Linear algebra & Numerical Analysis

Introduction to MATLAB

Marta Jarošová

http://homel.vsb.cz/~dom033/

Outline

- What is it MATLAB?
- MATLAB Environment and MATLAB Help
- Variables, matrices and vectors
- Strings
- .m files: scripts and functions
- Flow control
- 2D, 3D graphics
- Guide

What is it MATLAB?

- MATLAB = “MATrix LABoratory”
- a high-performance language for technical computing
  - *computation, visualization, and programming environment*

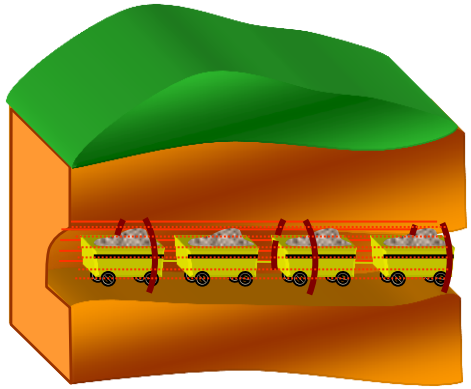
- a modern programming language environment
  - sophisticated *data structures*
  - built-in *editing* and *debugging tools*
  - support of *object-oriented programming*

➔ an excellent tool for teaching and research

Matlab tools

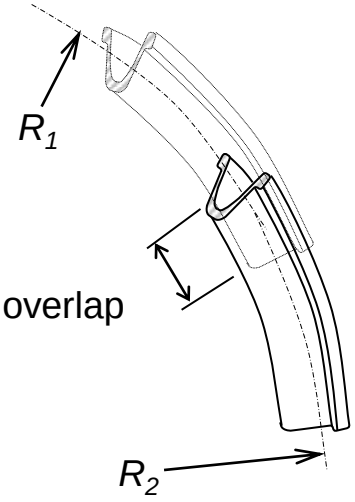
- powerful ***built-in routines*** enable a very wide variety of computations
- easy to use ***graphics commands*** that make the visualization of results immediately available
- specific applications are collected in ***toolboxes****:*
  - signal processing
  - symbolic computation
  - control theory
  - simulation
  - optimization
  - parallel computing
  - and several other fields of applied science and engineering

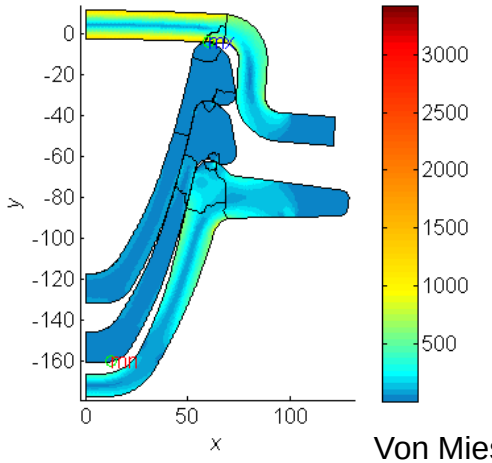
Real world problem: Mining industry

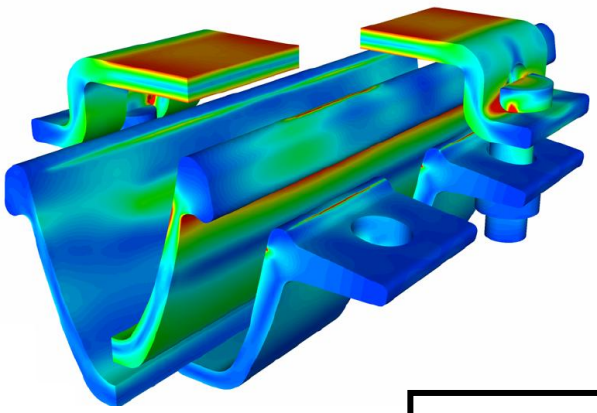


steel support

clamp joint



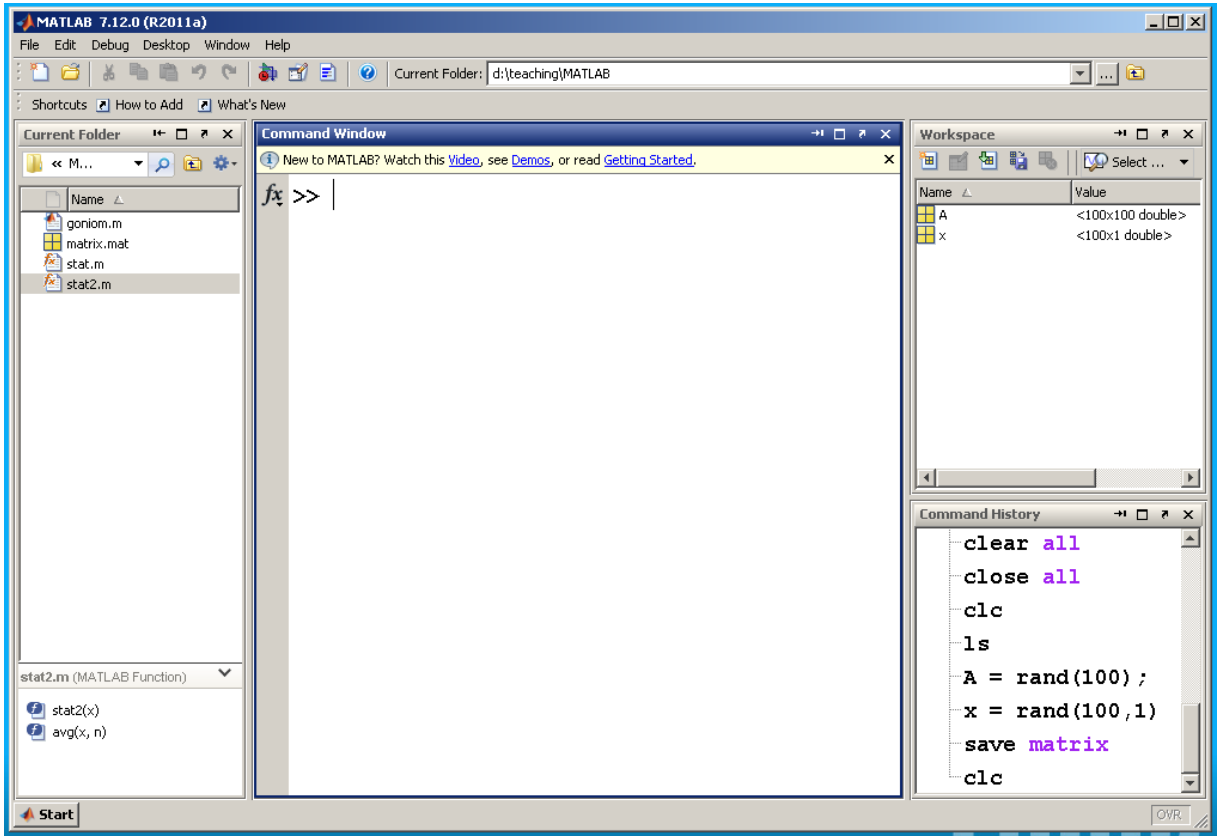




Von Mieses stress [MPa]

