Worksheet 2 Report

Mike Knee

Issue: 1

Date: October 28, 2016

School of Physics and Astronomy
University of Birmingham
Birmingham, B15 2TT

Contents

1 Trapezium and Simpsons Method

Two numerical methods, the trapezium method and then Simpson's rule, were used to evaluate the integral

$$\int_0^2 e^{-x} \sin x dx$$

This integral can be evaluated analytically. Which was done first in order to have some reference value which the results were compared to. Integration by parts was used to solve this integral

$$I = \int_0^2 e^{-x} \sin x dx$$

$$I = \left[-e^{-x} \cos x\right]_0^2 + \int_0^2 e^{-x} \cos x dx$$

applying integration by parts again

$$I = \left[-e^{-x} \cos x\right]_0^2 + \left[-e^{-x} \sin x\right]_0^2 - \int_0^2 e^{-x} \sin x dx$$

$$I = \left[-e^{-x} \cos x\right]_0^2 + \left[-e^{-x} \sin x\right]_0^2 - I$$

$$2I = \left[-e^{-x} \cos x\right]_0^2 + \left[-e^{-x} \sin x\right]_0^2$$

$$2I = 1 - e^{-2}(\cos 2 + \sin 2)$$

Which gives the value of the integral.

Figure 1 shows a plot of $e^{-x} \sin x$ against x.

