Matlab
lesson 3: programming

Instructor:
ir drs E.J Boks
Elecrical Engineering
Embedded Systems Engineering
phone:(026) 3658173
Room: D2.03
e-mail:ewout.boks@han.nl

Answers to lesson 2 questions

```
% plot exercise 1


% dataset 1

x1 = 10:40;

y1 = log(20*pi*x1);
```

```
% exercise 2

f = 3;

omega = 2*pi*f;

t = 0:(1/5*f):2;   % 5x oversapmling
y2 = 25*cos(2*pi*omega*t -0.25);

% plotting
h1 = figure;
h2 = figure;

% select the first
figure(h1);
plot(x1,y1);
% and another window
figure(h2);
plot(t,y2);
```

Answers to last week's questions

```
% Exercise 3
% MSCS course
% baseband signal
Freq = 1E3;
% Carrier
CA = 4;
% modulation
modfreq= 300;
b = 1;
```

Answers to last week's questions

```
% data sampling
sampling_freq = 4*Freq;
time = 0:(0.1/sampling_freq):5/Freq;    % we evaluate 4 periods of the signal
y_baseband = CA*cos(2*pi*Freq*time);
y_modulator = CA*cos(2*pi*modfreq*time);
y_am = CA*(1+b).*cos(2*pi*modfreq*time).*cos(2*pi*Freq*time);
y_fm = CA*cos(2*pi*Freq*time+b*cos(2*pi*modfreq*time));
% produce the plots
subplot(4,1,1),stem(y_baseband,'go'), xlabel('Baseband signal');
subplot(4,1,2),stem(y_modulator,'bo'), xlabel('Modulator');
subplot(4,1,3),stem(y_am,'bo'), xlabel('AM modulated');
subplot(4,1,4),stem(y_fm,'ro'), xlabel('FM modulated');
```

Programming outline

- Matlab programming is centered around the creation of .m files that become scripts or functions.
- A script simply executes a program, whereas a function accepts input data and produces output data.
- Script is simply a body of text containing matlab cmds
- Function definition adheres to a standard outline

.m file

The .m file can be created by typing edit <filename>. This opens the default editor. The file is saved on the current path .

The path is listed by typing pwd .

To see the particular path of a .m file, use the which cmd.

In order to make .m file executable, the addpath cmd can be used to modify the $PATH variable. for example:

addpath('/home/johan/work/matlab/thesis/mfiles') adds a path to this directory.

Note: on Unix and Mac OS X systems, use the / operator. Win32 requires the \ operator.

Function definition

- Function definition :

function output | [$output_1$ $output_2$ ... $output_n$] = name($input_1$,$input_2$,.....,$input_n$)

body of function. All variables declared and used in the body are local variables whose scope ends at the boundaries of the function

Global variables are defined by declaring them global , e.g :

global var;

Basic .m file layout

A basic .m file with a function definition should look like this:

```
function [ f  g ] = fact(n,m)          % function definition line
```

inputs

function name

output

Keyword

```
% Hello everybody this is what I do – H1 line
% Here goes the Help text
f = prod(n:m);                         % function body
g=f+5;
```

function example

```
function avg = avgscore(testscores, student,first,last)

global x,y;
for k=first:last
    scores(k) = x*testscores.(student).week(k-y);
end

avg = sum(scores)/(last-first+1);
```

Subfunction function within a function defintion

- A subfunction is declared in the body of the main function.
- All variables local to the main function are out of scope for the subfunction.
- Subfunctions can be used to subdivide your code in to easily digestable bits that can be debugged more easily than one solid block of code.

