Homework 2

1. For the Arenstorf problem, we have the following IVP

$$x'' = x + 2y' - \mu' \frac{x+y}{D_1} - \mu \frac{x-\mu'}{D_2}, \tag{1}$$
$$y'' = y - 2x' - \mu' \frac{y}{D_1} - \mu \frac{y}{D_2}, \tag{2}$$

where

$$D_1 = \left((x+\mu)^2 + y^2\right)^{\frac{3}{2}},$$
$$D_2 = \left((x-\mu')^2 + y^2\right)^{\frac{3}{2}}.$$

We have that $\mu = 0.012277471$, $\mu' = 1 - \mu = 0.987722529$, and

$$x(0) = 0.994$$
$$x'(0) = 0$$
$$y(0) = 0$$
$$y'(0) = -2.00158510637908522240537862224.$$

We also set $T_{max} = 17.0652165601579625588917206249$, the period of the orbit. Solving the system using the DOPRI5(4) method (code provided in the Github repository: link), we produce the plot below (Fig. 1):

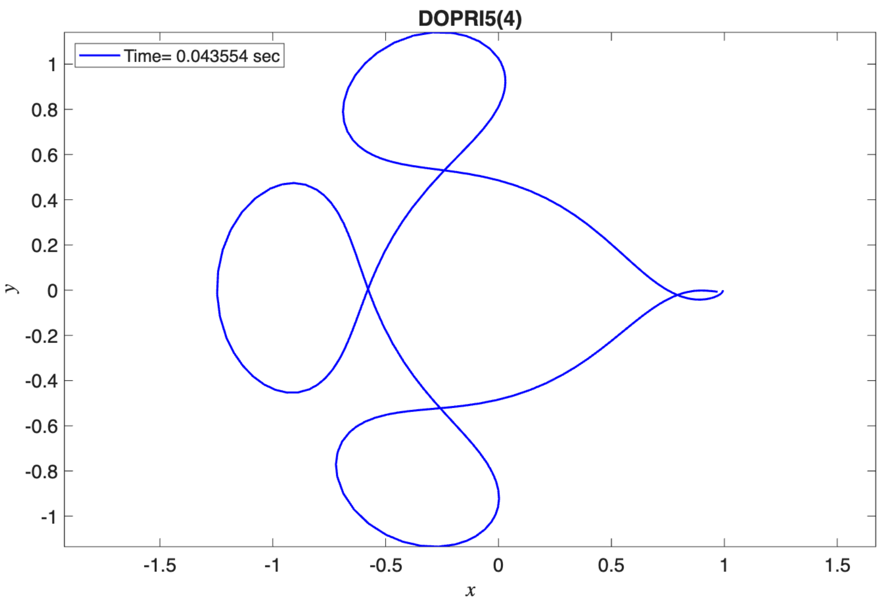

Figure 1: Plot of the numerical solution to the Arenstorf problem solved using DOPRI5(4) with $T_{max} \approx 17$.

Next, we consider $T_{max} = 100$ and keep the rest of the parameters fixed. We also solve the system using three built-in solvers in MATLAB: `ode45`, `ode78`, and `ode89`. The results are presented in Fig. 2.

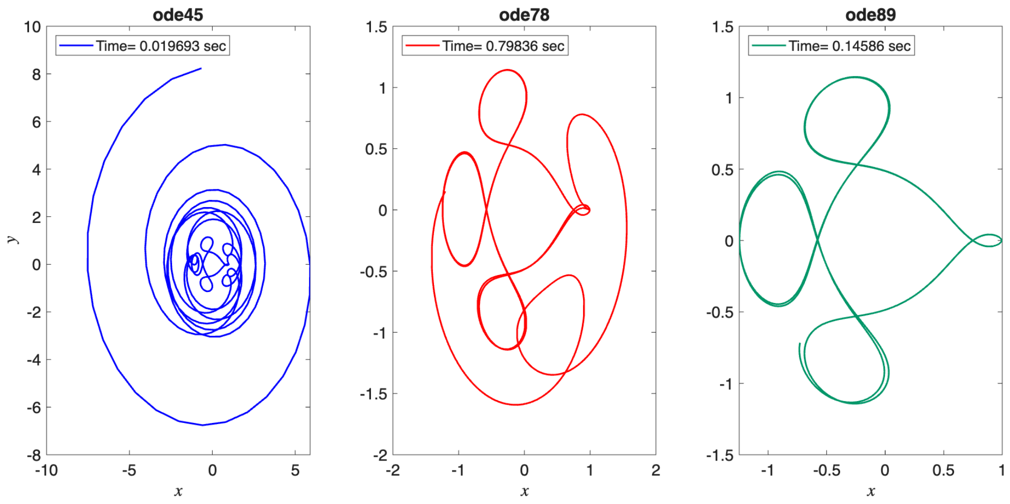

Figure 2: Plots of the numerical orbit of the Arenstorf problem using three different built-in solvers in MATLAB (`ode45`, `ode78`, and `ode89`). The legends feature the computation time of each solver with the error tolerance set at 10^{-12}.

