Math 5601 Midterm Project

Jacob Hauck

October 13, 2023

Throughout this project, we consider the IVP

$$y' = f(t, y), \qquad a \le t \le b \tag{1}$$
$$y(a) = g_a, \qquad g_a \in \mathbf{R}. \tag{2}$$

We also use the mesh with sample points $t_j = a + jh$, with $t_0 = a$, where $h > 0$ is the step size. Lastly, we assume that f is L-Lipschitz in y uniformly for $t \in [a, b]$ (so that the solution of (1-2) is unique).

Problem 1.

Using the Taylor expansion for y about t_j, we get

$$y(t_{j+1}) = y(t_j) + hy'(t_j) + \frac{h^2}{2}y''(t_j) + \mathcal{O}(h^3). \tag{3}$$

Similarly, expanding y about t_{j+1} gives

$$y(t_j) = y(t_{j+1}) - hy'(t_{j+1}) + \frac{h^2}{2}y''(t_{j+1}) + \mathcal{O}(h^3). \tag{4}$$

Further expanding y'' about t_j, we get

$$y(t_j) = y(t_{j+1}) - hy'(t_{j+1}) + \frac{h^2}{2}\left(y''(t_j) + \mathcal{O}(h)\right) + \mathcal{O}(h^3) \tag{5}$$
$$= y(t_{j+1}) - hy'(t_{j+1}) + \frac{h^2}{2}y''(t_j) + \mathcal{O}(h^3). \tag{6}$$

Rearranging (6) and (3) and substituting from (1), we get

$$\frac{y(t_{j+1}) - y(t_j)}{h} = y'(t_j) + \frac{h}{2}y''(t_j) + \mathcal{O}(h^2) \quad = f(t_j, y(t_j)) + \frac{h}{2}y''(t_j) + \mathcal{O}(h^2), \tag{7}$$
$$\frac{y(t_{j+1}) - y(t_j)}{h} = y'(t_{j+1}) - \frac{h}{2}y''(t_j) + \mathcal{O}(h^2) = f(t_{j+1}, y(t_{j+1})) - \frac{h}{2}y''(t_j) + \mathcal{O}(h^2). \tag{8}$$

If we take the average of both sides of (7) and (8), then we finally obtain

$$\frac{y(t_{j+1}) - y(t_j)}{h} = \frac{f(t_{j+1}, y(t_{j+1})) + f(t_j, y(t_j))}{2} + \mathcal{O}(h^2). \tag{9}$$

Thus, if $y_j = y(t_j)$ and we compute y_{j+1} using the trapezoidal scheme, that is, as the solution of

$$y_{j+1} = y_j + h \cdot \frac{f(t_{j+1}, y_{j+1}) + f(t_j, y_j)}{2}, \tag{10}$$

then y_{j+1} (assuming the solution of (10) is unique) will satisfy the estimate

$$|y_{j+1} - y(t_{j+1})| = \frac{h}{2} \cdot |f(t_{j+1}, y(t_{j+1})) - f(t_{j+1}, y_{j+1})| + \mathcal{O}(h^3). \tag{11}$$

Using the Lipschitz property of f, we obtain

$$|y_{j+1} - y(t_{j+1})| \leq \frac{hL}{2} \cdot |y_{j+1} - y(t_{j+1})| + \mathcal{O}(h^3), \tag{12}$$

so

$$|y_{j+1} - y(t_{j+1})| \cdot \left(1 - \frac{hL}{2}\right) \leq \mathcal{O}(h^3). \tag{13}$$

As $h \to 0$, the quantity $1 - \frac{hL}{2} \to 1$; therefore,

$$|y_{j+1} - y(t_{j+1})| = \mathcal{O}(h^3). \tag{14}$$

That is, the *local truncation error* of the trapezoidal scheme is of order 3, which means that the *global truncation error* is of order 2.

Problem 2.

Consider the Taylor expansion of y about t_{j+1} at the points t_{j-1}, t_j and t_{j+1}:

$$y(t_{j-1}) = y(t_{j+1}) - 2hy'(t_{j+1}) + 2h^2y''(t_{j+1}) + \mathcal{O}(h^3) \tag{15}$$

$$y(t_j) = y(t_{j+1}) - hy'(t_{j+1}) + \frac{h^2}{2}y''(t_{j+1}) + \mathcal{O}(h^3) \tag{16}$$

$$y(t_{j+1}) = y(t_{j+1}). \tag{17}$$

If we form the linear combination $3y(t_{j+1}) - 4y(t_j) + y(t_{j-1})$, then we get

$$3y(t_{j+1}) - 4y(t_j) + y(t_{j-1}) = \quad 3y(t_{j+1}) \tag{18}$$

$$- 4y(t_{j+1}) + 4hy'(t_{j+1}) - 2y''(t_{j+1})h^2 \tag{19}$$

$$+ y(t_{j+1}) - 2hy'(t_{j+1}) + 2y''(t_{j+1})h^2 + \mathcal{O}(h^3). \tag{20}$$

Therefore, canceling terms and substituting from (1), we have

$$3y(t_{j+1}) - 4y(t_j) + y(t_{j-1}) = 2hf(t_{j+1}, y(t_{j+1})) + \mathcal{O}(h^3) \tag{21}$$