another function example

The besselj function in Matlab looks like this:

```
function [w,ierr] = besselj(nu,z,scale)
%BESSELJ Bessel function of the first kind.
%   J = BESSELJ(NU,Z) is the Bessel function of the first kind, J_nu(Z).
%   The order NU need not be an integer, but must be real.
%   The argument Z can be complex.  The result is real where Z is positive.
```

another function example

And continued ....

```
%   $Revision: 5.17 $  $Date: 2002/04/09 00:29:45

if nargin == 2, scale = 0; end

[msg,nu,z,siz] = besschk(nu,z); error(msg);

%   Copyright 1984-2002 The MathWorks, Inc.

[w,ierr] = besselmx(real('J'),nu,z,scale);

if ~isempty(w) & all(all(imag(w) == 0)), w = real(w); end

w = reshape(w,siz);
```

Programming operators

All the usual operators can be used in Matlab programming:

- Arithmetic operators
- Relational operators
- logical operators

Programming tools

- Flow control : very similar to other high level languages.
  - If … else …. elseif … end . Conditional execution based on the outcome of a logic expression.
  - Switch. Execution of a body based on the value of a variable.
  - For loop. Defined loop under control of a pre-set range.
  - While. Undefined loop under control a logicical condition.
  - try-catch mechanism. This traps errors at runtime

if - else

An example :

```
if n< 0
    disp('The input is negative');
elseif rem(n,2) == 0 %  if n positive and even, divide by 2.
    A = n/2;
else
    A = (n+1)/2;
end
```

switch example

An example of the switch statement is:

```
switch var
    case 1
        disp('1');
    case {2,3,4}
        disp('Higher than 1 and less than 5');
    otherwise
        disp('5 or higher');
end
```

For loop example

An example of a defined loop:

```
for m=2:6
    x(m) = 2*x(m-1);
    for (n=m:12)
        y(n) = y(m-1)*x(n);
    end
end
```

While loop

An example of an undefined loop:

```
n=1;
while prod(1:n) < 1E10
    n=n+1;
end
```

an undefined loop may be terminated by using the break statement. The opposite continue statement enters the next iteration of the loop.

Try catch mechanism

An example of the try-catch mechanism is:

```
try
    statement1
    statement2
    .....
catch
    statementA
    statementB
end
```

Expressions

In Matlab, four types of expression evaluation exist:

- Evaluation at run-time from the .m file or the cmd line.
- String evaluation using the eval() or feval() functions.
- Shell escape functions. These are run by typing their name preceded by ! in the cmd window.
- Evalution through regular expressions – texprocessing using tokesn and operators.

Other data types

In addition to the standard scalar and matrix variables, Matlab offers a number of other data types that are used in programming:

- Multidimensional arrays – matrix with a time dimension
- Cell arrays – array of elements that can contain other Matlab data elements such as matrices or text.
- Characters and text
- Structures – similar to C language struct elements

Other datatypes

- $Array^k$ = (zeros|ones|rand|randn)(a,b,c,....k) creates a k dimensional array.
- Cell arrays are multidimensional arrays whose elements are copies of other arrays: array = { $array_1$ $array_2$ $array_3$ ..... $array_k$ }
- Characters: text = 'Hello world!'
- Structures: bodies of variables grouped in a structure – very similar to a C language structure.

Creation of a Multidimensional array

An example:

```
A = [5 3 9; 3 65 -5; 23 12 7];
A(:,:,2) = [12 -3 4; 5 4 66; -3 -100 -12];
```

The dimension (ie Timestamp or page) is given by the last parameter.

an element on page (dimension) 1 of the matrix is accessed by:

```
A(i,j,1) ;
```

Cell Array example

A cell array is a mechanism to store and retrieve large or diffuse amounts of data. The key identifier is are the curly brackets { and }. These are the cell array constructors (comparable with the [and] in normal matrices.

An example:

A(1,1) = { [1 2 3; 4 5 6; 7 8 9] };

A(1,2) = { 'Alexander the Great' };

A(2,1) = { 3-5j };

A(2,2) = { -pi: pi/25 : pi };

Cell elements can be accessed either by typing the cell name or using the celldisp or cellplot functions.

