1 Practice

Solution 1.1. While local truncation errors do not directly measure the error of an ode solver, it can be shown that they are directly proportional to the error for one-step methods. This makes them a very useful estimate for the actual error since we can compute the local truncation errors relatively straightforwardly, while the actual error is typically impossible to compute since we do not know an exact solution to the ode in general.

Solution 1.2. We introduce the two new variables u_1 and u_2:

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}.$$

Then the second order differential equation becomes the system

$$\begin{aligned} u_1' &= u_2 \\ u_2' &= -u_1(1+u_2^2)^{\frac{3}{2}}. \end{aligned}$$

Solution 1.3. The initial condition in our new variables is

$$\mathbf{u}_0 = \begin{pmatrix} 0 \\ \sqrt{3} \end{pmatrix}.$$

Now for $i = 1, 2, 3, 4$ we compute the updates

$$\mathbf{u}_i = \mathbf{u}_{i-1} + h\mathbf{f}(t_i, \mathbf{u}_{i-1})$$

$$\mathbf{u}_1 = \mathbf{u}_0 + .5\mathbf{f}(0, \mathbf{u}_0) = \begin{pmatrix} 0 \\ \sqrt{3} \end{pmatrix} + \begin{pmatrix} .5\sqrt{3} \\ 0 \end{pmatrix} = \begin{pmatrix} .5\sqrt{3} \\ \sqrt{3} \end{pmatrix}$$

$$\mathbf{u}_2 = \mathbf{u}_1 + .5\mathbf{f}(.5, \mathbf{u}_1) = \begin{pmatrix} .5\sqrt{3} \\ \sqrt{3} \end{pmatrix} + \begin{pmatrix} .5\sqrt{3} \\ -2\sqrt{3} \end{pmatrix} = \begin{pmatrix} \sqrt{3} \\ -\sqrt{3} \end{pmatrix}$$

$$\mathbf{u}_3 = \mathbf{u}_2 + .5\mathbf{f}(1, \mathbf{u}_2) = \begin{pmatrix} \sqrt{3} \\ -\sqrt{3} \end{pmatrix} + \begin{pmatrix} -.5\sqrt{3} \\ -4\sqrt{3} \end{pmatrix} = \begin{pmatrix} .5\sqrt{3} \\ -5\sqrt{3} \end{pmatrix}$$

$$\mathbf{u}_4 = \mathbf{u}_3 + .5\mathbf{f}(1.5, \mathbf{u}_3) = \begin{pmatrix} .5\sqrt{3} \\ -5\sqrt{3} \end{pmatrix} + \begin{pmatrix} -2.5\sqrt{3} \\ -.25\sqrt{3}(76)^{\frac{3}{2}} \end{pmatrix} \approx \begin{pmatrix} -3.4641016151377544 \\ -295.5539625781328 \end{pmatrix}$$

$$\begin{aligned} \mathbf{u}_5 = \mathbf{u}_4 + .5\mathbf{f}(2, \mathbf{u}_4) &\approx \begin{pmatrix} -3.4641016151377544 \\ -295.5539625781328 \end{pmatrix} + \begin{pmatrix} -147.7769812890664 \\ 44717595.61574259 \end{pmatrix} \\ &\approx \begin{pmatrix} -151.24108290420418 \\ 44717300.061780006 \end{pmatrix} \end{aligned}$$

Note in general that we will not be able to give exact values and instead will simply give their floating point approximations. For example, as the expressions become nastier and nasteir in the next steps, it would be much simpler just to give the floating point approximation instead, like we did for the last step. Furthermore, notice that the vectors are increasing in magnitude relatively quickly, which is generally not good behavior. This suggests that our step size is too large.

2 Coding

Some sample code has been provided in ode.py.

