Learning MATLAB by doing MATLAB*

December 10, 2005

Just type in the following commands and watch the output.

1. Variables, Vectors, Matrices

`>a=7`	a is interpreted as a scalar (or 1×1 matrix)
`>b=[1,2,3]`	after comma: new element in the same row, therefore we have $b \in \mathbb{R}^{1,3}$
`>c=[1+2,3,3]`	
`>d=[7 7 2]`	spaces have the same meaning as commas
`>e=[7 a 2]`	
`>f=[1;2;3;4]`	semicolon: starts a new row, therefore we have $f \in \mathbb{R}^{4,1}$
`>g=f(2)`	accesses the second element of the column vector f
`>E=[1 2 3;2 1 3]`	MATLAB distinguishes between lower and upper cases
`>h=E(1,2)`	accesses the $(1,2)$ element of E
`>E`	
`>F(3,4)=7`	MATLAB currently considers F to be a 3×4-Matrix. Void elements are set to zero.
`>F(4,3)=2`	Now we need a fourth row!
`>F(1,2)=3`	Sets the $(1,2)$ element of F to 3.
`>F(1:2,3:4)=[1 3;2 7]`	$1:2$ means row 1 to row 2; $3:4$ means column 3 to column 4
`>whos`	statistics of used variables
`>clear a b`	deletes variables a and b
`>whose`	
`>clear`	deletes all variables
`>help clear`	help for the command `clear`
`>A=[1 2];`	concluding the line with a semicolon suppresses the output
`>A`	
`>pi`	
`>A=eye(3)`	3×3 identity matrix
`>b=[1 2 3]`	
`>B=diag(b)`	diagonal matrix
`>C=diag([1 7 8])`	
`>D=diag([1;7;8])`	geht auch mit Spaltenvektoren
`>E=ones(4)`	4×4 matrix of all ones
`>F=ones(2,3)`	2×3 matrix of all ones
`>G=zeros(4)`	
`>H=zeros(2,3)`	
`>I=[A B;zeros(3) A]`	block matrix
`>C`	
`>C'`	transpose of C
`>w(3)=5`	MATLAB considers w to be a row vector. The third element of w is set to 5.
`>x=0:1/3:2`	row vector with entries from 0 to 2 in steps of 1/3
`>for i=1:10`	i runs from 1 to 10. (MATLAB waits for the final `end` kommt!)
`y(i)=2*i`	the index $i = 0$ is not possible here, since vectors have no 0th element!
`end`	
`>for i=1:10`	
`z(i)=2*i;`	
`end`	
`>z`	

*This is partly based on lecture notes by Christian Mehl and Andreas Steinbrecher, both TU Berlin.

2. Simple Operations

`>clear`	
`>A=[1 2 3;2 1 0]`	
`>B=[2 2;1 0;0 1]`	
`>C=[0 1 0;5 1 3]`	
`>size(A)`	Returns number of rows and columns of A in a row vector.
`>[m,n]=size(A)`	One way to access these numbers individually.
`>m=size(A,1)`	Another way.
`>b=[2 1 3]`	
`>x=[2;1;3]`	
`>A*B`	matrix multiplications
`>A*C`	error, dimensions do not fit!
`>A`	
`>C`	
`>A*C'`	$A \cdot C^T$
`>diag(A*C')`	returns the diagonal of a matrix
`>whos`	`ans` (for “answer”) is the is the unnamed output
`>D=A+C`	matrix addition
`>E=A+B`	error, dimensions do not fit!
`>E=A-B'`	$E = A - B^T$
`>g=A*x`	matrix times vector
`>g=A*b`	error!
`>A`	
`>b`	
`>B`	
`>f=b*B`	row vector times matrix
`>C=[1 2 3]'`	

