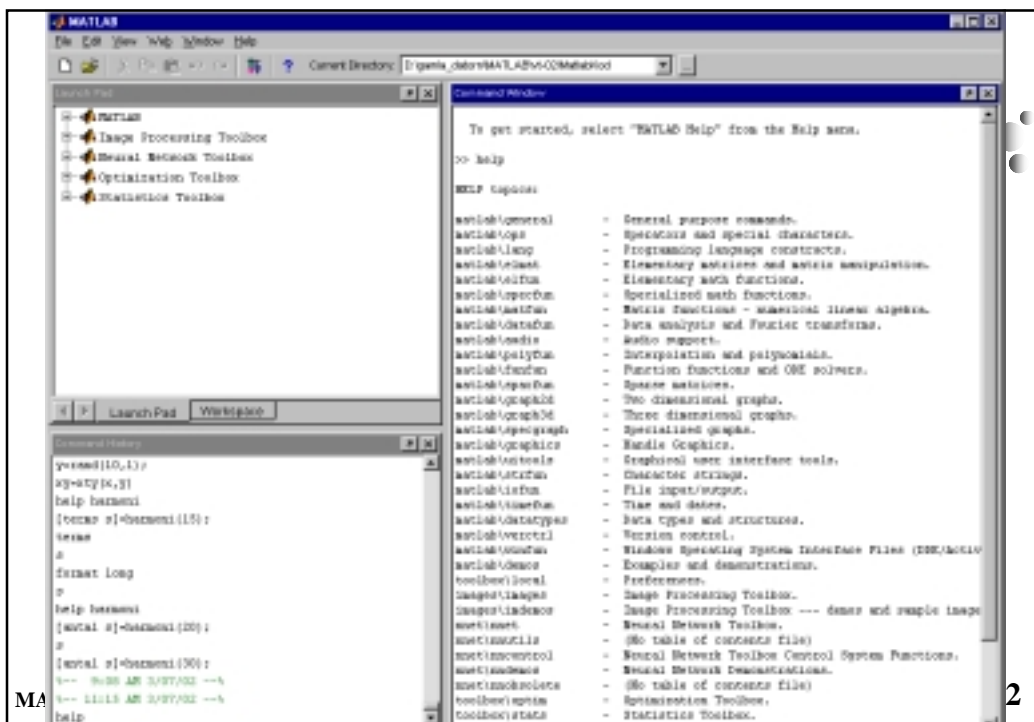
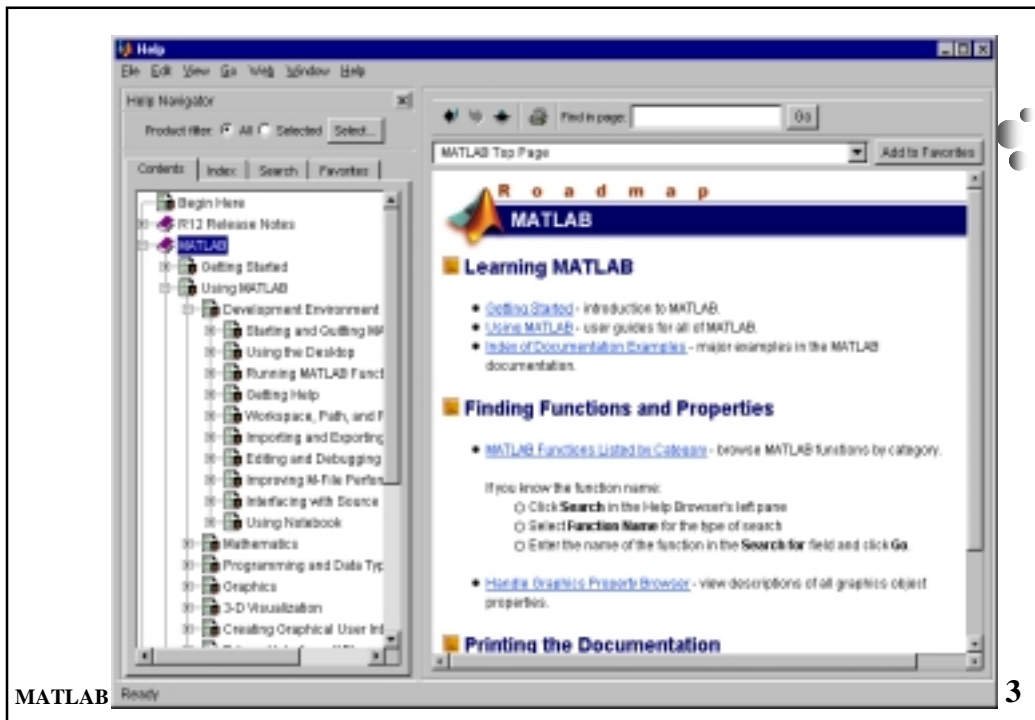


