Math 128A: Programming Assignment 4

Michael Pham

Summer 2024

Problems

Problem 1.1. Write down Newton's method for solving equation (3) for w_{i+1}, using the initial guess $w_{i+1}^{(0)} = w_i$.

Solution. To implement Newton's method, we observe that we first have:

$$
\begin{aligned}
w_{i+1} &= w_i + hf(t_{i+1}, w_{i+1}) \\
0 &= w_{i+1} - w_i - hf(t_{i+1}, w_{i+1}) = g(w_{i+1}) \\
g(x) &= x - w_i - hf(t_{i+1}, x) \\
g'(x) &= 1 - hf'(t_{i+1}, x)
\end{aligned}
$$

Then, using Newton's Method, we want to find the root to this function $g(w_{i+1})$ for some w_{i+1}. So, we have:

$$w_{i+1} = w_i' - \frac{g(x)}{g'(x)}$$

Note here that w_i' is different from the w_i used in $g(x)$; w_i' refers to the previous root estimate. ∎

Problem 1.2. Implement a MATLAB function `backeuler.m` of the form `function [t,w] = backeuler(f, dfdy, a, b, alpha, N, maxiter, tol)`.

Solution. Below is our code for `backeuler.m`:

Code: Code for `backeuler.m`

```
function [t, w] = backeuler(f, df, a, b, alpha, N, maxiter, tol)
h = (b-a)/N;
t = a;
w = alpha;

t_arr = zeros(1, N+1);
w_arr = zeros(1, N+1);

t_arr(1) = t;
w_arr(1) = alpha;

for i = 1:N
        fprintf(' j         w0              w              err              \n');
        fprintf('--------------------------------------------------------\n');
        j = 1;
        flag = 0;
        w0 = w;
        wi = w0;
        while (flag == 0)
                top = w - wi - h*f(t+i*h, w);
                bot = 1 - h*df(t+i*h, w);
                w = w0 - top/bot;
                fprintf('%2d  %12.8f %12.8f  %12.8f \n', j, w0, w, abs(w - w0));

                if (abs(w-w0) < tol)
                        flag = 1;
                else
                        j = j + 1;
                        w0 = w;
                        if j > maxiter
                                error("Maximum iterations reached without convergence.");
                        end
                end
        end
```

```
        t = a + i*h;
        w = wi + h*f(t, w);

        t_arr(i+1) = t;
        w_arr(i+1) = w;
end

t = t_arr;
w = w_arr;
end
```

■

Problem 1.3a. Predict the number of steps N that are required to solve the following equation:

$$y' = y^2(1-y), \quad 0 \leq t \leq 2000, \quad y(0) = 0.9$$

Solution. Since $\lambda \approx -1$ and we have that $h\lambda = -2.7853$, then we observe the following:

$$\begin{aligned} h &= \frac{h\lambda}{\lambda} \\ &\approx \frac{-2.7853}{-1} \\ &= 2.7853 \\ N &\approx \frac{2000-0}{2.7853} \\ &= 718.06 \end{aligned}$$

Then, rounding up, we thus get that $N \approx 719$. ■

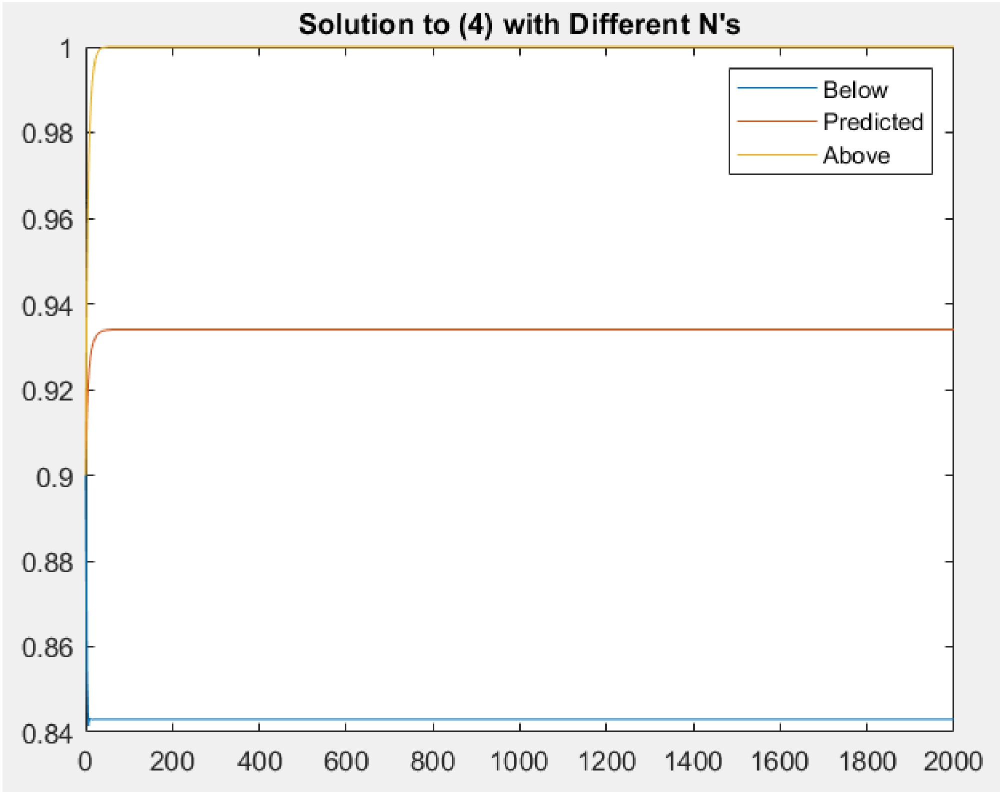
Problem 1.3b. Verify the estimate of N by solving with N about 10% above and below the predicted value. Plot the solutions and check that they give the expected value $y(2000) \approx 1$.

