Inlichtingenblad, matlab- en simulink handleiding en practicumopgaven IWS

3 Matlab

3.1 Fundamentals

MATLAB. The name MATLAB stands for *matrix laboratory.*

Main principle. MATLAB works with rectangular numerical matrices with possibly complex elements. Special meaning is attached to 1-by 1 matrices, which are scalars and to matrices with only one row or column, which are vectors. Operations and commands in MATLAB are intended to be natural in a matrix sense, not unlike how they might be indicated on paper.

Working mode. MATLAB is an interactive system.

MATLAB **expressions.** MATLAB is an *expression* language. Expressions typed by user are interpreted and evaluated by the MATLAB system. MATLAB statements are of the form: *variable* = *expression* or *expression*. *variable* is just a matrix name. *expression* is a composition of MATLAB function operators, special variables and other variables. If *expression* yields more than one matrix then *variable* should be a list of matrix names, separated by comas, enclosed by square brackets. In the case when *"variable ="* in MATLAB statement is omitted the result of *expression* is assigned to the default variable `ans`.

The created variables are stored in the MATLAB workspace. They can be listed by executing the command `who`.

Start & Quit. To start MATLAB session enter `matlab` (in the DOS or UNIX environment). To quit MATLAB use the command `quit`.

Help. A HELP facility is available, providing online information on most MATLAB topics. For example, `help eig` provides HELP information on the use of the eigenvalue function.

3.2 Matrices

Numbers. The conventional decimal notation is used, with optional decimal point and leading minus sign for numbers. For example 3, 9.68, -99, 1.602 E-20, 0.001, 6.0225 e23
Complex numbers are entered using the special functions `i` and `j`, f.e. $\mathtt{z} = 3 + 4 * \mathtt{i}$ or $\mathtt{z} = 5 * \mathtt{exp}(\mathtt{i} * 0.92)$.

Matrix generation. The basic method of entering matrices is to use an explicit list. The explicit list of elements is separated by blanks or comas, is surrounded by brackets, [and], and uses the semicolon ; to indicate the ends of the rows. For example, the statement $\mathtt{A} = [1, 2, 3; 4, 5, 6; \mathtt{sin}(5), \mathtt{exp}(8), 9^2]$ enters the matrix A which is following:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \sin(7) & \exp(8) & 9^2 \end{pmatrix}$$

The elements of the explicit list can be matrices of the proper dimensions. In this way big matrices can be created from little ones.
For special cases there exists a way of entering matrices which uses the colon. The statement `x=a:b:c` generates the matrix x consisting of numbers from `a` to `c` with the step `b`.

Subscripting. Consider a 10-by-10 dimensional matrix A.

`A(1,2)`	specifies an element of a matrix A from the first row and second column.
`A(1:5,7:10)`	specifies the 5-by-4 submatrix of elements from the first five rows and the last four columns of A.
`A(:,3)`	is the third column.
`A(v,w)`	is the matrix obtained by taking the elements of A with row subscripts from the vector v and column subscripts from the vector w. v and w have to consist of integer components.

Matrix operations.

$\mathtt{C} = \mathtt{A}'$	C is the complex conjugate transpose of A
$\mathtt{C} = \mathtt{A} + \mathtt{B}$ or $\mathtt{C} = \mathtt{A} - \mathtt{B}$	C is the sum (subtraction) of A and B
$\mathtt{C} + \mathtt{A} * \mathtt{B}$	C is the product of A and B
$\mathtt{C} = \mathtt{A} \setminus \mathtt{B}$	C is the solution of $A * C = B$
$\mathtt{C} = \mathtt{B}/\mathtt{A}$	C is the solution of $C * A = B$
$\mathtt{C} = \mathtt{A\hat{}p}$	C is the p-th power of the matrix A

Array operations. The term array operations refers to element-by-element arithmetic operations instead of the usual linear algebraic matrix operations denoted by symbols "$*$", "$\backslash$", "/", "^", "′". Proceeding an operator with a period "**.**" indicates an array or element by element operation, f.e. if `x=[1,2,3]; y=[4,5,6];` then `z=x.*y` results in `z= 4 10 18`

Special and utility matrices.

`compan(p)`	companion matrix; p is a vector of polynomial coefficients.
`diag(x,k)`	diagonal matrix with the elements of x on the k-th diagonal.
`hankel(c)`	the square Hankel matrix whose the first column is c.
`zeros(m,n)`	an m-by-n matrix of zeros.
`eye(m,n)`	an m-by-n matrix of ones on the diagonal.
`linspace(x1,x2,n)`	generates a matrix of linearly spaced n vectors between x1 and x2.
`rand(m,n)`	an m-by-n matrix with random entries.

3.3 Functions

Elementary matrix functions.

`poly(A)`	characteristic polynomial of A
`det(A)`	determinant of A
`inv(A)`	invers of A
`trace(A)`	trace of A
`expm(A)`	matrix exponential of A
`logm(A)`	matrix logarithm of A
`sqrtm(A)`	matrix square root of A
`size(A)`	row and column dimensions of A
`length(v)`	vector length of v
`[V,D] = eig(A)`	eigenvalues (D) and eigenvectors (V) of A.

Columnwise data analysis.

