PHY 607 Project 1

Will Martin

September 30 2025

Contents

1 Introduction

This project aims to compare various numerical methods with analytical results for a second order ordinary differential equation (ODE) and a definite integral to determine the accuracy and performance of those methods.

The ODE used for the method comparison is the standard form of one-dimensional cylindrical heat conduction, simplifying by assuming a steady state, no internal energy generation, and a constant value for thermal conductivity. The specific case used is of a hollow cylinder with specified temperature boundary conditions at the inner and outer walls of the cylinder. The methods used to numerically integral the ODE are the explicit Euler's method, a standard formulation of a fourth order Runge-Kutta method (RK4), and an initial value problem solver from the Python library SciPy. Setup of the ODE and the numerical methods are outlined in the Methods section.

The definite integral used for the method comparison is the equation for lift from an elliptical lift distribution on a finite wing, based on lifting-line theory assuming the flow is incompressible, inviscid, steady, has constant density, and has a uniform freestream. The methods used to numerically solve the definite integral are the "left" Riemann Sum, the trapezoidal rule, Simpson's 1/3 rule, and SciPy methods for the trapezoidal and Simpson's rules. Setup of the integral and the numerical methods are outlined in the Methods section.

2 Methods

2.1 ODE Formulation

The second order ODE for 1D cylindrical heat conduction assuming a steady state, no internal energy generation, and a constant value for thermal conductivity is given by Eq. 1. In this equation, T is temperature and r is radius.

$$\frac{d}{dr}\left(r\frac{dT}{dr}\right) = 0 \tag{1}$$

To solve this ODE, boundary conditions must be specified. There are several choices for boundary condition for a heat conduction problem, but for this case simple specified temperature conditions are used to fix the temperature at the inner (r_1) and outer (r_2) walls of the hollow cylinder. It is required for the cylinder to be hollow in this problem, as an inner radius of zero will cause divide by zero issues. The conditions are given by Eq. 2:

$$T(r_1) = T_1 \qquad T(r_2) = T_2 \qquad r > 0 \tag{2}$$

The exact (analytical) solution to this ODE is given by Eq. 3, the result of which will be compared with the results from the numerical methods.

$$T(r) = \frac{\ln(r/r_1)}{\ln(r_2/r_1)}(T_2 - T_1) + T_1 \tag{3}$$

For the numerical methods, the ODE is converted to state space form (Eq. 4). Due to the nature of the numerical methods, the boundary condition for $T(r_2)$ is dropped and a new one is set for the first derivative at the inner radius, which is given in Eq. 5. This allows the numerical methods to solve the ODE iteratively in outward steps of size Δr.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} T \\ \frac{dT}{dr} \end{bmatrix} \qquad \dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{dT}{dr} \\ -\frac{1}{r}\frac{dT}{dr} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{x_2}{r} \end{bmatrix} \tag{4}$$

$$x_1(r_1) = T_1 \qquad x_2(r_1) = \frac{T_2 - T_1}{r_1 \ln(r_2/r_1)} \tag{5}$$

2.2 Euler's Method

Using the state space form above, the system can be solved using the explicit Euler's method (Eq. 6), where $\Delta t = \Delta r$. This method has the benefit of being very simple, however it comes at the cost of accuracy, which will be discussed further in the Results & Validation section.

$$x_{n+1} = x_n + \dot{x}_n \Delta t \tag{6}$$

2.3 Runge-Kutta Method

Again using the state space form of the ODE, the system can be solved with a fourth order formulation of the Runge-Kutta method, given by Eq. 7, along with intermediate values discussed below. This method has the benefit of being more accurate than Euler's method, often by a large amount, but it more complicated and often more difficult to implement. The accuracy will be discussed more in the Results & Validation section.

$$x_{n+1} = x_n + (k_1 + 2k_2 + 2k_3 + k_4)\frac{\Delta t}{6} \tag{7}$$

The intermediate values (Eq. 8) used above are the values of the first derivative of the system (Eq. 9) taken at different times and locations. These get weighted in Eq. 7 to arrive at that more accurate than Euler's method.