Thus, if we know that $y_{j-1} = y(t_{j-1})$, and $y(t_j) = y_j$ and we compute y_{j+1} using the two-step backward differentiation scheme, that is, as the solution of

$$\frac{3y_{j+1} - 4t_j + y_{j-1}}{2h} = f(t_{j+1}, y_{j+1}), \tag{22}$$

then the local truncation error $|y_{j+1} - y(t_{j+1})|$ will satisfy

$$|y_{j+1} - y(t_{j+1})| = \frac{2h}{3} \cdot |f(t_{j+1}, y_{j+1}) - f(t_{j+1}, y(t_{j+1}))| + \mathcal{O}(h^3). \tag{23}$$

By the Lipschitz property of f,

$$|y_{j+1} - y(t_{j+1})| \leq \frac{2hL}{3} \cdot |y_{j+1} - y(t_{j1+})| + \mathcal{O}(h^3), \tag{24}$$

so

$$|y_{j+1} - y(t_{j+1})| \cdot \left(1 - \frac{2hL}{3}\right) \leq \mathcal{O}(h^3). \tag{25}$$

As $h \to 0$, the quantity $1 - \frac{2hL}{3} \to 1$; therefore,

$$|y_{j+1} - y(t_{j+1})| = \mathcal{O}(h^3). \tag{26}$$

That is, the *local trunction error* of the two-step backward differentiation scheme is of order 3, and the *global truncation error* is of order 2.

Problem 3.

Consider the Taylor expansions of $y(t_{j+1})$, $y(t_j)$, $y(t_{j-1})$, and $y(t_{j-2})$ about t_{j+1}:

$$y(t_{j+1}) = y(t_{j+1}) \tag{27}$$

$$y(t_j) = y(t_{j+1}) - hy'(t_{j+1}) + \frac{h^2}{2}y''(t_{j+1}) - \frac{h^3}{6}y'''(t_{j+1}) + \mathcal{O}(h^4) \tag{28}$$

$$y(t_{j-1}) = y(t_{j+1}) - 2hy'(t_{j+1}) + 2h^2y''(t_{j+1}) - \frac{4h^3}{3}y'''(t_{j+1}) + \mathcal{O}(h^4) \tag{29}$$

$$y(t_{j-2}) = y(t_{j+1}) - 3hy'(t_{j+1}) + \frac{9h^2}{2}y''(t_{j+1}) - \frac{9h^3}{2}y'''(t_{j+1}) + \mathcal{O}(h^4) \tag{30}$$

Then, for $\beta_1, \beta_2, \beta_3, \beta_4 \in \mathbf{R}$,

$$\beta_1 y(t_{j+1}) + \beta_2 y(t_j) + \beta_3 y(t_{j-1}) + \beta_4 y(t_{j-2}) = \tag{31}$$

$$(\beta_1 + \beta_2 + \beta_3 + \beta_4)y(t_{j+1}) \tag{32}$$

$$-(\beta_2 + 2\beta_3 + 3\beta_4)y'(t_{j+1})h \tag{33}$$

$$+\frac{1}{2}(\beta_2 + 4\beta_3 + 9\beta_4)y''(t_{j+1})h^2 \tag{34}$$

$$-\frac{1}{6}(\beta_2 + 8\beta_3 + 27\beta_4)y'''(t_{j+1})h^3 + \mathcal{O}(h^4). \tag{35}$$

To cancel the lower-order terms, we must choose β_2, β_3, and β_4 such that

$$\begin{aligned} -1 &= \beta_2 + \beta_3 + \beta_4 \\ 0 &= \beta_2 + 4\beta_3 + 9\beta_4 \\ 0 &= \beta_2 + 8\beta_3 + 27\beta_4, \end{aligned} \tag{36}$$

then we get

$$\frac{y(t_{j+1}) + \beta_2 y(t_j) + \beta_3 y(t_{j-1}) + \beta_4 y(t_{j-2})}{-(\beta_2 + 2\beta_3 + 3\beta_4)h} = y'(t_{j+1}) + \mathcal{O}(h^4) = f(t_{j+1}, y(t_{j+1})) + \mathcal{O}(h^3). \tag{37}$$

To satisfy (36), we must have $4\beta_3 + 9\beta_4 = 8\beta_3 + 27\beta_4$, so $\beta_3 = -\frac{9}{2}\beta_4$. Then $\beta_2 = 18\beta_4 - 9\beta_4 = 9\beta_4$, and $-1 = 9\beta_4 - \frac{9}{2}\beta_4 + \beta_4 = \frac{11}{2}\beta_4$, so $\beta_4 = -\frac{2}{11}$. Then $\beta_3 = \frac{9}{11}$, and $\beta_2 = -\frac{18}{11}$. Lastly, $-(\beta_2 + 2\beta_3 + 3\beta_4) = \frac{6}{11}$.

