Taylor Series and Numerical Methods for ODEs

Sergey Dutt

November 4, 2025

1 Tasks 1 - 2: Written Equations

1.1 The Second Derivative

Given the ordinary differential equation (ODE) $y'(t) = f(t, y)$, I use the chain rule to find the correct expression for $y''(t)$ in terms of $\frac{\partial f}{\partial t}$, $\frac{\partial f}{\partial y}$, and $f(t, y)$.

$$y''(t) = \frac{\partial f(t,y)}{\partial t} + \frac{\partial f(t,y)}{\partial y} f(t,y) \tag{1}$$

a. for $f(t, y) = t^2 + y$

$$y''(t) = t^2 + 2t + y \tag{2}$$

b. for $f(t, y) = t + y^2$

$$y''(t) = 1 + 2yt + 2y^3 \tag{3}$$

1.2 The Third Derivative

I now follow the same task for the $y'''(t)$

$$y'''(t) = \frac{d}{dt}\left(\frac{\partial f(t,y)}{\partial t}\right) + \frac{d}{dt}\left(\frac{\partial f(t,y)}{\partial y} f(t,y)\right) \tag{4a}$$

The differentiation of the first part of equation 4a gives us

$$\frac{d}{dt}\left(\frac{\partial f(t,y)}{\partial t}\right) = \frac{\partial^2 f(t,y)}{\partial t^2} + \frac{\partial^2 f(t,y)}{\partial y\, \partial t} f(t,y) \tag{4b}$$

and the differentiation of the second part of equation 4a gives us

$$\frac{d}{dt}\left(\frac{\partial f}{\partial y}f\right) = \frac{\partial^2 f}{\partial t\,\partial y}f + \frac{\partial^2 f}{\partial y^2}f^2 + \left(\frac{\partial f}{\partial y}\right)\left(\frac{\partial f}{\partial t}\right) + \left(\frac{\partial f}{\partial y}\right)^2 f \tag{4c}$$

where $f = f(t, y)$. Now combining parts 1 and 2, the final result is

$$y'''(t) = f\left(2\frac{\partial^2 f}{\partial y\,\partial t} + \left(\frac{\partial f}{\partial y}\right)^2\right) + f^2\frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial t^2} + \left(\frac{\partial f}{\partial y}\right)\left(\frac{\partial f}{\partial t}\right) \tag{4d}$$

a. for $f(t, y) = t^2 + y$

$$y'''(t) = t^2 + 2t + y + 2 \tag{5}$$

b. for $f(t, y) = t + y^2$

$$y'''(t) = 2y + 2(y^2 + t)(3y^2 + t) \tag{6}$$

2 Tasks 3 - 4: Numerical Equations

2.1 Computing Derivatives

Using a Jupyter worksheet, I constructed a function for creating derivatives. The function contains a loop for $y''(t)$ up to $y^{(8)}(t)$, and returns those derivatives as functions. Using the flexibility of symbolic functions and the python package SymPy, this function can construct derivatives for the expression $f(t, y)$, and specified functions of $f(t, y)$.

Using this function, I replicated the results of 1.1 and 1.2. Here is the code output:

```
Generically: y'' =  f(t, y)*Derivative(f(t, y), y) + Derivative(f(t, y), t)
Generically: y''' =  f(t, y)**2*Derivative(f(t, y), (y, 2)) + f(t, y)
*Derivative(f(t, y), y)**2 + 2*f(t, y)*Derivative(f(t, y), t, y)
+ Derivative(f(t, y), t)*Derivative(f(t, y), y) + Derivative(f(t, y), (t, 2))
The Second Derivative for (a) is y'' =  t**2 + 2*t + y
The Second Derivative for (b) is y'' =  2*y*(t + y**2) + 1
The Third Derivative for (a) is y''' =  t**2 + 2*t + y + 2
The Third Derivative for (b) is y''' =  2*y + 2*(t + y**2)*(t + 3*y**2)
```

2.2 Creating Taylor Series

With this derivative-creating function, I insert it into another function that creates Taylor series. I then use this taylor function to create 2nd, 4th, 5th, 6th, and 8th order taylor approximating polynomial for $y(t + h)$.

