# Solving Ordinary Differential Equations
# Using the Euler and Runge-Kutta Methods

Mark Brautigam
Math 370 – Spring 2025
10 May 2025

## 1. Introduction

Some ordinary differential equations have no explicit solution. In these cases, we rely on numerical methods to give us approximations. The Euler and Runge-Kutta methods are numerical methods for solving initial-value ordinary differential equations.

Differential equations often provide the first or second derivative at a particular point, but not the actual value of the function. If we know the value at one edge of the interval in question, we can use the known derivative(s) to calculate a value a small increment away from the edge value. Then we can continue calculating more values in terms of the values we have already calculated. Since this is an iterative process, error accumulates with each new approximation in the series. Such methods are prone to growing error with more iterations.

The Runge-Kutta methods are actually a family of iterative methods. They were developed around 1900 by Carl Runge and Wilhelm Kutta. Euler proposed his method in 1770. Its error is proportional to the step size $h$. Euler's method was later recognized as belonging to the broader Runge-Kutta family of solutions.

When we say Runge-Kutta method without any qualifiers, we are usually referring to the most widely known member of the family, RK4, which uses a weighted average of four different approximations: the slope at the beginning of the interval, the slope at the end of the interval, and two different approximations of the slope at the midpoint of the interval. The RK4 method is a fourth-order method, meaning the error is on the order of $h^4$, where $h$ is the step size.

One application of numerical methods to differential equations is simulation and analysis of infectious disease data, such as in the recent COVID outbreak (Iskandar, 2022). Another application is found in the film *Hidden Figures*, where Katherine Johnson uses the Euler method to calculate the re-entry of John Glenn from earth orbit (Khan, 2017).

I will analyze the error and convergence of both methods to determine what are the advantages of each.

## 2. Methods

### Euler Method

The Euler method approximates an unknown value using a known value and a derivative or derivatives that lead to the next point in the interval. From the derivative, we know the slope at the current point. From the slope, we can calculate an expected value at the next point.

The Euler method for computing a point $y_{n+1}$ from a known point $y_n$ and a derivative function $f$, is

$$y_{n+1} = y_n + h\, f(t_n, y_n)$$

where $h$ is the step size. Once we've computed $y_1$ from $y_0$, we can continue computing the remainder of the values using the same strategy, computing each term using the previous term. In programming this, we would use a loop. It's probably not practical to use indexing or other non-loop methods to solve for all points, since each point's calculation depends on the previous point's result. Here is some pseudo code for using Euler's method:

```
Define the differential function $f(x, y)$
Define initial conditions ($x_0$ and $y_0$)
Define number of steps $n$
Define maximum value to be evaluated, $x_n$
Calculate the step size: $h$ = ($x_n$ - $x_0$) / $n$
Loop from 0 to n-1
    $y_{i+1} = y_i + h * f(x_0 + i * h,\ y_0)$
    Store the values of $y_i$ for later plot and analysis
```

### Runge-Kutta 4 Method

The Runge-Kutta 4 method approximates the new value at four different points: the left edge of the interval, the right edge of the interval, and two different midpoint approximations. The formulas are:

$$y_{i+1} = y_i + \frac{h}{6}\,(k_1 + 2k_2 + 2k_3 + k_4)$$

$$x_{i+1} = x_i + h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + \frac{h}{2}, y_i + h\frac{k_1}{2})$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + h\frac{k_2}{2})$$

$$k_4 = f(x_i + h, y_i + h\, k_3)$$

In these equations,

- $k_1$ is the slope at the beginning of the interval;
- $k_2$ is the slope at the midpoint of the interval, using $k_1$ as a starting point;
- $k_3$ is the slope at the midpoint of the interval, using $k_2$ as a starting point;
- $k_4$ is the slope at the end of the interval.