If we set

$$\alpha_1 = \frac{1}{-(\beta_2 + 2\beta_3 + 3\beta_4)} = \frac{11}{6} \tag{38}$$

$$\alpha_2 = \frac{\beta_2}{-(\beta_2 + 2\beta_3 + 3\beta_4)} = -\frac{18}{6} \tag{39}$$

$$\alpha_3 = \frac{\beta_3}{-(\beta_2 + 2\beta_3 + 3\beta_4)} = \frac{9}{6} \tag{40}$$

$$\alpha_4 = \frac{\beta_4}{-(\beta_2 + 2\beta_3 + 3\beta_4)} = -\frac{2}{6}, \tag{41}$$

then by (37),

$$\frac{\alpha_1 y(t_{j+1}) + \alpha_2 y(t_j) + \alpha_3 y(t_{j-1}) + \alpha_4 y(t_{j-2})}{h} = f(t_{j+1}, y(t_{j+1})) + \mathcal{O}(h^3). \tag{42}$$

If we had $y_{j-2} = y(t_{j-2})$, $y_{j-1} = y(t_{j-1})$, and $y_j = y(t_j)$, and we computed y_{j+1} as the solution of

$$\frac{\alpha_1 y_{j+1} + \alpha_2 y_j + \alpha_3 y_{j-1} + \alpha_4 y_{j-2}}{h} = f(t_{j+1}, y_{j+1}), \tag{43}$$

then $|y_{j+1} - y(t_{j+1})|$ would satisfy

$$|y_{j+1} - y(t_{j+1})| = \frac{h}{\alpha_1} \cdot |f(t_{j+1}, y_{j+1}) - f(t_{j+1}, y(t_{j+1}))| + \mathcal{O}(h^4). \tag{44}$$

Using the Lipschitz property of f, we obtain

$$|y_{j+1} - y(t_{j+1})| \cdot \left(1 - \frac{hL}{\alpha_1}\right) \leq \mathcal{O}(h^4). \tag{45}$$

As $h \to 0$, the quantity $1 - \frac{hL}{\alpha_1} \to 1$; therefore,

$$|y_{j+1} - y(t_{j+1})| = \mathcal{O}(h^4). \tag{46}$$

That is, the implicit scheme

$$\frac{\alpha_1 y(t_{j+1}) + \alpha_2 y(t_j) + \alpha_3 y(t_{j-1}) + \alpha_4 y(t_{j-2})}{h} = f(t_{j+1}, y_{j+1}) \tag{47}$$

with $\alpha_1 = \frac{11}{6}$, $\alpha_2 = -\frac{18}{6}$, $\alpha_3 = \frac{9}{6}$, and $\alpha_4 = -\frac{2}{6}$ has 3rd-order global accuracy and 4th order local accuracy. Since we had to choose these values of α to cancel higher-order terms, these must be the coefficients in the third-order backward differentiation scheme.

We now consider Newton's method for finding the root of a function f:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{48}$$

Problem 4.

Suppose that f has a root z of multiplicity $m \geq 2$. Then, by definition, there exists a function r such $r(z) \neq 0$, and $f(x) = (x-z)^m r(x)$. Then $f'(x) = m(x-z)^{m-1} r(x) + (x-z)^m r'(x) = (x-z)^{m-1}(mr(x) + (x-z)r'(x))$. Then we can still safely define Newton's method despite the fact that $f'(z) = 0$ by setting

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{(x-z)^m r(x)}{(x-z)^{m-1}(mr(x) + (x-z)r'(x))} = x - \frac{(x-z)r(x)}{mr(x) + (x-z)r'(x)} \tag{49}$$

and observing that the denominator in the last expression is nonzero when $x = z$ because $r(z) \neq 0$. Then Newton's method becomes $x_{k+1} = g(x_k)$.

To apply the theory of convergence in the project description, we need to compute

$$g'(x) = 1 - \frac{(r(x) + (x-z)r'(x))(mr(x) + (x-z)r'(x)) - (x-z)r(x)(mr'(x) + r'(x) + (x-z)r''(x))}{(mr(x) + (x-z)r'(x))^2}$$

so that

$$g'(z) = 1 - \frac{m(r(z))^2}{(mr(z))^2} = 1 - \frac{1}{m} \tag{50}$$

since $r(z) \neq 0$. Since $g'(z) \neq 0$ if $m \geq 2$, but $|g'(z)| < 1$, it follows by the convergence theorem in the project description that Newton's method has *linear* convergence in this case.

Problem 5.

In the case that f has a root z of multiplicity $m \geq 2$, we saw that Newton's method defined by $x_{k+1} = g(x_k)$, where

$$g(x) = x - \frac{(x-z)r(x)}{mr(x) + (x-z)r'(x)} \tag{51}$$

had a linear convergence rate to the root z of f. We can fix this simply by adjusting Newton's method to

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)}, \tag{52}$$

that is, by replacing g by g_m, where

$$g_m(x) = x - m\frac{(x-z)r(x)}{mr(x) + (x-z)r'(x)}. \tag{53}$$

This method has at least quadratic convergence by the convergence theorem in the project description because

$$g_m'(x) = 1 - m\frac{(r(x) + (x-z)r'(x))(mr(x) + (x-z)r'(x)) - (x-z)r(x)(mr'(x) + r'(x) + (x-z)r''(x))}{(mr(x) + (x-z)r'(x))^2}$$

so that

$$g_m'(z) = 1 - m\frac{m(r(z))^2}{(mr(z))^2} = 0. \tag{54}$$

Then the iteration $x_{k+1} = g_m(x_k)$ converges at least quadratically to the root z of f by the convergence theorem in the project description.