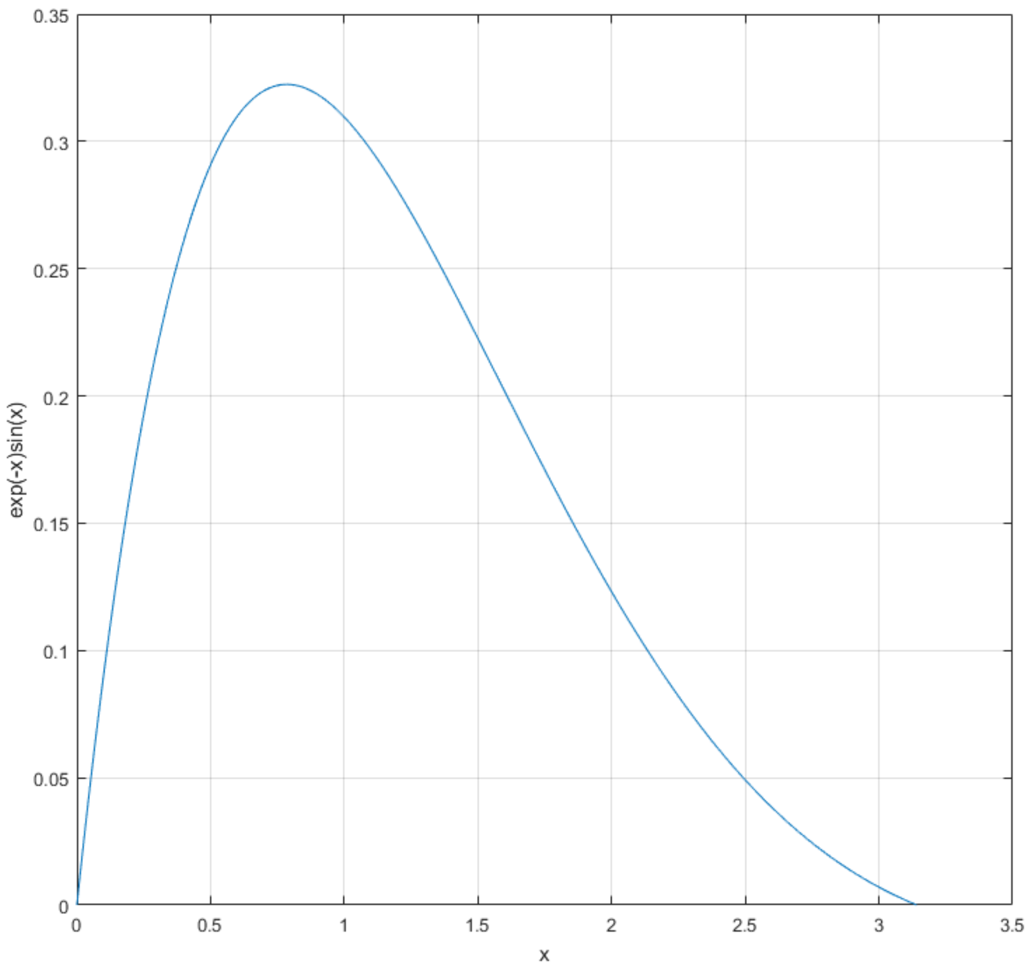

Figure 1: Plot of $f(x) = e^{-x} \sin x$, which we want to find the integral of.

1.1 Trapezium Method

Running either of the first two options in the program compiled from "question2.cpp" uses the trapezium method to calculate an estimation of our integral. The program will also output an analytic solution to the problem before any numerical calculations are made for comparison.

The error in the trapezium method result can be derived by taking the Newton-Cotes formula from [2] and noticing that the error for one slice is then given by the second part, ie.

$$\epsilon_s = \frac{1}{12} h^3 f''(\eta)$$

Where h is the width of the slice, and η maximises the value of f''. From this it is clear that the error for a single slice is proportional to the width of the slice cubed

$$\epsilon_s \propto h^3$$

So the error for the complete integral estimation will be n (the number of slices) times this

$$\epsilon_T \propto h^3 n$$

The width of each slice is given by

$$h = \frac{u - l}{n}$$

where u and l are the upper and lower bounds respectively. From this the width of each slice is proportional to $1/n$ and so

$$\epsilon_T \propto \frac{1}{n^2}$$

A log graph showing the relative error decreasing as the number of intervals used increases is shown in figure 2. The data for this plot was produced using option (2) in the program. It is clear that the graph does agree with our derived formula for the error, as at each interval the relative error is proportional to $1/n^2$ where n is the number of intervals. However, as the relative error decreases to around 10^{-13} the method starts to fail, and the error no longer decreases in the manner expected. This is because when a very high number of intervals is used the value for each slice becomes very small. This means the floating point error obtained by summing these very small numbers becomes more significant, and the fluctuations observed in the graph are produced.

1.2 Simpson's Method

Running any of the final three options in the compiled program will use the Simpson's method to calculate an estimate of the integral. As before the analytic solution is printed out beforehand for comparison.

