Inleiding programmeren en numerieke methoden

Exam

8/2-201~~3~~ 2

For this exam no books or graphical electronic calculators are allowed.
Location: X 5118.-156, Time: 14:00-17:00.

Norm:
The practicum score counts 33% of the final score.
Each of the four parts of the exam counts 16.5% of the final score.
Each subquestion within a part of the exam count equal.

1 MATLAB commands

Briefly explain what the following build-in MATLAB commands do. (Max. 5 lines for each command.) Write if the function is a numerical method command, or a symbolic toolbox command or a simple MATLAB command (such as function, for and if).

1. help
2. format
3. feval
4. end
5. size
6. fsearchmin
7. elseif
8. randperm
9. dsolve
10. \

2 Debugging and Bisection

1. The following program (bisect.m) should use the bisection method to find the root of a function defined in the file *myfun.m*:

```
1. %
2. % myfun.m
3. %
4. % This contains the function that we want to find the root of.
5. function [value]=myfun(x)
6. value==x-3*sin(x.^2);
```

However, the program does not work. Find the 5 bugs and correct them! (Line numbers are indicated on the left. You can use them to report the location of the bugs.) One or more or the errors might be located in the function above.

```
1.  %
2.  % bisect.m
3.  %
4. This function finds the root of a function using the bisection method
5.  function[answer,iflag]=bisect(fun,a,b,tolerance,maxits)
6.
7.  iflag=0;                 % Define control parameter to identify convergence
8.  iterations= 0;           % Initialize number of iteration variable
9.  f_a= feval(fun,a)        % Find initial value at point a
10.  f_b= feval(fun,b)        % Find initial value at point b
11.
12.                          % Loop as long as maximum number of iterations
13.                          % is not reached,the not tolerance met and there
14.                          % is a root in the interval
15. while((f_a*f_b<0) & iterations<maxits) & (b-a)>tolerance
16.    iterations= iterations+1;  % Update number of iterations
17.    c= (b+a)/2;            % Find center point
18.    f_c =feval[fun,c];     % Find value at center point
19.    if f_c*f_a<0           % Identify where the root is and update interval
20.        b=c;f_b =f_c;
21.    else f_b*f_c<0
22.        a=c;f_a =f_c;
23.    else                   % Identify if c is the answer
24.      iflag= 1;answer=c:
25.    end
26.
27. switch iterations       % Determine the reason that the while loop ended
28. case maxits             % Maximum number of iterations were met
29.    iflag= -1; answer=NaN;
30. case 0                  % There is not an odd number of roots in the interval
31.     iflag= -2; answer=NaN;
32.     otherwise         % The root was found in the given number of iterations
33.     iflag= iterations;answer=c;
34. end
```

2. Briefly explain how the bisection method works. Use a drawing for illustration.

3. What would you type in to MATLAB on the command line to find a possible root between -1 and 1 supposing that the debugged program is stored in an m-file?

3 Matrices

1. Write a code to construct the following matrix using <u>one</u> for loop and **not** using the *diag* or *while* commands.

$$\begin{bmatrix} 0 & 1 & \sqrt{2} & \sqrt{3} & \cdots & \sqrt{9} & \sqrt{10} \\ i & 1 & 0 & 0 & \cdots & 0 & 0 \\ \sqrt{-2} & 0 & 2 & 0 & \cdots & 0 & 0 \\ \sqrt{-3} & 0 & 0 & 3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sqrt{-9} & 0 & 0 & 0 & \cdots & 9 & 0 \\ \sqrt{-10} & 0 & 0 & 0 & \cdots & 0 & 10 \end{bmatrix} \tag{1}$$

2. Construct the same matix without using the *for* or *while* commands.
3. What are the commands for diagonalizing (finding the eigen values and eigen vectors) and inverting matrices numerically?

4 Build algorithm

1. Describe in brief the idea of the Simpson's 1/3 method. You are encouraged to make a drawing.
2. Make a step by step outline of an algorithm that use the Simpson's 1/3 method. Do not forget to consider input of integration range and other information from the user.
3. Write a code following your algorithm described in the point above, where you use one while loop. Assume that the name of the function that you want to integrate is called *func*.