Math 4610 Fundamentals of Numerical Analysis Tasksheet 3

The problems for the Tasksheet 03 are included below. The deadline for turning in your work on these problems will be posted on the repository. In addition, you will turn in your work through the math4610 repository. A directory will be constructed that will be used as a place to store your work.

Tasks

Task 1: Computationally verify that the central difference approximation for the second derivative approximation defined in Task Sheet 2, Task 5 is actually second order accurate. Using values of $h \neq 0$, compute the approximation and the difference between the exact value and the approximation. Print out the data in columns like those presented in the notes for the first order approximation of the the first derivative.

Task 2: Produce a log-log plot of the error from the data in Task 1. Discuss how the results show that the approximation is second order accurate over a range of value where $h > 0$. Also, determine when the approximation begins to fail due to finite precision in the number representation and/or finite precision of arithmetic. You will need to write a code that produces the plot. If needed, you can use the Python code in the notes.

Task 3: Write routines that will produce the machine epsilon as discussed in class. Write two separate routines, one for the single precision setting on your computer and one for double precision. Compile and test each of of these routines separately. This means writing a main() program in separate files.

Task 4: In this task you will be asked to begin a software manual for the various routines that you will be creating over the semester. The software manual will span the entire semester and should include a lot of your work. The following steps should help organize your work.

Within your terminal, create folders to include the software pages in the math4610 repository. Create a folder using something like

```
koebbe% mkdir software_manual
```

Then create a file that will contain a table of contents for pages that are created for individual routines. Maybe something like

```
koebbe% cd software_manual
koebbe% vim software_manual_toc.md
```

When you edit the file, create a unique link for each new software page added. There is a Markdown template that provides an example of what is expected. You can make a copy of the templacte file in the folder created above. Then edit the file to create a page for your code.
For example, create a page for the single precison version of the code from Task 3.

```
koebbe% cp softwareManual/softwareManualTemplate.md smaceps.md
```

or

```
koebbe% cp softwareManual/softwareManualTemplate.md smaceps.html
koebbe% vim smaceps.html
```

depending on whether or not you want to use MathJax for math notation.

Build the links to the individual pages in the table of contents file by adding an entry to the table for each routine.

Build the link from your README.md repository file at the top level. That is, add a line like

```
[Software Manual Table of Contents:](./software_manual/software_manual_toc.md)
```

There are a lot of ways to do this. The steps above should provide the structure you need. You might consider building a template for the pages from the page you can download and make copies from the template.

Task 5: This task will walk you through the development of a shared library for your routines from the two routines written and tested in this tasksheet. Suppose that the two routines are named, smaceps() and dmaceps().

Create two files that contain only the code for the routines for the machine epsilon computation. No main() method is needed.

Compile each of the files using an object name. For example

```
koebbe% gcc -c smaceps.c
koebbe% gcc -c dmaceps.c
```

or

```
koebbe% gcc -c *.c
```

This will produce two files:

```
koebbe% ls *.o
dmaceps.o  smaceps.o
koebbe%
```

Created a shared library,

```
koebbe% ar rcv mylib.a dmaceps.o smaceps.o
```

This is the archive command in Unix and the " cvf" informs the archive command to create, be verbose, and use the file name specified for the library. The output will be a file of the form

```
koebbe% ls mylib.a
mylib.a
```

Make the library a random access library.

```
koebbe% ranlib mylib.a
```

This last step allows a link/load command to access the routines in any order. You can check the result using the following command.

```
koebbe% ar tv mylib.a
```

This gives a table of contents for the archive/shared library.

Task 6: Search the internet for discussions of shared libraries. Write a brief summary of what you find including the pros and cons of shared libraries. Your write up should be a brief paragraph (3 or 4 sentences) that describe your findings. Include links to the sites you cite.

Previous Table of Contents — Next
