What is MATLAB and how to start it up?

- MATLAB LABoratory
- Object-oriented high-level interactive software package for scientific and engineering numerical computations
- Enables easy manipulation of matrix and other computations
- Can handle symbolic computation, but is not specifically tailored to do that.

To start up MATLAB on CDF (teach.cs) type

%m atlab -nodesktop or %m atlab &

at the Unix shell prompt.

The matlab -nodesktop command starts up MATLAB in your current window (most likely an xterm), and can work on any ASCII terminal; it gives you a simple and light ASCII interface, still including plots, assuming you are running Xwindows. The matlab command starts up MATLAB in a new window of the console you are on, assuming you are running Xwindows; it gives you a more powerful, but also complex and heavy interface, the desktop.

The standard MATLAB prompt is >> (while in Maple it is a single > ).

In the following, whatever follows >> on a line starting with >> is typed by the user at the MATLAB prompt.

To quit MATLAB, type

>> quit or >> exit

An example of MATLAB code

\[
A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}; \quad x = \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix}; \quad b = A \cdot x \quad \text{(2x1 vector)}
\]

\[
b = \begin{bmatrix} -7 \\ -16 \end{bmatrix}
\]

• The assignment operator is = (In Maple, it is :=).
• The semicolon (;) causes suppression of echoing the result of assignment/computations. (In Maple, the semicolon causes echoing the result and the colon (:) causes suppression of echoing). (See exception)
• The newline (<return> or <enter>) denotes the end of a statement. (The comma can also be used to denote the end of a statement.) If no semicolon precedes the newline, the result of the statement will be echoed. (In Maple, a statement ends with colon or semicolon. The newline does not denote anything specific.) (See exception)

Using MATLAB interactively

Recall that whatever follows >> on a line starting with >> is typed by the user at the MATLAB prompt. The rest is output echoed by MATLAB.

\[
A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix};
\]

\[
x = \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix};
\]

\[
b = A \cdot x
\]

\[
b = \begin{bmatrix} -7 \\ -16 \end{bmatrix}
\]

\[
A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}
\]

\[
x = \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix}
\]
MATLAB tutorial 5 © C. Christara, 2001-2021

• Exponentiation is carried by the hat symbol ^ (In Maple, it is carried by the hat or the double star ** symbol.)

• MATLAB can be used as a simple calculator, i.e. it is not needed to assign the result of a computation to a variable. A default variable, named ans, is used to save the result of the latest computation or expression. (In Maple, the above is also true, but the name of the default variable is percent % in Maple 7 and doublequote " in Maple V.)

• MATLAB has all the standard mathematical functions, e.g. exp, log (stands for ln), log10, log2, sin, cos, tan, asin, acos, atan, sqrt, sinh, cosh, abs, max, min, etc, and many more built-in functions that compute a variety of mathematical results. (In Maple, the above is also true, but the names of some functions are different, e.g. ln (same as log), arcsin, arccos, etc.)

• Complex numbers are handled just as easily as real ones. Complex numbers are output in the form a + bi. The variables i and j are by default set to the value $\sqrt{-1}$. If, though, the user overwrites them, the value $\sqrt{-1}$ is lost. It can be recovered through i = sqrt(-1) and j = sqrt(-1).

• The variable pi is by default set to the value of the number $\pi$. If, though, the user overwrites it, then the value of $\pi$ is lost. It can be recovered through pi = 4*atan(1). (In Maple, pi is Pi.)

• To clear the variable name a use clear a, while to clear all variable names use clear. (In Maple, use unassign.)

MATLAB tutorial 6 © C. Christara, 2001-2021

• To clear the variable name a use clear a, while to clear all variable names use clear. (In Maple, use unassign.)

MATLAB tutorial 7 © C. Christara, 2001-2021

• To clear the variable name a use clear a, while to clear all variable names use clear. (In Maple, use unassign.)

MATLAB tutorial 8 © C. Christara, 2001-2021

• To clear the variable name a use clear a, while to clear all variable names use clear. (In Maple, use unassign.)
In MATLAB, it is easy to extract parts of the matrix. In general,

```
>> A(i:j, k:l)
gives the submatrix of A defined by rows i through j and columns k through l of matrix A. If the index is just a colon (:), without numbers that define a range, all rows or all columns of the matrix are included. For example,
```

```
>> A(:, k:l)
gives the submatrix of A defined by columns k through l of all rows of matrix A.
```

In general, the colon can be used to generate a sequence of numbers forming a vector (or one-dimensional matrix). Thus,

```
>> m:k:n
gives a list of numbers, starting from m and ending to n with stepsize k and
```

```
>> m:n
gives a list of numbers, starting from m and ending to n with stepsize 1.
```

Matrix concatenation can also be easily done. If A and B are two matrices of the same number of rows,

```
>> C = [A B]
creates a matrix C, with the same number of rows as A and B, and number of columns the sum of the numbers of columns of A and B. The columns of C consist of the columns of A (to the left) followed by the columns of B (to the right). On the other hand, if A and B are two matrices of the same number of columns,
```

```
>> C = [A; B]
creates a matrix C, with the same number of columns as A and B, and number of rows the sum of the numbers of rows of A and B. The rows of C consist of the rows of A (top) followed by the rows of B (bottom).
```

Examples:

```
>> A = [
    1  2  3
    4  5  6
    7  8  9
    0  0 10];
>> B = A(1:2, 2:3)
B =
    2  3
    5  6
>> C = [-1; -4]
C =
    -1
    -4
>> D = [B C]
D =
    2  3  -1
    5  6  -4
```