3 Testing your code

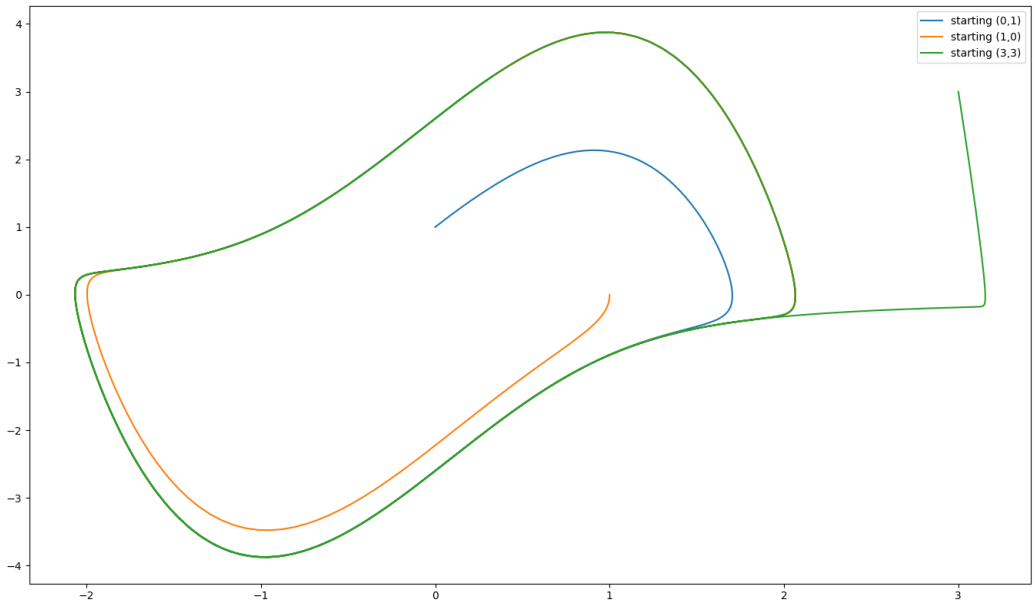
Solution 3.1. Introducing the variables

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix},$$

we obtain the new system

$$\begin{aligned} u_1' &= u_2 \\ u_2' &= 2(1 - u_1^2)u_2 - u_1. \end{aligned}$$

I chose $(0, 1)$, $(1, 0)$, and $(3, 3)$ as starting values.



You should observe that no matter your starting values (as long as they are reasonably close to the origin), your solutions will all converge to a *limit cycle*. In the graph above, this is where all three graphs are basically overlapping. This is known as an attractor and is a very interesting phenomenon in differential equations with a deep history of study.