Now we consider the implementation of the backward Euler method for (1, 2):

$$y_0 = y(a) = g_a, \qquad y_{j+1} = y_j + hf(t_{j+1}, y_{j+1}) \quad \text{if} \quad j \in \{0, 1, \ldots, J-1\}. \tag{55}$$

Problem 6.

Suppose that y_j, t_{j+1} and h are known at the jth step of the backward Euler method. Then y_{j+1} can be obtained by solving the nonlinear equation

$$y_{j+1} = y_j + hf(t_{j+1}, y_{j+1}) \tag{56}$$

for y_{j+1}. Since our methods for numerically solving nonlinear equations work only for equations of the form $\tilde{f}(x) = 0$ (for the bisection, Netwton's and secant methods) and $\tilde{g}(x) = x$ (for the fixed point method), we need to recast (56) in these forms. In other words, we need to define $\tilde{f}$ and $\tilde{g}$ such that

$$\tilde{f}(y_{j+1}) = 0 \iff y_{j+1} = y_j + hf(t_{j+1}, y_{j+1}) \iff \tilde{g}(y_{j+1}) = y_{j+1}. \tag{57}$$

There are many ways to do this, but perhaps the simplest is to choose

$$\tilde{f}(x) = x - y_j - hf(t_{j+1}, x), \qquad \tilde{g}(x) = y_j + hf(t_{j+1}, x). \tag{58}$$

Problem 7.

Suppose that we wanted to use the bisection method or the secant method to solve $x = y_j + hf(t_{j+1}, x)$ for x.

(a) To use the bisection method, we would need to know

(1) the initial interval $[a, b]$ that contains the root, and

(2) the stopping conditions (error tolerances and maximum iterations).

As mentioned in the project description, the stopping conditions can be set according to the accuracy requirements determined by the step size h. The initial interval, however, would be more difficult to determine.

(b) To use the secant method, we would need to know

(1) the initial points x_0 and x_1, and

(2) the stopping conditions (error tolerances and maximum iterations).

As with the bisection method, the stopping conditions here can be determined fairly easily. The two initial points, however, would be more difficult to choose. Perhaps $x_0 = y_{j-1}$ and $x_1 = y_j$ would work, but we would still need to answer the question of how to choose x_0 when $j = 0$.

Problem 8.

In this problem, we attempt to use the backward Euler method to solve the following special case of (1, 2):

$$y' = e^{2t}y^2, \qquad y(0) = 0.1, \qquad 0 \le t \le 1. \tag{59}$$

(a) In order to avoid duplicate code, I have implemented an abstract version of the backward Euler method that takes as input the initial condition g_a, the interval $[a, b]$, and the step size h. The function f is specified implicitly by the function argument solver, defined by

$$\text{solver}(x_0, t, h) = \text{solution } x \text{ of } \big[x = x_0 + hf(t, x)\big]. \tag{60}$$

I define two solver functions, one that uses Newton's method, and one that uses the fixed point method.

In Listing 1 is the abstract backward Euler method implementation (copied from `backward_euler.m`). In Listings 2 and 3 (copied from `be_newton.m` and `be_fixed.m`) are functions that construct suitable solver functions that use Newton's method and the fixed point method to solve the equation $x = x_0 + hf(t, x)$ for x by converting the equation into $\tilde{f}(x) = 0$ and $\tilde{g}(x) = x$, where $\tilde{f}$ and $\tilde{g}$ are the same as in Problem 6; note that we need to take $x_0 = y_j$ and $t = t_{j+1}$ so that $\tilde{f}(x) = 0$ and $\tilde{g}(x) = x$ are equivalent to $x = x_0 + hf(t, x)$ (see line 16 of Listing 1).

Listing 1: Abstract backward Euler method

```
1  function [t, y] = backward_euler(solver, g_a, a, b, h)
2
3  % get as close to b as possible without going past on the last step
4  % add one step for the initial value
5  num_steps = 1 + floor((b - a) / h);
6
7  t = zeros(1, num_steps);
8  y = zeros(1, num_steps);
9
10 t(1) = a;
11 y(1) = g_a;
12
13 for j = 2:num_steps
14     t(j) = t(j-1) + h;
15
```

```
    % solver : (x_0, t, h) -> solution of x = x_0 + h * f(t, x)
    y(j) = solver(y(j-1), t(j), h);
end
```

Listing 2: solver using Newton's method

```
function result = be_newton( ...
    f, dfdy, epsilon, epsilon_f, epsilon_f_prime, max_it, log_iterations ...
)

% creates a backward Euler solver function for y' = f(t,y)
% that maps (x_0, t, h) to the solution of x = x_0 + h * f(t,x)
% using Newton's method with initial point x_0
result = @(x_0, t, h) newton( ...
    @(x) x - x_0 - h * f(t, x), @(x) 1 - h * dfdy(t, x), ...
    x_0, epsilon, epsilon_f, epsilon_f_prime, max_it, log_iterations ...
);
```