$$\begin{aligned} k_1 &= f(t_n, x_n) \\ k_2 &= f\left(t_n + \tfrac{\Delta t}{2}, x_n + k_1 \tfrac{\Delta t}{2}\right) \\ k_3 &= f\left(t_n + \tfrac{\Delta t}{2}, x_n + k_2 \tfrac{\Delta t}{2}\right) \\ k_4 &= f(t_n + \Delta t, x_n + k_3 \Delta t \end{aligned} \tag{8}$$

$$f(t, x) = \dot{x} \tag{9}$$

2.4 Integral Formulation

The definite integral for lift for a finite wing with an elliptical lift distribution is given by Eq. 10, using lifting-line theory and assuming the flow is incompressible, inviscid, steady, has constant density, and has a uniform freestream. In this equation, L is lift, ρ is density, V is freestream velocity, Γ_0 is vortex strength at $y = 0$, y is spanwise position, and b is the total wingspan.

$$L = \rho V \Gamma_0 \int_{-b/2}^{b/2} \sqrt{1 - \left(\frac{2y}{b}\right)^2}\, dy \tag{10}$$

To obtain the exact solution, the following substitution (Eq. 11) can be made to simplify the integral.

$$y = \frac{b}{2}\cos\theta \tag{11}$$

The exact (analytical) solution to this integral is given by Eq. 12, the result of which will be compared with the results from the numerical methods.

$$L = \rho V \Gamma_0 b \frac{\pi}{4} \tag{12}$$

For the numerical methods, the integrand is extracted from the lift integral, which is given by Eq. 13, and is used in the following numerical methods.

$$f(y) = \rho V \Gamma_0 \sqrt{1 - \left(\frac{2y}{b}\right)^2} \tag{13}$$

2.5 Riemann Sum

Using the integrand from above, a Riemann Sum can be performed to approzimate the lift from the definite integral. Specifically, the "left" Riemann Sum is used, which is given below in Eq. 14.

$$L = \sum_{i=1}^{n} f(y_{i-1})\Delta y_i \tag{14}$$

This method calculates the area under a curve with series of rectangles, with the top left corner of each rectangle coinciding with the curve and the bottom of each rectangle coinciding with the horizontal axis. This is visualized in the Results & Validation section.

2.6 Trapezoidal Rule

To improve upon the Riemann Sum, the rectangle areas under the curve can be modified by connecting the top left corners of adjacent areas, creating a series of trapezoids. This is appropriately called the Trapezoidal Rule, which is given below in Eq. 15. This is once again visualized in the Results & Validation section.

$$L = \sum_{i=1}^{n} \frac{f(y_{i-1}) + f(y_i)}{2}\Delta y_i \tag{15}$$

While approximating the integral better than the Riemann Sum, the Trapezoidal Rule still approximates a non-linear curve with linear segments.

2.7 Simpson's Rule

To better represent the non-linear curve than the Trapezoidal Rule is able to, Simpson's Rule connects points along the curve with quadratic segments. This is given by Eq. 16, which uses quadratic interpolation to obtain a midpoint, with which the quadratic segment can be found.

$$L = \sum_{i=1}^{n} \frac{y_i - y_{i-1}}{6}\left[f(y_{i-1}) + 4f\left(\frac{y_{i-1} + y_i}{2}\right) + f(y_i)\right] \tag{16}$$

This method is not visualized in this report, however many examples are available on websites such as Wikipedia.

3 Results & Validation

3.1 ODE Conditions

For the first ODE case investigated in this report, the boundary conditions chosen are as follows in Eq. 17:

$$\begin{array}{ll} r_1 = 1 & r_2 = 8 \\ T_1 = 10 & T_2 = 20 \end{array} \tag{17}$$

Additionally, a step size for the radius was needed for the numerical solutions, which was chosen to be $\Delta r = 0.5$. For The second ODE case, the same step size was used, but the boundary conditions were changed to those listed in Eq. 18:

$$\begin{array}{ll} r_1 = 1 & r_2 = 15 \\ T_1 = 10 & T_2 = 15 \end{array} \tag{18}$$

For the third ODE case, the boundary conditions were kept the same as case two, but the step size was decreased to $\Delta r = 0.1$.

3.2 ODE Solutions

Using the above conditions in the exact solution and numerical methods, the exact and approximate temperature distributions were found and are plotted in Figures 1-3.

