

Short Introduction into MATLAB

Christian Schulz
christian.schulz@cma.uio.no

CMA/IFI - UiO

5 September 2006



Centre of
Mathematics for
Applications



**UNIVERSITY
OF OSLO**

1. Basics

- Startup
- Variables
- Standard Operations

2. Next Steps

- Plots
- Built in Functions
- Write/Plot to File

3. Programming

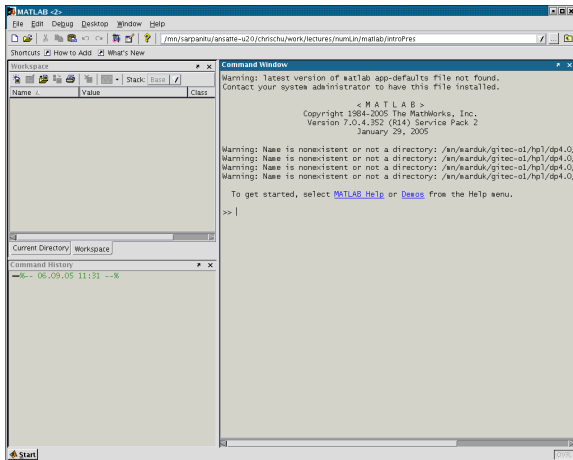
- Scripts and Functions
- Flow Control
- Built in Functionality vs. Programming

4. Now What?

5. Comments/Conventions

Starting MATLAB

- MATLAB is installed at computers of Dep. of Informatics as well as Dep. of Mathematics
- Just type in
`> matlab &`



Definition of Variables

- Numbers:


```
>> alpha = 5
alpha =
      5
```
- Row-Vectors/Arrays:


```
>> b = [1 2 3] % or b = [1,2,3]
b =
      1      2      3
```
- Matrices:


```
>> A = [1 1 1; 2 2.5 3; 4 5 6]
A =
      1.0000      1.0000      1.0000
      2.0000      2.5000      3.0000
      4.0000      5.0000      6.0000
```
- Matrices are basic data elements in MATLAB (MATrix LABoratory)

Variables cont.

- Specifying variable with “;” at end omits output:

```
>> C = [1 3 5; 2 4 6; -1 -2 -3];
```

```
>>
```

- Command “whos” prints information about all defined variables:

```
>> whos
```

Name	Size	Bytes	Class
A	3x3	72	double array
C	3x3	72	double array
alpha	1x1	8	double array
b	1x3	24	double array

Grand total is 22 elements using 176 bytes

Variables in Workspace

The screenshot shows the MATLAB environment with the following components:

- Workspace:** A table listing variables in the workspace.

Name	Value	Class
A	[1 1 2 2.5 3 4 5 6]	double
C	[1 3 5; 2 4 6; -1 -2 -3]	double
alpha	5	double
b	[1 2 3]	double
- Command Window:** Displays the commands entered and their results.


```

      To get started, select MATLAB Help or Demos from the Help menu.

      >> alpha = 5
      alpha =
           5

      >> b = [1 2 3]
      b =
           1     2     3

      >> b = [1,2,3]
      b =
           1     2     3

      >> A = [1 1 1; 2 2.5 3; 4 5 6]
      A =
           1.0000    1.0000    1.0000
           2.0000    2.5000    3.0000
           4.0000    5.0000    6.0000

      >> C = [1 3 5; 2 4 6; -1 -2 -3];
      >> whos
      Name           Size           Bytes  Class
      A               3x3              72  double array
      C               3x3              72  double array
      alpha           1x1               8  double array
      b               1x3              24  double array

      Grand total is 22 elements using 176 bytes

      >>
      
```
- Command History:** Shows a list of previous commands for review.


```

      --%-- 06.09.05 11:31 --%
      alpha = 5
      b = [1 2 3]
      b = [1,2,3]
      A = [1 1 1; 2 2.5 3; 4 5 6]
      C = [1 3 5; 2 4 6; -1 -2 -3];
      whos
      
```