MatSol

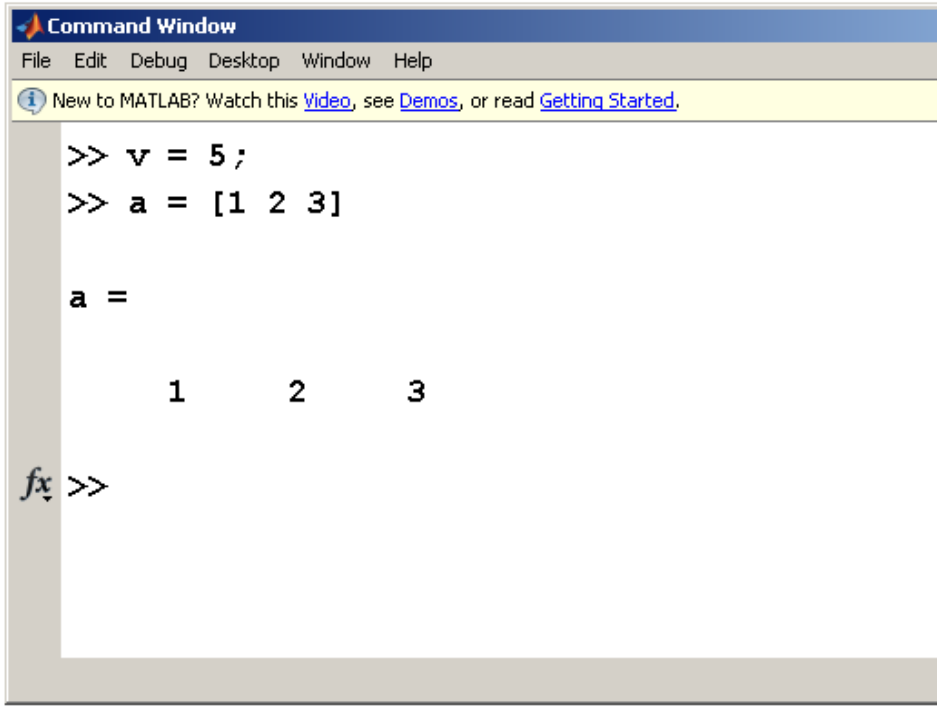
MATLAB Environment



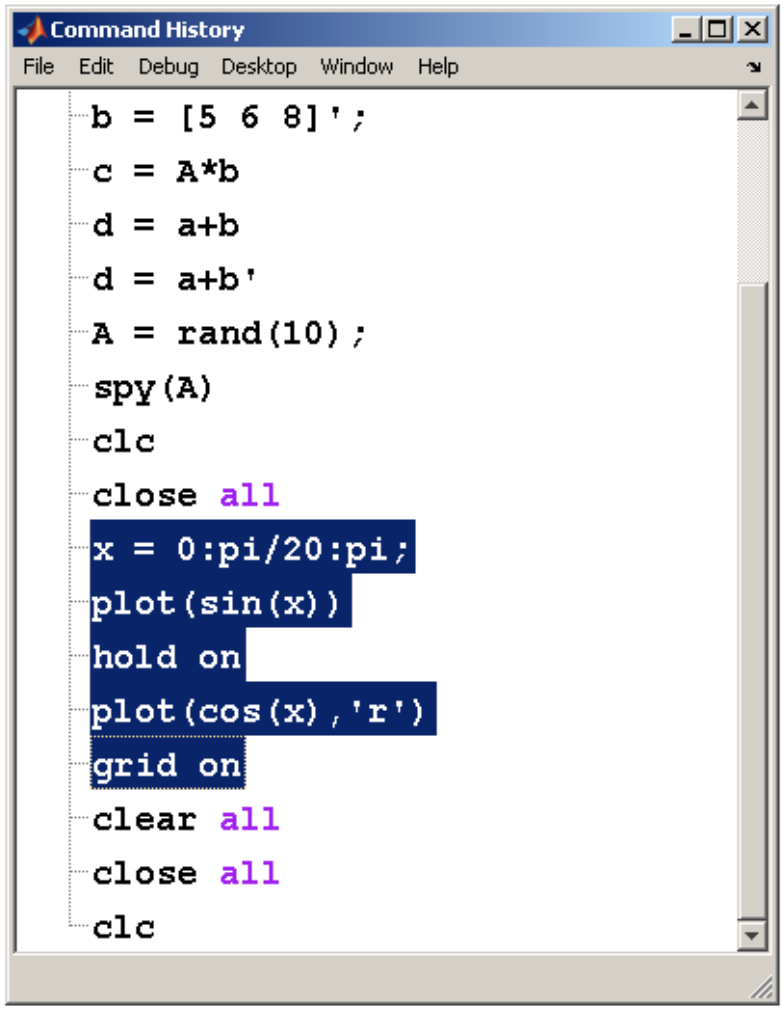
Command Window

- Use the Command Window to enter variables and to run MATLAB functions and scripts. MATLAB displays the results.

- Press the up arrow key ↑ to recall a statement you previously typed. Edit the statement as needed, and then press **Enter** to run it.



