95 Matlab summary and exercises

We first give a summary of Matlab/Octave topics that you should be familiar with now. Then we give a number of simple test questions. These are examples of the kind of Matlab/Octave question that will be on the exam. Note that these are in the obligatory part of the exam (basic questions) and you must answer all of them correctly.

1. Summary

Matlab/Octave is supposed to be a natural mathematical tool for you to use in the mathematics courses and in the engineering courses. You will be expected to use Matlab/Octave without help in these courses. Everone of the students must be able to do this, so here is a list of basic topics that I want to emphasize now and that you must check that you are familiar with. Make sure that you understand all of them.

- The use of semicolon `;`.
- How to enter a matrix.
- How to enter a text string.
- How to enter floating point numbers, e.g., `1.55e3`.
- Know the difference between matrix multiplication and elementwise multiplication, i.e., `A*B` and `A.*B` .
- The command `plot`.
- The command `feval`.
- Iteration with two kinds of loop: `while ... end` and `for ... end`.
- The conditional command: `if ... end`
- Understand how functions work.
- Use the commands `help` and `helpdesk`.
- Two kinds of m-files: function file and script file.
- Finding simple test problems.
- Methods for finding errors (debugging); for example, compute the steps of the program by hand, remove all semicolons and compare with what the program actually does.
- Never use numerical values in the program; use variables instead, initialize the variables at the beginning of the computation. Simple example:

```
>> f='funk3'; h=0.1; I=[0, 1]; u0=0;
>> [x,U]=my_ode(f,I,u0,h);
```

Two more topics that we have not used yet (to be done in later courses; you need not know this yet):

- Use breakpoints for debugging.
- The command `global`.

2. Exam questions

Make sure that you can do **all** of the following. (The `>>` denotes the prompt in the command window.)

95.1. The file `funk1.m` is:

```
function z=funk1(x,y)
z=x/2+y/3;
```

What are the values of `a` and `size(a)` after the following:

```
>> x=3; y=6; z=8;
>> y=funk1(z,x);
>> a=[x, y, z];
```

95.2. Add the missing parts of the following program (marked by `??`):

```
function [x,U]=my_int(f,I,ua,h)
% my_int - solves the initial value problem u'(x)=f(x), u(a)=ua
%
%   Syntax:
%           [x,U]=my_int(f,I,ua,h)
%   Arguments:
%           f   - string containing the name of a function file,
%                 for example, f='funk'
%           I   - 1x2 matrix, specifying an interval I=[a b]
%           ua  - real number, the  initial value
%           h   - positive number, the stepsize
%   Returns:
%           x -  a vector, the set of nodes x(i)
%           U -  a vector, U(i) is the approximate solution at
%                the point x(i)
%   Description:
%           The program computes an approximate solution of the initial
%           value problem  u'(x)=f(x), a<x<b; u(a)=ua, according to
%           the algorithm in the Fundamental Theorem of Calculus.
a=I(1);
b=I(2);
??
x(i)=a;
U(i)=ua;
while x(i)<b
  i=i+1;
  x(i)=??
  U(i)=??
end
x=x';
U=U';
```

95.3. The file `my_trig.m` is:

```
function [t,W]=my_trig(int,w0,h)
% my_trig - the solves initial value problem for the system of
%          ordinary differential equations w'=Aw, A=[0 1;-1 0]
%   Syntax:
%           [t,W]=my_trig(int,w0,h)
```

```
%   Arguments:
%           int - 1x2 matrix specifying a time interval int=[a,b]
%           w0  - 2x1 matrix specifying an initial value
%           h   - positive number, the stepsize
%   Returns:
%           t - nx1 matrix containg the time points with t(1)=a
%           W - nx2 matrix containing the approximate solution
%   Description:
%           The program computes an approximate solution of the intial
%           value problem  w'=Aw, a<t<b; w(a)=w0, A=[0 1;-1 0].  Here w
%           and w0 are column vectors of dimension 2x1.
a=int(1);
b=int(2);
A=[0 1;-1 0];
i=1;
t(1)=a;
W(:,1)=w0;
while t(i)<b
  i=i+1;
  t(i)=t(i-1)+h;
  W(:,i)=W(:,i-1)+h*A*W(:,i-1);
end
t=t';
W=W';
```

What are the values of `x` and `U` after the following:

```
>> I=[0 0.1]; h=1e-1; u0=[0;1];
>> [x,U]=my_trig(I,u0,h);
```

95.4. The file `funk3.m` is:

```
function y=funk3(t,x)
c=5
y=c*x;
```

The commands

```
>>f='funk3'; h=0.1; I=[0, 1]; u0=0;
>>[x,U]=my_ode(f,I,u0,h);
```

give the following output in the command window:

```
c=5
c=5
c=5
c=5
c=5
```

Correct the error.

95.5. Correct the error in the following:

```
>> f=funk3; h=0.1; I=[0, 1]; u0=0;
>> [x,U]=my_ode(f,I,u0,h);
```

95.6. We first write:

```
>> a=[1;2;3]; b=[4;5;6]; c=[7;8;9]; A=[a,b,c]; B=[a,b];
```

Which of the following are not correct?

```
>> a+b
>> a*b
>> a.*b
>> A*b
>> A.*b
>> a'*b
>> a*b'
>> A*A
>> A.*A
>> A*B
>> A.*B
>> B*A
>> B*B
```

95.7. We want to compute $\exp(x)$ and write the file `funk4.m`:

```
function y=funk4(u)
% Function file for computing exp with my_ode.
y=u;
```

and the command:

```
>> f='funk4'; h=0.1; I=[0, 1]; u0=0;
>> [x,U]=my_ode(f,I,u0,h);
```

Correct the two errors. What output do you get from the command

```
>> help funk4
```

95.8. Complete the following program. What is the result of the command `>> z=iteration(2,3)`?

```
function y=iteration(x,n)
% iteration - computes iterated squares
%   Syntax:
%           y=iteration(x,n)
%   Arguments:
%           x - a real number, the initial value
%           n - an integer, the number of iterations
%   Returns:
%           y - nx1 matrix containg the iterated numbers
%   Description:
%           The program computes n steps of the iteration y(i+1)=y(i)^2
%           starting with the initial value x.
```

3. Answers

95.1. `a=[3, 5, 8]`, `size(a)=[1,3]`

95.2. `i=1;`
`x(i)=x(i-1)+h;`
`U(i)=U(i-1)+h*feval(f,x(i-1));`

95.3. `x=[0;0.1]`, `U=[0 1;0.1 1]`

95.4. `c=5;`

95.5. `f='funk3';`

95.6. The following are wrong:

```
>> a*b
>> A.*b
>> A.*B
>> B*A
>> B*B
```

95.7. `function y=funk4(x,u)`

```
u0=1;
```

`Function file for computing exp with my_ode.`

95.8.

```
function y=iteration(x,n)
% iteration - computes iterated squares
%   Syntax:
%           y=iteration(x,n)
%   Arguments:
%           x - a real number, the initial value
%           n - an integer, the number of iterations
%   Returns:
%           y - nx1 matrix containg the iterated numbers
%   Description:
%           The program computes n steps of the iteration y(i+1)=y(i)^2
%           starting with the initial value x.
y(1)=x;
for i=1:n
  y(i+1)=y(i)^2;
end
y=y';
```

The result is `z=[2;4;16]`.

2003-12-08 /stig