3. Matrix Manipulations

`>clear`	
`>A=[1 2;3 4]`	
`>A(3,2)=7`	Adds a third row!
`>A(1:2,2)`	`(1:2,2)`: 1st to 2nd element of column 2
`>A(3,1:2)`	`(3,1:2)`: 1st to 2nd element of row 3
`>B(3:4,3)=[5;6]`	MATLAB creates a matrix B with the 3rd and 4th elements of column 3 being 5 and 6, respectively. All other entries are zero.
`>C(4:5,4)=A(1:2,2)`	
`>B(:,3)`	3rd column of B
`>d=C(1,:)`	1st row of C
`>E=[1 2 3;4 5 6;7 8 9;10 11 12]`	
`>E(1:2:4,3)`	`(1:2:4,3)`: Picks every second element in column 3 from 1 to 4.
`>F=[1 2 3 4 5;6 7 8 9 10;11 12 13 14 15]`	
`>G=F(1:2:3,1:2:5)`	Picks every second element in columns 1,3,5 from 1 to 3.
`>G=F(1:2:end,1:2:end)`	Alternative notation if you forgot the dimensions.
`>H=[1 3;9 11]`	
`>H^(-1)`	The inverse.
`>inv(H)`	Also the inverse.
`>H\A(1:2,1:2)`	Always use the backslash for computing $H^{-1}A$. Don't use `inv(H)*A`!
`>det(H)`	The determinant.
`>b=[99 100 101]`	
`>F(1,1:3)=b`	
`>A`	
`>A(1,1:3)=b`	Elements $(2,3)$ and $(3,3)$ are added.

4. Subprograms, m Files

For the following subprograms you have to create files with the name of the function appended by ".m", for example "test1.m". (You may use any editor to this, MATLAB comes with a built-in editor which also offers debugging.) The m files have to be in the current working directory. Try `help pwd` and `help cd` for more infos about working directories.

```
% This is my first m file.                % You can add comments behind %.
% It is called test1.m and computes the sum of all elements of a matrix A.
function y=test1(A)
[m,n]=size(A);
y=0;                                      % Init
for i=1:m                                 % i runs from 1 to m
  for j=1:n                               % j runs from 1 to n
    y=y+A(i,j);
  end
end
```

Save under `test1.m`.

```
>clear
>help test1
>A=[1 2;3 4]
>s=test1(A)
```

```
% The program test2.m computes the sum and the product of all elements of a matrix A,
% as well as the trace if the matrix is square.
function [y,p,t]=test2(A)
[m,n]=size(A);
summe=0;
p=1;
t=0;
for i=1:m
  if (m==n)                               % if m = n, then ...
    t=t+A(i,i);
  end
  for j=1:n
    y=y+A(i,j);
    p=p*A(i,j);
  end
end
```

Save under `test2.m`.

```
>clear, help test2
>A=[1 2;3 4]
>test2(A)                 Only the first output argument is displayed.
>[su,pr,sp]=test2(A);
>su,pr,sp
>B=[1 2 3;4 5 6]
>[su,pr,sp]=test2(B)
```

MATLAB functions can have several input variables (e.g., `function ausgabe=test3(A,B,C)`) or require no input/output at all (e.g., `function []=test4()` or you just omit the whole function declaration, call with `test4`). Besides the 'for' loop, there is also a 'while' loop. See `help while`.

5. Graphics

```
>for i=1:10, x(i)=i/10; y(i)=x(i)^2; z(i)=sqrt(x(i)); end
>plot(x,y)
>plot(x,z)
>clf
>plot(x,y)
>hold on
>plot(x,z)
>plot(x,2*z,'r')
>plot(x,y+z,'g*')
>hold off
>plot(x,y-z,'k+')
>help plot
>title('My plot')
>xlabel('x axis')
>ylabel('y axis')
>axis([0,20,-5,50])
>box
>grid
>clf
>subplot(3,2,1)          The plot has 3 · 2 = 6 subplots. The 1st subplot is active.
>plot(x,y)
>subplot(3,2,2)
>plot(x,z,'k')
>subplot(3,2,5)
>plot(x,z+y,'mo')
>hold on
>plot(x,z,'k')
>subplot(3,2,1)
>plot(x,z,'k')
>subplot(3,2,4)
>title('leer')
>subplot(3,1,2)          We have only 3 subplots, one in each row. The 2nd is now active.
>plot(y)
>orient tall
>help orient
>print -dps test1.ps     Creates a pos file test1.ps of this plot (more options in the menu File/Save As..)
>help print
```