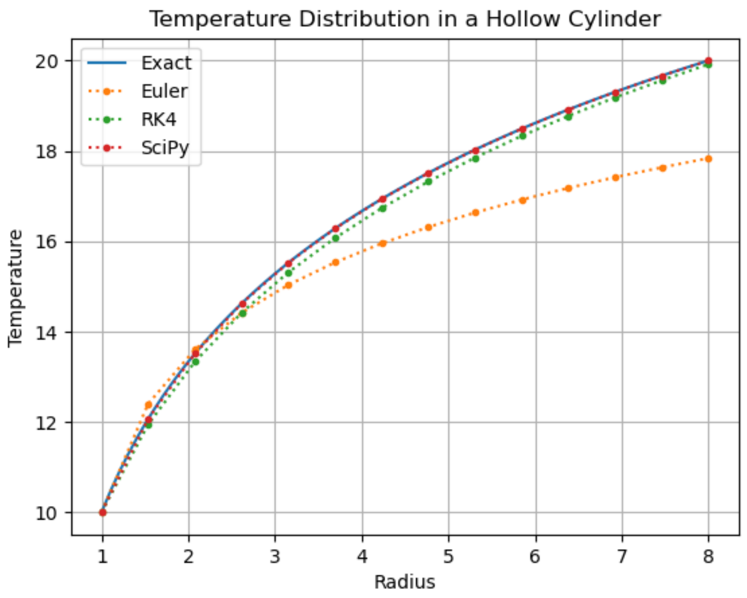

Figure 1: Case 1 Temperature Distributions

In the results from the first case (Fig. 1), the clear outlier is Euler's method, as its temperature at the outer radius is over 20% off from the exact result. Even though the outer radius temperature is too low using Euler's, the temperature at the second radial step is too high. This overshoot highlights the linear approximation of Euler's method and how it can introduce major error into a system. Additionally in this case, it is apparent that even though the RK4 method performs much better than Euler's, its result is still slightly off from the exact solution. However, rather than having a compounding error as the radius increases, the RK4 error actually reduced towards the outer radius. Finally, when looking at the SciPy method results there appears to be no error at all. This is because SciPy integration methods are adaptive based on result error. It will locally decrease step size if the provided step size causes too much error.

In the second case results (Fig. 2), the behavior of Euler's method is mostly the same as the first case. Due to the linear approximation, there is again a large amount of error in the results. As for the RK4 results, the error is much less noticeable, however this is because of the scale of the graph. Upon closer inspection, the RK4 results undershoot prior to $r = 8$ as happened in case 1, but now overshoot after $r = 8$ and end with an outer temperature slightly higher than the exact solution.

For the third case, with the decreased step size, the results (3) show signs of improvement. The Euler's results have much less error than in case 2, less than a 10% error rather than around 20%. Although it is difficult to tell, the RK4 method also improved, no longer noticeably under or overshooting the exact result.

Two key physical behaviors of cylindrical heat conduction are that the temperature profile should be exactly logarithmic, as prescribed in Eq. 3, and heat flux (first derivative of the ODE) decreases as r increases. The first of the behaviors is difficult to verify, however due to the error introduced especially by Euler's method, it is safe to say that the logarithmic shape is not perfectly maintained. As for the second behavior, the first derivative of the ODE is easily visible as it is the slope of the temperature distributions. It is clear that the slope decreases as the radius increases, therefore the second behavior is maintained.

From these numerical results, it is clear that the most accurate way to approximate the exact solution of an ODE is by using an adaptive higher order method. If a low order method like Euler's must be used, decreasing the step size is the best way to improve accuracy of the result.