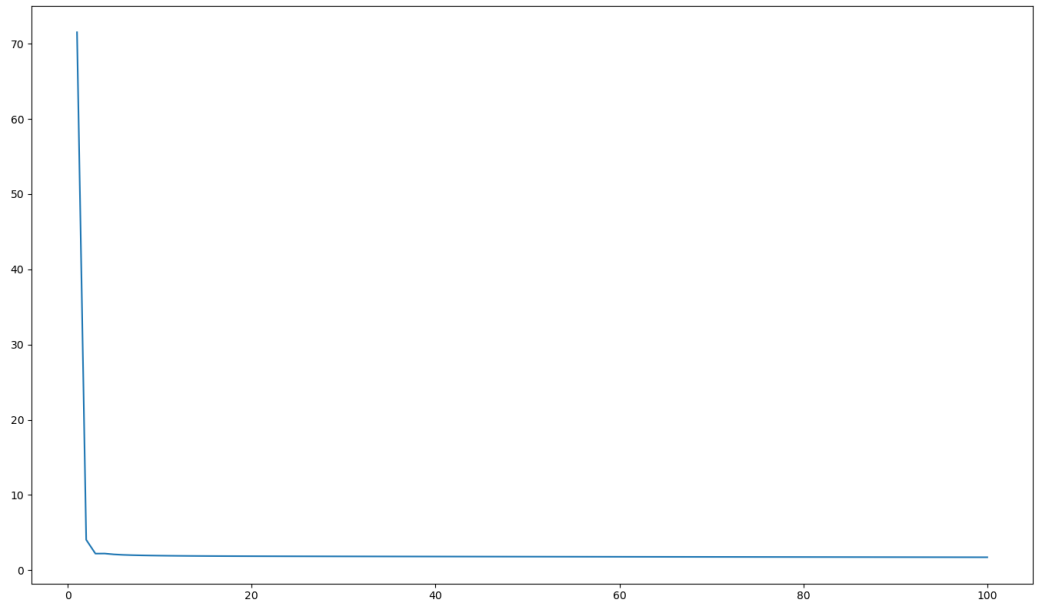
Solution 3.2. Introducing the variables

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix},$$

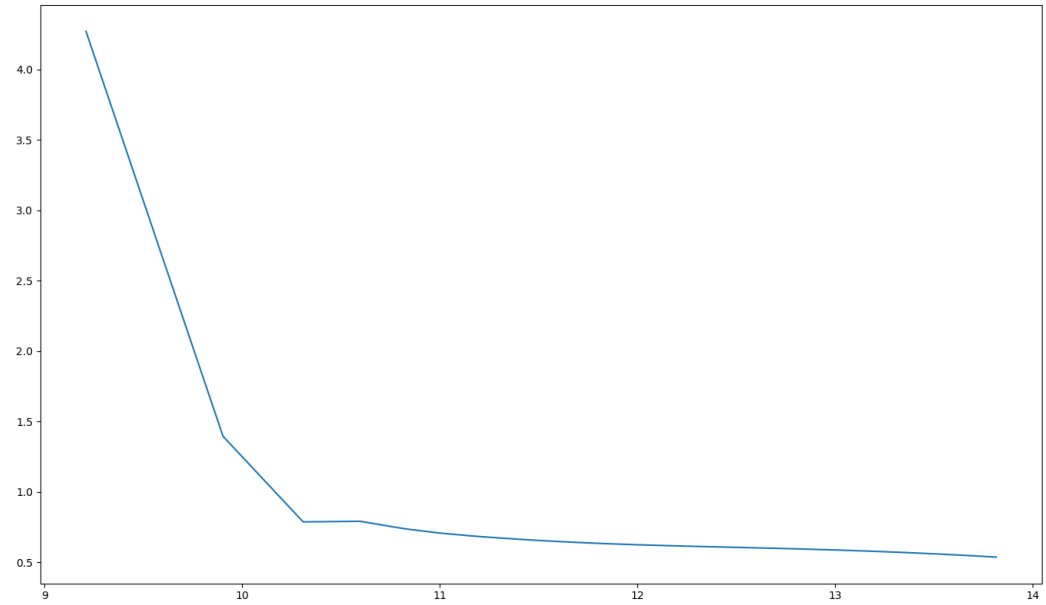
we obtain the new system

$$
\begin{aligned}
u_1' &= u_2 \\
u_2' &= u_1 + 2u_4 - d\frac{u_1 + c}{((u_1 + c)^2 + u_3^2)^{\frac{3}{2}}} - c\frac{u_1 - d}{((u_1 - d)^2 + u_3^2)^{\frac{3}{2}}} \\
u_3' &= u_4 \\
u_4' &= u_3 - 2u_2 - d\frac{u_3}{((u_1 + c)^2 + u_3^2)^{\frac{3}{2}}} - c\frac{u_3}{((u_1 - d)^2 + u_3^2)^{\frac{3}{2}}}.
\end{aligned}
$$

