Numerical Methods (NUM101) — Coursework 3

This is the third of three courseworks which combine for 50% of the credits for this unit. The maximum number of marks for this coursework is 10, worth 16% of the overall credits. For this coursework you can choose between two questions. Submit only the answer to **one** of the questions. Beware that Question 1 gives only a **maximum mark of 5**. Parameter h of the coursework problem is **personalised** in each of the questions.

Instructions and rules

- Deadline: 17 May 2010.
- Material to be handed in:
  - (**Hardcopy**) A printed document containing the output generated by your script `cw31.m` or `cw32.m` (see below for instructions) and comments (no essay!),
  - (**Hardcopy**) printout of all program codes with comments.
  - (**Victory**) Upload the following `m`-files to the Victory assignment `CW3`.
    * If you choose Question 1:
      upload the function files `TrapezRule.m` and `SimpsonRule.m`, the script `cw31.m`, which has to recover the printed output, and, possibly, other functions that you created and which are needed to run `cw31.m`.
    * If you choose Question 2:
      upload the function files `SimpsonSolve.m`, the script `cw32.m`, which has to recover the printed output, and, possibly, other functions that you created and which are needed to run `cw32.m`.

  The working Matlab files (function and script) account for 70% of the credits. The graph of the hardcopy (see question) accounts for 30%, the remainder of the printout is only there to show the output, the code, and give additional comments.
- Credit for code part of question:

  100% code performs computation correctly and efficiently, is well structured and commented;

  ≥80% code performs computation correctly and efficiently;

  ≥60% code performs computation correctly but has problems[1];

  ≥40% code does not perform computations correctly but could be made to work with minor corrections;

  ≥20% the intentions behind the code are discernible with some effort.
- This is **individual** coursework. If you copy code from other students you will be turned in for plagiarising.
- For questions, clarifications and further help contact:

  Jan Sieber (`jan.sieber@port.ac.uk`, office LG.146).

[1] Examples for problematic code:
- code works correctly most of the time but fails for some valid arguments;
- code "grows" arrays in inner loops or has wrong order of complexity;
- magic numbers spread throughout the code;
- one part of the code is a repetition of another part.

Question 1: Submit either this question or Question 2 **This question gives only 5 marks**

Trapezoidal rule vs Simpson rule Compare the accuracy of trapezoidal rule and Simpson rule for three different functions. In Victory folder `Coursework 3` you find the m-file `cw31.m`, which defines the three functions after you insert your student ID (six-digit number). You can use this file as the start of your script.

(a) Write two functions, `TrapezRule` and `SimpsonRule` that integrate an function f over an arbitrary interval $[a, b]$ using N grid points. The first line of `TrapezRule.m` (integration using composite trapezoidal rule) should look like this:

```
function y=TrapezRule(f,a,b,N)
```

and the first line of `SimpsonRule.m` (integration using composite Simpson rule) should look like this:

```
function y=SimpsonRule(f,a,b,N)
```

Inputs:

- `f`: user-supplied function f, the integrand;
- `a,b`: lower and upper boundary of the integration interval;
- `N`: number of grid points.

Output:

- `y`: (approximated) value of $\int_a^b f(x)\mathrm{d}x$.

[3 marks]

(b) Write a script `cw31.m` which compares the accuracy of the Simpson rule and the trapezoidal rule for numbers of grid points varying between $N = 5$ and $N = 301$ (in steps of 2 such that N is always odd). For comparison we choose the integrals

$$(1)\ \int_0^{2\pi} \mathrm{e}^{x/h}\mathrm{d}x \qquad (2)\ \int_0^{2\pi} \mathrm{e}^{h\sin^3 x}\mathrm{d}x \qquad (3)\ \int_0^{2\pi} \sin\left(\mathrm{e}^{-x/h}\right)\mathrm{d}x$$

where h is your student ID number times 10^{-5} (see start of script `cw31.m` on Victory). Plot the error that both methods give for each function versus $\delta = (b-a)/(N-1)$ (the step-size) on a $\log_{10}$-$\log_{10}$ scale to show which method is more accurate for a given number . [2 marks]

Total for Question 1: 5 marks

Hints:

- If you do not find an analytical expression for the integral to obtain the error you have to find an estimate (for example by comparing the result for N grid points with the result on a much finer grid).
- Matlab has a built-in plotting function `loglog`, which helps with plotting on a logarithmic scale.

