MA261 Assignment 1 Tom Dalby U1903831 Part 1.

1.1. Denote the solution to the equation as $\underline{Y}$.

$$\underline{Y}'(t) = \lambda \underline{Y}(t) \quad \text{so} \quad \underline{Y}(t) = Ae^{\lambda t}.$$
$$\underline{Y}(0) = A$$
Thus, $\underline{Y}(t) = \underline{Y}(0)e^{\lambda t}$.

We then have that
$$T_n = \underline{Y}(t_{n+1}) - \underline{Y}(t_n) - P(\lambda h)\underline{Y}(t_n).$$
$$= \underline{Y}(0)e^{\lambda t_{n+1}} - \underline{Y}(0)e^{\lambda t_n} - P(\lambda h)\underline{Y}(0)e^{\lambda t_n}$$

Notice $\forall \mu \in \mathbb{R}$, $e^{\mu t}-1 \in C^{\infty} \subseteq C^{m+1}$ so, by Taylor's Theorem,

$\cancel{P(\lambda)}$ $e^{\lambda h}-1 = P(\lambda h) + R_m(\lambda h)$ where $R_m(\lambda h) = O(h^{m+1})$ is the remainder term as $P(\lambda h)$ is the m^{th} order Taylor polynomial of $e^{\lambda h}-1$ and λ is a constant independent of hs.
~~So $R_m(\lambda h)$~~

So, subbing in $P(\lambda h) = e^{\lambda h} - 1 \cancel{- O(h^{m+1})} - R_m(\lambda h)$ into the equation for T_n, we obtain

$$T_n = \underline{Y}(0)e^{\lambda t_{n+1}} - \underline{Y}(0)e^{\lambda t_n} + \underline{Y}(0)e^{\lambda t_n} - \underline{Y}(0)e^{\lambda t_n}e^{\lambda h}$$
$$+ R_m(\lambda h)\underline{Y}(0)e^{\lambda t_n}$$
$$= \underline{Y}(0)e^{\lambda t_{n+1}} - \underline{Y}(0)e^{\lambda(t_n+h)} + R_m(\lambda h)\underline{Y}(0)e^{\lambda t_n}$$

But $t_{n+1} = t_n + h$ by definition, thus

$$T_n = R_m(\lambda h)\underline{Y}(0)e^{\lambda t_n}$$

t_n is fixed (t_{n+1} is the variable dependent on h, not t_n) thus we can regard $\underline{Y}(0)e^{\lambda t_n}$ as ~~a con~~ independent of h.
Therefore, by definition of the remainder term $R_m(\lambda h) = O(h^{m+1})$ as λ is independent of h, we get
$$T_n = O(h^{m+1}).$$

3

1.2. Fix some $h > 0$ and $\theta \in [0,1]$.

Then for some $x \in \mathbb{R}$ define $y = x + \theta h$.

It follows, as $u \in C^3(\mathbb{R})$, that there exists $\xi_1 \in [x, x+\theta h]$ and $\xi_2 \in [x+\theta h, x+h]$ s.t.

$$u(x) = u(y) + u'(y)(-\theta h) + \frac{1}{2}u''(y)(\theta h)^2 + \frac{1}{6}u'''(\xi_1)(-\theta h)^3$$

$$u(x+h) = u(y) + u'(y)(1-\theta)h + \frac{1}{2}u''(y)(1-\theta)^2h^2 + \frac{1}{6}u'''(\xi_2)(1-\theta)^3h^3$$

Thus, $\frac{u(x+h)-u(x)}{h}$

$$= \frac{1}{h}\Big[u'(y)h + \frac{1}{2}u''(y)h^2 - u''(y)\theta h^2 + \frac{1}{6}u'''(\xi_2)(1-\theta)^3h^3 - \frac{1}{6}u'''(\xi_1)(-\theta h)^3\Big]$$

$$= u'(y) + \left(\frac{1}{2}-\theta\right)u''(y)h + \frac{1}{6}u'''(\xi_2)(1-\theta)^3h^2 - \frac{1}{6}u'''(\xi_1)(-\theta)^3h^2$$