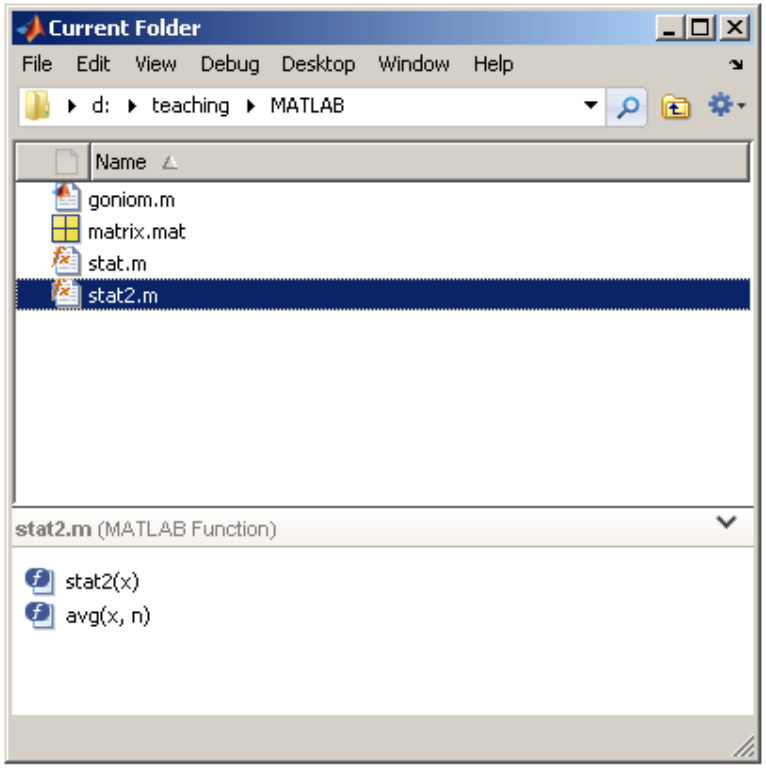
Command History



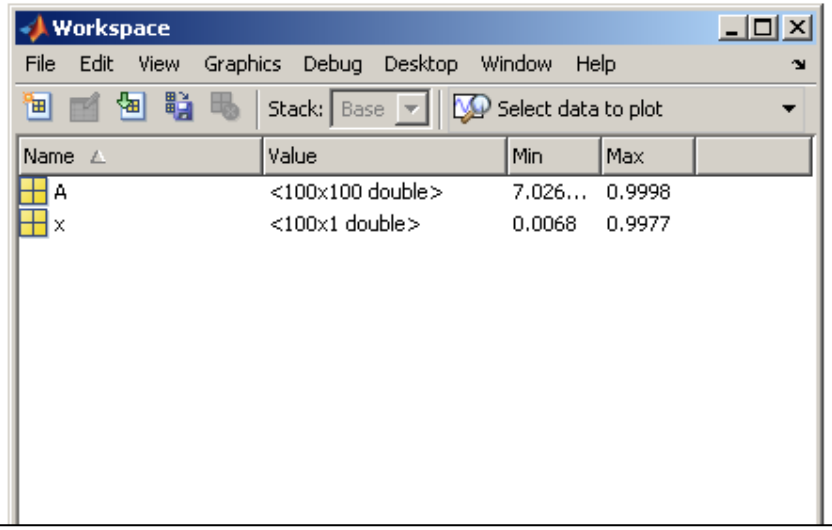
- Statements you enter in the Command Window are **logged** in the Command History.
- You can **view and search** for previously run statements, as well as copy and execute selected statements
- You can also **create a file** from selected statements.

Current Folder

- MATLAB limits where it looks for files so it can locate them more quickly.
- The file must be in one of these locations:
  - MATLAB current folder
  - A folder that is on the MATLAB search path
- The Current Folder browser is a tool for managing files.



Workspace

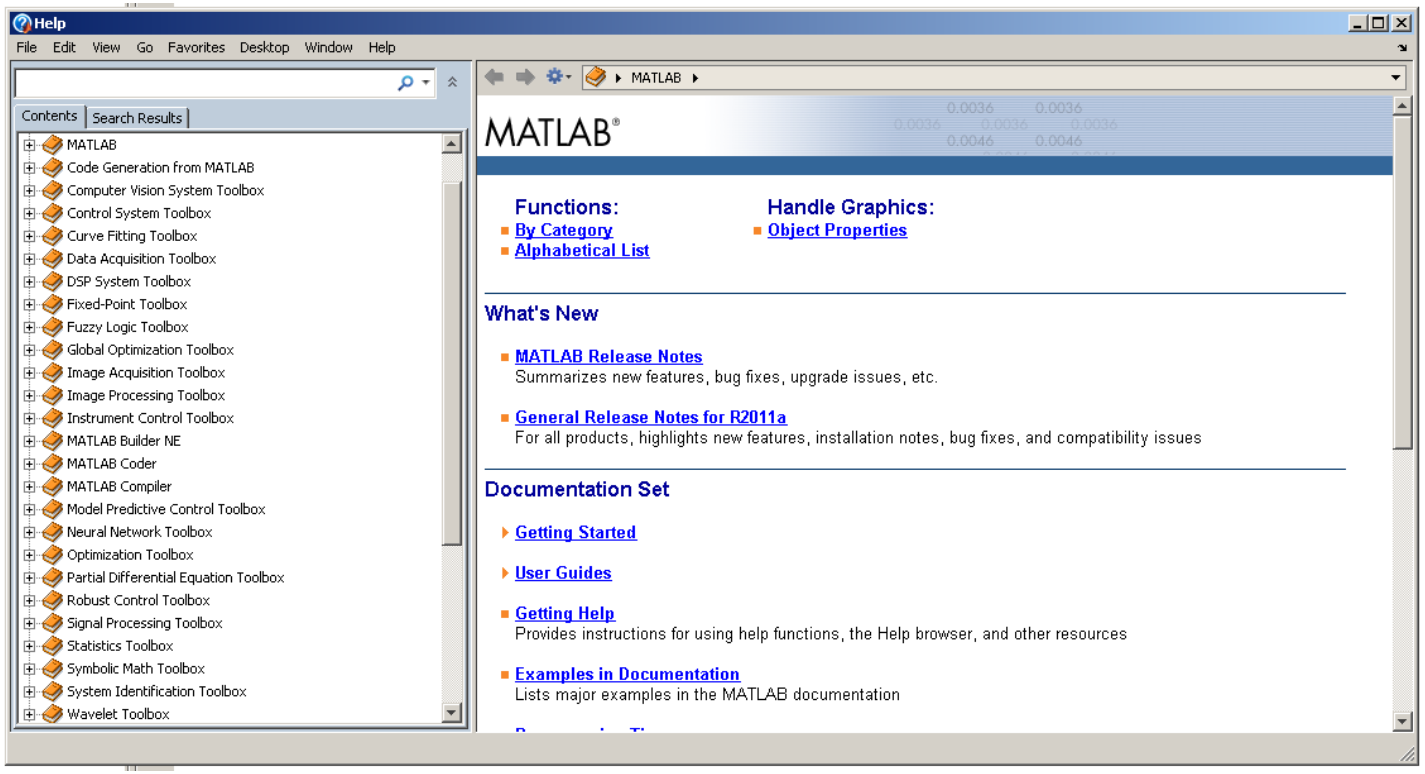


- The Workspace consists of the set of variables stored in memory.
- You add variables to the workspace by using functions, running function and script files, and loading saved workspaces.

```
>> who
Your variables are:
A  x

>> whos
  Name          Size                Bytes  Class     Attributes
  A           100x100               80000  double
  x           100x1                   800  double
```