## MATRix LABoratory

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





## The language - the command syntax

- Resembles 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)

The image shows the MATLAB Optimization Toolbox interface. At the top, a plot displays a function with a red dot indicating the current optimization point. Below the plot is the Command Window with the following code:

```

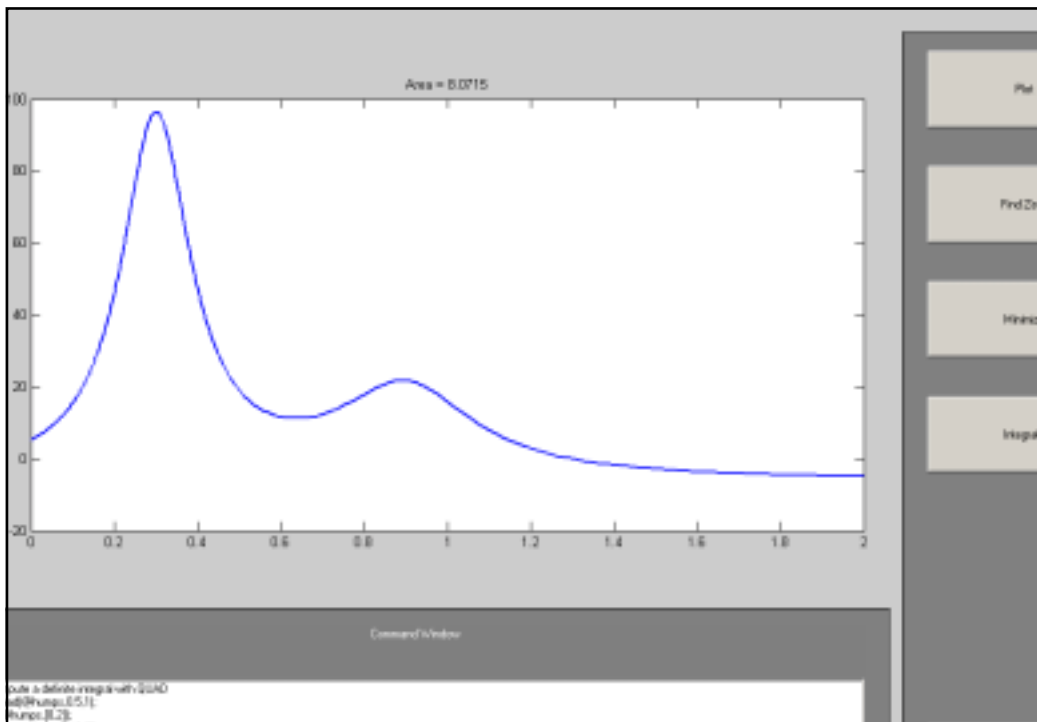
% Find a zero of fzero() near 1 with fzero
z = fzero(@funp,1,'options(@display','off)');
plot(funp(z),z);
hold on; plot(z,1); hold off

```

To the right of the Command Window is a vertical toolbar with buttons for 'Minimize', 'Integrate', 'Help', and 'Done'. Below the Command Window is a Help browser window showing the 'General Demo/Helper functions' section, which includes a list of functions and their descriptions:

- General Demo/Helper functions.**
  - `cadlines` - Demo gateway routine for plotting coarsened line demos.
  - `cadsetup` - Set up for coarsened line demos.
  - `cadcleanup` - Clean up after coarsened line demos.
  - `finddemo` - Find demos available for individual toolboxes.
  - `helpfun` - Utility function for displaying help text conveniently.
  - `plotmat` - Display a matrix in a figure window.
- MATLAB/Helper functions.**
  - `looky` - The graph of the Submarine Faller gradient demo.
  - `peaks` - A sample function of two variables.
  - `webdemo` - Generate MathWorks' logo.