Structure

an example of building a structure is:

patient.name = 'Fred Flintstone'

patient.address = 'Bedrock”

patient.yob = 1966;

In the example, patient can have and index:

patient(3)

Accessing the structure data field is done through the dot (.) operator:

patient.name;

This field can either be static or dynamic (evaluated at runtime)

Vectorisation

A typical pitfall for engineers accustomed to other programming languages is that they omit using Matlab's matrix abilities.

Code example:

```
x = 0.01;
for k=1:1001
    y(k)=log10(x);
    x = x + 0.01;
end
```

This code is quite allright but not very efficient.

Vectorisation

The right way to accomplish this calculation:

```
x = 0.01 : 0.01: 10 ;
y = log10(x);
```

- This code computes the same result but much faster and leaner.
- Use Matlab matrix engine whenever possible.

Function handles

- A function can be *referenced* to by using a function handle – analogous to a pointer to a function in the C language. The handle is defined by the @ sign.

Example: fhandle = @sin ;

This handle can then be used in other calculations using the feval (function evaluation) cmd:

plot(data, feval(fhandle,data));

Function functions

- In order to perform a specific evaluation on a function, so-called function functions exist in Matlab. In this class, function functions perform:
  - Zero finding
  - Optimisation
  - Quadrature
  - solving of ordinary differential equations

Function function example

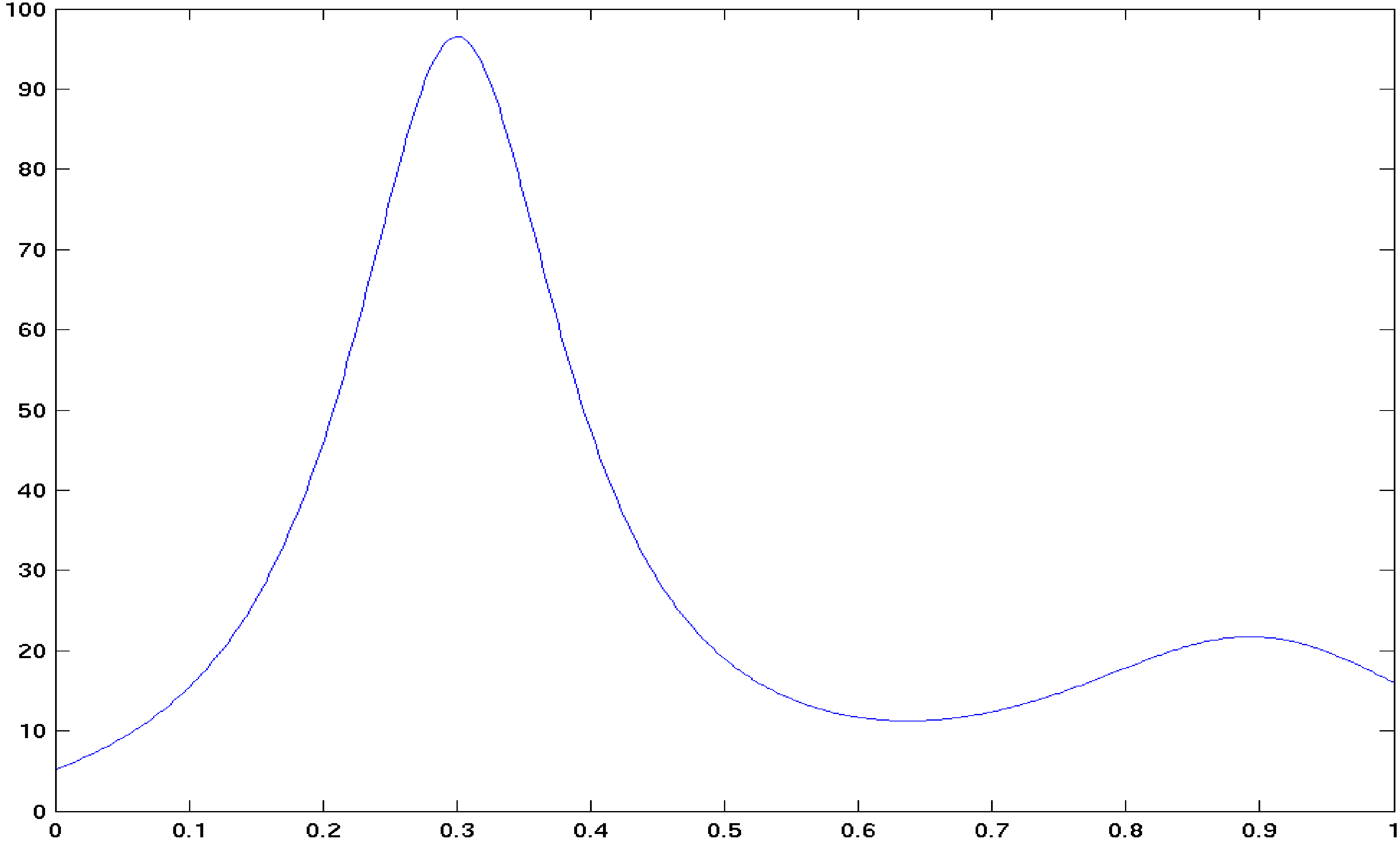
```
Function y = humps(x)
y = 1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6;

x = 0:0.002:1;
y=humps(x);

plot(x,y)
```

