Math 151B Final

Nick Monozon

September 8, 2022

Question 2

Consider the initial value problem given by

$$y' = \frac{y^2 + y}{t},\ 1 \le t \le 3,\ y(1) = -2.$$

We will approximate the solution to this IVP using the four-step Adams-Bashforth method with a step size of $h = 0.2$, hence requiring $N = (b-a)/h = (3-1)/0.2 = 10$ iterations. Using the Python implementation of this method as attached below (ab4.py), we obtain as output the following table of approximations.

i	t_i	w_i
0	1.0	-2.0000000
1	1.2	-1.7142452
2	1.4	-1.5555229
3	1.6	-1.4545197
4	1.8	-1.4040130
5	2.0	-1.3489404
6	2.2	-1.3149037
7	2.4	-1.2781767
8	2.6	-1.2538815
9	2.8	-1.2302235
10	3.0	-1.2124970

ab4.py

```
# Imports
import numpy as np
import pandas as pd
import math

# For more decimal places
pd.set_option("display.precision", 7)

# Function
f = lambda t, y: (y**2 + y)/t
# Left endpoint
a = 1
# Right endpoint
b = 3
# Step size
h = 0.2
# First initial condition
alpha = -2

N = int((b-a/h))
t = np.arange(a, b+h, h)
w = np.zeros(len(t))
w[0] = alpha

```

```python
# For order 4 Runge-Kutta approximations
rk = np.zeros(4)
rk[0] = alpha

# Runge-Kutta order 4 approximations for alpha1, alpha2, alpha3
for i in range(0, 3):
  k1 = h*f(t[i], rk[i])
  k2 = h*f(t[i] + h/2, rk[i] + 1/2*k1)
  k3 = h*f(t[i] + h/2, rk[i] + 1/2*k2)
  k4 = h*f(t[i+1], rk[i] + k3)
  rk[i+1] = rk[i] + 1/6*(k1 + 2*k2 + 2*k3 + k4)

w[0] = alpha
w[1] = rk[1]
w[2] = rk[2]
w[3] = rk[3]

# Adam-Bashforth 4-step approximations
for i in range(3, len(t) - 1):
  w[i+1] = w[i] + h/24*(55*f(t[i], w[i]) - 59*f(t[i-1], w[i-1]) + 37*f(t[i-2], w[i-2]) - 9*f(t[i
    -3], w[i-3]))

# Output
df = pd.DataFrame({('t_i'): t, 'w_i': w})
print(df)
```

Question 3

Consider the boundary value problem

$$y'' = -\frac{4}{x}y' - \frac{2}{x^2}y + \frac{2\ln x}{x^2},\ 1 \le x \le 2,\ y(1) = \frac{1}{2},\ y(2) = \ln 2.$$

We will approximate the solution to this BVP using the linear shooting method with a step size of $h = 0.1$, performing a total of $N = (b-a)/h = (2-1)/0.1 = 10$ iterations. Using the Python implementation of the linear shooting method (linear_shooting.py) attached below, we obtain as output the following table of approximations.

i	x_i	u_{1i}	v_{1i}	w_i
0	1.0	0.50000000	0.00000000	0.50000000
1	1.1	0.49612933	0.08262895	0.57876151
2	1.2	0.48786815	0.13886862	0.62674220
3	1.3	0.47833198	0.17749440	0.65583333
4	1.4	0.46911769	0.20406284	0.67318852
5	1.5	0.46101418	0.22220559	0.68322847
6	1.6	0.45437322	0.23436059	0.68874298
7	1.7	0.44930890	0.24220221	0.69152058
8	1.8	0.44580769	0.24690313	0.69272048
9	1.9	0.44378996	0.24929868	0.69309839
10	2.0	0.44314476	0.24999263	0.69314718

linear_shooting.py

```python
# Imports
import numpy as np
import pandas as pd
import math

# For more decimal places
pd.set_option("display.precision", 8)

# Function
p = lambda x: -4/x
q = lambda x: -2/x**2
r = lambda x: 2*np.log(x)/x**2
# Left endpoint
a = 1
# Right endpoint
b = 2
# Step size
h = 0.1

# Left endpoint condition
alpha = 1/2
# Right endpoint condition
beta = np.log(2)

N = int((b-a)/h)
xs = np.arange(a, b+h, h)
w1 = np.zeros(len(xs))
u1 = np.zeros(len(xs))
u2 = np.zeros(len(xs))
v1 = np.zeros(len(xs))
v2 = np.zeros(len(xs))
w1[0] = alpha

# Initial conditions for IVPs
u1[0] = alpha
u2[0] = 0
v1[0] = 0
v2[0] = 1