Clearly, using `ode45`, we experienced numerical blow up in finite time. Similarly, `ode78` also began to show inaccuracy in its solution, as the simulated path of the celestial object veered off the expected path of the orbit at later times in the time interval $[0, 100]$. This solver was also the slowest of the three, and by a lot! The `ode78` was nearly 80 times slower than the `ode45` solver and more than 5 times slower than the `ode89` solver. The simulation using the `ode89` solver performed the best (visually). The orbit over the entire time interval is contained in the box given by $[-1.5, 1] \times [-1.5, 1.5]$. Further, the numerical solution does not stray off the path of the expected orbit by much. Based on these results, we would choose to use `ode89` to solve this IVP for a large time interval, since it has a good balance of speed and accuracy.

2. Using the code provided in the Github repo, we plot the regions of absolute stability for the explicit Runge-Kutta methods (Explicit Euler; Midpoint Ruler with Euler predictor; 3-stage, 3rd order Runge-Kutta; 4-stage, 4th order Runge-Kutta; DOPRI5(4)) in Fig. 3.

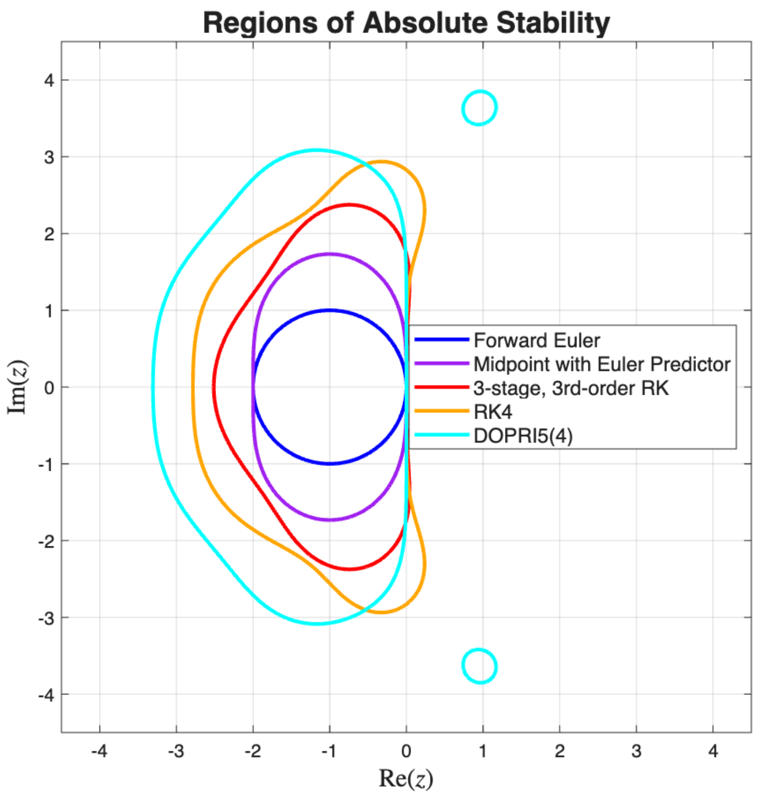

Figure 3: The regions of absolute stability for five explicit Runge-Kutta methods.

3. We construct the Forward Euler method (Scheme (3))

$$u_{n+1} = u_n + hf(t_n, u_n), \tag{3}$$

to numerically solve the general IVP (4),

$$y' = f(t, y),\ \ y(0) = y_0. \tag{4}$$

From the construction of scheme (3), we must have

$$u_{n+1} - u_n - hf(t_n, u_n) = 0. \tag{5}$$

The local truncation error for this method can be written as

$$\tau_{n+1} = y_{n+1} - y_n - hf(t_n, y_n). \tag{6}$$

Then, subtracting equation (5) from (6), we have

$$\begin{aligned} \tau_{n+1} &= (y_{n+1} - u_{n+1}) - (y_n - u_n) - h\left(f(t_n, y_n) - f(t_n, u_n)\right) \\ \iff (y_{n+1} - u_{n+1}) &= \tau_{n+1} + (y_n - u_n) + h\left(f(t_n, y_n) - f(t_n, u_n)\right) \\ \iff \|y_{n+1} - u_{n+1}\| &= \|\tau_{n+1}\| + \|y_n - u_n\| + h\|f(t_n, y_n) - f(t_n, u_n)\| \end{aligned}$$