Then, when we plot this function, we observe a minimum around x=0.63

Y=humps(x) plot



Humps minimum

- Create a function handle:

```
humpshandle = @humps;
```

- Then, enter the handle into the minimiser function function:

```
minx = fminsearch(humpshandle, 0.5)

minx =

     0.6370

miny = humps(minx)

ans =

   11.2528
```

Programming ticks & tricks

- Planning the Program
- Using Pseudo-Code
- Selecting the Right Data Structures
- General Coding Practices
- Naming a Function Uniquely
- The Importance of Comments
- Coding in Steps
- Making Modifications in Steps
- Functions with One Calling Function
- Testing the Final Program

When proper programming still results in errors ......

Then Matlab offers the .m file debugger.

From the editor, start the graphical debugger. This behaves like other debuggers, ie one can:

- set breakpoints
- stop execution
- evaluate variables
- locate runtime errors

Matlab programming

Break

Questions

- Problem 1 :

given the signal :

$$V(t) = 20 * \cos^2(\omega t) - 15\sin^3((3\omega - \frac{2}{3}\pi) * t) \quad (V)$$

where $\omega = 2\pi f$.

write a Matlab program function that enables you to easily plug in multiple values of f and t.

(*Remember the difference between matrix and pointwise multiplications)*

Questions (2)

- Problem 2:

You generate the following test signal:

  - $f = 4$ Hz
  - The time interval $t = 0$ to 3 seconds

What is an applicable time interval spacing (sampling time)?

Write a script in which you compute all time periods where the signal has an amplitude larger than 30.

Questions (3)

- Problem 3 (advanced):

The signal V(t) (see problem 2) is received from a real source with undesired noise added. The noise has an rms value of 20V . Add this noise to your Matlab signal.

Write a function or script that determines the original frequencies back from the polluted signal V(t) + noise.

- Analyse the plot: what are the identifiable frequency components in the signal ?
- Do they correspond with the given frequencies of the signal function ?

Questions (4)

Some tips for solving this exercise :

- An N-dimensional noise matrix can be made with the rand(N) function.
- For the given angular frequency and time domain, compute the Fourier Transform using Matlab's built-in FFT function. For help, type help fft.

Y = fft(X,n) where X is a set of data and n specifies the number of FFT points. Use n=512 in this case.

- Then, compute the power spectrum of the signal. The power spectrum of signal Y is equal to Y * Y conjugated. Use Matlab's built-in function conj() for this purpose.
- Plot the power spectrum for the applicable frequency range ($0 .. 0.5 * F_{sampling}$)

Matlab programming

End of lesson three