`max(x)`	the largest element in x.
`min(x)`	the smallest element in x.
`sort(x)`	sorts each column of x in ascending order.
`sum(x)`	the sum over each column of x.
`prod(x)`	the product over each column of x.

Elementary math functions. A set of elementary mathematical functions are applied on element by element basis to arrays. The usual elementary functions are:

`abs`	absolute value	`angle`	phase angle
`sqrt`	square root	`real`	real part
`imag`	imaginary part	`conj`	complex conjugate
`round`	round to nearest integer	`sign`	signum function
`exp`	exponential base e	`log`	natural logarithm

Available functions include the trigonometric functions: `sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh`

Functions. The set of available MATLAB functions can be extended by users.
Consider arbitrary function *fun* of an input argument $(in_1, \dots, in_n)$ which yields the output vector $(ou_1, \dots, ou_m)$ to be the new MATLAB function. The following steps has to be done:
1. Open the new file *fun.m* in the directory available by MATLAB.
2. Write in the first line of the file `function [ou1,...,oum]=fun(in1,...,inm)`
3. In the sequel define the output variables `ou1,...,oum` in terms of the input variables `in1,...,inm`, using existing MATLAB expressions, (additional auxiliary variables) ending each expression by ; (except the first line).
For example,

$$fun(x) = \frac{1}{(x-0.3)^2+0.1} + \frac{1}{(x-0.9)^2+0.04} - 6$$

The file *fum.m* would be following:

```
function y=fun(x)
y=1./((x-.3).^2+.01)+1./((x-.9).^2+.04)-6;
```

The statement `fun(3)` in the MATLAB environment results the value of *fun* at $x = 3$
All variables of fun.m file are local. I/O variables `in, ou` can be any matrices.

3.4 Graphics

Plot.

`plot(x,y,'line-type')`	plots the vector `x` versus the vector `y`
`plot(x1,y1,'line-type1', x2,y2,'line-type2',...)`	a way of producing of multiple lines
`print`	print graph screen
`clg`	clear graph screen

Line type. **-** solid, – dashed, **:** dotted, **-.** dashdot.

Plot description.

`title('text')`	plot title: *text*
`xlabel('text')`	x-axis label: *text*
`ylabel('text')`	y-axis label: *text*
`grid`	grid lines

Example.

```
t=0:.05:4*pi;
y=sin(t);
plot(t,y,'-.), xlabel('t'), ylabel('sin(t)'), title('plot 1'), grid;
```

3.5 Miscelaneous

General.

`demo`	run demonstrations
`who`	list variables in memory
`what`	list M-files on disk
`clear`	clear workspace
`clc`	clear command screen

Control flow.

FOR loops:	`for` *variable=expression* , *statement*; ... *statement*; `end`
	For example: `for k=1:n, x(i)=0; end;`
WHILE loops:	`while` *expression*, *statement*; ... *statement*; `end;`
	For example: `eps=1; while (1+eps) > 1, eps=eps/2; end;`
IF statement:	`if` *expression*, *statement*; ... *statement*;
	`else` *statement*; ... *statement*; `end;`
	For example: `if a>b, x=a; else x=b; end;`

Relational operations. $<$ less than; $<=$ less than or equal; $>$ greater than; $>=$ greater than or equal; $==$ equal; = not equal.
The comparison is done between the pairs of corresponding elements. The result is a matrix of ones and zeros, with one representing TRUE and zero FALSE.

Logical operations. **&** AND; | OR; ˜ NOT.

M-files. M-file is a disk file of which the name has the extension **.m**. It consists of a sequence of normal MATLAB statements. There are two types of M-files: *scripts* and *functions*. When *script* is invoked MATLAB simply executes the commands found in the file. (To run a *script* M-file type filename without the extension **.m**.) The statements in a *script file* operate globally on the data in the workspace.
Functions files are useful for extending MATLAB. They are briefly discussed in the previous section.

Useful commands for M-files:

`pause`	pause until key press
`%`	comment line

Disk files.

`save` *filename variable1 ...* `/ascii`	save *variable1,...* in ascii code in the disk file *filename.m.*
`load` *filename*	load the disk file *filename.m*

3.6 Control system toolbox

Model representation.

`den=[`$1, a_{n-1}, \ldots, a_0$ `];` `num=[`$b_n, \ldots, b_0$ `];`	transfer function $\frac{b_{n-1}s^{n-1}+\cdots+b_0}{s^n+a_{n-1}s^{n-1}+\cdots+a_0}$
`A,B,C,D`	state space model $\begin{array}{l}\dot{x} = Ax + Bu \\ y = Cx + Du\end{array}$

Model conversions.

`[num,den] = ss2tf(A,B,C,D,iu)`	state-space to transfer function (`iu` - input number)
`[A,B,C,D] = tf2ss(num,den)`	transfer function to state-space

Analysis functions

`[y,x] = lsim(A,B,C,D,u,t,x0)`	simulation with an arbitrary input `u`.
`[y,x] = lsim(num,den,u,t)`	`u` must have as many columns as there are inputs u and as many rows as length of `t`. The vector `t` specifies the time axis for simulation, f.e. `t=(0:0.1:10)`. `x0` -initial state.
`step(A,B,C,D,iu,t)`	step response.
`step(num,den,t)`	
`bode(num,den)`	Bode plots
`ctrb(A,B)`	controllability matrix.
`obsv(A,C)`	observability matrix.