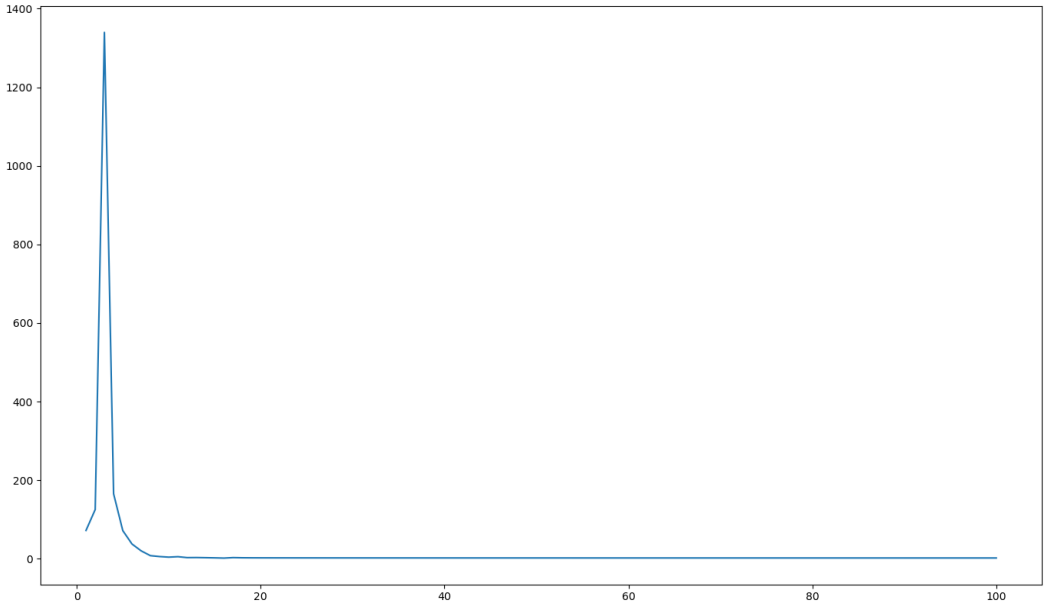
Plotting the error vs n gives

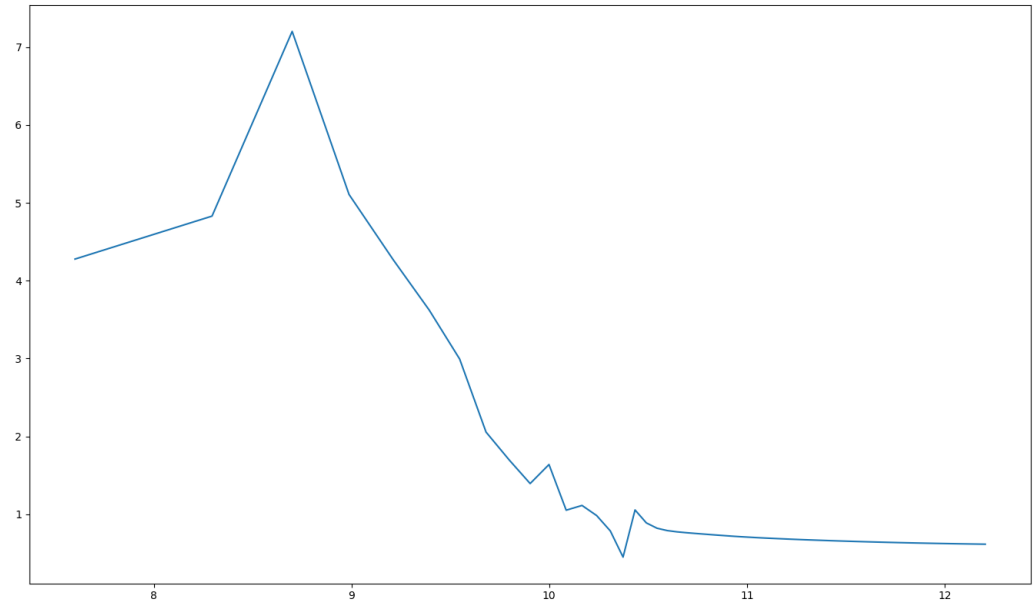


while plotting the log error vs log N gives

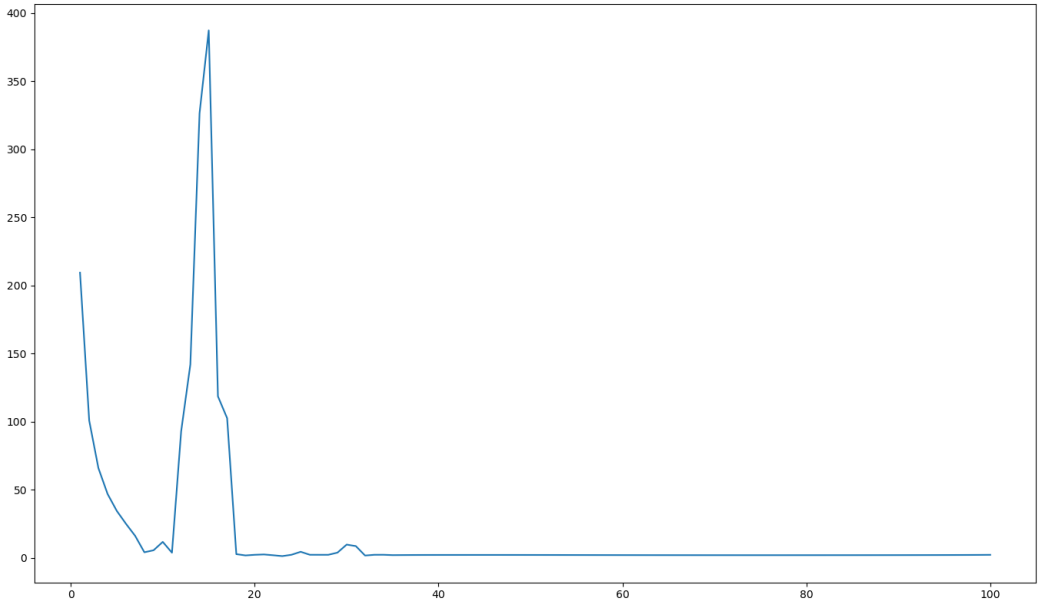


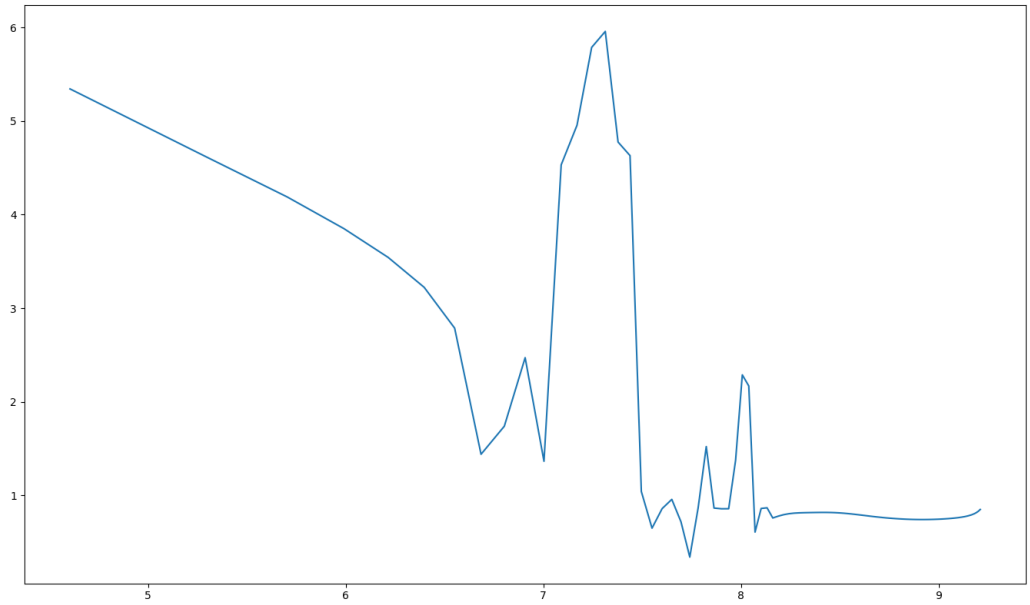
Computing some more data points to get more detail





The graphs don't have a very consistent trend, so I accepted any reasonable approximation of the slope. Repeating this process for the Runge-Kutta method gives the following error graphs.



