P3800 Project 1: Numerical Differentiation and Integration

Your Friendly Neighbourhood Professors

January 2024

1 Latex YES!

Just by getting to the point of being able to compile this document, you've come quite a way! The project reports for this course are to be typeset in LaTeX. It's easy to make mistakes and things can get pretty frustrating, especially in the beginning. But it's good for you! Most scientific writing for journal submissions (and books) is done using Latex. And once you get the hang of it, writing out equations is much faster than using other menu-driven typesetting programs. You can quickly find documentation on the web for Latex. One useful website is

`http://en.wikibooks.org/wiki/LaTeX`

2 Makefiles

It may be easy to keep track of compiling simple programs, e.g. with

```
gcc myprogram.c -o myprogram
```

or

```
gfortran myprogram.f90 -o myprogram
```

but once there are many source files using different libraries and the compilation line gets a little busier, it helps to use makefiles to keep track of everything. The makefile also keeps track of what needs to be updated so that not all the source code needs to be compiled every time you make a change in one source file.

You can read more about the command **make** and makefiles at

`http://www.gnu.org/software/make/manual/make.html`

or by typing

```
man make
```

Also, having a makefile makes it easier for others (like people marking your assignments) to compile your code. For example, there is a file called *Makefile* in this directory that governs how a piece of code called *differentiate.f90* is compiled by the *make* utility. Just type **make differentiate** and the utility will use the contents of *Makefile* to produce the executable *differentiate*. To run the compiled program, just type **./differentiate**. If all is well, running the code should produce a file called a file called *diff_results.dat*.

3 Numerical Differentiation

3.1 Differentiating a known function

The program *differentiate.f90* calculates the derivative of $\sin(x)$ at $x = 1$ rad, using a forward difference and a central difference. The program outputs the absolute error |numerical result − exact| of the calculations as a function of step size h to a file called *diff_results.dat*. The results are graphed in Fig. 1. The graphic was produced using a program called **Grace**, freely available for Unix-like systems.

`http://plasma-gate.weizmann.ac.il/Grace/`

You can recreate most of the the plot by typing

```
xmgrace -log xy -nxy diff_results.dat
```

The `-nxy` tells **Grace** to use the first column in the data file as the x values (horizontal axis), and all the other columns as different sets of y values. Otherwise, only the first two columns are plotted (the first on the horizontal, the second on the vertical). The `-log xy` tells **Grace** to use logarthmic scaling for the x and y axes. By double clicking on the graph in the **Grace** window, you will get a menu of options for plotting symbols. By double clicking near an axis, you can add a title to that axis, play with tick marks and labels, etc. You can also access these features through the various menus. To generate an `eps` (encapsulated postscript) file used by Latex for making figures, select `Print setup` from the `File` menu. There, choose `eps` from the `Device` list and choose an appropriate name for the file (default is usually okay). Then, when you print (`Print` from the `File` menu), an `eps` file will be generated.

You can convert an eps file to a pdf file, which is often friendlier, say, as an email attachment, with the utility `epstopdf`

```
epstopdf diff_results.eps
```

You should also do this if you are compiling with `pdflatex` instead of `latex`, although `pdflatex` may do the conversion for you.

Programs like **Grace** are called what-you-see-is-what-you-get (WYSIWYG). You edit the look of the end result pretty directly. This is pretty convenient at