MATLAB has many built-in functions and operators for matrices. For example,

```
>> iA = inv(A);
computes the inverse of matrix A (if A is already defined as a square matrix and if the inverse exists), and saves it in variable iA. Also,
```

```
>> A'
displays the transpose of A, while
```

```
>> I3 = eye(3, 3);
saves the 3 × 3 identity matrix in variable I3.
```

MATLAB takes a high-level approach when asked to solve linear systems. If A is a solvable n × n matrix and b a n × 1 vector,

```
>> x = A\b;
will compute the solution x to the system Ax = b.
```
For loops

The most typical syntax of the for loop is of the form
for i = j:k:l
    statements
end
where i, the loop counter, starts from j, and proceeds to l, with stepsize k. If k is missing, i.e. the for statement looks like for i = j:l, the stepsize takes the default value 1. Note that none of the i, j, k and l variables needs to be integer.

Examples: What values does a take in the following loops?
for a = 0.1:0.1:0.5, statements, end
0.1, 0.2, 0.3, 0.4, 0.5
for a = 0.1:0.2:0.6, statements, end
0.1, 0.3, 0.5
for a = 0:-2:-10, statements, end
0, -2, -4, -6, -8, -10
for a = 0:-2:10, statements, end
None! To be precise, it takes the value of the empty matrix. Thus the statements of the loop (whatever they were), do not get executed at all.

If (conditional) statement, relational and logical operators

The general form of the if statement is
if expression
    statements
elseif expression
    statements
else
    statements
end
The statements following each elseif are executed, if the expression of the respective branch is true and none of the previous branches resulted in execution. The else branch is executed if none of the previous branches resulted in execution. The elseif and else parts are optional. Zero or more elseif parts can be used as well as nested if’s.
A simple form of the expression is \texttt{expr relop expr} where relop (relational operator) is \texttt{==, <, >, <=, >=, or \textasciitilde=}.
(In Maple, the respective relops are \texttt{=, <, >, <=, >=, or <\textasciitilde>}.)

Example:

```matlab
if I == J
    A(I, J) = 2;
elseif abs(I-J) == 1
    A(I, J) = -1;
else
    A(I, J) = 0;
end
```

What would the result of the above statement be, if it is inserted in a two-level nested for loop with both counters I and J starting at 1 and proceeding to 7 with stepsizes 1?

The standard logical operators are \&, |, and \textasciitilde. 
(In Maple, they are \texttt{and, or, and not}, respectively.)

With the help of logical operators, we can build more complex expressions for if statements.

Example:

```matlab
if (I <= J) \& (J-3 < I)
    A(I, J) = 7-J+I;
end
```

What would the result of the above statement be, if it is inserted in a two-level nested for loop with both counters I and J starting at 1 and proceeding to 7 with stepsizes 1?

Consider also the conditional statements

```matlab
if (I <= J) \& (J-3 < I)  
    A(I, J) = 7-J+I;
elseif (J-4 < I)  
    A(I, J) = -1;
else
    A(I, J) = 0;
end
```

What would the result of the above statement be, if it is inserted in a two-level nested for loop with both counters I and J starting at 1 and proceeding to 7 with stepsizes 1?
Functions and scripts in MATLAB

Functions in MATLAB can be either built-in or user-defined. Functions are saved in M-files, called so because the filename ends in .m. Functions may have one or more input parameters and may return one or more output variables. Currently, only one function (and its subfunctions, if any) can be defined per M-file.

A M-file does not always need to be a function. It can also be a (possibly long) sequence of MATLAB statements, i.e. a script. Scripts have neither formal input parameters, nor formal output variables.

- To execute a function M-file, type (after the MATLAB prompt) the name of the file without the .m extension, followed by the input arguments in parentheses.
- Should you need to save the output variables, precede the call by the output variables (in square brackets if more than one) and the '=' sign.
- To execute a script M-file, just type (after the MATLAB prompt) the name of the file without the .m extension.

Examples:

Assume the following is in file trochoid1.m in the current directory

```matlab
function [ix, iy] = trochoid1(a, r, U, n)
du = U/n;
x = 0; % x = a*u0 - r*sin(u0);
y = r; % y = r*cos(u0);
u = 0; % u = u0;
ix(1) = round(x);
iy(1) = round(y);
for i = 2:n+1
    u = u + du;
    x = a*u - r*sin(u);
    y = r*cos(u);
    ix(i) = round(x);
    iy(i) = round(y);
end
```

Assume also the following is in file script.m in the current directory

```matlab
a = 25; r = 100;
U = 6*pi; n = 200;
[ixl, iyl] = trochoid1(a, r, U, n);
plot(ixl, iyl, '-'); xlabel('x'); ylabel('y');
print -depsc trochoid.eps % print trochoid.ps
```

Assume now we start MATLAB, and type

```matlab
>> script
```

What will happen?
- Variables a, r, U and n are set to the respective values.
- pi gets the default value of the number π.
- The trochoid1 function is called with input arguments a, r, U, n, and output arguments ixl, iyl.
- ixl, iyl get their value through the call to the function trochoid1.
- a, r, U and n keep the value they had before the call to trochoid1.
- All other variables used in the function trochoid1 are local by default (i.e. they do not keep their value outside the scope of trochoid1).

On-line help

Type

```matlab
>> help
```
to get all the help topics and

```matlab
>> help topic
```
or

```matlab
>> doc topic
```
to get information on any specific MATLAB topic. For example,

```matlab
>> help inv
```
will tell you how to use the inv built-in function that gives the inverse of a matrix. To get on-line help through a html winodw environment, type

```matlab
>> helpwin
```