Below the Help browser window is a section titled 'See also SIMDEMO' and a description of the `demo` function: `demo` is both a directory and a function. `DEMO Demo list for the Image Processing Toolbox.`



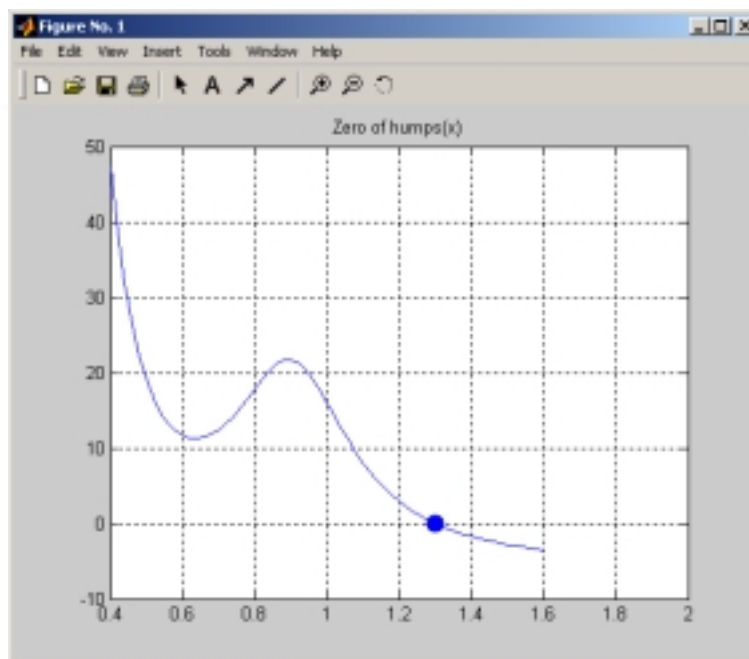
```

>> type humps
function [out1,out2] = humps(x)
%HUMPS A function used by QUADDEMO, ZERODEMO and FPLOTDDEMO.
% 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 FPLOTDDEMO.
% 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 nargin==2,
    out1 = x; out2 = y;
else
    out1 = y;
end
>> zerodemo

```

MATLAB

7



MATLAB

8

## 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**

MATLAB

9

...more

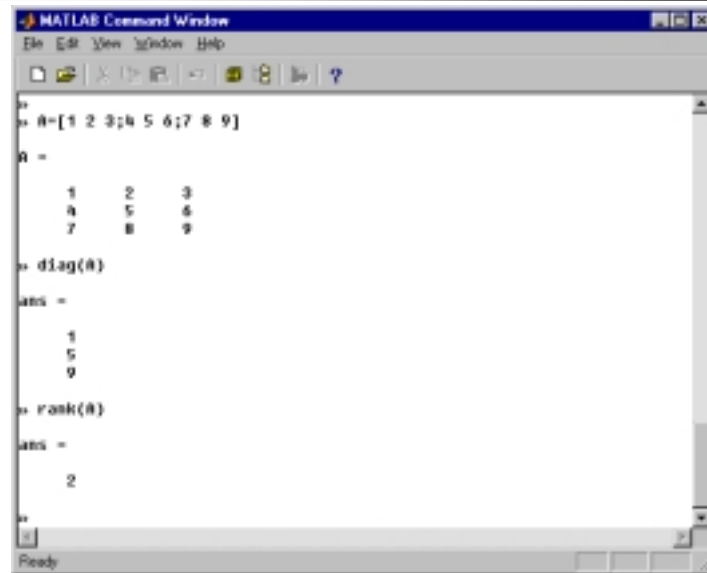
---

- **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**

MATLAB

10

## Create a matrix



```
MATLAB Command Window
File Edit View Window Help
A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> diag(A)
ans =
     1
     5
     9
>> rank(A)
ans =
     2
```