Listing 3: solver using the fixed point method

```
function result = be_fixed(f, epsilon, max_it, log_iterations)

% creates a backward Euler solver function for y' = f(t,y)
% that maps (x_0, t, h) to the solution of x = x_0 + h * f(t,x)
result = @(x_0, t, h) fixed(@(x) x_0 + h * f(t,x), x_0, epsilon, max_it, ...
    log_iterations);
```

The two types of solver constructed by `be_newton` and `be_fixed` call the `newton` and `fixed` functions, which implement Newton's method and the fixed point method and were defined in Homework 1 and 2. For reference, they can be found in Listings 4 and 5 (copied from `newton.m` and `fixed.m`. I modified them slightly for this project – they no longer output the state at each iteration step, but now provide an option to log the total number of steps, which we will need later).

Listing 4: Newton's method

```
function result = newton( ...
    f, f_prime, x0, epsilon, epsilon_f, epsilon_f_prime, max_it, log_iterations ...
)

x_next = x0;

for k = 0:max_it
    xk = x_next;
    fk = f(xk);
    f_primek = f_prime(xk);

    % check f_prime not zero *before* dividing by it
    if abs(f_primek) <= epsilon_f_prime
        fprintf("Failed. f' too small.\n");
        break;
    end

    % now we can update x_next and compute Cauchy error
    x_next = xk - fk / f_primek;
    cauchy_error = abs(x_next - xk);

```

```
    if cauchy_error < epsilon || abs(fk) < epsilon_f
        break;
    end
end

if log_iterations
    fprintf("Newton iterations = %d\n", k);
end

result = xk;
```

Listing 5: The fixed point method

```
function result = fixed(g, x0, epsilon, max_it, log_iterations)

x_next = x0;

for k = 0:max_it
    xk = x_next;
    x_next = g(xk);
    cauchy_error = abs(x_next - xk);

    if cauchy_error < epsilon
        break;
    end
end

if log_iterations
    fprintf("Fixed point iterations = %d\n", k);
end

result = xk;
```

(b) In Table 1 (summarized from `p8_output.txt`) are the numerical values of $y(1)$ (that is, y_J) output by the backward Euler method using Newton's method and the fixed point method for the nonlinear equation, with step size $h \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\right\}$.

	y_J	
h	Fixed	Newton
$\frac{1}{4}$	0.235418015201	0.235440531032
$\frac{1}{8}$	0.165331430895	0.165335563041
$\frac{1}{16}$	0.154566102258	0.154571002483
$\frac{1}{32}$	0.150463955586	0.150469141443
$\frac{1}{64}$	0.148618358067	0.148639382119
$\frac{1}{128}$	0.147724451062	0.147777319070

Table 1: Backward Euler y_J values

(c) We can solve (59) directly by separation of variables.

$$\frac{y'}{y^2} = e^{2t} \implies -y^{-1} = \frac{1}{2}e^{2t} + C, \tag{61}$$

where C is a constant. Since $y(0) = .1$, it follows that $-10 = \frac{1}{2} + C$, so $C = -\frac{21}{2}$. Therefore,

$$y(t) = \frac{2}{21 - e^{2t}} \tag{62}$$

is the solution of (59).

In Table 2 (summarized from `p8_output.txt`) are the errors between $y(1)$ and the numerical approximation y_J output by the backward Euler method using Newton's method and the fixed point method for the nonlinear equation, with step size $h \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\right\}$.

We see that the error decreases by a factor of roughly 2 each time h decreases by a factor of 2. This is consistent with backward Euler method's linear convergence: $|y_J - y(1)| = \mathcal{O}(h)$.

	$\lvert y(1) - y_J \rvert$	
h	Fixed	Newton
$\frac{1}{4}$	8.847743e-02	8.849995e-02
$\frac{1}{8}$	1.839085e-02	1.839498e-02
$\frac{1}{16}$	7.625522e-03	7.630422e-03
$\frac{1}{32}$	3.523375e-03	3.528561e-03
$\frac{1}{64}$	1.677777e-03	1.698801e-03
$\frac{1}{128}$	7.838704e-04	8.367384e-04

Table 2: Backward Euler errors at $t = b = 1$

(d) We can use the `log_iterations` option of our `newton` and `fixed` functions to log the number of iterations n_j used by the solvers on the jth step of the time iteration. The final number of iterations n_J for $h \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\right\}$ are given in Table 3 (summarized from `p8_output.txt`).

	n_J	
h	Fixed	Newton
$\frac{1}{4}$	61	5
$\frac{1}{8}$	9	2
$\frac{1}{16}$	5	2
$\frac{1}{32}$	4	2
$\frac{1}{64}$	3	1
$\frac{1}{128}$	2	1

Table 3: Number of nonlinear solver iterations on last time step

Looking at n_J when $h = \frac{1}{4}$ and $h = \frac{1}{8}$, we see that the fixed point method requires more iterations to converge than the Newton's method. With a small step size, the difference between y_j and y_{j+1} is larger than with a small, making the initial guess to these methods not as good. Therefore, both methods will require several steps to converge. Over the course of many steps, Newton's method should converge faster than the fixed point method because it is a second order method, and the fixed point method is only first order.