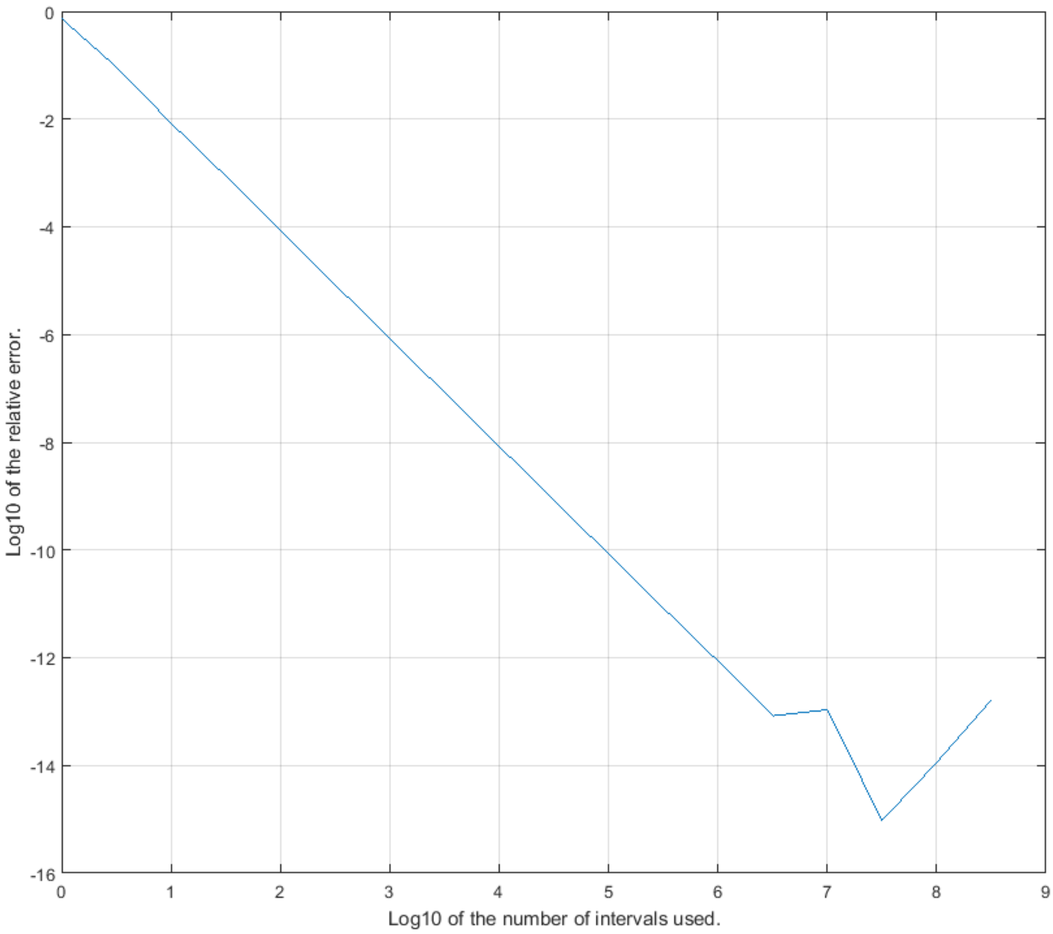

Figure 2: Log plot showing how the error decreases with the number of intervals used in the trapezium method.

The error in the Simpson's method can be shown analytically in a similar manner to the trapezium method. The formula for the remainder given in [1] is

$$\epsilon_s = \frac{1}{90} h^5 f^{(4)}(\eta)$$

Where as above h is the slice width and η is the value that gives the highest value for $f^{(4)}$ for that slice. Using the same line of reasoning as above it can be shown that the error for the total estimate is

$$\epsilon_T \propto \frac{1}{n^4}$$

when the Simpson's method is used. This means the error will diverge much more quickly than the Trapezium rule, and vastly less iterations are needed to produce a comparable level of precision.

Option (4) in the program will allow the user to specify a number of significant figures. In order to check the accuracy of the result the calculated result is compared to the analytic result at each iteration, we then stop the program and output the answer once the difference between the actual answer and the estimated answer is less than desired significant figures.

Similar to the one for the trapezium method a log graph of the relative error against the number of intervals used is shown in 3. It is immediately clear that the error converges much quicker than that of the trapezium rule, with the maximum possible precision (as dictated by the size of double variables) being reached fair quicker. Again this agrees with our analytic error analysis as we expected the error to be proportional to n^{-4} instead of n^{-2} as in the trapezium rule. It is worth noting that the relative error of value for 10^4 intervals is not generated properly. This is likely due to the relative error being a number with significant figures smaller than can be stored in a double variable, so it is rounded to 0. This means when the log is taken an undefined result is given, and the program returns -inf instead of a number.

After using a number of intervals greater than 10^4 the relative error begins to fluctuate and increase again. This is due to similar reasons as for the trapezium rule. When high numbers of intervals are used the floating point error of adding the estimate for each slice becomes significant and produces the fluctuations and increase in error that is observed.

The error increases again as we use a much higher number of intervals. This is because.