The slopes at the midpoint are given greater weight. This is similar to Simpson's Rule for approximating integrals. Here is some pseudo code for using RK4:

```
Define the differential function $f(x, y)$
Define initial conditions ($x_0$ and $y_0$)
Define number of steps $n$
Define maximum value to be evaluated, $x_n$
Calculate the step size: $h = (x_n - x_0) / n$
Loop from 0 to n-1
      k1 = h * f (x, y)
      k2 = h * f (x+h/2, y+k1/2)
      k3 = h * f (x+h/2, y+k2/2)
      k4 = h * f (x+h, y+k3)
      y = y + 1/6 * (k1 + 2*k2 + 2*k3 + k4)
      x = x + h
      Save values of x and y for later plot and analysis
```

I generally used code that I found on the Jupyter web site, but there was one problem I had to work through. The Jupyter code for Euler expected the differential equation function to be of the form $dy/dx = f(y, x)$ and the RK4 code expected the DE function to be of the form $dy/dx = f(x, y)$. I didn't notice this at first and I got one batch of bad approximations. After fixing this problem by switching the order of the coordinates for RK4, the absolute error reduced into the expected range.

Jupyter's RK4 code came in the form of the calculation for just one iteration. Instead of adding the loop into the existing function, I wrote a new function that called Jupyter's function in a loop. My preference is to leave existing good code as is and make use of it by calling it.

I wrote test functions to make sure my copied code was doing the right thing. I did find some bugs. I had the code print out lots of arrays to help pinpoint where any problems might be. The final iterations of the code will print out only plots and the necessary numbers to determine the error and order of convergence.

**Error Bounds and Order of Convergence**

Expected error for the Euler method is on the order of $h^2$ for each step (the local error), where $h$ is the step size, and on the order of $h$ for the outcome after evaluating all steps (the global error), each step which depends on the previous step and is influenced by the error at all previous steps. An exact error bound is hard to pin down, because we may not be able to calculate the necessary maximum value of $f''(t)$ given only a differential equation (Pizer, 1998). We can say that the local error is $k_1 h^2$ where $k_1$ is an unknown constant, and the global error is $k_2 h$ where $k_2$ is an unknown constant. Since the global error is a constant factor times the step size, we expect the grid order of convergence to be linear, or near 1.

The RK4 method is a fourth-order method. The error per step (local error) is on the order of $h^5$, where $h$ is the step size (Singh, 2018). The error is on the order of $h^4$ after evaluating all steps (the global error), each depending on all previous steps. So we expect the grid order of convergence to be 4.

**Evaluating the Methods**

To evaluate the effectiveness of the Euler method and the RK4 method, I looked at approximations of a simple differential equation. The equation is a population equation,

$$\frac{dp}{dt} = kp$$

where the constant $k = 0.5$ and the initial population $p_0 = 2$.

For purpose of error checking, I computed that the actual solution to this equation is

$$p = p_0 \, e^{kt}$$

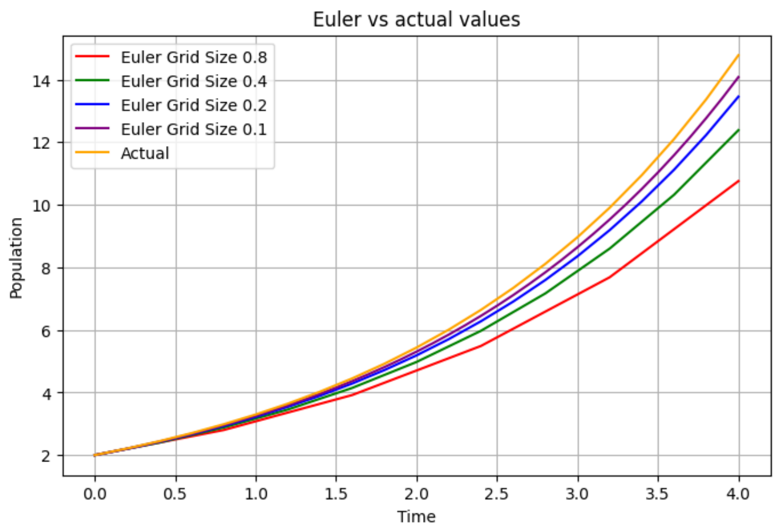
In general, we use a numerical method to solve these differential equations because we don't know the actual solution. But in this case, we use a differential equation for which we know the solution, so we can evaluate the performance of the numerical methods.

## 3. Results

The programming and results are found in the Colab Notebook called *Mark_370_Final.ipynb.*

### Euler error and grid convergence

I used step sizes of 0.8, 0.4, 0.2, 0.1, and for a final analysis, 0.05.



