UCB Math 128A, Spring 2020: Programming Assignment 4

Due May 2

Consider the initial-value problem

$$y' = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha. \tag{1}$$

The *Backward Euler method* is given by

$$w_0 = \alpha \tag{2}$$

$$w_{i+1} = w_i + hf(t_{i+1}, w_{i+1}) \tag{3}$$

Since f depends on the unknown w_{i+1}, (3) is an equation that must be solved at each time step, and in general f might be a non-linear function. In this assignment, we will implement the backward Euler method in MATLAB using Newton's method for solving (3). The solver will then be used to solve a stiff differential equation.

1. Write down Newton's method for solving (3) for w_{i+1}, using the initial guess $w_{i+1}^{(0)} = w_i$. The iterations will depend on the partial derivative $\partial f/\partial y = f_y(t, y)$.

2. Implement a MATLAB function `backeuler.m` of the form

   ```
   function [t,w] = backeuler(f, dfdy, a, b, alpha, N, maxiter, tol)
   ```

   which implements the backward Euler method using the Newton's method from problem 1. Here, `f` is the function $f(t, y)$, `dfdy` is the function $f_y(t, y)$, `N` is the number of time steps, `maxiter` is the maximum number of iterations in the Newton solver (the code should break if this is exceeded without convergence), and `tol` is the tolerance between two successive Newton iterations. At each time step, print the value of the Newton updates to monitor the convergence (similar to the `newton_table.m` function on the course web page).

   Use the following simple test to check your solver:

   ```
   f = @(t,y) -y * sin(y);
   df = @(t,y) -sin(y) - y*cos(y);
   a = 0; b = 3; alpha = 1; N = 20; maxiter = 20; tol = 1e-12;
   [t1,w1] = backeuler(f, df, a, b, alpha, N, maxiter, tol);
   [t2,w2] = rk4(f, a, b, alpha, N);
   plot(t1,w1, t2,w2)
   legend('Backward Euler', 'Runge-Kutta 4')
   ```

   The two curves should be close to each other, and if the number of steps N is increased they should get even closer (the RK4 method is much more accurate and can be considered the true solution). Do not include this test in your report, it is only meant as a verification of your backward Euler method.

Turn page $\longrightarrow$

3. Now consider a simple combustion model equation:

$$y' = y^2(1-y), \quad 0 \le t \le 2000, \quad y(0) = 0.9. \tag{4}$$

(a) Predict the number of steps N that are required to solve (4) using RK4. Since $y(t) \approx 1$, the stability can be estimated from the test equation $y' = \lambda y$ where $\lambda = \partial f/\partial y = 2y - 3y^2 \approx -1$. Note that the stability region of RK4 crosses the real axis at $h\lambda = -2.7853$.

(b) Verify your estimate of N in part (a) numerically, by solving with N about 10% below and 10% above your prediction. Plot the solutions and check if they give the expected value $y(2000) \approx 1$.

(c) Show that the backward Euler method is A-stable. What does this tell about the number of steps N required for stability?

(d) Solve (4) using your `backeuler` solver with $N = 1$, $N = 5$, and $N = 10$. Use the same values of `maxiter` and `tol` as in the example above. Plot the solutions. Does the method appear to be stable?

Reporting

Your report should contain the answers to the questions, the MATLAB codes, the MATLAB commands, the plots in (3bd), and the convergence of the Newton solver in (3d).