Let $\tau = \max_n \|\tau_n\|$ and recall that we assume f is Lipschitz with Lipschitz constant L, meaning

$$\|f(t_n, y_n) - f(t_n, u_n)\| \le L\left(\|y_n - u_n\|\right).$$

Also, let $e_n = \|y_n - u_n\|$. Then,

$$\begin{aligned} e_{n+1} &\le \tau + e_n + hLe_n \\ &= \tau + (1+hL)e_n. \end{aligned}$$

This can be viewed as the recurrence relation

$$\begin{aligned} x_{n+1} &= (1+hL)x_n + \tau \\ \Rightarrow x_n &= (1+hL)^n x_0 + \frac{(1+hL)^n - 1}{hL}\tau. \end{aligned}$$

Given this fact, we have an upper bound for e_{n+1}, namely,

$$e_{n+1} \le (1+hL)^n e_0 + \frac{(1+hL)^n - 1}{hL}\tau.$$

Note that $(1+hL) \le e^{hL}, \forall hL \ne 0$, meaning

$$\begin{aligned} e_{n+1} &\le (1+hL)^n e_0 + \frac{(1+hL)^n - 1}{hL}\tau \\ &\le e^{hL} e_0 + \frac{e^{nhL} - 1}{hL}\tau \end{aligned}$$

Now, consider the coefficient of τ; taking $h \to 0$, we'll have

$$\lim_{h\to 0} \frac{e^{nhL} - 1}{hL} = \lim_{h \to 0} ne^{nhL} = n.$$

It follows that

$$\begin{aligned} & e_{n+1} \le e^{hL} e_0 + \frac{e^{nhL} - 1}{hL}\tau \\ \iff & e_{n+1} \le e_0 + n\tau \end{aligned}$$

Since τ is $\mathcal{O}(h^2)$, we have $e_{n+1} = \mathcal{O}(e_0) + \mathcal{O}(h)$.

4. For linear stability analysis, we consider the test problem (IVP (7))

$$y' = \lambda y, \ y(0) = y_0, \tag{7}$$

using the DIRK method with Butcher array

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1 & 1-\gamma & \gamma \\ \hline & 1-\gamma & \gamma \end{array}$$

where $\gamma = 1 - \frac{1}{\sqrt{2}}$. With this Butcher array, we construct the numerical scheme

$$k = \lambda\left(\vec{1}\cdot u_n + hAk\right) \tag{8}$$

$$u_{n+1} = u_n + hb^\top k. \tag{9}$$

We first solve for k:

$$\begin{aligned}
k &= (I - \lambda hA)^{-1}\left(\lambda u_n \cdot \vec{1}\right) \\
&= \left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} - \lambda h\begin{bmatrix}\gamma & 0\\1-\gamma & \gamma\end{bmatrix}\right)^{-1}\left(\lambda u_n \cdot \vec{1}\right) \\
&= \begin{bmatrix}1-h\lambda\gamma & 0\\h\lambda(\gamma-1) & 1-h\lambda\gamma\end{bmatrix}^{-1}\left(\lambda u_n \cdot \vec{1}\right) \\
&= \frac{1}{(1-h\lambda\gamma)^2}\begin{bmatrix}1-h\lambda\gamma & 0\\h\lambda(1-\gamma) & 1-h\lambda\gamma\end{bmatrix}\left(\lambda u_n \cdot \vec{1}\right).
\end{aligned}$$