Arithmetic Operations

- Standard Matrix Operations are done by just typing them in:

```
>> ApC = A+C; AtC = A*C; AexpAlpha = A^alpha;  
>> d = A*(b');
```

- Conjugate complex transpose is given by “A' ”
- If you need just transpose, use “A.' ”
- Element wise operations are denoted by an additional “.”:

```
>> At_elemC = A.*C; Aelem_expAlpha = A.^alpha;
```

- Solving linear systems in form of “ $A*x = d$ ” can be solved using left division:

```
>> x = A\d;
```

- Also possible to solve “ $y*A = b$ ”:

```
>> y = b/A;
```

Accessing Elements/Submatrices

- Indices of matrices/vectors/arrays start with 1

- Element of matrix: `>> A(1,2);`

- Create a sequence:

```
>> 2:.5:4
```

```
ans =
```

```
2.0000
```

```
2.5000
```

```
3.0000
```

```
3.5000
```

```
4.0000
```

- Variable “ans” contains value of last operation with unstored result

- Column i in matrix: `>> A(:,i);`

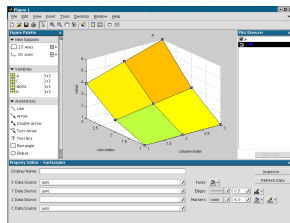
- First two rows: `>> A(1:2,:);`

- Can also use array with indices: `>> A([1 3 2],:);`

Plots

- Command “plot(x,y)” plots vector y against x
- “title('plot-title');” sets title
- Can label axis with “xlabel('x-axis-label');”
- Plot several things into one figure: “hold on;”
- Delete old plot when plotting new: “hold off;”
- “plottools on;” gives you a editing tool for a figure
- More plot commands (e.g. surf1 or polar)

```
>> surf1(A);  
>> title('A');  
>> xlabel('column-index');  
>> ylabel('row-index');  
>> zlabel('value');  
>> plottools on;
```



Built in Functions

- MATLAB has a lot of built in functions
- A function can return more than one value: `>> [m,n] = size(A);`
- Many functions normally applied to scalar values (e.g. `sin`, `cos`) can also be applied to matrices (are applied element wise)

To get more information about what functions are built in and syntax/usage of specific functions use MATLAB built in help or/and documentation

- Help: `>> help command`
- Start Documentation: `>> doc`
- Documentation for a command: `>> doc command`
- `">> lookfor text"` searches text in (first line of) help of functions

File Input/Output

Store/Load Data

- “save filename;” stores all variables in workspace in binary file “filename.mat”
- “save filename x y z;” saves x y z into “filename.mat”
- “load filename;” correspondingly loads variables stored in “filename.mat” into workspace
- “clear A;” removes variable “A” from workspace
- “clear;” removes all variables

Plots

Use “print” to ‘plot’ into a file

- Create figure and store returned handle: `>> figID = figure`
- (“figure(figID2);” makes figID2 current figure)

File Input/Output cont.

Plots cont.

- Do Plots (will be done in current figure)
- Print: `>> print (figId, '-dps', 'figure.ps')`

Formatted Output

- `"fid = fopen(filename, permissions);"` opens file `"filename"` with corresponding permissions
- `"fprintf(fid, format, a,b,...);"` writes formatted output into file associated with `"fid"` (behaves as `fprintf` in C)
- `"fclose(fid);"` closes corresponding file
- Be aware how matrices/vectors/arrays and `fprintf` interact when using it (see `">> help fprintf"`)!