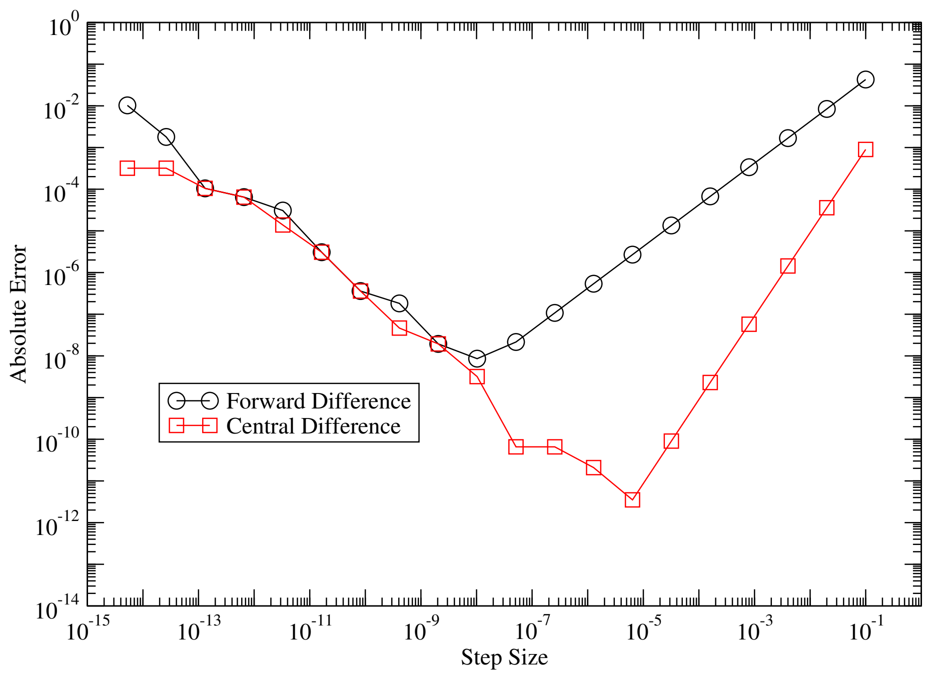

Figure 1: Absolute error as a function of step size h for forward and central difference method in calculating the derivative of $\sin x$ at $x = 1^r$. Note the minimum in the error. Also note the slopes of the curves on the high side of the minimum. The $O(h)$ method (forward difference, circles) has slope one in logarthmic units, while the $O(h^2)$ method (central difference, squares) has slope two.

times. However, it is often more efficient to have a script that generates your graph, as when using the PyPlot package in Python, particularly when you need to update a plot when the data have changed. While **Grace** provides scripting capability, we will not explore it further. We will use **Grace** for this project just to introduce you to one light-weight plotting program. Included in the project directory is the Python script `plot_diff_results.py` that generates a plot similar to the one produced by `xmgrace`. I found that some features of the plot were tricky to implement, but once implemented, easy to reuse.

TASK #1: Modify *differentiate.f* so that it calculates the derivative of $\sin(x)$ at $x = 1$ rad using the fourth order method shown in class

$$f'(x) = \frac{1}{12h}\left(f[x-2h] - 8f[x-h] + 8f[x+h] - f[x+2h]\right) + \mathcal{O}(h^4).$$

Plot the results together with those already shown in Fig. 1. In the main body of your report, **briefly** discuss your results. In particular, comment on the minimum errors attained for the different methods, and the values of h at which they occur. Do the results match theoretical estimates obtained in class?

To do (5 subtasks):

- Modify *differentiate.f* so that the results of the fourth order method are reported by the program.
- Plot the fourth order results along with what is already shown in Fig. 1. Make sure to edit the figure caption.
- In the figure, include a power-law fit to the appropriate part of the fourth-order results. It should be a straight line with a particular slope.
- Comment on what the slope should be theoretically. Does this match the numerical result?
- What are the minimal errors for each of the three methods? Do they match theoretical predictions? Show your work for the fourth order method.

3.2 Note on plots

Plots are crucial to writing reports and papers. Here are a few requirements.

- All plots should have captions. Captions describe what each curve is, and may briefly state conclusions.
- All plots should be referenced in the main text. Make sure to describe what's going on in the plots in the main text.
- Do not include a title; the equivalent information should be given in the caption.
- Include a legend if there is more than one curve.
- Graphs should be framed, as in Fig. 1.
- ALL TEXT SHOULD BE LARGE ENOUGH TO BE READ EASILY... BY OLD PEOPLE! (like me)
- Axes labels should indicate units (in brackets) unless the quantity is dimensionless.
- Use appropriate tick labels and tick marks so that the reader can reasonably estimate values on a curve.

3.3 Differentiating data

In the project directory, you should also find a file called *experiment.dat*. Column 1 is the temperature T (on the Kelvin scale), while column 2 reports a dimensionless quantity s related to the sound velocity in the compound $\mathrm{UNi_2Si_2}$. Phase transitions are marked by sudden changes in the derivative of s with respect to T. The data are not equally spaced in T.