We then use this formulation of k in equation (9) to get

$$\begin{aligned}
u_{n+1} &= u_n + \frac{h}{(1-h\lambda\gamma)^2}[1-\gamma\ \gamma]\begin{bmatrix}1-h\lambda\gamma & 0\\h\lambda(1-\gamma) & 1-h\lambda\gamma\end{bmatrix}\cdot\begin{bmatrix}\lambda u_n\\\lambda u_n\end{bmatrix} \\
&= \left(1 + \frac{h\lambda}{(1-h\lambda\gamma)^2}[1-\gamma\ \gamma]\begin{bmatrix}1-h\lambda\gamma & 0\\h\lambda(1-\gamma) & 1-h\lambda\gamma\end{bmatrix}\cdot\begin{bmatrix}1\\1\end{bmatrix}\right)u_n \\
&= \left(1 + \frac{h\lambda}{(1-h\lambda\gamma)^2}\Big[(1-\gamma)(1-h\lambda\gamma) + (1-\gamma)h\lambda\gamma,\ \gamma(1-h\lambda\gamma)\Big]\cdot\begin{bmatrix}1\\1\end{bmatrix}\right)u_n \\
&= \left(1 + \frac{h\lambda}{(1-h\lambda\gamma)^2}\Big[(1-\gamma),\ \gamma(1-h\lambda\gamma)\Big]\cdot\begin{bmatrix}1\\1\end{bmatrix}\right)u_n \\
&= \left(1 + \frac{h\lambda}{(1-h\lambda\gamma)^2}\left((1-\gamma) + \gamma(1-h\lambda\gamma)\right)\right)u_n \\
&= \left(1 + \frac{h\lambda}{(1-h\lambda\gamma)^2}\left((1-\gamma) + \gamma(1-h\lambda\gamma)\right)\right)u_n \\
&= \left(1 - \frac{(h\lambda)^2\gamma^2}{(1-h\lambda\gamma)^2}\right)u_n \\
&= \frac{(1-h\lambda\gamma)^2 - (h\lambda)^2\gamma^2}{(1-h\lambda\gamma)^2}u_n \\
&= \underbrace{\frac{1-2h\lambda\gamma}{1-2h\lambda\gamma+(h\lambda)^2\gamma^2}}_{R(h\lambda)}u_n.
\end{aligned}$$

Let $z = h\lambda$, meaning

$$R(z) = \frac{1-2z\gamma}{1-2z\gamma+z^2\gamma^2},$$

which is clearly always less than 1. Then,

$$\mathsf{RAS} = \mathbb{C} \supset \{z \in \mathbb{C} : \Re(z) < 0\}.$$

Thus, the method is A-stable (the RAS is also plotted in Fig. 4).

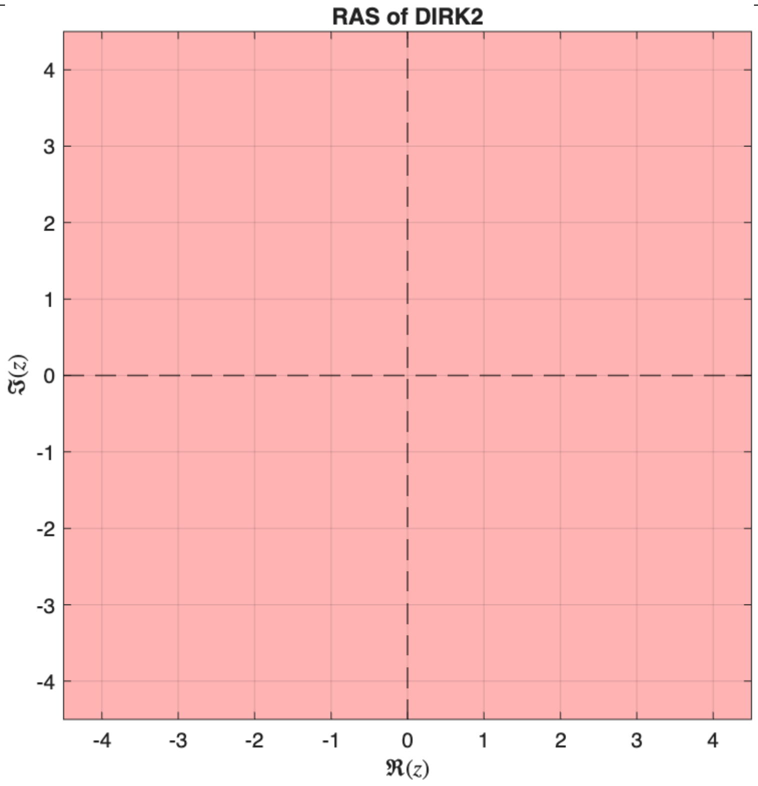

Figure 4: The region of absolute stability of numerical scheme (9) (shaded). Note that the entire left half of the complex plane is included in the RAS, meaning the scheme is A-stable. Code used to generate this figure can be found in the Github repo.

Next, we see that the method is L-stable as well; let $N(z) = 1 - 2z\gamma$ and $D(z) = 1 - 2z\gamma + (z)^2\gamma^2$ and notice

$$\mathsf{deg}(N(z)) = 1 < 2 = \mathsf{deg}(D(z)).$$