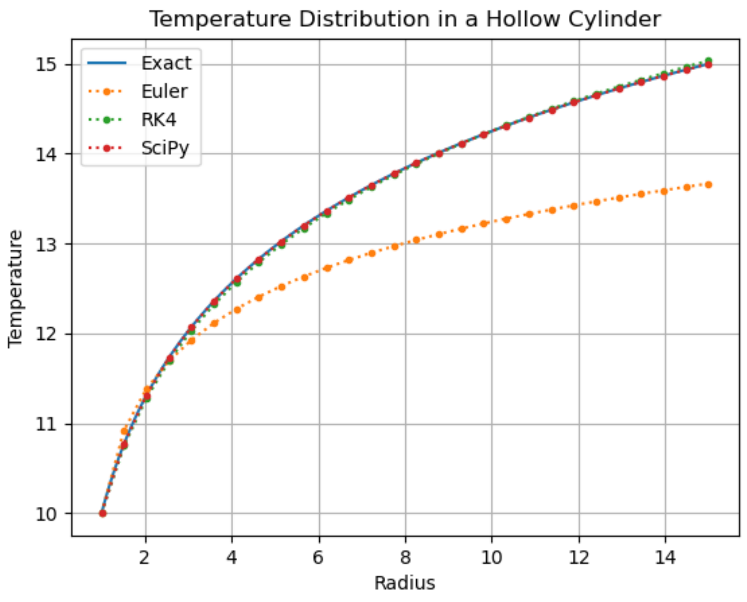

Figure 2: Case 2 Temperature Distributions

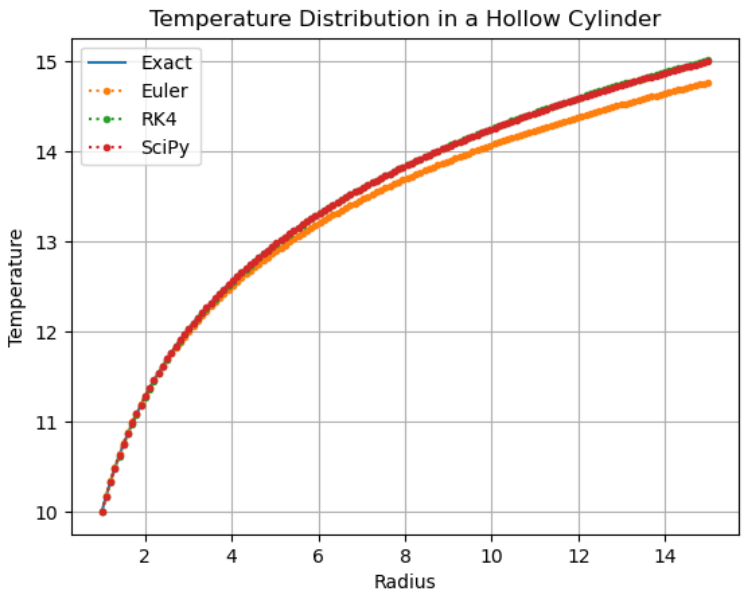

Figure 3: Case 3 Temperature Distributions

3.3 Integral Conditions

For the first integral case, the conditions chosen are given in Eq. 19.

$$\begin{gathered}\rho = 1.225 \\ V = 10 \\ \Gamma_0 = 1 \\ b = 5 \\ \Delta y = 0.5\end{gathered} \tag{19}$$

For the second integral case, the conditions are the same as in case 1 except for the step size, which was changed to $\Delta y = 0.1$.

3.4 Integral Solutions

Visualization of the Riemann Sum and the Trapezoidal Rule is given in Figures 4 and 5, using the conditions for case 1.

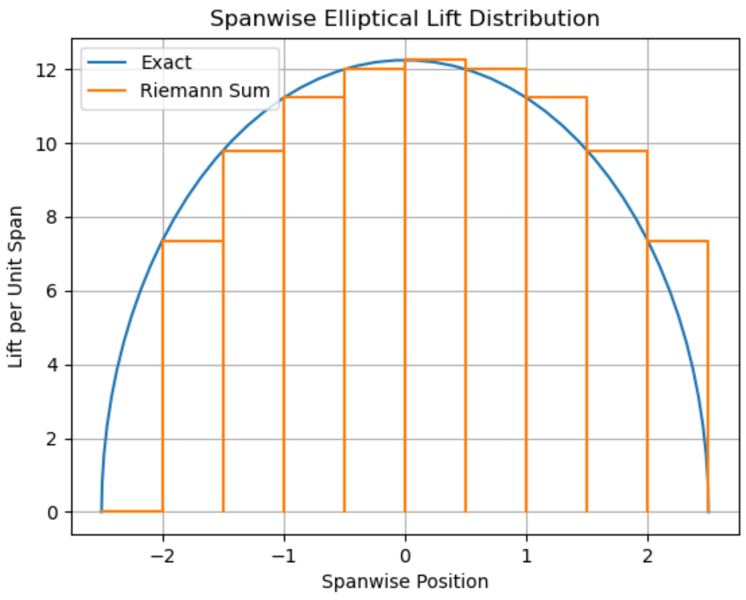

Figure 4: Case 1 Lift Distribution

Upon first inspection of the Riemann Sum and Trapezoidal Rule, the Trapezoidal Rule looks like it should be much more accurate. However, in the case of this integral, they are in fact equal. This is because the integral is symmetric, so any undershoot and overshoot by the Riemann Sum effectively cancels out, resulting in the Trapezoidal Rule. This is also reflected in the values of lift produced by each method, for both case 1 and 2, listed below in Table 1.