MATLAB

11

## Matrices

### assignment

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

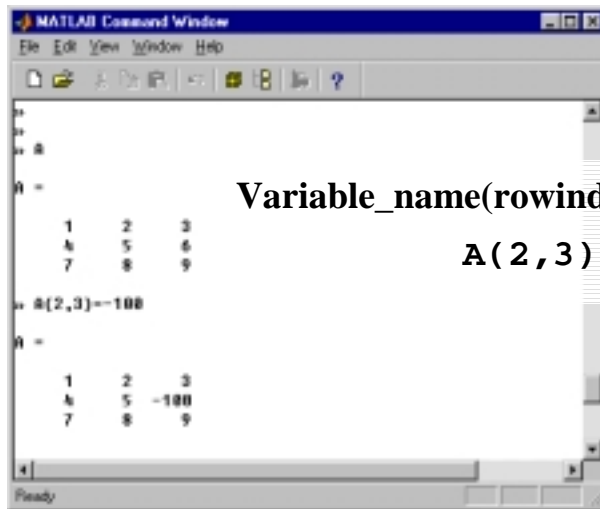
### Printing values

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

MATLAB

12

## Access to individual elements in matrix



```
MATLAB Command Window
File Edit View Window Help
A =
     1     2     3
     4     5     6
     7     8     9

A(2,3) = -100

A =
     1     2     3
     4     5    -100
     7     8     9
```

Variable\_name(rowindex,columnindex)

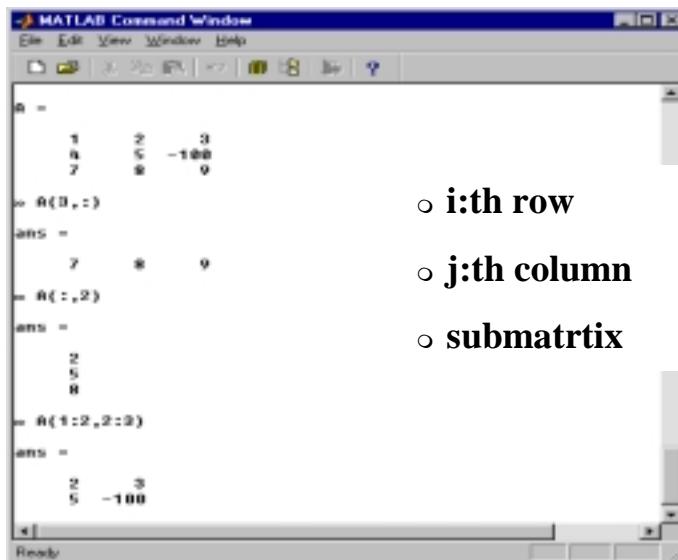
$A(2,3)$

Note! Difference between assignment and accessing

MATLAB

13

## Part of a matrix



```
MATLAB Command Window
File Edit View Window Help
A =
     1     2     3
     4     5    -100
     7     8     9

>> A(i,:)
ans =
     7     8     9

>> A(:,j)
ans =
     2
     5
     8

>> A(i:j,k:l)
ans =
     2     3
     5    -100
```