The following is the output of T2, the second order taylor polynomial. The other polynomials would be displayed similarly, but due to their length I will emit showing them.

```
T2 for y(t+h) =  h**2*(f(t_0, y_0)*Derivative(f(t_0, y_0), y_0) +
Derivative(f(t_0, y_0), t_0))/2 + h*f(t_0, y_0) + y_0
```

Note that, if $t = t_0$ and $y = y_0$, the following equation is equivalent to the above output:

$$y(t+h) = y(t) + y'(t)\,h + y''(\xi)\,\frac{h^2}{2} \tag{7}$$

3 Tasks 5 - 7: Numerically Solving IVP's

3.1 RK4, T4, and T5

Using Runge-Kutta (RK4), T4, and T5, with step size $h = 0.1$, I can solve for $y(1)$ of the following IVP.

$$\text{IVP:} \quad \begin{cases} \dfrac{dy}{dt} = t^4, \\ y(0) = 1 \end{cases} \tag{8}$$

t	RK4_y	RK4_error	T4_y	T4_error	T5_y	T5_error
0.000e+00	1.000	0.000e+00	1.000	0.000e+00	1.000	0.000e+00
0.100	1.000	8.333e-08	1.000	2.000e-06	1.000	0.000e+00
0.200	1.000	1.667e-07	1.000	4.000e-06	1.000	0.000e+00
0.300	1.000	2.500e-07	1.000	6.000e-06	1.000	0.000e+00
0.400	1.002	3.333e-07	1.002	8.000e-06	1.002	0.000e+00
0.500	1.006	4.167e-07	1.006	1.000e-05	1.006	0.000e+00
0.600	1.016	5.000e-07	1.016	1.200e-05	1.016	2.220e-16
0.700	1.034	5.833e-07	1.034	1.400e-05	1.034	4.441e-16
0.800	1.066	6.667e-07	1.066	1.600e-05	1.066	6.661e-16
0.900	1.118	7.500e-07	1.118	1.800e-05	1.118	6.661e-16
1.000	1.200	8.333e-07	1.200	2.000e-05	1.200	6.661e-16

Figure 1: Results and Error for numerical approximations to Equation (8)

T4 has the largest error, RK4 has the second largest, and T5 has the lowest error, with 9 orders of magnitude more accurate than RK4. In fact, T5 matches the exact solution for the first 5 steps, before it becomes very slightly off. Also notice that within 10 steps, RK4's

error looses an order of magnitude after 1 step, T4 looses an order of magnitude after 4 steps, but T5 never changes it's order.

3.2 RK4 and T4

Similarly, I use RK4 and T4, with step size $h = 0.1$, to solve for $y(2)$ of

$$\text{IVP:} \quad \begin{cases} \dfrac{dy}{dt} = -ty, \\ y(1) = 1 \end{cases} \tag{9}$$

t	RK4_y	RK4_error	T4_y	T4_error
1.000	1.000	0.000e+00	1.000	0.000e+00
1.100	0.900	9.304e-08	0.900	4.774e-07
1.200	0.803	2.186e-07	0.803	7.615e-07
1.300	0.708	3.798e-07	0.708	8.626e-07
1.400	0.619	5.772e-07	0.619	8.047e-07
1.500	0.535	8.084e-07	0.535	6.220e-07
1.600	0.458	1.068e-06	0.458	3.540e-07
1.700	0.389	1.347e-06	0.389	4.159e-08
1.800	0.326	1.635e-06	0.326	2.775e-07
1.900	0.271	1.919e-06	0.271	5.712e-07
2.000	0.223	2.185e-06	0.223	8.156e-07

Figure 2: Results and Error for numerical approximations to Equation (9)

The errors for RK4 and T4 are almost of similar magnitude. Knowing that RK4 is an order 4 method, we could qualitatively conclude that T4 is also approximately order 4, but here is some more concrete proof.

Using the following relationship for error (E) and order (p), for any positive integer n

$$E(h/n) \propto (h/n)^p \tag{10}$$

I derive an equation for determining the order of a numerical method.

$$p = \frac{\ln\left(\frac{E(h)}{E(h/2)}\right)}{\ln(2)} \tag{11}$$

When I plug in the error of $y(2)$ for T4 using $h = 0.1$ and $h = 0.05$, I get $p = 3.977 \approx 4$. Thus, T4 is indeed an order 4 method.