Scripts and Functions

General

- For bigger tasks one wants to reuse code
- MATLAB program-files have extension “.m”
- Two possibilities: script-files and function-files
- Functions/scripts are called/executed by typing in filename (without “.m”)

Script-Files

- Contain a series of MATLAB commands
- Commands are executed as if they were typed in
- All variables are global $\Rightarrow$ interact with workspace
- Problem: must have matching variable-names / name overlapping

Function-Files

Structure

- Begin with declaration of function
`function [out1, out2, ...] = funname(in1, in2, ...)`
 - `in1, in2, ...`: input parameter
 - `out1, out2, ...`: output parameter
- Followed by some rows of comments (comment starts after “%”) which are displayed when entering “>> help funname”
- Body of function follows
- After Body subfunctions may follow

General

- Used variables are local
- Subfunctions are not visible outside $\Rightarrow$ can structure code
- Name file as “funname.m”

Flow Control

Like programming languages MATLAB has branching and loops

Branching

- Mostly branching is done using `if`:

```
if condition1
    program1
elseif condition2
    program2
else
    program3
end
```

- A condition is as normal something like
(`count < limit`) & ((`height - offset`) $\geq$ 0)
- Check operators for possible conditions/concatenations
- Be careful when using matrices in conditions (`A ~= B`)
- MATLAB also supports `switch`

Flow Control cont.

Loops

- For-loop: Executes block of code specified number of times

```
for variable = expression  
    statements  
end
```
- While-loop: Repeatedly executes statements while condition is true

```
while condition  
    statements  
end
```
- Example:

```
for i = [1,2,4,5]; A(i) = i; end
```
- More Flow Control for loops like `break`, `continue`, etc

Built in Functionality vs. Programming

- Built in functions are optimized
- Use built in functions / matrix operations whenever possible (instead of loops,...)

Example

```
function [R,success] = makeTriangular(A)

% uses row operations to make A upper triangular
% use:  makeTriangular(A)
% in:   A n x n Matrix
% out:  R n x n upper triangular Matrix (if successful)
%       success 0 (false) if failed, 1 (true) if successful

success = 0;
[m, n] = size(A);
if (m ~= n)
    return; % returns from function
end
```

Built in Functionality vs. Programming

Example cont. - Loops

```
for k = 1:n-1 % row operations
    for ii_row = k+1:n
        factor = A(ii_row,k) / A(k,k);
        A(ii_row,k) = 0;
        for ii_column = k+1:n
            A(ii_row,ii_column) = A(ii_row,ii_column) - ...
                                   factor * A(k,ii_column);
        end
    end
end
success = 1;
R = A;
```

Built in Functionality vs. Programming

Example - Matrix Operations

- Replace for-loops by matrix operations

```
for k = 1:n-1 % row operations
    factor(k+1:n,k) = A(k+1:n,k)/A(k,k);
    A(k+1:n,k) = 0;
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - ...
                        factor(k+1:n,k)*A(k,k+1:n);
end
```

But (Of Course)

- If an exercise says you should implement an algorithm doing xyz, you should not just call the corresponding MATLAB function xyz

How To Continue

- Make intense use of MATLAB `help` and `doc` commands
- You can use the MATLAB editor for programming, just type
`>> edit`
- For further information about MATLAB google for it
- Some tutorials I have found:
 - <http://www.cyclismo.org/tutorial/matlab/>
 - <http://web.ew.usna.edu/mecheng/DESIGN/CAD/MATLAB/usna.html>
 - <http://www.math.ufl.edu/help/matlab-tutorial/>
 - <http://www.nd.edu/hlin1/ee550/page4.html>

Comments, Conventions

- Structure your code, reuse code
- **Comment your code!**
 - Describe what, how, why; do not repeat code
 - When defining a function, write the comments for that function, including description of the input, output parameter and what the function does
- Use indentation (MATLAB editor does this)
- Use appropriate naming, e.g `relativeSize` or `relSize` and `absSize` instead of `size1`, `size2` or even `s1`, `s2`
- Use names starting with capital letter for matrices, with small letter for vectors, scalars
- Do not redefine MATLAB commands or predefined variables

Compulsory Exercises

- One report per student
- Report written in English
- Write in TeX or LaTeX, not Word
- All source files have to be handed in
- **Use names given in assignment** for functions, variables, files.
Do not change upper/lower case. (Scripts are used to test your code)
- Function names and corresponding .m filenames must be identical
- Send packed solutions to me

Thank you for your attention.

Have fun with the exercises.