Question 2: Submit either this question or Question 1

Solving an integral equation Consider the equation for $x(t)$

$$x(t) = r(t) + \int_a^b K(t,s)x(s)\,\mathrm{d}s, \tag{1}$$

where the functions $K(t,s)$ and $r(t)$ are given functions taking arguments in the interval $[a,b]$, and $x(t)$ is the unknown function. Find an approximate solution $x(t)$ by following the steps listed below.

(a) Write a function `SimpsonSolve` that solves equations of the type (1) approximately for any given functions $K(t,s)$ and $r(t)$ and on any interval $[a,b]$ by replacing the integral in (1) with the Simpson formula. The first line of `SimpsonSolve.m` should be:

```
function x=SimpsonSolve(K,r,a,b,N)
```

Inputs:

- `K` (the *kernel*): user-supplied function K, the factor in front of x in the integral in (1). The function K takes two arguments, both in the interval $[a,b]$ (that is, `K` is of the type `function y=K(t,s)`).
- `r` (the *inhomogeneity*): user supplied function r, the additional term in the right-hand-side of (1). The function r takes a single argument from the interval $[a,b]$ (that is, `r` is of the type `function y=r(t)`).
- `a,b`: lower and upper boundary of the integration interval;
- `N`: (odd) number of grid points used when approximating the integral in (1) with the Simpson formula.

Output:

- `x`: a function that takes a single argument from $[a,b]$, the approximate solution of (1). After the call `x=SimpsonSolve(K,r,a,b,N)` it should be possible to plot `x` by saying

```
t=a:0.001:b;
plot(t,x(t));
```

[7 marks]

(b) The script `cw32.m` and the function file `rfunc.m` on Victory provide an example for the functions `K(t,s)` and `r(t)` (constructed from `rfunc` and your student ID number) on the interval $[a,b]=[0,1]$. Plot your solution $x_{\mathrm{ref}}(t)$ for this example using a large N (say, $N=2000$). [1 mark]

(c) Estimate the error $d=\max_{t\in[0,1]}|x(t)-x_{\mathrm{ref}}(t)|$ of the solution x for grid sizes N between 11 and 801 (in steps of 10) by comparing it to your reference solution on the fine mesh obtained in part (b). Plot the error d versus $\delta=(b-a)/(N-1)$ (the step-size) on a $\log_{10}$-$\log_{10}$ scale. The error should have the form

$$|d| = C\delta^p$$

where $p>0$ is an integer. Estimate p and C from your results for `N=11:10:801`. [2 marks]

Total for Question 2: 10 marks

Question 2 continues overleaf

Hints:

- Solution strategy:
  - replace the integral with the Simpson formula on a grid $a = t_1 < t_2, \ldots < t_N = b$ ($t_{k+1} = a + k\delta$);
  - write down the resulting equation on paper for each t_k (that's N equations);
  - replace $x(t_k)$ by x_k. The x_k are unknowns (that's N unknowns);
  - this gives a linear system of N equations for N unknowns (x_k). Write this system in the form $Ax = b$ and solve this system with Matlab: assemble a matrix `A`, the right-hand-side `b`, then `x=A\b;`;
  - create a function from the vector `x` by using Matlab's `interp1` and `ppval` functions.
- Matlab has a built-in plotting function `loglog`, which helps with plotting on a logarithmic scale.
- Matlab has simple statistical functions that help you to fit the C and p from your results (for example, `polyfit`)