But recalling $y = x + \theta h$, $u'(y) = \frac{d}{dx}u(x+\theta h)$ so

$$\frac{d}{dx}u(x+\theta h) = \frac{u(x+h)-u(x)}{h} + \left(\theta - \frac{1}{2}\right)u''(x+\theta h)h + \frac{1}{6}u'''(\xi_1)(-\theta)^3h^2 - \frac{1}{6}u'''(\xi_2)(1-\theta)^3h^2.$$

~~Then if $\theta \neq \frac{1}{2}$, $(\theta-\frac{1}{2})u''(x+\theta h)h$ does not vanish~~

Both u'' and u''' are continuous functions on the closed interval $[x,h]$ hence are bounded. So regardless of our earlier choice of $\theta \in [0,1]$, $|u''(x+\theta h)| \le M_2$ for some $M_2 > 0$ ~~thus~~ and ~~similarly~~ also $|\frac{1}{6}u'''(\xi_1)(-\theta)^3h^2| \le M_3$

~~and~~ $|\frac{1}{6}u'''(\xi_1)(-\theta)^3| \le C_3|(-\theta)^3| \le C_3$ where $C_3 = |\frac{1}{6}u'''(\xi_1)|$

and finally $|\frac{1}{6}u'''(\xi_2)(1-\theta)^3| \le C_3'|(1-\theta)^3| \le C_3'$ where

$C_3' = |\frac{1}{6}u'''(\xi_2)|$.

So we obtain the result $|(\theta - \frac{1}{2})u''(x+\theta h)h| \leq |\theta - \frac{1}{2}|M_2 h$

$\leq M_2 h$ for $M_2 > 0$ independent of h.

so $(\theta - \frac{1}{2})u''(x+\theta h)h = O(h)$.

Also, $|\frac{1}{6}u'''(\xi_1)(-\theta)^3 h^2| \leq C_3 h^2$, $C_3 > 0$ independent of h so $\frac{1}{6}u'''(\xi_1)(-\theta)^3 h^2 = O(h^2)$.

Finally, $|\frac{1}{6}u'''(\xi_2)(1-\theta)^3 h^2| \leq C_3' h^2$, $C_3' > 0$ independent of h so $\frac{1}{6}u'''(\xi_2)(1-\theta)^3 h^2 = O(h^2)$.

If $\theta \neq \frac{1}{2}$, $(\theta - \frac{1}{2})u''(x+\theta h)h$ does not vanish (necessarily) so

$$\frac{d}{dx}u(x+\theta h) = \frac{u(x+h) - u(x)}{h} + (\theta - \frac{1}{2})u''(x+\theta h)h + \frac{1}{6}u'''(\xi_1)(-\theta)^3 h^2 - \frac{1}{6}u'''(\xi_2)(1-\theta)^3 h^2$$

6/6

$$= \frac{u(x+h) - u(x)}{h} + O(h) + O(h^2) - O(h^2)$$

$$= \frac{u(x+h) - u(x)}{h} + O(h) \text{ by big-O rules.}$$

If on the other hand $\theta = \frac{1}{2}$, we can find an even higher p as $(\theta - \frac{1}{2})u''(x+\theta h)h = 0$ thus

$$\frac{d}{dx}u(x+\theta h) = \frac{u(x+h) - u(x)}{h} + \frac{1}{6}u'''(\xi_1)(-\theta)^3 h^2 - \frac{1}{6}u'''(\xi_2)(1-\theta)^3 h^2$$

$$= \frac{u(x+h) - u(x)}{h} + O(h^2) - O(h^2)$$

$$= \frac{u(x+h) - u(x)}{h} + O(h^2) \text{ by big-O rules.}$$

1.3. We know $h \geq 0$. $\tau(t;0) = \underline{Y}(t) - \underline{Y}(t) - 0 = 0$ so $|\tau(t;0)| = 0$ so from now can assume $h > 0$.