In general, for a function $f(x)$ sampled at a set of (unequally) spaced points, a second-order formula for the derivative is

$$f'_i = \frac{h_{i-1}^2 f_{i+1} + (h_i^2 - h_{i-1}^2) f_i - h_i^2 f_{i-1}}{h_i h_{i-1}(h_i + h_{i-1})} + O(h^2),$$

where $f_i = f(x_i)$, $h_i = x_{i+1} - x_i$ and h is the larger of h_i and h_{i-1}.

TASK #2 Write a program that uses the above formula to calculate the slope $\frac{ds}{dT}$. Plot your result. Ignoring the initial noise at T near zero, how many phase transitions do you see? Make sure your plot focusses on the important features of the data.

4 Numerical Integration

In the study of blackbody radiation, there is a result, Stefan's Law, that states that the power emitted from a perfect blackbody per unit area is

$$j = \sigma T^4,$$

where T is the absolute temperature and σ is a constant that can be expressed as,

$$\sigma = \frac{2\pi k^4}{c^2 h^3} \int_0^\infty \mathrm{d}u \frac{u^3}{e^u - 1}.$$

It turns out that the integral in the above expression can be done analytically, and has a value of $\pi^4/15$.

TASK #3: As an exercise in numerical integration, calculate the integral

$$\int_0^\infty \mathrm{d}u \frac{u^3}{e^u - 1},$$

using Gauss-Laguerre integration (see handout from Klein and Godunov). The parameters for $N = 2, 4, 8, 16 \ldots 32$ are provided in the files labelled *weights.dat_N*. These files were obtained from the website quoted in the handout. Check the website to see what the columns of the files are. Report the absolute value of the relative error, (exact - num val)/exact, as a function of N in a table. Look at Table 1 as a template. **Briefly** discuss the results. Note that the parameters are given to 12 digits of precision, while a double precision real number generally has 14 decimal digits of precision.

Order	value of integral	relative error
2	818	h
4	2.319556	e
8	704	l
16	46	l
20	0	o
24	0	o
28	0	o
32	35	o

Table 1: Sample table. Use this table as a template to report the values of the integral to be done using Gauss-Laguerre integration, as well as the absolute relative error.

The program *integrate.f90* provides a start. Compile the code by typing **make integrate**. Examine the program source code as well as the script *run_integrate.sh* to figure out how you might obtain the required table of values. The shell script *run_integrate2.sh* is an alternate script that accomplishes the same task but in a slightly different way. To run a script, just type **./run_integrate2.sh**. When writing your own script, make sure to change permissions to make it executable.

TASK #4: Write your own program to evaluate the integral using Simpson's rule. When writing your code, define a separate function for your integrand. This will enable you and others to easily modify your code to integrate some other function. One difficulty in using Simpson's rule is that you can not integrate out to infinity, but rather must choose an upper limit to the integral. The value of the integral should be insensitive to within desired precision to the upper limit of integration, i.e., once the upper limit is large enough, making it larger won't change the value of your answer. Part of your task, therefore, is to explore how using Simpson's rule for the integral at hand depends on step size and this upper limit of integration.

Concretely, carry out the integral

$$\int_0^x \mathrm{d}u \frac{u^3}{e^u - 1},$$

using $n = 11$, 101, 1001, $10^4 + 1$, $10^5 + 1$, $10^6 + 1$, and $10^7 + 1$ points at which to evaluate the integrand (remembering that Simpson's rule requires an odd number of points), for $x = 10$, 100, 1000 and 10000. Your program, or program/script combination, should produce four data files, each containing absolute relative error as a function of n for a given x. Using **grace**, plot the four curves use logarithmic axes. **Briefly** discuss your results and compare them to the Gauss-Laguerre method.

5 YOUR SUBMISSION

You need to submit a pdf of your report to D2L that includes graphs, tables and discussion related to each task. All tasks carry the same weight for marking purposes.

Please submit all relevant source code, scripts, latex, eps and pdf files electronically as shown in class. We should be able to simply type **latex project1.tex** to reproduce your report. **Please write and include a script called** ***run_project1.sh*****, which when run should compile and run code and output any relevant data (not necessarily all possible output!) to the screen.** Use the *run_project1.sh* provided as a template if you wish. The idea is that by running *run_project1.sh*, we should quickly see that all your code compiles, that it produces meaningful output, and whether your numbers are right.