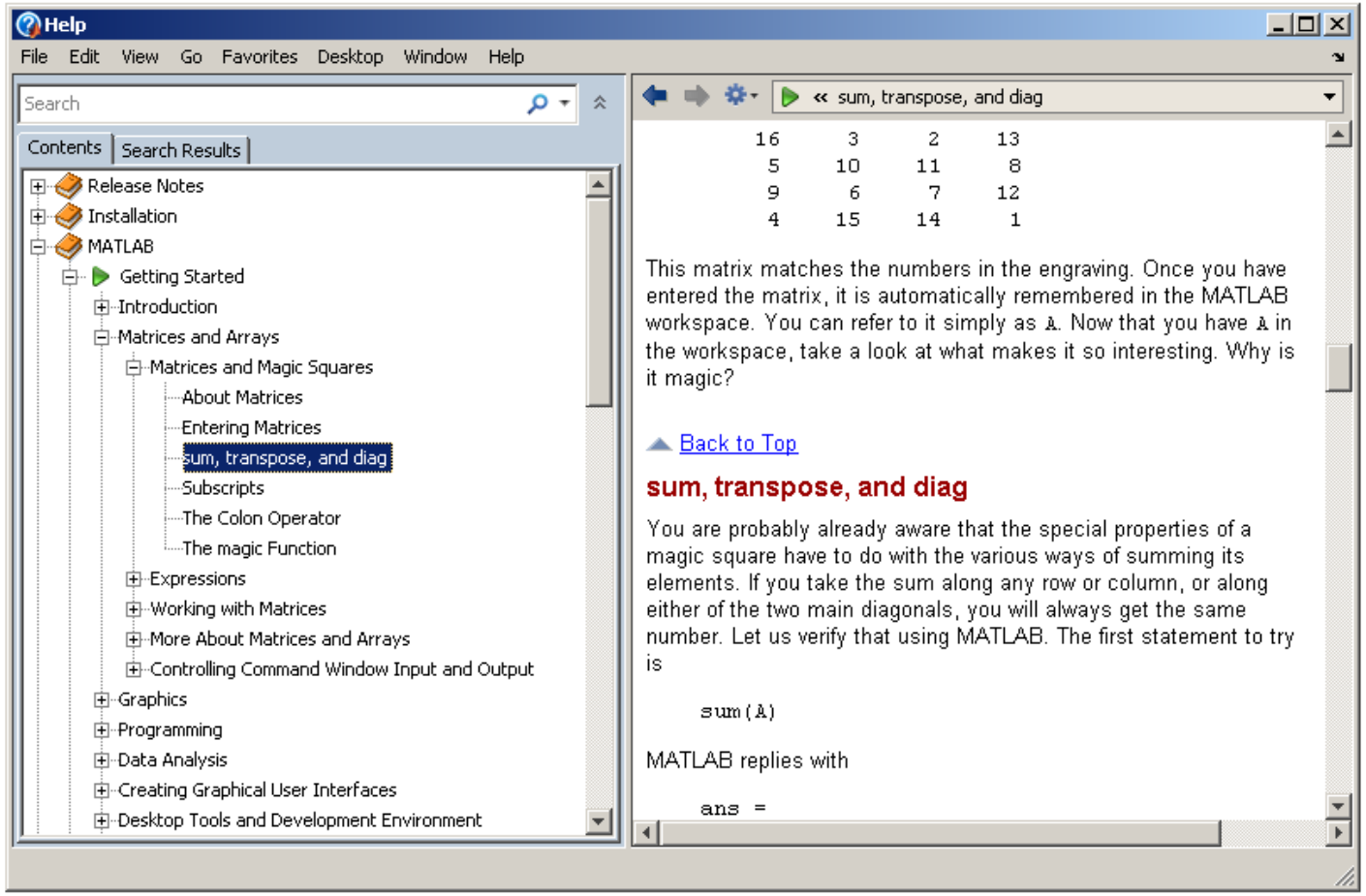
Help and Documentation

- There are different ways to get help, depending on your needs.