(e) Referring to Table 3 once again, we see that the number of iterations used by the nonlinear solvers on the last step ($j = J$) decreases to just a handful very quickly as h decreases. Recall the estimates for the error of the fixed point method

$$|x_k - z| \lesssim \tilde{L}^k |x_0 - z|, \tag{63}$$

where $\tilde{L}$ is the Lipchitz constant for $\tilde{g}$, and for Newton's method

$$|x_k - z| \lesssim \left(\frac{M}{2m} |x_0 - z| \right)^{2^k}, \tag{64}$$

where M and m are upper and lower bounds on $\tilde{f}'$ and $\tilde{f}''$. From these estimates, we see that the error of the fixed point method and Newton's method depends on how close the initial value x_0 is to the root z. In particular, if x_0 is closer to z, then the bound goes below a fixed tolerance ε in fewer steps, suggesting that the methods both use fewer steps when their initial point is closer to the target value z.

On the last step of the iteration, we choose $x_0 = y_{J-1}$ as the initial point of the iteration to solve for $z = y_J$. Since $|y_{J-1} - y(1-h)| = \mathcal{O}(h)$ by the first order convergence of the backward Euler method, and since $y(1-h) \to y(1)$ as $h \to 0$ by the continuity of y, it follows that $y_{J-1} \to y(1)$ as $h \to 0$. On the other hand, the first order convergence of the backward Euler method also implies that $|y_J - y(1)| = \mathcal{O}(h)$. Therefore, $y_{J-1} \to y_J$ as $h \to 0$. That is, as we choose a smaller step size, our initial guess y_{J-1} approaches the root y_J. By the discussion of the estimates above, this means that the nonlinear solvers should use fewer iterations on the last time step as $h \to 0$, which is exactly what we observe numerically.

Problem 9.

(a) We consider using the backward Euler method to solve

$$y' = e^{2t} y^2, \qquad y(0) = 0.2, \qquad 0 \le t \le 1. \tag{65}$$

Repeating the separation of variables in Problem 8 (c), we can find that

$$y(t) = \frac{2}{11 - e^{2t}}. \tag{66}$$

Running the same experiments from Problem 8 (checking the error at $t = 1$ and the number of fixed point and Newton iterations for various values of h), I notice that the nonlinear solvers fail to converge for smaller step sizes ($\frac{1}{4}$, $\frac{1}{8}$, and $\frac{1}{16}$), and generally the error between y_J and $y(1)$ is higher than it was in Problem 8, despite all other parameters being equal (see `p9_output.txt` for details). In Figure 1, we see that the numerical solution (blue, $h = \frac{1}{128}$) seems to be close to the true solution (orange), but there is still a noticeable error at $t = 1$.

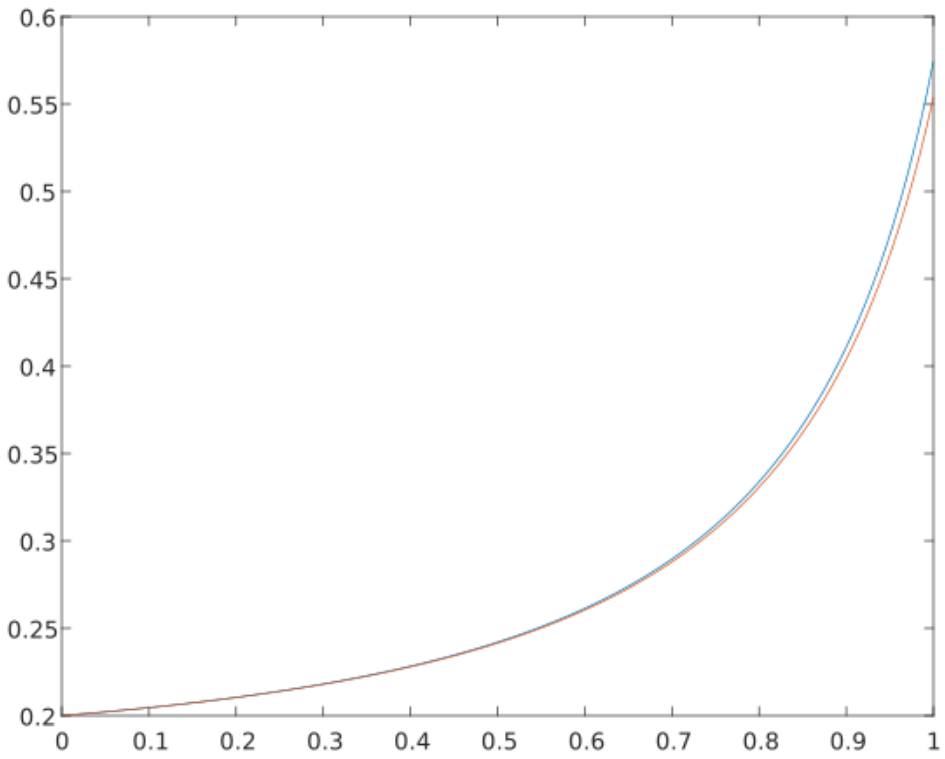

Figure 1: The solution of (65) (orange) and the numerical solution with $h = \frac{1}{128}$ (blue)