○ i:th row

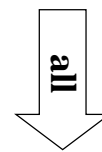
$A(i, :)$

○ j:th column

$A(:, j)$

○ submatrix

$A(i:j,k:l)$



MATLAB

14

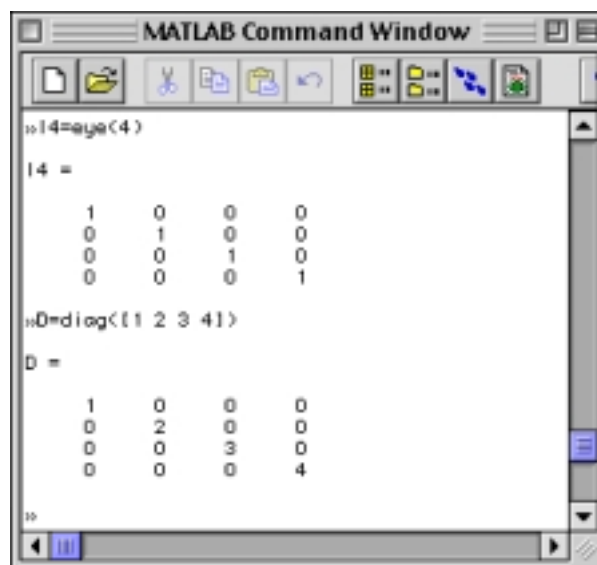
## Special matrices

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

MATLAB

15

## eye & diag



```

MATLAB Command Window
>> I4=eye(4)
I4 =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> D=diag([1 2 3 4])
D =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
>>

```

MATLAB

16



## : 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`

MATLAB

17

## Matrix operations

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

i matrisoperatorer!

MATLAB

18

## 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))
```

MATLAB

19

## 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
}
```

MATLAB

20

## ...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...

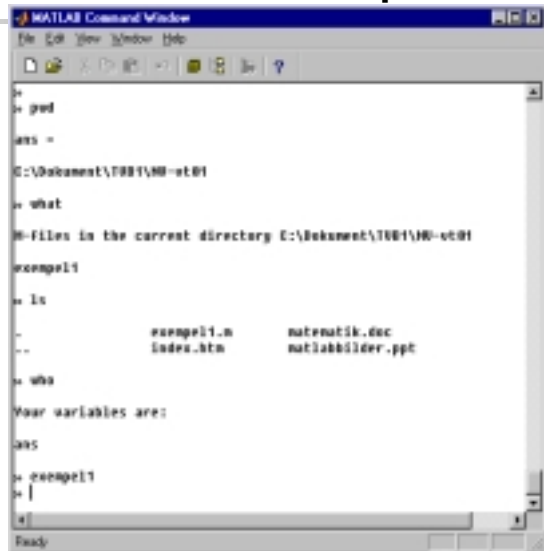
---

```
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
```

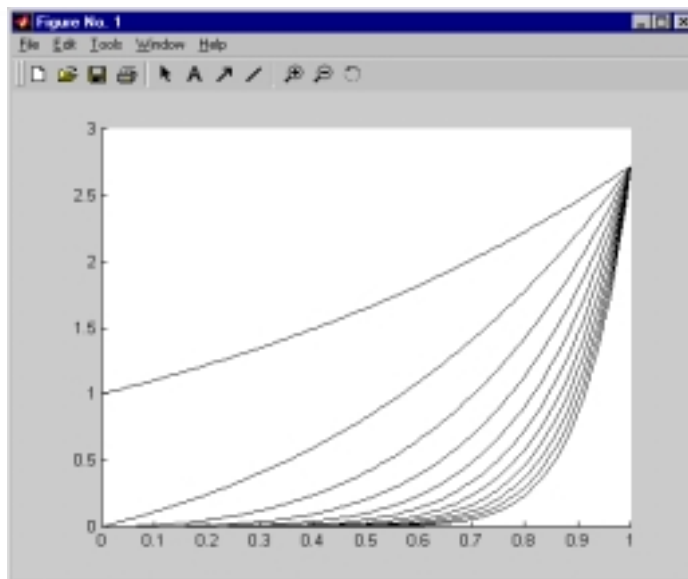


```
MATLAB Command Window  
File Edit View Window Help  
> pwd  
ans =  
C:\dokument\T001\NO-et01  
> what  
M-files in the current directory C:\dokument\T001\NO-et01  
exempel1  
> ls  
--  
--          exempel1.m      matematik.doc  
--          index.htm      matlabbilder.ppt  
> who  
Your variables are:  
ans  
> exempel1  
=> |  
n |  
0 |  
1 |  
2 |  
3 |  
4 |  
5 |  
6 |  
7 |  
8 |  
9 |  
10|  
Ready
```

