Part I. Theory

Problem 1 (Iserles, Exercise 3.4). Restricting your attention to scalar autonomous equations \( y' = f(y) \), prove that the ERK method with the tableau

\[
\begin{array}{c|ccccc}
0 & 1 \\
1/2 & 1/2 & 0 \\
1/2 & 0 & 1 \\
1 & 0 & 0 & 1 \\
\hline
1/6 & 1/3 & 1/3 & 1/3 & 1/3
\end{array}
\]

is of order four. (Note: This is a long calculation, so start early!)

Problem 2 (Iserles, Exercise 3.7). Write the theta method, (1.13), as a Runge–Kutta method.

Problem 3 (Iserles, Exercise 4.6). Evaluate explicitly the function \( r \) for the following Runge–Kutta methods:

- a. \[
\begin{array}{c|ccc}
0 & 0 & 0 \\
2/3 & 1/3 & 1/3 \\
1/4 & 3/4 & 1/4 \\
\hline
& & & \\
\end{array}
\]

- b. \[
\begin{array}{c|ccc}
1/6 & 1/6 & 0 \\
5/6 & 5/6 & 1/6 \\
1/2 & 1/2 & 2/3 \\
\hline
& & & \\
\end{array}
\]

- c. \[
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 1/2 & 0 \\
1/6 & 2/3 & 1/6 & 1/6 \\
\hline
& & & \\
\end{array}
\]

Part II. Programming

Download the sample code, hw3.py, which contains implementations of Euler’s method (euler) and the explicit trapezoid rule (etrap). For etrap, observe that after computing \( x1 = \xi_1 \) and \( x2 = \xi_2 \), we immediately evaluate and store the function values \( f1 = f(t_n + c_1 h, \xi_1) \) and \( f2 = f(t_n + c_2 h, \xi_2) \). This lets us reuse these function values at later stages without having to evaluate \( f \) again. (Remember: function evaluation is expensive!)

Create functions implementing the following ERK methods:
- explicit midpoint: \( \text{emid}(f, t0, y0, h, N) \)
- classical 3-stage Runge–Kutta: \( \text{rk3}(f, t0, y0, h, N) \)
- RK4 (tableau given in Problem 1): \( \text{rk4}(f, t0, y0, h, N) \)

Problem 4. For this problem, you will be solving the scalar IVP \( y' = y, \quad y(0) = 1 \), numerically on the interval \([0, 1]\). The exact solution \( y(t) = e^t \) has \( y(1) = e = 2.7182818284590 \ldots \). Approximate \( e \) by solving this IVP with \( h = 0.01 \) for each of the following explicit Runge–Kutta methods:
Problem 5. The function `errorPlot(method)` applies method (which can be any function for solving ODEs) to solve the IVP from Problem 4 for various choices of $h$, then creates a log-log plot of the absolute error vs. $h$.

Create error plots for the methods `euler`, `emid`, `rk3` and `rk4`. What is the relationship between the plot and the order of each method?

Problem 6. The **Lorenz system** is a famous system of nonlinear ODEs, whose study (numerically, at first) helped launch Chaos Theory. Consider the system of ODEs

\[
\begin{align*}
x' &= 10(y - x), \\
y' &= x(28 - z) - y, \\
z' &= xy - \frac{8}{3}z,
\end{align*}
\]

which is a special case of the Lorenz system. If $\mathbf{y} = (x, y, z)$, then this is in the usual form $\mathbf{y}' = f(t, \mathbf{y})$. Create a function `fLorenz(t,y)` corresponding to this $f$.

The function `lorenzPlot` uses your `rk4` and `fLorenz` to solve the Lorenz system for $t \in [0, 100]$, with $\mathbf{y}_0 = (0, 2, 20)$, and creates a 3D plot of the numerical solution. Run `lorenzPlot()`, and print out your plot.