(1) Write a MATLAB routine to solve a two-point boundary-value problem,

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

using the shooting method where the underlying one-step method is the explicit fourth-order Runge–Kutta method. You will need to extend the RK4 method that you developed for Project 1 so that it can be used on systems of first-order ordinary differential equations.

For the root-finding for the initial velocity, use either the secant method or the Newton method with the complex-step derivative approximation, with error tolerance \( TOL \), and maximum number of iterations \( \text{MaxIters} \).

The resulting function should be written so that it can be called in MATLAB by typing:

\[ [x, t] = \text{BVP\_shooting}(\text{@f, a, b, alpha, beta, h, TOL, MaxIters}) \]

Then, use your method to solve the following two-point boundary-value problem,

\[ y'' = e^t + y \cos t - (t + 1)y', \quad y(0) = 1, \quad y(1) = 3, \]

for \( h = 0.1, \ h = 0.05 \).

(2) Write a MATLAB routine to solve linear two-point boundary-value problem,

\[ y'' = u + vx + wy', \quad y(a) = \alpha, \quad y(b) = \beta, \]

using the finite-difference method. The resulting function should be written so that it can be called in MATLAB by typing:

\[ [x, t] = \text{BVP\_finitediff}(\text{@u, @v, @w, a, b, alpha, beta, h}) \]

You may use the built-in MATLAB routines to perform the matrix inverse. Apply your method to the same example problem as in Problem 1, and compare your results.