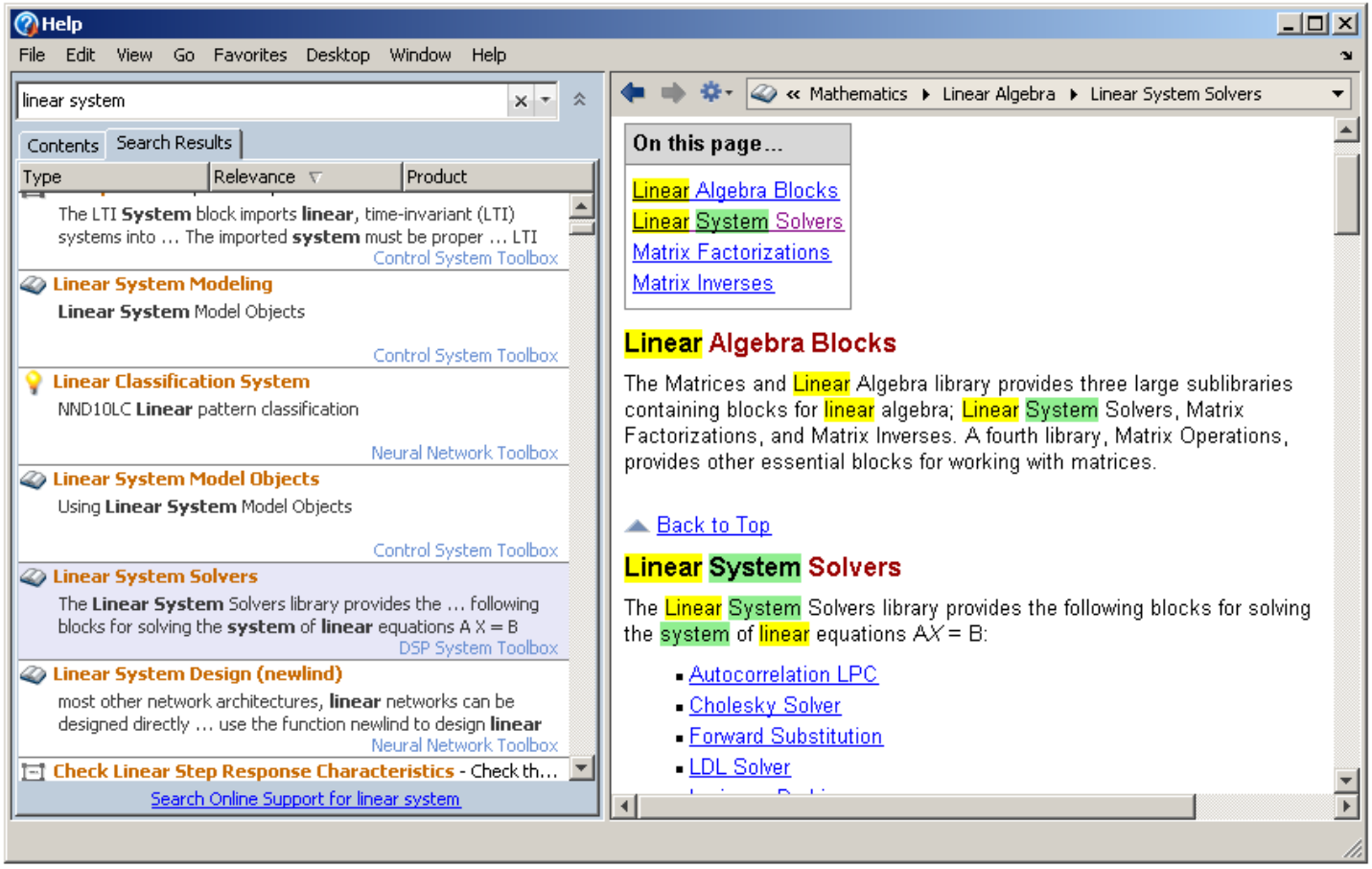
Help: Contents

- Look for getting started guides, code examples, demos, and more.



Help: Search Results

- In the Help browser **Search** field, enter the words you want to look for.



MATLAB Help

- From command window

```
>> help spones
 SPONES Replace nonzero sparse matrix elements with ones.
    R = SPONES(S) generates a matrix with the same sparsity
    structure as S, but with ones in the nonzero positions.

    See also spfun, spalloc, nnz.

    Reference page in Help browser
       doc spones
```

Matrices and vectors

Run in MATLAB Command Window

```
>> echodemo vectors_matrices
```

Useful matrix functions

- **`A'`** – *transpose of matrix A. Also* transpose(A).
- **`det(A)`** – *determinant of A*
- **`eig(A)`** – *eigenvalues and eigenvectors*
- **`inv(A)`** – *inverse of A*
- **`svd(A)`** – *singular value decomposition*
- **`norm(A)`** – *matrix or vector norm*
- **`find(A)`** – *find indices of elements that are nonzero. Can also pass an expression to this function, e.g.* find(A > 1) *finds the indices of elements of A greater than 1.*

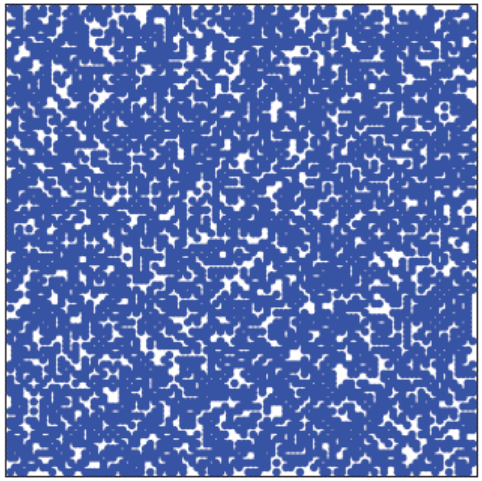
Useful matrices

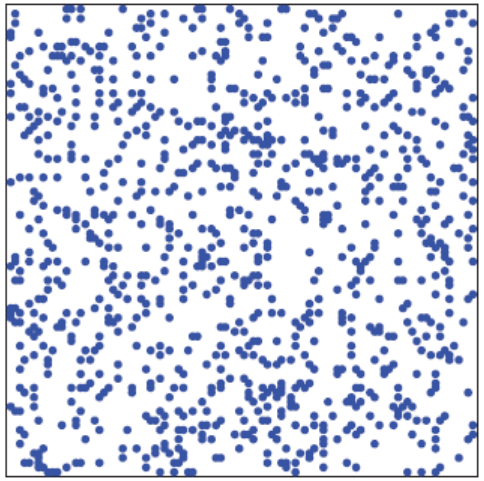
A few other useful matrices are:

- **`zeros`** – *create a matrix of zeros*
- **`ones`** – *create a matrix of ones*
- **`rand`** – *create a matrix of random numbers*
- **`eye`** – *create an identity matrix*

Sparse matrices

Sparse matrix have the large number of zero elements





The sparse attribute allows MATLAB to:

- Store only the nonzero elements of the matrix, together with their indices.
- Reduce computation time by eliminating operations on zero elements.

Sparse matrices

Example

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \\ 0 & 4 & 5 & 0 \\ 0 & 0 & 6 & 7 \end{pmatrix}$$

```
i = [1  2  3  4  1  3]'; %indices of rows
j = [1  2  2  3  4  4];  %indices of columns
v = [1  3  4  6  2  7]'; %values
```

```
A=[1 0 2 0 0; 0 1 0 0 1; 2 0 0 2 0; 3 1 2 0 0] %saved as full
B = sparse(A)          % B saved as sparse
C = full(B)            % C saved as full
```

Sparse matrices

```
n=5; e=ones(n,1);   %vector of ones
A = spdiags([-e 2*e -e], -1:1, n, n); %sparse matrix nxn with
% 2's on diagonal and -1 on subdiagonale and superdiagonale

[I,J,V]=find(S); %returns a vector V containing the values
%  that correspond to the row and column indices I and J.

I=[1 1 2 3]; J=[1 3 2 4]; V=[1 1.5 2 3.7]; m=5; n=6;
S=sparse(I,J,V,m,n); %generate mxn sparse matrix from I,J,V

spy(S)          %plots the sparsity pattern of the matrix S.
speye(5,4);     %sparse identity
nnz(S)          %number of nonzero elements

sprand, sprandn, sprandsym
```

Strings

```
str = 'Dr. John Doe';  %create string

%join 2 strings
str1 = strcat(str,', ','1970')    %ignore spaces
str2 = [str,', ','1970']    %do not ignore the  spaces

T=1323.56;
sprintf(,Temperature T=%10.4fK', T) %format data to string
ans =
Teperature T= 1323.5600K

strcmp('hello','Hello') %compare 2 strings
ans =
     0
```

