)=9/5Temp(C)+32
 Låt användaren mata in starttemp. i C, ökning mellan varje rad
 i tabellen och antalet rader i tabellen


>> CelFah
Konverteringstabell mellan Celsius och Fahrenheit
Ge starttemp i Celsius: -10
Ge steget (mellan rader)i Celsius: 5
Hur många rader i tabellen: 10
Celsius        Fahrenheit
        -10.00         14.00
         -5.00         23.00
             0         32.00
          5.00         41.00
         10.00         50.00
         15.00         59.00
         20.00         68.00
         25.00         77.00
         30.00         86.00
         35.00         95.00
```

```
************************Med FPRINTF blir
utskriften
Celsius        Fahrenheit
    -10.00      14.00
     -5.00      23.00
      0.00      32.00
      5.00      41.00
     10.00      50.00
     15.00      59.00
     20.00      68.00
     25.00      77.00
     30.00      86.00
     35.00      95.00
>>
```



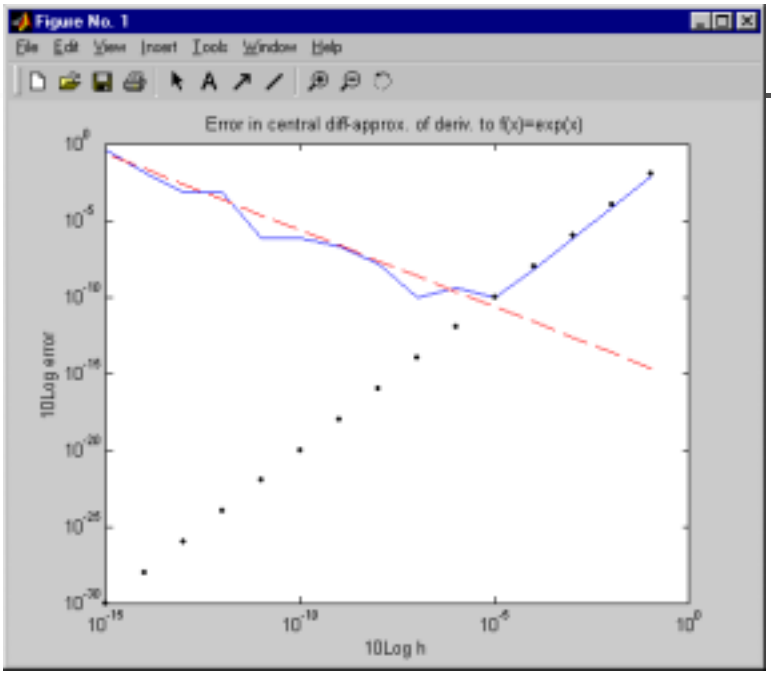
## Graphics - 2D

- `plot()` – draws a set of ordered pairs of points with or without lines between (marks, colours)
- `fplot()` – draw curve of a one-dimensional function `hold` – next plot is drawn in same plot window as before
- `title(), xlabel(), ylabel()` – commands for text in plot windows
- `contour()` – draws contour plots of 2-dim functions

```
% Study the error when the derivative of f(x)=exp(x)
% is approximated by
% fprim(x)= (f(x+h)-f(x-h))/(2h)
%
format compact
x=input('Give x-value for approximation: ')
%
%Define all h-values in an array
%
t=0.1;
i=1;
while t>eps,
 h(i)=t;
 i=i+1;
 t=t/10;
end %of while
fxph=exp(x+h);
fxmh=exp(x-h);
deriv=(fxph-fxmh)./(2*h);
error=exp(x)-deriv;
format short e
minerr=min(abs(error))
min_ind=find((error == minerr)|(error == -minerr));
```

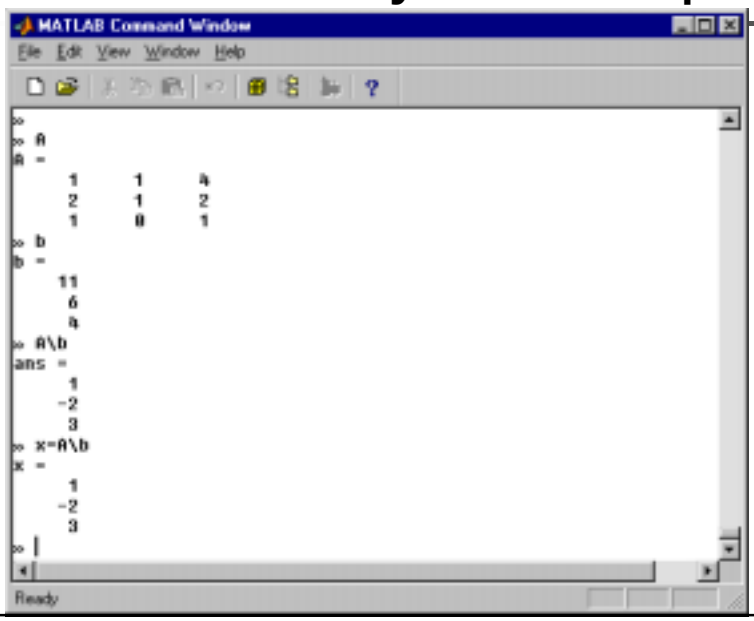