So,

$$\frac{\tau(t;h)}{h} = \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - \varphi(t, \underline{Y}(t); h)$$

$$= \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - f(t,y) + f(t,y) - \varphi(t, \underline{Y}(t); h),$$

so by the triangle inequality,

$$\frac{|\tau(t;h)|}{h} \leq \left| \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - f(t, \underline{Y}(t)) \right| + \left| f(t, \underline{Y}(t)) - \varphi(t, \underline{Y}(t); h) \right|$$

Fix $\varepsilon > 0$

◦ Recall $f(t, \underline{Y}(t)) = \underline{Y}'(t)$ so

$$\left| \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - f(t, \underline{Y}(t)) \right| = \left| \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - \underline{Y}'(t) \right|$$

As $\underline{Y} \in C'(0,T)$, by Taylor's Theorem $\exists\, \xi_1 \in [t, t+h]$ s.t.

$$\underline{Y}(t+h) = \underline{Y}(t) + \underline{Y}'(\xi_1) h, \text{ thus}$$

$$\left| \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - \underline{Y}'(t) \right| = |\underline{Y}'(\xi_1) - \underline{Y}'(t)|.$$

$\underline{Y}'$ is continuous on $(0,T)$ so $\exists H_1$ s.t. ~~$\forall\, 0<h<H_1$~~, $\forall\, x \in (t, H_1)$,

$|\underline{Y}'(x) - \underline{Y}'(t)| < \frac{\varepsilon}{2}$. If we let ~~$h<H_1$~~ $0 < h < H_1$, it follows that $\xi_1 \in (t,h) \subseteq (t, H_1)$ so $|\underline{Y}'(\xi_1) - \underline{Y}'(t)| < \frac{\varepsilon}{2}$.

◦ Recall $f(t, \underline{Y}(t)) = \varphi(t, \underline{Y}(t); 0)$, so

$$|f(t, \underline{Y}(t)) - \varphi(t, \underline{Y}(t); h)| = |\varphi(t, \underline{Y}(t); 0) - \varphi(t, \underline{Y}(t); h)|$$

φ is continuous in the h argument so $\exists H_2 > 0$ s.t. $\forall x \in (0, H_2)$,

$$|\varphi(t, \underline{Y}(t); x) - \varphi(t, \underline{Y}(t); 0)| < \frac{\varepsilon}{2}.$$

So if we let $H = \min\{H_1, H_2\}$ and then for any $h < H$, ~~$h \in (0$~~ and any $t \in [0, T-h]$, as $h \in (0, H_2)$ and $h \in (t, H_1)$, we get

$$\frac{|T(t;h)|}{h} \leq \left| \frac{\underline{Y}(t+h) - \underline{Y}(t)}{h} - f(t, \underline{Y}(t)) \right| + \left| f(t, \underline{Y}(t)) - \varphi(t, \underline{Y}(t); h) \right|$$

$$< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

i.e. if $h < H$ and $t \in [0, T-h]$ then

$$|T(t;h)| < \varepsilon h.$$

6/6

Adi Jayakumar (1905687), Tom Dalby (1903831)

MA261 Modelling and Differential Equations Assignment sheet 1

2.0)
The required functions were implemented as follows:

```
def forwardEuler(f, Df, tn, yn, h):
        """Computes 1 step of the Forward Euler method"""
        return yn + h * f(tn, yn)

def evolve(phi, f, Df, t0, y0, T, N):
        """Solves the IVP y(t) = f(y(t)) with y(t0) = y0"""
        y = [y0]
        h = T/N
        tn = t0

        while(tn < T):
        curVal = phi(f, Df, tn, y[-1], h)
        tn += h
        y.append(curVal)

        return y

def computeEocs(herr):
        """Computes the EOCs of a 2 x m matrix of the form [[h1, e1], ..., [hm, em]]"""
        ans = []
        for i in range(1, len(herr)):
        curEoc = log(herr[i][1]/herr[i - 1][1])/log(herr[i][0]/herr[i - 1][0])
        ans.append(curEoc)
        return ans
```