Scripts

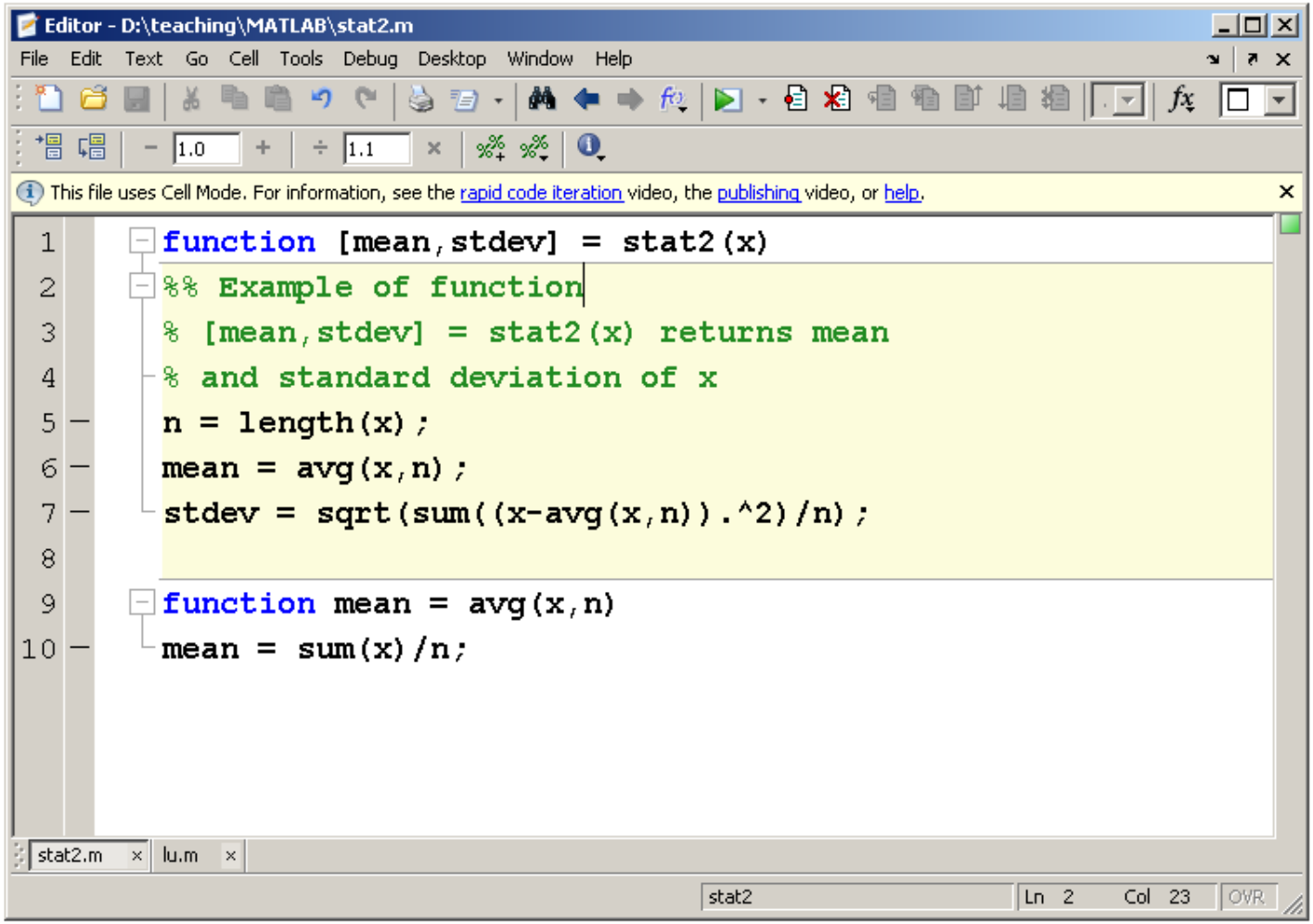
- external files, have a filename extension of .m
- **the simplest MATLAB programs**, a sequence of statements and comments
- useful for automating blocks of MATLAB commands, such as computations you have to perform repeatedly from the command line
- **operate on existing data in the workspace**
- **do not return output arguments** – any variables that they create remain in the workspace

goniom.m

```
% Script example:
% Evaluating goniometric
% functions in pi/2
x=pi/2;
s=sin(x); c=cos(x);
t=tan(x); co=cot(x);
disp([s,c,t,co]); %show results
```

```
>> goniom
  1.0e+016 *

 0.0000   0.0000   1.6331   0.0000
```

Editor


Editor - D:\teaching\MATLAB\stat2.m
File Edit Text Go Cell Tools Debug Desktop Window Help
This file uses Cell Mode. For information, see the rapid code iteration video, the publishing video, or help.
1 function [mean,stdev] = stat2(x)
2 %% Example of function
3 % [mean,stdev] = stat2(x) returns mean
4 % and standard deviation of x
5 n = length(x);
6 mean = avg(x,n);
7 stdev = sqrt(sum((x-avg(x,n)).^2)/n);
8
9 function mean = avg(x,n)
10 mean = sum(x)/n;
stat2.m
lu.m
stat2
Ln 2 Col 23
OVR

Functions

- external files, have a filename extension of .m
- First line: function declaration with **input and output** arguments

```
function [out1, out2, ...] = myfun(in1, in2, ...)
```

- The variables within the body of the function are **all local** variables.

- **Anonymous Functions**
- **Primary and Subfunctions**
- **Nested Functions**

stat.m

```
function [mean,stdev] = stat(x)
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2/n));
```

```
>> [mean stdev] = stat([52/4 5.2 7.9])
mean =
    8.7000
stdev =
    3.2342
```

Functions

Anonymous Functions

- a simple form of the MATLAB function that is defined within a single statement.
- You can define an anonymous function right at the command line, or within a function or script.

```
>> sqr = @(x) x.^2;
>> sqr(7)

ans =

    49
```

```
>> f = @(x) 5*x^2 + 3*x + 5;
>> f(0)

ans =

     5
```

Functions

Primary and Subfunctions

- Any function (except anonymous) must be defined within a file.
- Each such function file contains a required *primary function* that appears first, and any number of *subfunctions* that may follow the primary.
- Primary functions can be called from outside of the file that defines them, while subfunctions cannot. Subfunctions are visible only to the primary function and other subfunctions within their own file.

stat2.m

```
function [mean,stdev] = stat2(x)
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);

function mean = avg(x,n)
mean = sum(x)/n;
```

```
>>[mean stdev]=stat2([1 2 5])

mean =
    2.6667

stdev =
    1.6997
```