(b) We consider using the backward Euler method to solve

$$y' = e^{4t}y^2, \qquad y(0) = 0.1, \qquad 0 \le t \le 1. \tag{67}$$

Repeating the separation of variables in Problem 8 (c), we can find that

$$y(t) = \frac{4}{41 - e^{4t}}. \tag{68}$$

Noting that $y(t) \to \infty$ as $t \to \frac{\log(41)}{4} \approx 0.9284 < 1$, we see that the solution y of (67) actually only exists on an interval $\left[0, \frac{\log(41)}{4}\right)$, so when using the backward Euler method to solve on $[0, 1]$, we expect something bad (or at least unpredictable) to happen near $t = 0.9284$.

Indeed, running the numerical simulation, the numerical values start off approximating y well, but then fail around $t = 0.9284$. Smaller values of h allow the approximation to stay good for longer, but none of them make it past $t = 0.9284$. In Figure 2 we can see the numerical solution with $h = \frac{1}{128}$ using Newton's method as the nonlinear solver. In Figure 3, we can see that the numerical solution stays faithful a little longer when $h = \frac{1}{1000}$.

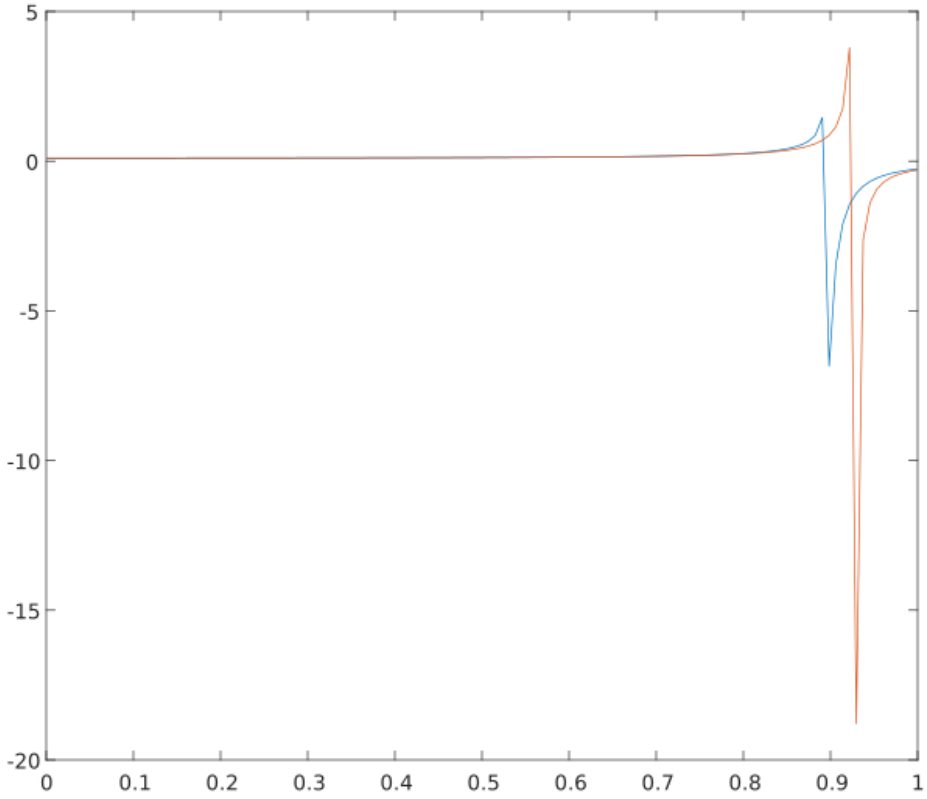

Figure 2: The solution of (67) (orange) and the numerical solution with $h = \frac{1}{128}$ (blue), using Newton's method as the nonlinear solver.