Turn on Tools/Edit Plots and right click to change properties of the plots. For inclusion of plots in presentations and papers it is often a good idea to increase the font sizes and thicken the lines.

6. Other useful commands

`load`	–	load variables from files
`save`	–	save variables in files
`blkdiag`	–	create block diagonal matrices
`kron`	–	Kronecker product
`A(:)`	–	vec(A)
`.* ./`	–	elementwise multiplication and division of two matrices
`chol`	–	Cholesky decomposition of matrices
`cond`	–	matrix condition number
`eig`	–	eigenvalues of matrices and matrix pencils
`hess`	–	Hessenberg form of matrices
`norm`	–	matrix norm
`qr`	–	QR decomposition of matrices
`schur`	–	Schur form of matrices
`svd`	–	singular values of matrices

7. Using the Control Toolbox

First, let us consider a simple example for demonstrating the basic functionality.

```
>A = [0 1; -1/4 -1], B = [ 0; 1/2 ]
>C = [0 1], D = 0        Create the system matrices of a simple system.
>sys1 = ss(A,B,C,D)      The object sys1 contains this system in state space form.
>sys1.a
>eig(sys1)               poles of the system
>tf1 = tf(sys1)          form transfer function
>eig(tf1)                surprise, surprise
>zero(tf1)
>ss(tf1)                 Can you tell the difference?
>bode(sys1)              Try to guess the H^∞ norm!
>norm(sys1,inf)
>20*log10(ans)           Converted to dB.
>norm(sys1,2)
>svd(ctr(sys))           Kalman test for controllability
>svd(obsv(sys))          Kalman test for observability
>A = [4 3; -9/2 -7/2], B = [ 1; -1 ],
>C = [3 2], D = 0,       Create the system matrices of another system.
>sys2 = ss(A,B,C,D)
>eig(sys2)
>big = sys1 + sys2       Convenient way to form interconnected systems.
>zero(big), eig(big)
>norm(big,inf)
>norm(big,2)
>svd(obsv(big))
>[Ao,Bo,Co,Qo,ko] = obsvf(big.a,big.b,big.c)
>svd(ctrb(big))
>[Ac,Bc,Cc,Qc,kc] = ctrbf(big.a,big.b,big.c)
>bigmin = ss(big,'min')
>zero(bigmin), eig(bigmin)
>sys1.d = 1              Make first system invertible.
>tf(sys1)
>sysi = inv(sys1)        Form inverse of system.
>tf(sysi)
>s = tf('s')             Convenient for creating transfer functions directly.
>H = (s-1)*(s+4)/(s^2+2*s+10)
```

Type “help control” for a list of available functions in the control toolbox.

Now, let us consider a bigger example.

1. Create a working directory if you have not done so far. Download the benchmark examples available from `http://web.math.hr/~kressner/nlacontrol/` into this directory.
2. Store the system matrices of the benchmark example 1.9, which represents an aeroelastic model of parts of a Boeing B-767 airplane, in a system object in state space form (you can ignore the identity matrix E).
3. Investigate the stability of this system by computing its poles.
4. Have a look at the coefficients of the transfer function.
5. Form the controllability matrix and plot its singular values with `semilogy`.
6. Test controllability/obserability using the Hautus test.
7. Determine the number of controllable poles using `ctrbf`. Investigate the structure of the state matrix in controllability form using `spy(abs(Ac)>0.00000001)`.
8. Determine the number of observable poles using `obsvf`.