3.3 Graphical Comparison

$$\text{IVP:} \quad \begin{cases} \dfrac{dy}{dt} = sin(5\pi t) - cos(5\pi t) + y(t), \\ y(0) = 1 \end{cases} \tag{12}$$

Finally, I will compute the analytical solution to equation (12), along with RK4, T4, T6, and T8, with step size $h = 0.2$.

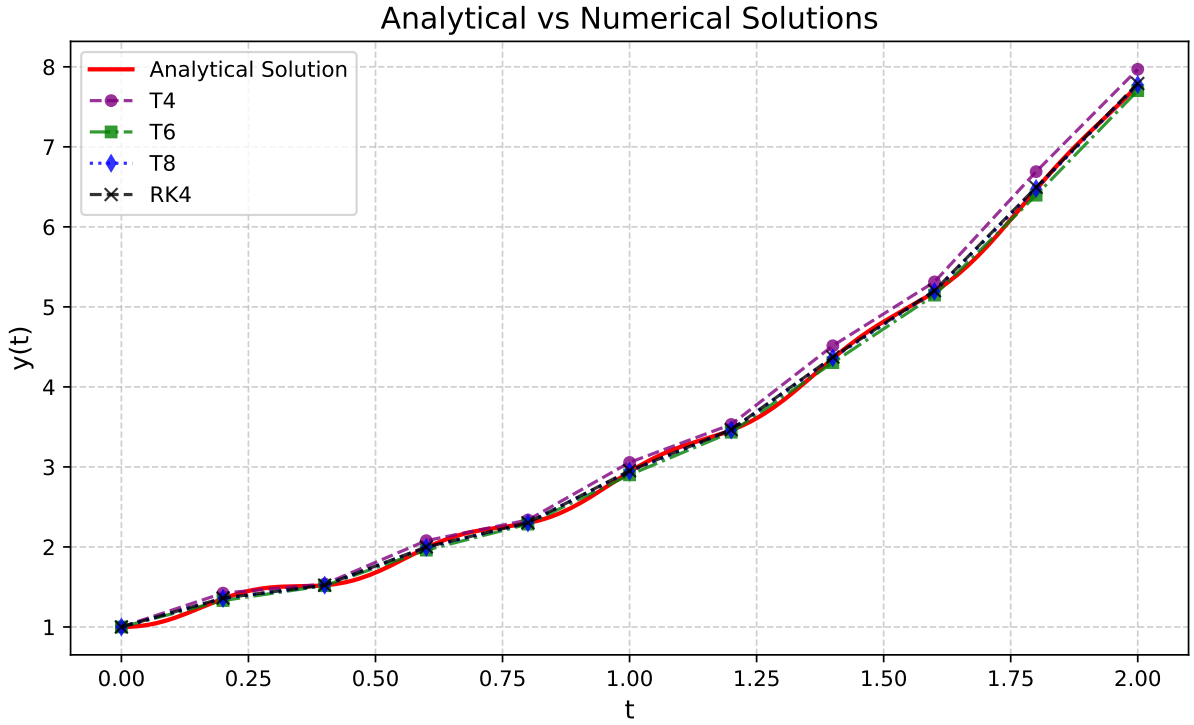

Figure 3: Analytical Solution and Numerical Approximations for $h = 0.2$.

Looking closely at Fig. 3, T4 overestimates the true solution, while T6 slightly underestimates it. On the other hand, RK4 and T8 are the best approximations, and it is hard

to distinguish between the two. In fact, the black-dashed line and blue dots for the two numerical methods take almost the exact path, both going over and under the red line an equal amount of times, averaging the true solution.

This may come across as surprising for two reasons. Firstly, it takes T8 nine terms of a taylor series and eight derivatives to achieve the same prediction as RK4, which only has two-terms and uses the first derivative. Also, RK4 is order 4 and T8 is order 8, yet they come to nearly the same answer for $y(2)$, meaning they have similar error.

This is an explanation for why RK4 is widely used for numerically approximating differential equations. It can match the accuracy of an 8th order taylor polynomial by calculating the slope at four carefully chosen locations per each step. It is way more simplistic than T8. As for the discrepancy in order, when you have a small enough h value,

$$h^4 \approx h^8$$

If we were to increase h, T8 would slowly become more accurate than RK4. But for the sake of simplicity and quicker computing, its better to select a small step size and use RK4.