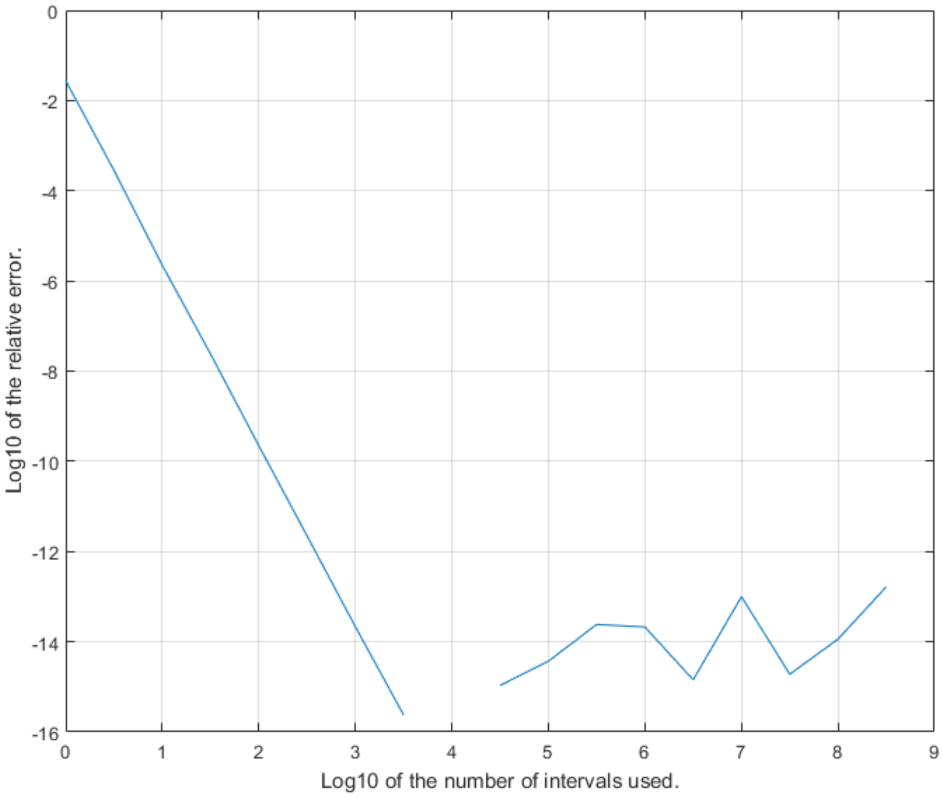

Figure 3: Log plot showing the relative error decreasing as the number of intervals used increases.

2 Gnu Scientific Library

2.1 Basic Integration

For this part routines from the Gnu Scientific Library (GSL) were used to estimate the integral from before. The source code for this is given in "question6.cpp", and uses the QAG adaptive integration routine from the GSL.

The output of the program shows the result, error and size of the workspace used by the routine. The size of this workspace corresponds to the number of times the routine had to use the algorithm. As the 41 point Gauss-Kronrod rules are used the function is called 41 times, to evaluate the function at each of the Kronrod points (see reference [3], chapters 4.6 and 4.7). This is hugely less than the trapezium or Simpson rules, which calls the function 2 and 3 times respectively for each slice. To get a high accuracy in those rules the function must be called a very large number of times, in the order 10^7, whereas in the GSL routine a high accuracy is achieved with only 41 calls.

2.2 Oscillatory Integration

Next we consider the value of the integral

$$\int_0^{2\pi} x \sin 30x \cos x$$

Figure 4 shows a plot of $f(x) = x \sin 30x \cos x$. The problem with this function is that it is oscillatory, which makes it more difficult for a computer to estimate numerically. Luckily the GSL provides us with a routine specifically designed for estimating integrals of oscillatory functions, called "gsl_integration_qawo". The source code "question7.cpp" uses this routine to estimate the value of the integral above.