MATLAB

23

## Figure window



MATLAB

24

```

D:\agencia_dotcom\MATLAB\vt-ID\MatlabKod\CeFah.m
File Edit View Insert Debug Breakpoints Web Window Help
1 %Konverteringstabell mellan Celsius och Fahrenheit
2 %Formel: Temp(F)=9/5Temp(C)+32
3 %Låt användaren mata in starttemp. i C, ökning mellan varje rad
4 %i tabellen och antalet rader i tabellen
5 %
6 -
7 - disp('Konverteringstabell mellan Celsius och Fahrenheit');
8 - start=input('Ge starttemp i Celsius: ');
9 - incr=input('Ge steget (mellan rader)i Celsius: ');
10 - total=input('Hur många rader i tabellen: ');
11 - stop=start+(total-1)*incr;
12 - C=start:incr:stop;
13 - F=9/5*C+32;
14 - more on
15 - format bank
16 - disp('Celsius          Fahrenheit');
17 - disp([C(:) F(:)]);
18 - disp('*****Med FPRINTF blir utskriften');
19 - disp('Celsius          Fahrenheit');
20 - fprintf('%10.2f %10.2f \n', [C;F]);
Ready

```

MATLAB

25

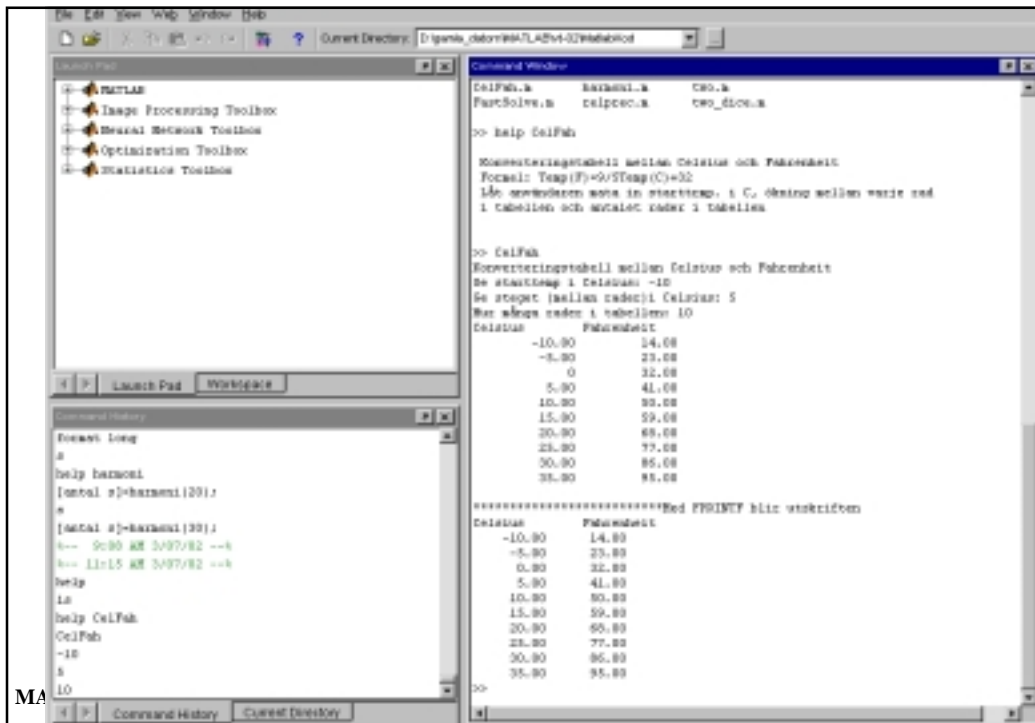
```

%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]);

```

MATLAB

26



MA

```

>>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
  
```

MATLAB

28

```

*****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));