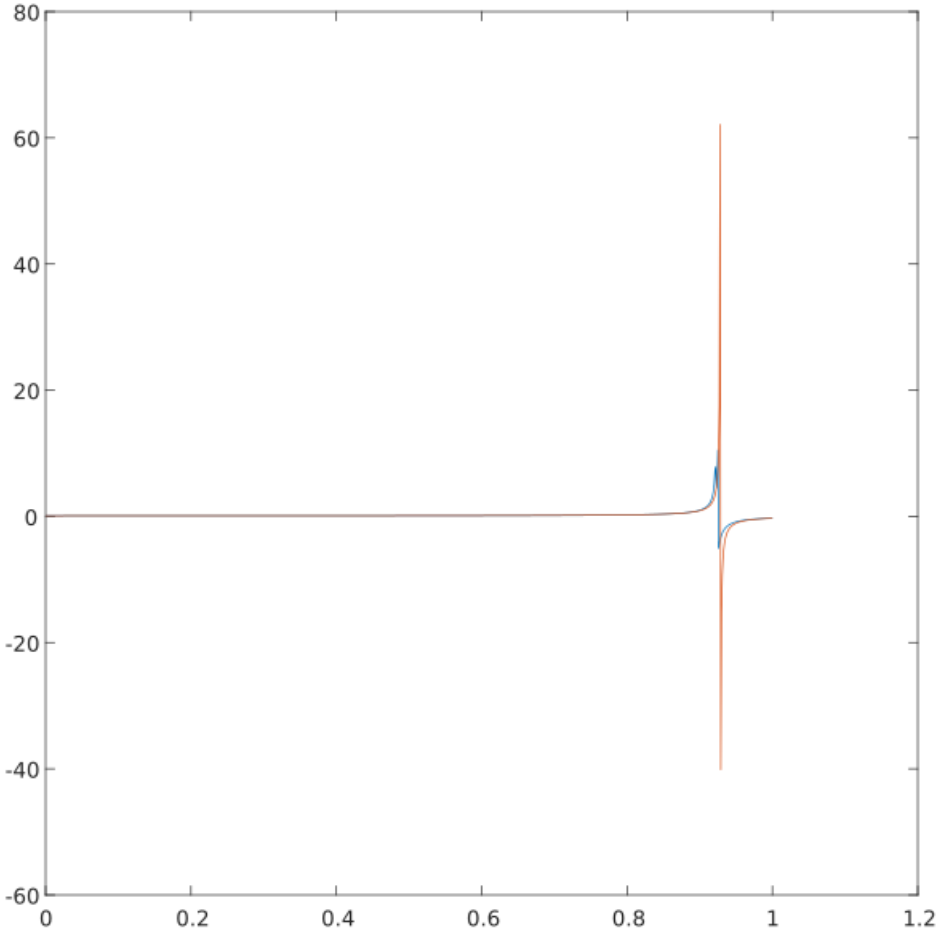

Figure 3: The solution of (67) (orange) and the numerical solution with $h = \frac{1}{1000}$ (blue), using Newton's method as the nonlinear solver.

Using the fixed point method, the numerical solution blows up near the singularity (see Figure 4). Interestingly, however, when using Newton's method and a sufficiently small step size h, the numerical solution doesn't blow up but jumps down near $t = 0.9284$ and afterwards "recovers" and starts approximating a solution of the ODE again (albeit with a different initial condition). Newton's method fails to converge only on one step of the iteration when this happens. In the blow-up cases, the nonlinear solvers fail to converge on every step after $t = 0.9284$ (see `p9_output.txt` for details).

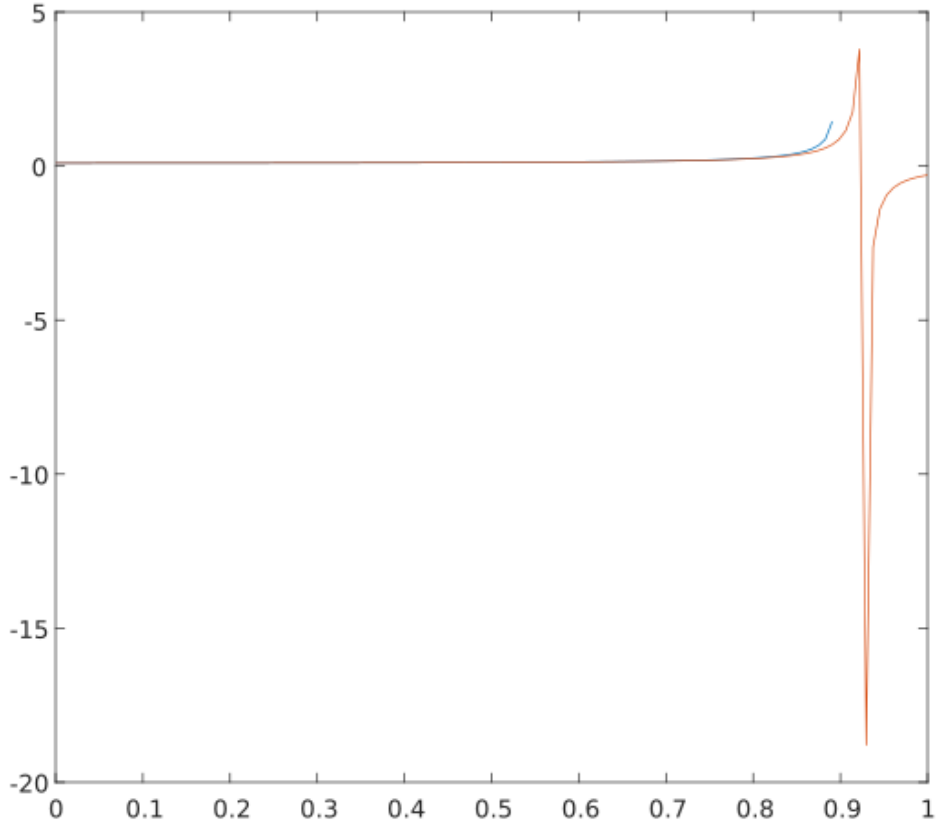

Figure 4: The solution of (67) (orange) and the numerical solution with $h = \frac{1}{128}$ (blue), using the *fixed point* method as the nonlinear solver. The numerical solution blows up to infinity near $t = 0.9284$, so its graph is cut off on the step where it blew up.