NOTE: For each homework assignment please follow the “Guidelines for Numerical Analysis HW”. For any problems calling for a MATLAB function or script, you must submit these, with a comment on top including your name, the date, and “MATH 471”. And please clearly label any and all plots (title, x-label, y-label, and legend).

1. **SOURCE CODE:**

Write separate MATLAB functions for solving the initial value problem (IVP):

\[
y'(t) = f(t, y), \quad a \leq t \leq b \\
y(a) = \alpha
\]

using the

(a) Euler method:

\[
[w, t] = \text{Euler}(@f, a, b, \text{alpha}, n)
\]

(b) Backward Euler method (using fzero):

\[
[w, t] = \text{BackEuler}(@f, a, b, \text{alpha}, n, \text{tolX})
\]

In both of the above functions, the inputs \( f, a, b, \text{alpha} (= \alpha) \) are as in the IVP, and \( n \) is the number of steps to be taken, so that the step-size \( h = (b - a) / n \). The outputs \( w, t \) represent the numerical solution of the IVP, with \( t(i) = a + (i - 1) * h, \, i = 1 : n + 1 \) (using MATLAB index conventions here), and \( w \) the approximations of \( y(t) \).

In \text{BackEuler}, \text{tolX} is the tolerance used in the root-finding problem that must be solved at each step. Be sure to set up this root-finding problem correctly! Use MATLAB’s \text{fzero} function to solve it, thus:

\[
w_{\text{next}} = \text{fzero}(g, x0, \text{options}, p1, p2, ...)
\]

with

\[
\text{options} = \text{optimset}('\text{TolX}', \text{tolX})
\]

Here, \( g \) is a function such that \( x = w_{\text{next}} \) is a root of \( g(x, p1, p2, ...) = 0 \), and \( x0 \) is an initial estimate for this root – think about what a good choice of \( x0 \) could be. Note that \( g \) depends on \( f \), so that \( f \) should be an argument of \( g \).

2. Consider the following IVP:

\[
y' = -5y + 5 \cos(ty), \quad 0 \leq t \leq 10 \\
y(0) = 1.
\]

(a) Apply your MATLAB function \text{Euler} to this IVP with step-sizes \( h = 0.4, 0.2, 0.1, \) and 0.05. For each value of \( h \) make a plot of the approximate solution versus time. Use the ‘-o’ plotting option so that grid points are displayed. From the plots, for what values of \( h \) do you obtain a stable approximate solution?

(b) Repeat part (a) using your \text{BackEuler} function, with \text{tolX} = 1e-14.
3. As we know, the Euler method and the Backward Euler methods

\[
\text{Euler: } w_{i+1}^E = w_i + hf(t_i, w_i)
\]

\[
\text{Backward Euler: } w_{i+1}^B = w_i + hf(t_{i+1}, w_{i+1})
\]

are both \(O(h)\) accurate.

One alternative to these two methods is just to average them:

\[
w_{i+1}^{\text{new}} = \frac{1}{2} (w_{i+1}^E + w_{i+1}^B)
\]

\[
= w_i + \frac{h}{2} (f(t_i, w_i) + f(t_{i+1}, w_{i+1}))
\]

The resulting method is often referred to as the \textbf{Trapezoidal method}.

(a) Is the Trapezoidal method \textit{explicit} or \textit{implicit}? Explain.

(b) **Accuracy:** The local truncation error for this method is given by

\[
\tau_{i+1} = \frac{y_{i+1} - y_i}{h} - \frac{1}{2} \left( f(t_i, y_i) + f(t_{i+1}, y_{i+1}) \right).
\]

Show that \(\tau_{i+1} = \mathcal{O}(h^2)\).

(c) **Stability:** Determine the \textit{stability region} of the Trapezoidal method, by considering, as in class, the linear differential equation \(y'(t) = -\lambda y, \lambda > 0\).

4. Consider the following IVP:

\[
y' = y - t^2 + 1, \quad a \leq t \leq b
\]

\[
y(a) = \alpha.
\]

Write down \textbf{Taylor’s method of order 3} applied to the above IVP. Simplify your answer as much as possible. Your solution should be in the form:

\[
w_0 = \alpha
\]

\[
w_{i+1} = \text{(an expression involving only } h, t_i, \text{ and } w_i \text{) \quad for } i = 0, 1, \ldots, n - 1.
\]

\textit{continued}...
5. **SOURCE CODE:**

Write a MATLAB function for solving the IVP:

\[
y'(t) = f(t, y), \quad a \leq t \leq b \\
y(a) = \alpha
\]

using the Fourth-Order Runge-Kutta method:

\[ [w, t] = \text{RK4}(\Phi f, a, b, \alpha, n) \]

The inputs and outputs of the this function are as in problem 1.

6. Use your MATLAB function \text{RK4} defined in problem 5 to solve the following two IVPs and show that the results of the method on both problems demonstrate \( O(h^4) \) convergence to the corresponding exact solution.

**IVP 1:**

\[
y' = y \\
y(0) = 1 \\
t \in [0, 10]
\]

**Exact Solution:**

\[ y(t) = e^t \]

**IVP 2:**

\[
y' = t/y \\
y(0) = 1 \\
t \in [0, 10]
\]

**Exact Solution:**

\[ y(t) = \sqrt{1 + t^2} \]

(a) Run \text{RK4} with \( n = 50, 100, 200, 400, 800, 1600, 3200 \). Turn in a table for each of the two IVPs that contains 3 columns:

- the various \( h \) values,
- the error at the final time \( t = 10 \),
- the ratio of the previous/current error

(b) After you have constructed these tables, write a brief statement explaining how the results in the tables demonstrate that the Fourth-Order Runge-Kutta method gives \( O(h^4) \) convergence to the exact solution.