Method	Case 1 Lift	Case 2 Lift
Exact	48.1056	48.1056
SciPy Generic Solver	48.1056	48.1056
Riemann Sum	46.5048	47.9618
Trapezoidal Rule	46.5048	47.9617
Trapezoidal Rule (SciPy)	46.5048	47.9617
Simpson's Rule	47.8823	48.0857
Simpson's Rule (SciPy)	47.8823	48.0857

Table 1: Lift Resutls

The results in Table 1 show firstly that the manually implemented methods produce the same results as the corresponding SciPy methods. The also show that equivalence between the Riemann Sum and Trapezoidal Rule for this symmetric integral. Lastly, the results show which methods are more accurate. Specifically, the SciPy generic solver produced an almost perfect result, only differing after 13 decimal places. For the purposes of this example, that is more than close enough to the exact solution. Next in accuracy was Simpsons Rule, then Riemann Sum and Trapezoidal Rule.

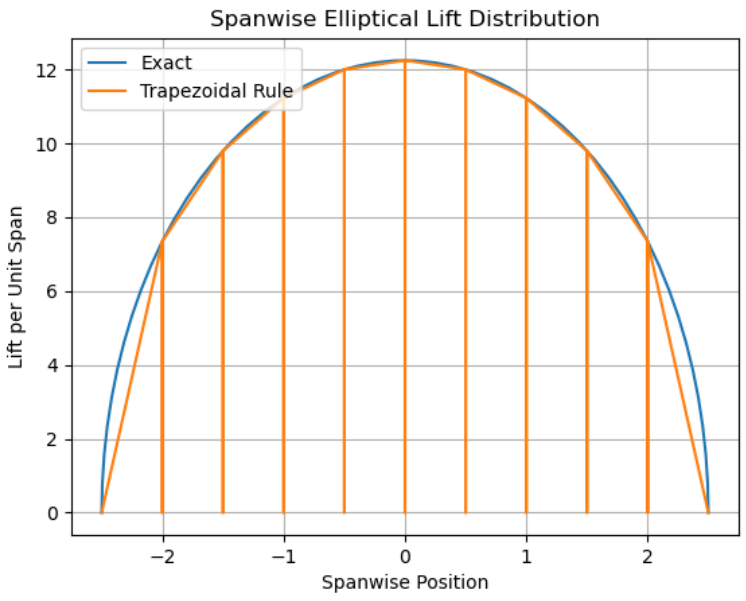

Figure 5: Case 1 Lift Distribution

From these results, it is clear that decreasing the step size produces a more accurate result from numerical methods. Beyond that, Simpson's Rule understandably obtains a more accurate solution, as it more accurately represents the curve of the integrand.

Two straightforward but important physical behaviors of the lift integral is that for zero span there should be zero lift, and for a wing with some nonzero span, the maximum lift should be at the center of the wing and lift should be zero at the ends of the wing. The first behavior can be verified by inspection of Eq. 12, where zero span directly cancels all other parameters, resulting in zero lift. The second behavior can be verified by the visualization in Figure 5, where the maximum lift is at the center of the wing and the ends of the wing have zero lift. On the contrary, this is not preserved by the Riemann Sum in Figure 4. The left end of the wing does have zero lift, but the right does not and the maximum lift is offset slightly to the right. In the case of an odd number of divisions, the maximum lift would be located at the center of the wing, but the right end would still not be zero. For this reason, The accuracy of the Riemann Sum can be considered worse than the Trapezoidal Rule, even for this symmetric integral.

4 Conclusion

The comparisons made for both the ordinary differential equation and the definite integral were key to determining how different numerical solution methods behave and how their accuracies compare.

From the ODE comparisons, it was determined that a pre-made algorithm such as one from the SciPy library for the Python coding language is the most accurate no matter the boundary conditions or step size, due to the adaptive behavior of such algorithms. The fourth order Runge-Kutta method was the next most accurate, not straying far from the exact solution even with large step sizes. With small step sizes, the RK4 result becomes very close to the exact result. Lastly is Euler's method, which struggles to maintain accuracy without very small step sizes. This method should only be the choice for a numerical solution if the function is close to being linear, and a small step size can be used.

For the integral comparisons, it was once again determined that premade algorithms can outperform the simple forms of common numerical methods. The generic definite integral solver from the SciPy library was able to solve the integral exactly down to a more than satisfactory precision, however the other methods were all dependent on step size for their accuracy. The best of the other methods was Simpson's Rule, as it was able to represent segments of the integrand curve with quadratic curves. In the case of this specific integral, the Riemann Sum and Trapezoidal Rule were equally as accurate, however for non-symmetric solutions, Riemann Sum will perform the worst, as it can significantly undershoot or overshoot the integrand curve.