```

```
# Approximating equations for system of ODEs
for i in range(0, len(xs) - 1):

  x = a + i*h

  k11 = h*u2[i]
  k12 = h*(p(x)*u2[i] + q(x)*u1[i] + r(x))
  k21 = h*(u2[i] + 1/2*k12)
  k22 = h*(p(x+h/2)*(u2[i] + 1/2*k12) + q(x+h/2)*(u1[i] + 1/2*k11) + r(x+h/2))
  k31 = h*(u2[i] + 1/2*k22)
  k32 = h*(p(x+h/2)*(u2[i]+1/2*k22) + q(x+h/2)*(u1[i] + 1/2*k21) + r(x+h/2))
  k41 = h*(u2[i] + k32)
  k42 = h*(p(x+h)*(u2[i] + k32) + q(x+h)*(u1[i] + k31) + r(x+h))
  u1[i+1] = u1[i] + 1/6*(k11 + 2*k21 + 2*k31 + k41)
  u2[i+1] = u2[i] + 1/6*(k12 + 2*k22 + 2*k32 + k42)

  kp11 = h*v2[i]
  kp12 = h*(p(x)*v2[i] + q(x)*v1[i])
  kp21 = h*(v2[i] + 1/2*kp12)
  kp22 = h*(p(x+h/2)*(v2[i] + 1/2*kp12) + q(x+h/2)*(v1[i] + 1/2*kp11))
  kp31 = h*(v2[i] + 1/2*kp22)
  kp32 = h*(p(x+h/2)*(v2[i] + 1/2*kp22) + q(x+h/2)*(v1[i] + 1/2*kp21))
  kp41 = h*(v2[i] + kp32)
  kp42 = h*(p(x+h)*(v2[i] + kp32) + q(x+h)*(v1[i] + kp31))
  v1[i+1] = v1[i] + 1/6*(kp11 + 2*kp21 + 2*kp31 + kp41)
  v2[i+1] = v2[i] + 1/6*(kp12 + 2*kp22 + 2*kp32 + kp42)

# Initializing arrays
w1 = np.zeros(len(xs))
w2 = np.zeros(len(xs))
W1 = np.zeros(len(xs))
W2 = np.zeros(len(xs))

# Conditions
w1[0] = alpha
w2[0] = (beta - u1[-1])/v1[-1]

# Approximations
for i in range(0, len(xs)):
  W1[i] = u1[i] + w2[0]*v1[i]
  # For first derivative (not outputted here)
  W2[i] = u2[i] + w2[0]*v2[i]

# Output
df = pd.DataFrame({('x_i'): xs, 'u_1i': u1, 'v_1i': v1, 'w_i': W1})
df
```

Question 4

Consider the matrix

$$\begin{bmatrix} 4 & 2 & 1 \\ 0 & 3 & 2 \\ 1 & 1 & 4 \end{bmatrix}$$

along with $\boldsymbol{x}^{(0)} = (1,2,1)^\top$. We will find the first eigenvalue-eigenvector approximations for the first three iterations of the power method. Performing the first iteration:

$$\begin{aligned}
\boldsymbol{x}^{(0)} &= \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, p_0 = 2 \\
\boldsymbol{y}^{(1)} = \boldsymbol{A}\boldsymbol{x}^{(0)} &= \begin{bmatrix} 4 & 2 & 1 \\ 0 & 3 & 2 \\ 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 8 \\ 7 \end{bmatrix} \\
\mu^{(1)} &= y^{(1)}_{p_0} = 8, p_1 = 1 \\
\boldsymbol{x}^{(1)} &= \frac{1}{y^{(1)}_{p_1}} \boldsymbol{y}^{(1)} = \frac{1}{9} \begin{bmatrix} 9 \\ 8 \\ 7 \end{bmatrix} = \begin{bmatrix} 1 \\ 8/9 \\ 7/9 \end{bmatrix}
\end{aligned}$$