```
if length(min_ind) == 1
 disp('for h= '), disp(h(min_ind))
else
 disp('There are more than one h-valu that give minerror')
 min_ind
end %of if
error=abs(error);
RXF=eps./h;
RT=h.^ 2;
clf
loglog(h,error)
title('Error in central diff-approx. of deriv. to f(x)=exp(x)')
xlabel('10Log h')
ylabel('10Log error')
hold on
loglog(h,RXF,'r--')
loglog(h,RT,'k.')
```



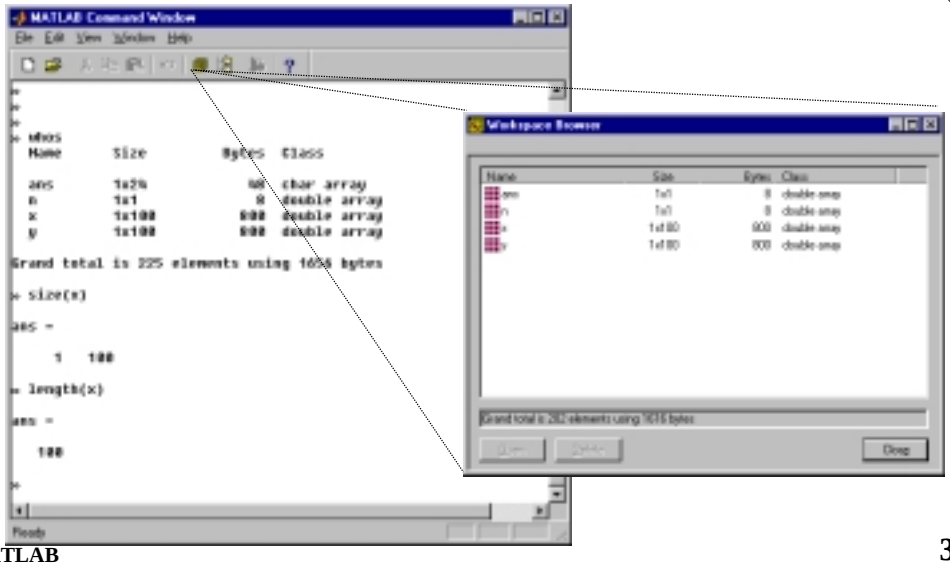

MATLAB 33

## Linear systems of equations




MATLAB 34

## who/whos






## what/pwd

```
» what
M-files in the current directory
  C:\Dokument\TVB1\NV-vt01
exempel1
» pwd
ans =
C:\Dokument\TVB1\NV-vt01
»
```

## File/New/M_file




## Functions

- **One function – one file** (normally same name on function and file)
- **The first line should start with `function`**

  **(otherwise a command file** approx.= **function main)**

  ```
  function [out-params.] = name(in-params.)
  ```
- **Zero, or more in-parameters – call by value**
- **Call by `name(parameters)`**
- **Comment rows directly after first line are printed with `>>help filename`**

```
function s = xty(x, y)
%Call: s=xty(x,y)
%Compute the scalar product between the vectors x and y
%
if length(x) ~= length(y)
   error('The vectors should have the same length');
end
s=0; /*This is the C-version */
for k=1:length(x)
   s=s+x(k)*y(k);
end
%A faster version
s1=x(:)'*y(:); % Make sure x and y are column vectors
%Use the predefined function dot
s2=dot(x,y);
if (s==s1) & (s==s2)
   disp('All three sums are equal')
else
   disp('Different sums'); /* No compound statement */
   format long
   [s s1 s2]
end
```



```
function [terms, sum] = harmoni(upper)
%Call: [terms sum] = harmoni(upper)
%Compute the sum of 1/k; k=1,2,3,4,5,.........
%until that sum exceeds upper.
%terms = no. of terms in the sum
%sum = the actual sum
%Also there is a printout for each time the sum
%exceeds "the next" integer.
%
sum=0;
k=0;
oldsum=-1;
heltal=1;
while (sum <= upper) & (oldsum ~= sum)
   k=k+1;
   oldsum=sum;
   sum=sum+1/k;
   if sum >= heltal
      fprintf(1,'%d terms needed to exceed %d \n', k, heltal);
      heltal=heltal+1;
   end %of if
end %of while
terms=k;
```

```
>> [termer summa]=harmoni(15);
1 termer behövs för att överskrida 1
4 termer behövs för att överskrida 2
11 termer behövs för att överskrida 3
31 termer behövs för att överskrida 4
83 termer behövs för att överskrida 5
227 termer behövs för att överskrida 6
616 termer behövs för att överskrida 7
1674 termer behövs för att överskrida 8
4550 termer behövs för att överskrida 9
12367 termer behövs för att överskrida 10
33617 termer behövs för att överskrida 11
91380 termer behövs för att överskrida 12
248397 termer behövs för att överskrida 13
675214 termer behövs för att överskrida 14
1835421 termer behövs för att överskrida 15
>> termer
termer =
        1835421
>> format compact
>> summa
summa =
    15.0000
>> format long
>> summa
summa =
   15.000000378267 23
```



```
function [pos, neg] = split1(x)
%Call: [pos neg]=split1(x)
%x= a vector with numeric values
%pos= a vector containing the positive values of x
%neg= a vector containing the negative values of x
%
%This is the straight forward solution in Matlab
indpos=0;
indneg=0;
for k=1:length(x)
   if x(k)<0
      indneg=indneg+1;
      neg(indneg)=x(k);
   else
      indpos=indpos+1;
      pos(indpos)=x(k);
   end %of if
end %of for k=......

>> type split2
function [pos, neg] = split2(x)
%Call: [pos, neg]=split2(x)
%x= a vector with numeric values
%pos= a vector containing the positive values of x
%neg= a vector containing the negative values of x
%
%This is the short, but not obvious solution in Matlab
indpos=find(x>=0);
indneg=find(x<0);
pos=x(indpos);
neg=x(indneg);
```

## Moreover

- **Easy to create GUIs**
  - **buttons**
  - **radio buttons**
  - **pop-up menues**
  - **fields for input**
- **TOOLBOXES for optimization, neural networks, image analysis and more are avilable**