Functions

Nested Functions

- You can define functions within the body of another function. These are said to be *nested* within the outer function.
- A nested function has **access to the workspaces of all functions inside of which it is nested**. A variable that has a value assigned to it by the primary function can be read or overwritten by a function nested at any level within the primary.

```
function x = A(p1, p2)
...
   function y = B(p3)
   ...
   end
...
end
```

Flow Control:
Conditional Control: `if-else-elseif`

```
% Generate a random number
a = randi(100, 1);
% If it is even, divide by 2
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end


a = randi(100, 1);
if a < 30
    disp('small')
elseif a < 80
    disp('medium')
else
    disp('large')
end
```

Conditional Control: `switch`

```
mynumber = input('Enter a number:');

switch mynumber
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
        disp('other value');
end
```

Loop Control: `for`

- The **`for`** loop repeats a group of statements a fixed, predetermined number of times. A matching end delineates the statements.

```
for n = 3:32
    r(n) = rank(magic(n));
end
r
```

It is a good idea to indent the loops for readability, especially when they are nested:

```
for i = 1:m
    for j = 1:n
        H(i,j) = 1/(i+j);
    end
end
```

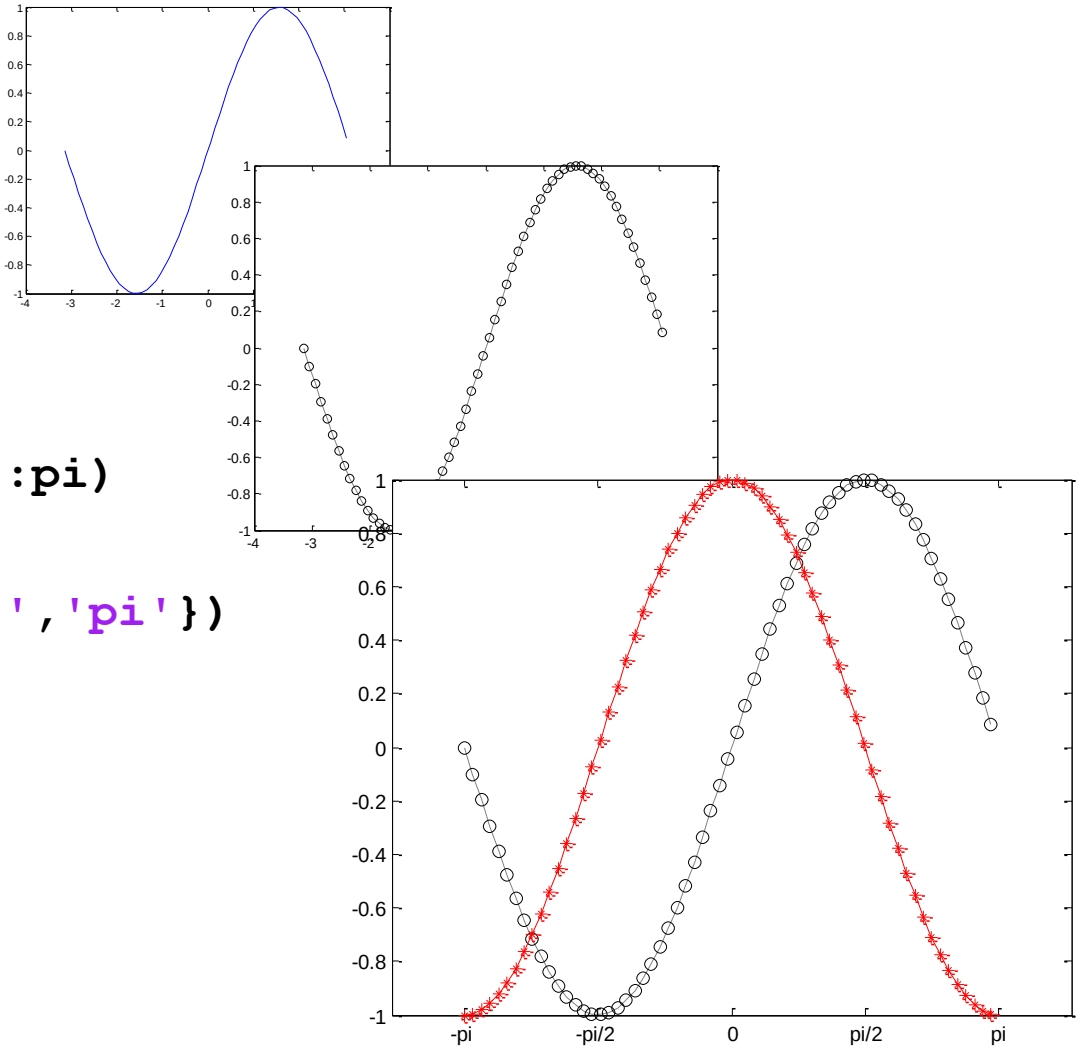
Loop Control: `while`

- The **`while`** loop repeats a group of statements an indefinite number of times under control of a logical condition. A matching end delineates the statements.

```
n = 5;
fact = 1;
while (n>1)
    fact = fact*n;
    n = n-1;
end
```

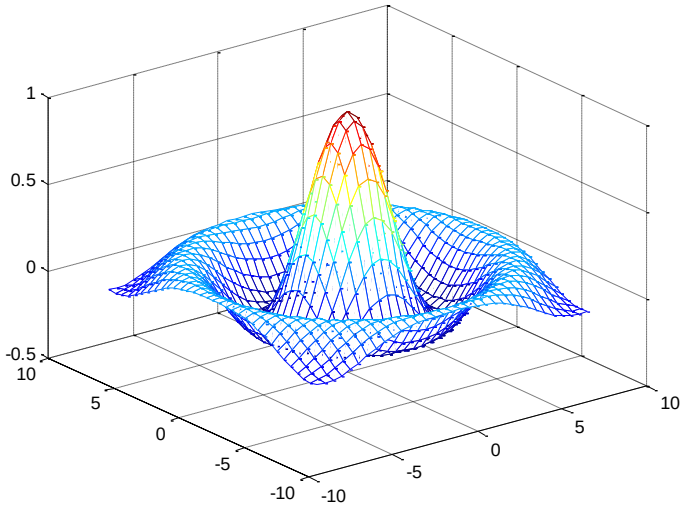
2D graphics: `plot`

```
x = -pi:.1:pi;
y = sin(x);
plot(x,y)

plot(x,y,'ko:')

set(gca,'XTick',-pi:pi/2:pi)
set(gca,'XTickLabel',…
{'-pi','-pi/2','0','pi/2','pi'})

hold on
z = cos(x);
plot(x,z,'r-*')
```