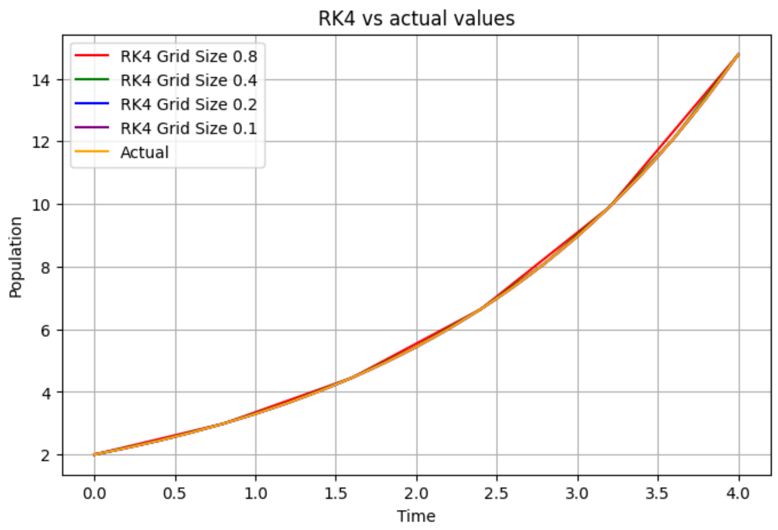
The above plot of the Euler approximations clearly shows that the error is proportional to step size. The step size changes by half for each successive approximation, and the error also changes by about half. The error and order of convergence numbers look like this:

| Step size h | Max error | 2nd step size | 2nd error | Order of convergence |
|---|---|---|---|---|
| 0.8 | 4.022 | 0.4 | 2.395 | 0.748 |
| 0.4 | 2.395 | 0.2 | 1.323 | 0.856 |
| 0.2 | 1.323 | 0.1 | 0.698 | 0.922 |
| 0.1 | 0.698 | 0.05 | 0.359 | 0.960 |

The table shows that the error is roughly halved when the step size is halved. The order of convergence seems to approach 1 as step size decreases, as predicted. I added one more approximation at step size $h = 0.05$ in order to see the order of convergence get even closer to 1.

**RK4 error and grid convergence**

I used step sizes of 0.8, 0.4, 0.2, 0.1, and for a final analysis, 0.05.

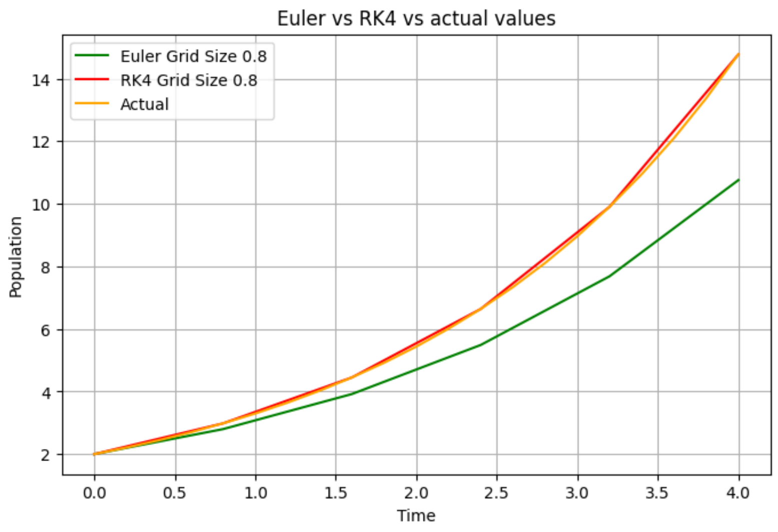


The above plot of the RK4 approximations clearly shows that while the error is related to step size, the error is so small it's hard to see on the plot, at even larger step sizes. The error and order of convergence numbers look like this:

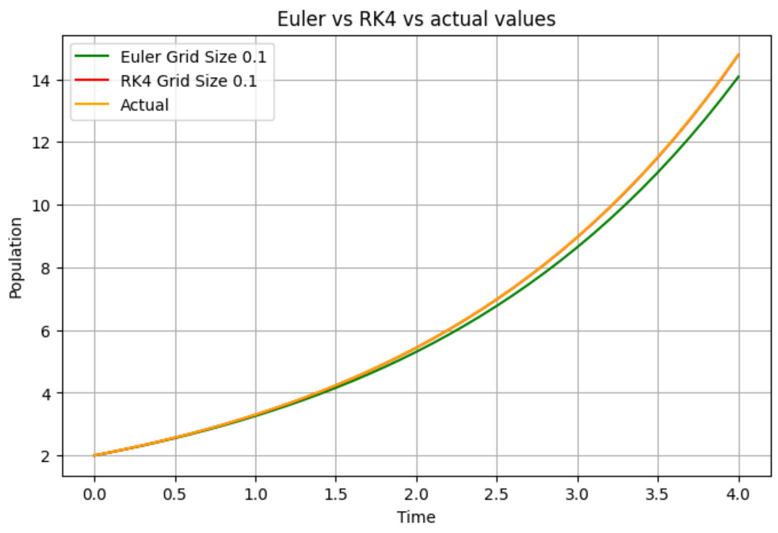
| Step size h | Max error | 2nd step size | 2nd error | Order of convergence |
|---|---|---|---|---|
| 0.8 | $4.525 \times 10^{-3}$ | 0.4 | $3.337 \times 10^{-4}$ | 3.761 |
| 0.4 | $3.337 \times 10^{-4}$ | 0.2 | $2.266 \times 10^{-5}$ | 3.880 |
| 0.2 | $2.266 \times 10^{-5}$ | 0.1 | $1.477 \times 10^{-6}$ | 3.940 |
| 0.1 | $1.477 \times 10^{-6}$ | 0.05 | $9.423 \times 10^{-8}$ | 3.970 |

Even with larger step sizes, the maximum error for RK4 starts out 3 orders of magnitude less than the maximum error for Euler. The order of convergence approaches 4 as step size decreases, as predicted.
I added one more approximation at step size $h = 0.05$ in order to see the order of convergence get even closer to 4.

**Euler vs. RK4**



The above plot shows Euler vs. RK4 with a relatively large grid size of 0.8. The numbers are in the tables in the previous section, but you can see that Euler has a much larger error than RK4. RK4 is really only visible in this plot because its straight segments are visible against the smooth curve of the actual values.



The above plot shows Euler vs. RK4 with a smaller grid size of 0.1. The numbers are in the tables in the previous section, but you can see that Euler still has a larger error than RK4. The difference is smaller in appearance on the plot, but the numbers say that Euler error is 3 orders of magnitude larger than RK4 error, which is not even visible on the plot.

## 4. Discussion

### Euler vs. RK4

Runge-Kutta converges much more quickly than Euler, and even with large step sizes, its results are much more accurate. RK4 was three orders of magnitude more accurate with a step size of 0.8 than Euler was with a step size of 0.05, more than ten times smaller. The number of calculations necessary to achieve an accurate approximation could be an order of magnitude smaller using RK4.

### Other Runge-Kutta methods

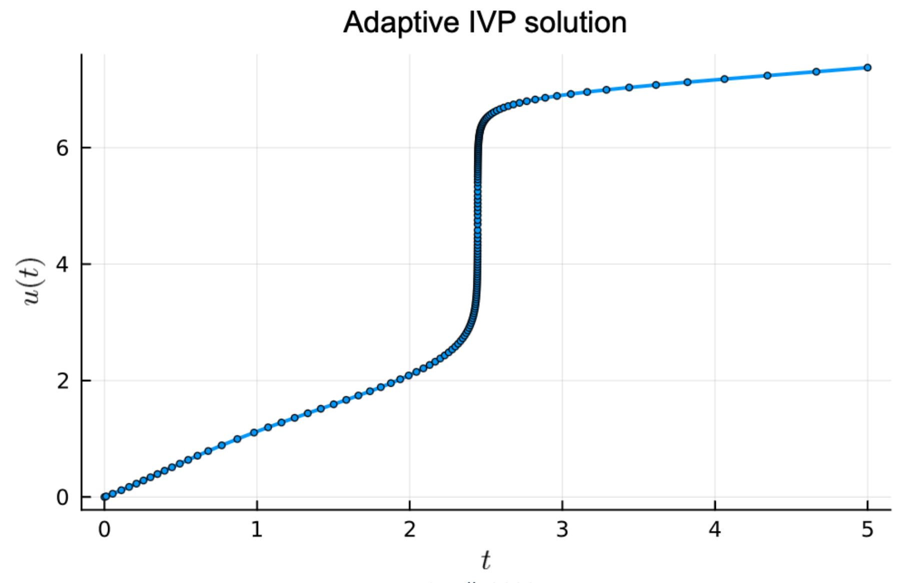
Runge-Kutta is a family of iterative methods, and there are other specific calculations that may in some circumstances be even more accurate than the RK4 method.