```

MATLAB

31

```

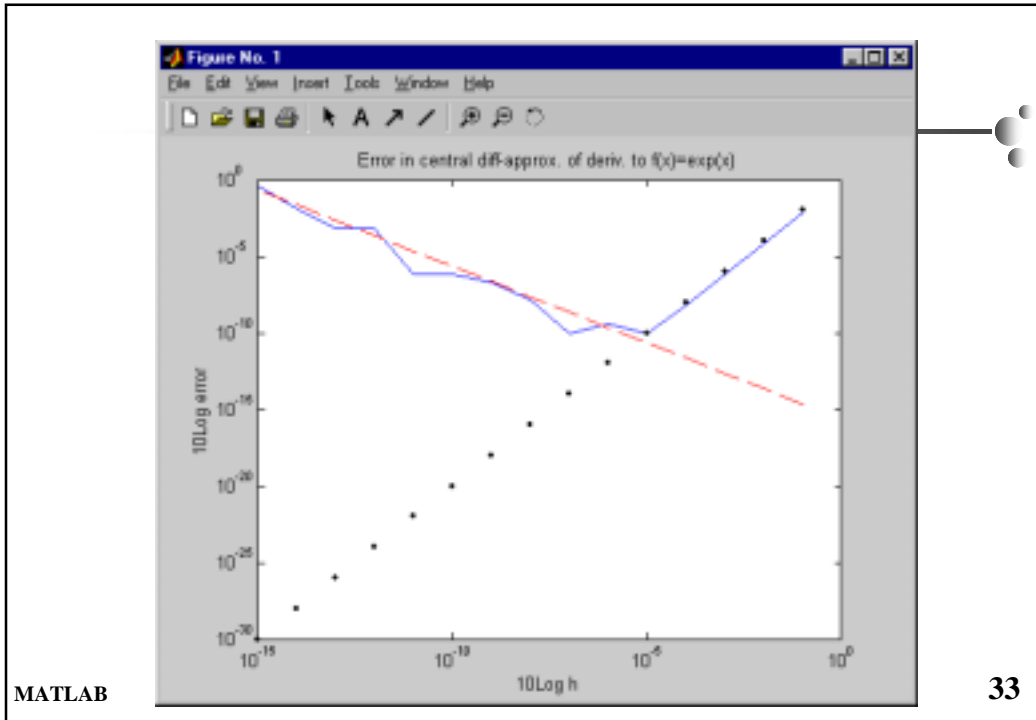
if length(min_ind) == 1
    disp('for h= '), disp(h(min_ind))
else
    disp('There are more than one h-valu that give minerror')
    min_ind
end %of if
error=abs(error);
RXF=eps./h;
RT=h.^ 2;
clf
loglog(h,error)
title('Error in central diff-approx. of deriv. to f(x)=exp(x)')
xlabel('10Log h')
ylabel('10Log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

```

MATLAB

32





### Linear systems of equations

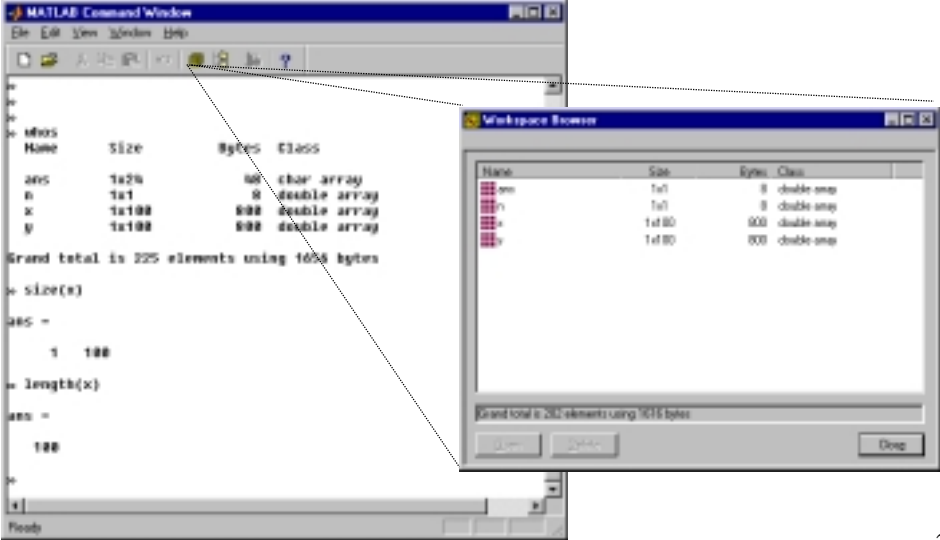
```

>> A
A =
     1     1     4
     2     1     2
     1     0     1
>> b
b =
    11
     6
     4
>> A\b
ans =
     1
    -2
     3
>> x=A\b
x =
     1
    -2
     3
  