Solution. To begin with, we do the following to plot the graph:

Code: Code to generate plot of different choices of N's

```
>> f = @(t, y) (y^2)*(1-y);
a = 0;
b = 2000;
alpha = 0.9;
N1 = 648;
N2 = 719;
N3 = 791;
[t1, w1] = rk4(f, a, b, alpha, N1);
[t2, w2] = rk4(f, a, b, alpha, N2);
[t3, w3] = rk4(f, a, b, alpha, N3);
plot(t1, w1, t2, w2, t3, w3)
legend("Below", "Predicted", "Above")
title("Solution to (4) with Different N's")
```

This yields us the following graph:



And we see that, indeed, we get around $y(2000) \approx 1$. ∎

Problem 1.3c. Show that the backward Euler method is A-stable. What does it tell us about the number of steps N required for stability?

Solution. We observe the following when applying Backwards Euler to the test equation:

$$\begin{aligned} w_{i+1} &= w_i + h\lambda w_{i+1} \\ w_{i+1} - h\lambda w_{i+1} &= w_i \\ w_{i+1}(1 - h\lambda) &= w_i \\ w_{i+1} &= \frac{w_i}{1 - h\lambda} \end{aligned}$$

Then, we see that $Q(h\lambda) = \frac{1}{1-h\lambda}$. Thus, for $\mathrm{Re}(h\lambda) < 0$, we have $|Q(h\lambda)| < 1$; it is indeed A-stable.

Because Backward Euler is A-stable, this means that the number of steps N we choose will not have an effect on stability; that is, changing it won't lead to wildly differing results. ∎

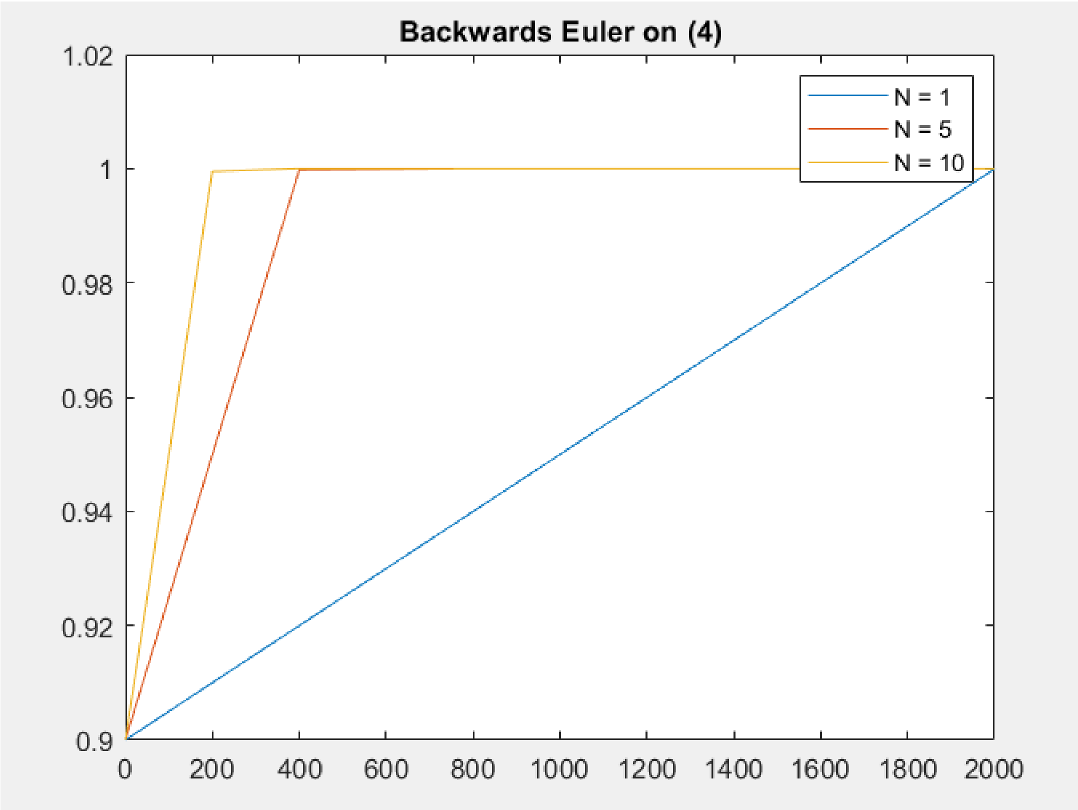
Problem 1.3d. Use the `backeuler` solver to solve the equation given with $N = 1$, $N = 5$, and $N = 10$. Plot the solutions. Does the method appear to be stable?

Solution. First, we introduce to the code to plot the solutions below:

Code: Code to plot solutions.

```
>> f = @(t, y) (y^2)*(1-y);
df = @(t, y) 2*y*(1-y) + (y^2)*(-1);
a = 0;
b = 2000;
alpha = 0.9;
maxiter = 20;
tol = 1e-12;
N1 = 1;
N2 = 5;
N3 = 10;
[t1, w1] = backeuler(f, df, a, b, alpha, N1, maxiter, tol);
[t2, w2] = backeuler(f, df, a, b, alpha, N2, maxiter, tol);
[t3, w3] = backeuler(f, df, a, b, alpha, N3, maxiter, tol);
plot(t1, w1, t2, w2, t3, w3)
legend("N = 1", "N = 5", "N = 10")
title("Backwards Euler on (4)")
```

Then, from here, we get the following plot:



Then, from the graph, we see that as all of the graphs converge to the right value of $y(2000) = 1$, we see that, indeed, Backwards Euler is stable. ■