One variation is called the **3/8 rule** and it uses slightly different coefficients. For the main calculation we substitute:

$$y_{i+1} = y_i + \frac{h}{8}\,(k_1 + 3k_2 + 3k_3 + k_4)$$

so the midpoint approximations get weight 3/8, which is slightly more than the 1/3 weight used in the more commonly used RK4 method. The 3/8 rule has smaller error, but it is more compute intensive, so it requires more powerful processors or more time.

If the solution to the differential equation changes very quickly, oscillates, or has sections of steep slope, even the RK4 method may generate larger errors that we would like. In this case, we could use an **adaptive** Runge-Kutta method that estimates the error at each step; if the error exceeds a defined threshold, the step is repeated with a smaller step size. The following differential equation requires smaller steps in the center portion near $t = 2.4$ as the slope increases dramatically (Driscoll, 2022).



*Driscoll, 2022*

**Advantages and disadvantages**

RK4 is generally considered the best choice for most uses. We always use it in preference it to Euler. Sometimes we will use a specialized Runge-Kutta such as an adaptive method instead of RK4.

The only real rationale for using Euler is that it is easier to understand, simpler to program, and may use less computing power per iteration. But if RK4 converges with fewer iterations, RK4 may use fewer computations overall. Even the adaptive methods are relatively easy to program and require no more computing power than RK4 (Xinyu, 2018), and the code is available online. So we should always choose a Runge-Kutta method when it is available and suitable.

## 5. References

Driscoll, T and Braun, R (2023). Adaptive Runge-Kutta. *Fundamentals of Numerical Computation.* Society of Applied and Industrial Mathematics, Philadelphia, PA. https://tobydriscoll.net/fnc-julia/ivp/adaptive-rk.html

Geeks for Geeks (2023). *Runge-Kutta 4th order method to solve differential equation.* https://www.geeksforgeeks.org/runge-kutta-4th-order-method-solve-differential-equation/

Iskandar, D and Tiong, O (2022). The application of Runge Kutta fourth order method in SIR model for simulation of COVID-19 cases. *Proceeding of Science and Mathematics,* 10, 61–70. https://science.utm.my/procscimath/wp-content/uploads/sites/605/2022/10/61-70-Danial-Iskandar_Ong-Chee-Tiong.pdf

Jupyter (2020a). Runge-Kutta method. *ESE Jupyter Material.* Imperial College London. https://primer-computational-mathematics.github.io/book/c_mathematics/numerical_methods/5_Runge_Kutta_method.html

Jupyter (2020b). Heun's method. *ESE Jupyter Material.* Imperial College London. https://primer-computational-mathematics.github.io/book/c_mathematics/numerical_methods/4_Heuns_method.html

Khan, A (2017). Meet the 'Hidden Figures' mathematician who helped send Americans into space. *Los Angeles Times.* Jan. 9, 2017. https://www.latimes.com/science/sciencenow/la-sci-sn-hidden-figures-katherine-johnson-20170109-story.html

Pizer, S (1998). Local truncation error for the Euler Method [Class notes, University of North Carolina at Chapel Hill]. http://www.cs.unc.edu/~smp/COMP205/LECTURES/DIFF/lec17/node3.html

Singh, V (2018). Estimation of error in Runge-Kutta fourth order method. *Journal of Emerging Technologies and Innovative Research.* https://www.jetir.org/papers/JETIR1802301.pdf

Wikipedia (nd). *Runge-Kutta methods.* https://en.wikipedia.org/wiki/Runge–Kutta_methods

Wikipedia (nd). *Euler method.* https://en.wikipedia.org/wiki/Euler_method

Xinyu, F (2018). Learning the Runga-Kutta method. 2. Adaptive step size [Video]. *YouTube.* https://www.youtube.com/watch?v=JcRsGD2pKlA