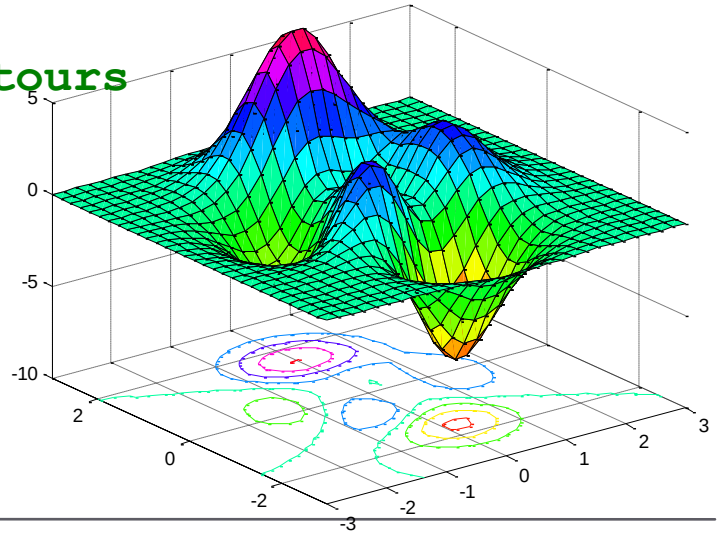


3D graphics: mesh, surf

```
figure
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
```



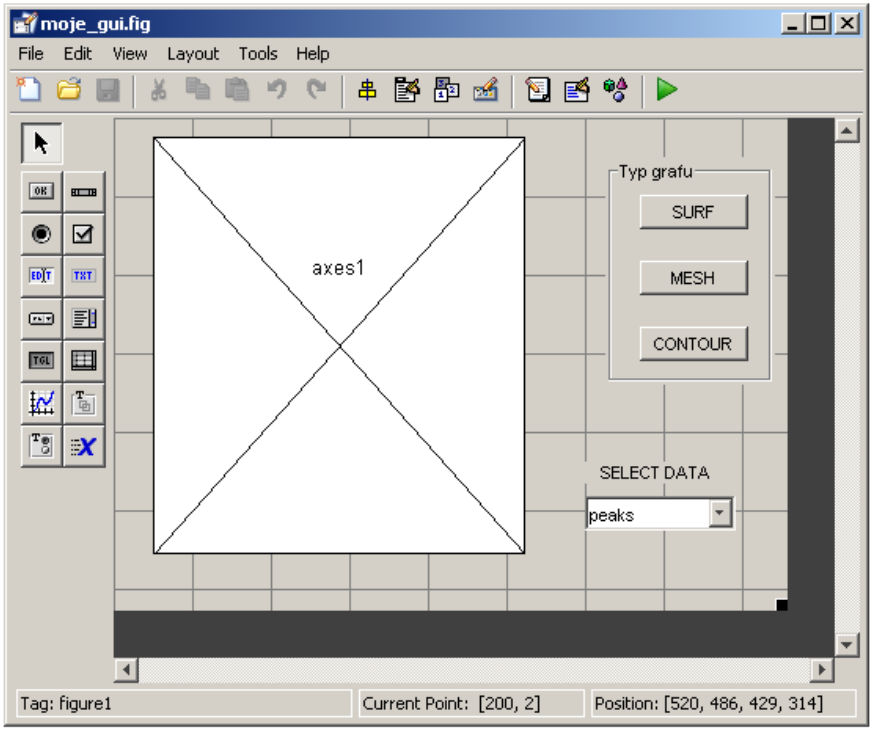

```
figure
[X,Y,Z] = peaks(30);
surf(X,Y,Z)     %surfc(X,Y,Z) with contours
colormap hsv
axis([-3 3 -3 3 -10 5])
```



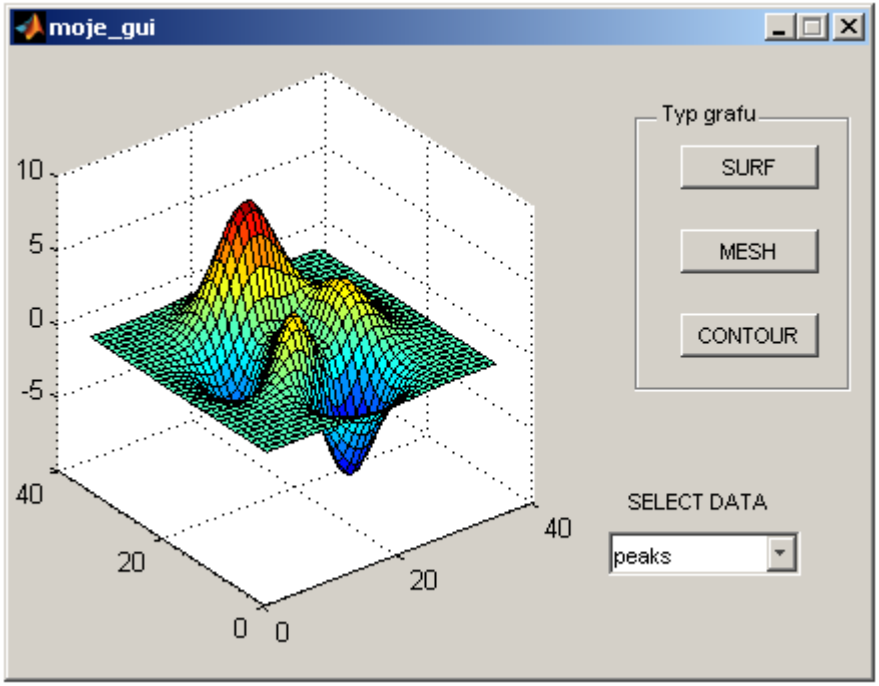
Creating GUI with GUIDE

- GUIDE, the MATLAB Graphical User Interface Development Environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools greatly simplify the process of laying out and programming GUIs

- When you open a GUI in GUIDE, it is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. The following figure shows the Layout Editor with a blank GUI template.



Creating GUI with GUIDE




```
>> moje_gui
```

- video file (11 min): http://www.mathworks.com/support/2011a/matlab/7.12/demos/CreatingaGUIwithGUIDE_viewlet_swf.html

References

- **Matlab: Instructions to download:**
  http://homel.vsb.cz/~dom033/predmety/NMM/matlab_download
- **Introduction to MATLAB:**
  http://web.gps.caltech.edu/classes/ge11d/doc/matlab_Resource_Seminar.pdf
- David Houcque, **INTRODUCTION TO MATLAB FOR ENGINEERING STUDENTS:**
  http://www.mccormick.northwestern.edu/docs/efirst/matlab.pdf
- **Getting Started Guide:**
  http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf