Matlab

Matlab is a language for technical computing[3]. It is widely used in the industry as well as in the academic world. Here at HUT Matlab is available for students and researchers in Unix and Windows networks. Unfortunately, there are not any off-line student versions available, but there is a free GNU Octave[4], which is mostly compatible with Matlab.

How to Read

This is basically a short reference for using Matlab in digital signal processing. This is not a tutorial. There are Matlab demo exercises in the autumn course that explain a lot more. These and a lot more can be found in `http://www.cis.hut.fi/Opinnot/T-61.246/Matlab/`.

Probably the best help can be found from Matlab itself. It contains all information on the commands and lots of examples and tutorials. There are some functions listed in a section from Page 121.

Using Matlab at HUT

The newest version of Matlab is Matlab 7. There are lots of graphical user interfaces and helps, but all commands can be written also in the prompt in Command Window. Matlab provides excellent help pages with several examples. Type `help function` in order to get a quick help for the operation of the function `function`.

When working in Unix environment, start by typing `use matlab` and open Matlab then with `matlab`. Listening of sound files may not work directly inside Matlab, but you have to write them to a WAV format (`wavwrite`) and listen with a player in Unix. The command `exit` terminates the session.

You should write down all your code into a file, which can be re-run or modified later. Use the Matlab editor, which can be opened by typing `edit` in Command Window. Save your files in the subdirectory `Z:\MYDSP\` (or corresponding) in Windows Network and set your Current Directory (a box containing a directory path in top of Matlab Window) to point to the same directory.

Basic Elements and Operations

Basic elements and operations are introduced in Matlab exercise I. Matlab operates with matrices of different sizes. By typing:

```
A = [3  -2  1  1  5; 2  4  0  -1  -3;  5  5  9  -2  -4]
```

we get a matrix of size $N \times M$, where N is the number of rows and M is the number of columns.

```
A =
     3    -2     1     1     5
     2     4     0    -1    -3
     5     5     9    -2    -4
```

We can easily get vectors with colon notation or by picking numbers from a matrix:

[3] `http://www.mathworks.com`
[4] `http://www.octave.org`


```
t = [3 : 0.5 : 4.5]  % [start : interval : end]
y = A(2, 2:5);       % second row, columns 2 to 5
% no feedback on the screen if semicolon (;) in the end of line
```

Matrix calculations can be made if the dimensions are proper.

```
t*y    % error, dimensions do not hold [1x4]*[1x4]
t.*y   % itemwise multiplication, [1x4].*[1x4]=[1x4]
t*y'   % inner product, ' == (complex conjugate) transpose, [1x4]*[4x1]=[1x1]
t'*y   % outer product, [4x1]*[1x4]=[4x4]
```

You can find the size of the matrix using `size(A)` and the length of the vector using `length(y)`.

Matlab provides basic operations for scalar, vectors or matrices, as well as for real and complex numbers, e.g. `exp(j*pi)`, `cos(3.14)`, `sqrt(25)`, ... (see Table ??).

Scripts and functions

Scripts and functions are both Matlab files ending with `.m`. They have slight but important differences althought they look like similar. You can add any comments to your files after % sign.

A **script** is a set of commands executed in a batch. The variables are found in the Matlab Workspace. Consider a script, which draws a circle with red color on the screen. The following lines are in a file called `drawcircle.m`.

```
w  = [0 : pi/64 : 2*pi];
C  = exp(j*w);                % i and j are both imaginary units in Matlab
plot(real(C), imag(C),'r');  % 'r' == red color
```

The script is called from Command Window by `drawcircle`, i.e. without the extension `.m`.

```
>> clear       % removes all variables in Matlab Workspace
>> whos        % --> no variables
>> drawcircle  % executes commands and draws a circle on screen
>> whos        % --> variables C and w available!
```

A **function** may receive input variables and produce output variables which are seen only in the memory area of the function.The first line of the function starts with a word `function` and contains names of input and output variables. Consider an example on a function which computes the angle in degrees. The filename for the function is `getdeg.m` and contains:

```
function [deg] = getdeg(z)
% GETDEG computes the angle of the complex number z = x + yj in degrees
x   = real(z);
y   = imag(z);
deg = 180 * atan(y/x) / pi;
```

If several input or output arguments, they are listed with the comma (,) seperator. The second line (and adjacent %lines) are printed when `help getdeg`.

The function is called from Command Window with `getdeg`

```
>> clear         % removes all variables in Matlab Workspace
>> whos          % --> no variables
>> z1 = 3 + 4*j; % a complex-valued z1
>> [degrees] = getdeg(z1);
>> whos          % --> variables z1 and degrees available, but no x, y nor deg
```


Somehow a function is a safer way to work because variables cannot be changed anywhere else but in the code. Assume that you have used a variable `j` as a counter, and its value is now 100. Then if you run `drawcircle.m` you will not get a circle while `C = exp(100*w);`! On the other hand, reading an array of numbers from a file to Matlab workspace is useful to do only once, not in a function that is called 100 times. In this case an array is read and then a script is used or the variable is given into a function as an input.

Error messages

If a syntax error occurs, it is informed in red color with the line and column. Often an error is easily found. Typical errors are a wrong number or type of parentheses, a comma instead of a dot, small/CAPITAL letter instead of CAPITAL/small, wrong number of arguments for a function, etc.

Sometimes a logical error happens, which means that no error messages are reported but the code does not work as hoped. If scripts are used, one can remove all variables by `clear all`, and try to run the code again.

Plotting and Printing

Basic plotting and printing commands are introduced in Matlab exercise I. Matlab provides easy tools for plotting figures. The basic command is `plot`, normally with two inputs `X` and `Y`. It is nice that you add labels and titles for each figures, `grid on`, `title`, `xlabel`, `legend`, etc.

You can print your figure directly from the Matlab window, but it is probably nice to export it into a file. Depending on which operating system you are using:

```
print -dmeta myfig.emf  % Windows Metafile for Word
print -dpng  myfig.png  % Portable Network Graphics for web browsers
print -deps  myfig.eps  % Encapsulated PostScript for LaTeX
```

Signal Processing Toolbox

All the commands related to Signal Processing Toolbox can be found by typing `help signal`. There are some nice demos, e.g. a DTMF demo with `phone`, and some GUI tools like `sptool` (Matlab exercise II).

Digital signal is represented as a vector in Matlab. For example, $x[n] = 2\delta[n] - 3\delta[n-2]$ can be written using the coefficients `x = [2 0 -3]`. An audio signal can be read with `[x, fs, nbits] = wavread('Z:\MYDSP\audio.wav');` Audio vectors can be exported with `wavwrite`.

An impulse response sequence $h[n] = \delta[n] - \delta[n-4]$ is expressed by `h = [1 0 0 0 -1]`. A digital filter $H(z) = B(z)/A(z)$ is represented in Matlab using the coefficients of the numerator polynomial $B(z)$ and that of the denominator polynomial $A(z)$. For example, in case of second-order IIR filter $H(z) = (1 - z^{-2})/(1 + 0.81z^{-2})$, the vectors and some analysis functions are

```
B = [1  0  -1];          % coefficients of B(z) in H(z)=B(z)/A(z)
A = [1  0  0.81];        % coefficients of A(z) in H(z)=B(z)/A(z)
figure(1); freqz(B, A);  % amplitude and phase responses
figure(2); zplane(B, A); % pole-zero plot
figure(3); impz(B, A);   % impulse response
```


In Matlab II there are examples on computing the spectrum and spectrogram of the signal. There are also demos on analyzing a LTI system and filtering. Examples on filter design (system synthesis) can be found in Matlab III. An important part here is to scale the frequencies correctly between $[0 \dots 1]$:

$$\frac{f_c}{f_T} = \frac{f_{c,Matlab}}{2} \qquad \text{Example: } \frac{3500 \text{ Hz}}{20000 \text{ Hz}} = \frac{f_{c,Matlab}}{2} \quad \Rightarrow \quad f_{c,Matlab} = 0.35$$

Some Matlab Commands

.

General commands and notations
`quit`, terminates Matlab session
`help`, lists all function directories available
`help` *function*, gives help on *function*, e.g. input arguments
`type` *function*, shows the code of *function*, which are not build-in
`pause`, waits until the user presses any key
`pause(s)`, waits s seconds
`more`, pauses scrolling
`who` or `whos`, lists all variables
`diary`, copies user commands into a file
`ver`, lists all toolboxes (versions) available
`!` *OS-command*, calls *command* in operating system
`%`, starts the comment till the end of line
`;`, does not print anything on screen
`disp`, prints strings of text nicely

File I/O
`load`, opens a MAT binary file containing variables
`save`, saves the variables into a MAT binary file
`textread`, reads a formatted text file into Matlab workspace
`dlmread`, reads a formatted file of numbers into Matlab workspace
`fopen`, opens a file
`fclose`, closes a file
`fprintf`, writes into a file
`fprintf(1,'...')`, writes into a file handle 1 = on screen

Useful commands
`size`, gives dimensions of the matrix (vector)
`length`, gives the length of the vector
`fliplr, flipud`, flips the order of items in an array, $x[n] \rightarrow x[-n]$
`[xstart : interval: xstop]`, creates a vector starting from $xstart$
`linspace`, creates a vector similarly to [* : * : *] notation

Elemenary functions
`exp`, `cos`, `sin`, `atan`, `sqrt`, `log`, `log10`, ...
`abs`, absolute value of a complex number, e.g. $|H(e^{j\omega})|$
`angle`, angle of a complex number, e.g. $\angle H(e^{j\omega})$
`real`, real part of a complex number
`imag`, imaginary part of a complex number
`roots`, calculates roots of a polynomial

`sum`, sums elements column-wise

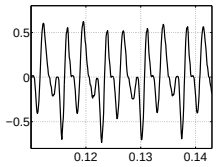
Plotting figures
`plot`, plots continuous signals
`stem`, plots sequences
`clf`, clears the current figure
`cla`, clears the current axis
`shg`, the active window is brought on top
`close all`, closes all windows
`subplot`, creates several axis in a window
`grid on`, inserts a grid on figure
`title`, title for a figure
`xlabel`, title for x-axis
`ylabel`, title for y-axis
`legend`, creates a legend for a figure
`axis`, zoom the axis, `axis([xmin xmax ymin ymax])`
`print`, exports a figure into a file or prints it to a printer
`get`, gets values of an object, e.g. `p = plot(..); get(p)`
`set`, sets values of an object, e.g. `p = plot(..); set(p,'LineWidth',2)`

Commands for audio
`soundsc` or `sound`, plays a vector as sound (sc = scaled)
`wavread`, reads a WAV file into Matlab
`wavwrite`, writes a WAV file from Matlab

DSP functions, see Signal Processing Toolbox
`fft`, fast Fourier transform
`ifft`, inverse Fourier transform
`dftmtx`, computes a matrix W for DFT
`unwrap`, eliminates jumps in phase angles
`conv`, linear convolution of two sequences
`conv`, polynomial multiplication
`filter`, filters signal x with a filter given
`impz`, impulse response for a discrete-time finite-dimensional system
`freqz`, draws frequence response, magnitude and phase response
`zplane`, plots a pole-zero-diagram
`residuez`, partial-fraction expansion of z-trasform $H(z)$
`tf2zp`, converts transfer function to corresponding zeros and poles
`tf2sos`, converts transfer function to corresponding set of second-order systems
`buttord, cheb1ord, ellipord`, etc., estimates the minimum order for fulfilling specifications
`butter, cheby1, ellip`, etc., computes filter coefficients
`fir1`, one of FIR filter design procedures, window method
`remez`, one of FIR filter design procedures, Parks-McClellan

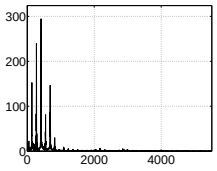
Some Matlab Examples

Read an audio file, plot and listen to it.



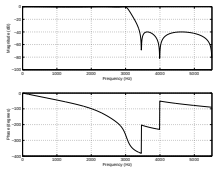

```
[y, fs, nbits] = wavread('vowel_o.wav');
M  = length(y);
t  = [0 : M-1]/fs;
plot(t, y); grid on;
title('My vowel /o/'); xlabel('time (s)');
axis([0.1 0.14 -0.8 0.8]);
soundsc(y, fs);
```

Compute a FFT $X(e^{j\omega})$ from the whole sequence and plot the spectrum.



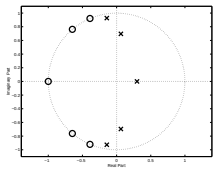

```
yF = fft(y);
M  = length(yF);
w  = fs * [0 : M-1]/M;
plot(w, 20*log10(abs(yF))); grid on;
title('Spectrum'); xlabel('freq. (Hz)');
set(gca, 'XLim', [0 fs/2]);
```

Design an elliptic IIR lowpass filter with cut-off 0.3π and plot the frequency response $|H(e^{j\omega})|$ and a pole zero diagram.



```
Wp = 0.3;  Ws = 0.36;
Rp = 0.5;  Rs = 40;
[N, Wn] = ellipord(Wp, Ws, Rp, Rs);
[B, A]  = ellip(N, Rp, Rs, Wn);
freqz(B, A);
zplane(B, A);
```

Design a FIR filter, cut-off at 3300 Hz and $f_T = 16$ kHz, with Hamming window of order 15, plot the frequency response $|H(e^{j\omega})|$.



```
Wp = 3300 / (16000/2);
N  = 15;
[B, A] = fir1(N, Wp, hamming(N+1));
[H, w] = freqz(B, A, 1024, 16000);
plot(w, abs(H));
```