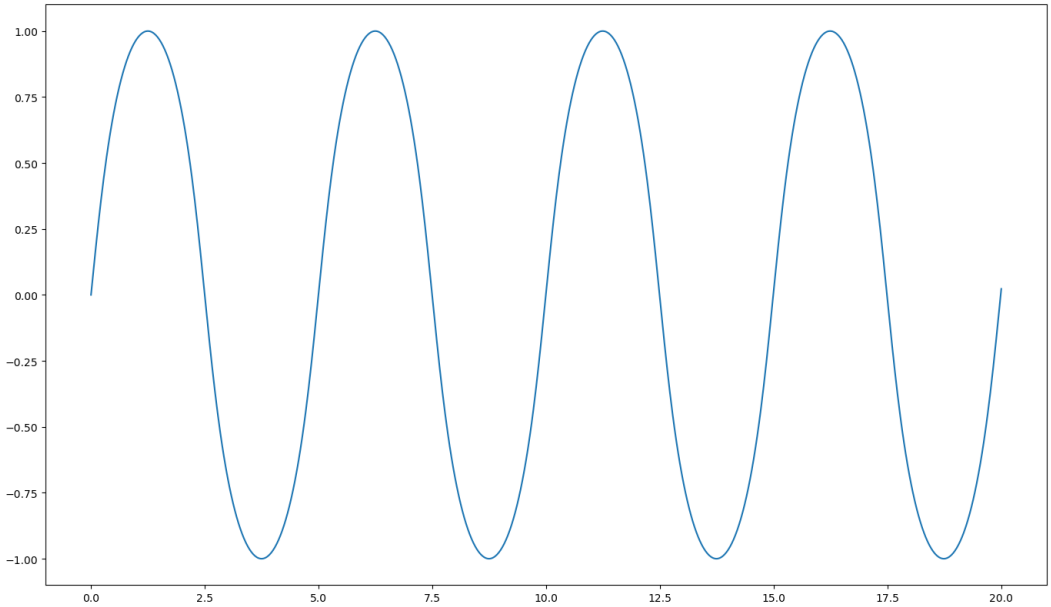


Again, there's not much clarity on the overall behavior of the errors here.

In both cases, the error decreases at first before a slight bit of (probably numerical) chaos ensues followed by a region of relative stability in the error. If the methods all behaved as expected (for example, if f were Lipschitz and y'' were bounded, which they are not) then we would expect to see a roughly linear region of the graph with slope corresponding to the order of the local truncation errors. However, what we see is relatively inconsistent behavior due to the fact that our theoretical assumptions in deriving the error bounds were not satisfied.

We are able to see, however, that the explicit Runge-Kutta method of order 4 greatly outperforms the Euler method. To give a fair comparison, we should note that Runge-Kutta requires four time more evaluations of the function $f(t, y)$, so we should compare Runge-Kutta with n steps to Euler method with $4n$ steps. However, even maing this comparison, we see very quickly that Runge-Kutta still vastly outperforms Euler's method. This highlights the effectiveness of higher order methods in general. Since the dependence on the number of timesteps is better for higher order methods, we see a much better return for increasing the number of timesteps vs lower order methods.

Solution 3.3. Using the exact same transformation as in the practice section to get a first order system, we then plug this first order system into our Runge-Kutta solver and obtain the following plot. We obtain a reasonably nice periodic function which is a bit sharper than a cosine curve but has many similar features.



Be careful to plot $y(x)$ vs. x rather than $y'(x)$ vs $y(x)$!