```

MATLAB

34

## who/whos



The image shows two MATLAB windows. The 'MATLAB Command Window' displays the output of the 'whos' command, listing variables 'ans', 'n', 'x', and 'y' with their sizes and classes. Below this, the 'size(x)' and 'length(x)' commands are executed, showing the dimensions of variable 'x'. The 'Workspace Browser' window shows a table of workspace variables: 'ans' (1x1 char array), 'n' (1x1 double array), 'x' (1x100 double array), and 'y' (1x100 double array). The total size is 225 elements using 1638 bytes.

```
ans      Size      Bytes  Class
-----
ans      1x26      88    char array
n         1x1       8     double array
x         1x100    800   double array
y         1x100    800   double array

Grand total is 225 elements using 1638 bytes

> size(x)
ans =
     1    100

> length(x)
ans =
    100
```

Name	Size	Bytes	Class
ans	1x1	8	double array
n	1x1	8	double array
x	1x100	800	double array
y	1x100	800	double array

Grand total is 225 elements using 1638 bytes

MATLAB 35

## what/pwd

» what

M-files in the current directory

C:\Dokument\TVB1\NV-vt01

exempell

» pwd

ans =

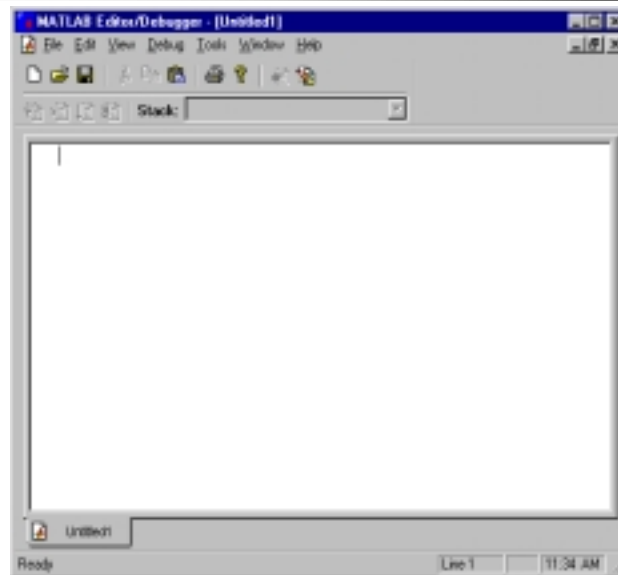
C:\Dokument\TVB1\NV-vt01

»

MATLAB

36

## File/New/M\_file



MATLAB

37

## 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`

MATLAB

38

```

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

```

MATLAB

39

```

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;

```

MATLAB

40

```

>> [termer summa]=harmoni(15);
1 termér behövs för att överskrida 1
4 termér behövs för att överskrida 2
11 termér behövs för att överskrida 3
31 termér behövs för att överskrida 4
83 termér behövs för att överskrida 5
227 termér behövs för att överskrida 6
616 termér behövs för att överskrida 7
1674 termér behövs för att överskrida 8
4550 termér behövs för att överskrida 9
12367 termér behövs för att överskrida 10
33617 termér behövs för att överskrida 11
91380 termér behövs för att överskrida 12
248397 termér behövs för att överskrida 13
675214 termér behövs för att överskrida 14
1835421 termér behövs för att överskrida 15
>> termér
termér =
1835421
>> format compact
>> summa
summa =
15.0000
>> format long
>> summa
summa =
15.00000037826723

```

MATLAB

41

```

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);

```

MATLAB

42

## Moreover

---

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