The result and error estimate are printed when the program is run, with the result being $-0.20967...$ (run program for output to full 15 signifcant figures) and the error estimate being 3.505×10^{-9}. The size of workspace used (shown under intervals in the output) is 32, meaning the algorithm has run multiple times in order to produce our desired accuracy.

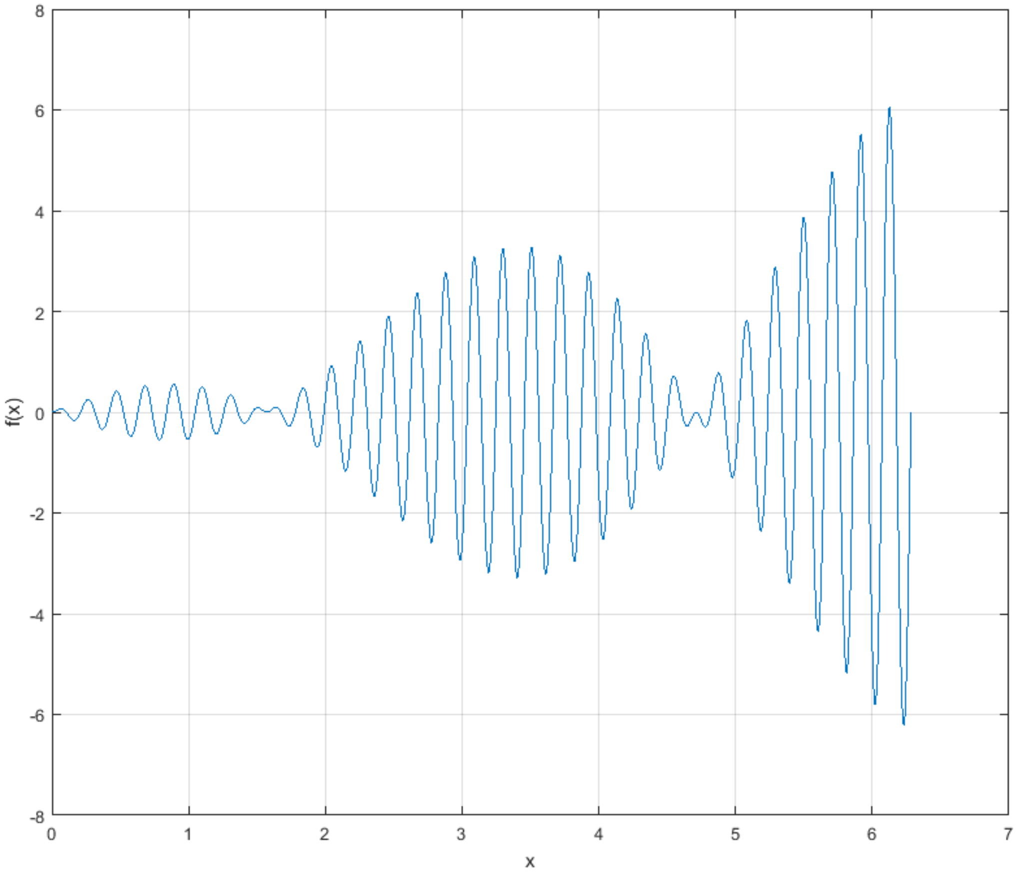

Figure 4: Plot of $f(x)$ against x, where $f(x) = x \sin 30x \cos x$.

References

[1] Eric W. Weisstein. *Simpson's Rule.* Visited on: 27/10/2016. URL: http://mathworld.wolfram.com/TrapezoidalRule.html.

[2] Eric W. Weisstein. *Trapezoidal Rule.* Visited on: 27/10/2016. URL: http://mathworld.wolfram.com/TrapezoidalRule.html.

[3] William T. Vetterling Brain P. Flannery Willian H. Press Saul E. Teukolsky. *Numerical Recipes (third edition).* Cambridge University Press. URL: http://apps.nrbook.com/empanel/index.html#.