2.1)

The Python function that was passed into '*evolve*' and the exact solution in the case $c = 1.5$ were implemented as follows:

```
def y_prime(_, y):
        """ODE system first parameter is time t"""
        return (1.5 - y) ** 2

def exact_soln(t):
        """Exact solution to the above ode with c = 1.5"""
        return (1 + t * 1.5 * (1.5 - 1))/(1 + t * (1.5 - 1))
```

We then had a function '*compute_eoc_matrix_and_errors*' which computed the matrix to be passed into '*computeEocs*' and the errors at time T for a given solution method which was implemented as follows

```
def compute_eoc_matrix_and_errors(method_func):
        """Computes the matrix fed into 'computeEocs' for the method 'method_func'"""
        N0 = 20
        T = 10
        t0 = 0
        y0 = 1

        eoc_m = []
        exact_val = exact_soln(10)
        errors = []

        # Computing EOCs and errors for the method
        for _ in range(0, 11):
                y = evolve(method_func, y_prime, lambda y: -2 * (1.5 - y), t0, y0, T, N0)
                cur_error = abs(exact_val - y[-1])
                errors.append(cur_error)
                cur_h = T / N0
                eoc_m.append([cur_h, cur_error])
                N0 *= 2

        return eoc_m, errors
```

For this question, the above function was called with the argument being '*forwardEuler*' and list containing the data for the EOC was passed into '*computeEocs*' and the '*errors*' list was printed out to console.

We observed that the experimental order of convergence was converging to 1 and therefore concluded that the Forward Euler method converges linearly in the case of this function. Furthermore, we were able to see that the errors seemed to roughly halve as the step size halved as well.

The observed EOCs and errors were

j such that *h_j = T/N_02^j*	EOCs	Errors
0	N/A	0.006430
1	1.027162	0.003155
2	1.010797	0.001566
3	1.004920	0.000780
4	1.002359	0.000389
5	1.001156	0.000195
6	1.000572	0.000097
7	1.000285	0.000049
8	1.000142	0.000024
9	1.000071	0.000012
10	1.000035	0.000006

5

2.2)
We implemented the method detailed in the question as the following Python functions

```
def method_2(f, Df, tn, yn, h):
        """Computes 1 step of the second method as detailed
        in Q2.2 caching the value of f(tn, yn) for efficiency"""
        val = f(tn, yn)
        return yn + (h / 2) * (val + f(tn + h, yn + h * val))
```

In order to generate the convergence data for this method, '*compute_eoc_matrix_and_errors*' was called with '*method_2*' as an argument. The data it generated was as follows

j such that *h_j = T/N_02^j*	EOCs	Errors
0	N/A	0.000411
1	2.085088	0.000097
2	2.049271	0.000023
3	2.025652	0.000006
4	2.013009	0.000001
5	2.006542	0.000000
6	2.003280	0.000000
7	2.001642	0.000000
8	2.000822	0.000000
9	2.000412	0.000000
10	2.000204	0.000000

5

We concluded that as the EOC seemed to be converging to *2* that this method converges quadratically. This also seems to follow in the errors, which seem to decease by a factor of 4 each iteration. The comparison of the 2 methods is presented below in Q2.3.

2.3)
The Forward Euler method has an experimental order of convergence of *1* meaning that for this particular function, it converges linearly. On the other hand, the method detailed in Q2.2 has an experimental order of convergence of *2* for this particular function and therefore converges quadratically. However, the Forward Euler method requires N evaluations of the derivative function f and the second method requires $2N$ evaluations of the derivative function f (once optimised). They both require $O(N)$ evaluations, and therefore one can conclude that the second method is better as it gives an extra order of convergence while only requiring a scalar multiple increase in the number of evaluations.

5

Plot # function calls vs. errors