So we obtain $\boldsymbol{x}^{(1)} = (1, 8/9, 7/9)^\top$ and $\mu^{(1)} = 8$. Onto the second iteration:

$$\begin{aligned}
\boldsymbol{x}^{(1)} &= \begin{bmatrix} 1 \\ 8/9 \\ 7/9 \end{bmatrix}, p_1 = 1 \\
\boldsymbol{y}^{(2)} = \boldsymbol{A}\boldsymbol{x}^{(1)} &= \begin{bmatrix} 4 & 2 & 1 \\ 0 & 3 & 2 \\ 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 8/9 \\ 7/9 \end{bmatrix} = \begin{bmatrix} 59/9 \\ 38/9 \\ 5 \end{bmatrix} \\
\mu^{(2)} &= y^{(2)}_{p_1} = 59/9, p_2 = 1 \\
\boldsymbol{x}^{(2)} &= \frac{1}{y^{(2)}_{p_2}} \boldsymbol{y}^{(2)} = \frac{9}{59} \begin{bmatrix} 59/9 \\ 38/9 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 38/59 \\ 45/59 \end{bmatrix}
\end{aligned}$$

So we obtain $\boldsymbol{x}^{(2)} = (1, 38/59, 45/59)^\top$ and $\mu^{(2)} = 59/9$. Finally, proceeding with the third iteration:

$$\begin{aligned}
\boldsymbol{x}^{(2)} &= \begin{bmatrix} 1 \\ 38/59 \\ 45/59 \end{bmatrix}, p_2 = 1 \\
\boldsymbol{y}^{(3)} = \boldsymbol{A}\boldsymbol{x}^{(2)} &= \begin{bmatrix} 4 & 2 & 1 \\ 0 & 3 & 2 \\ 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 38/59 \\ 45/59 \end{bmatrix} = \begin{bmatrix} 357/59 \\ 204/59 \\ 277/59 \end{bmatrix} \\
\mu^{(3)} &= y^{(3)}_{p_2} = 357/59, p_3 = 1 \\
\boldsymbol{x}^{(3)} &= \frac{1}{y^{3}_{p_3}} \boldsymbol{y}^{(3)} = \frac{59}{357} \begin{bmatrix} 357/59 \\ 204/59 \\ 277/59 \end{bmatrix} = \begin{bmatrix} 1 \\ 4/7 \\ 277/357 \end{bmatrix}
\end{aligned}$$

And we therefore have that $\boldsymbol{x}^{(3)} = (1, 4/7, 277/357)^\top$ and $\mu^{(3)} = 357/59$.

Question 5

Consider the multistep method

$$w_{i+1} = 2w_{i-1} - w_i + \frac{h}{2}\left[5f(t_i, w_i) + f(t_{i-1}, w_{i-1})\right]$$

for $i = 1, \ldots, N-1$ and starting values w_0, w_1 that can be used to approximate the solution to IVPs of the form

$$y' = f(t, y),\ a \le t \le b,\ y(a) = \alpha.$$

We want to find the local truncation error of this method. Per a definition in the textbook, the local truncation error for this multistep method is given by

$$\tau_{i+1}(h) = \frac{y(t_{i+1}) + y(t_i) - 2y(t_{i-1})}{h} - \frac{5}{2}f\big(t_i, y(t_i)\big) - \frac{1}{2}f\big(t_{i-1}, y(t_{i-1})\big).$$

Assume that $y \in \mathcal{C}^3$. We first find the Taylor expansion of $y(t_{i+1})$ about t_i to be

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(\xi_{i1})$$

for some $\xi_{i1} \in (t_i, t_{i+1})$. Next, we find the Taylor expansion of $y(t_{i-1})$ about t_i to be

$$y(t_{i-1}) = y(t_i) - hy'(t_i) + \frac{h^2}{2}y''(t_i) - \frac{h^3}{6}y'''(\xi_{i2})$$

for some $\xi_{i2} \in (t_{i-1}, t_i)$. Lastly, we find the Taylor expansion of $f\big(t_{i-1}, y(t_{i-1})\big)$ to be

$$f\big(t_{i-1}, y(t_{i-1})\big) = y'(t_{i-1}) = y'(t_i) - hy''(t_i) + \frac{h^2}{2}y'''(\xi_{i3})$$

for some $\xi_{i3} \in (t_{i-1}, t_i)$. Substituting these into the expression for $\tau_{i+1}(h)$, we have that

$$\begin{aligned}
\tau_{i+1}(h) &= \frac{y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(\xi_{i1}) + y(t_i) - 2\left(y(t_i) - hy'(t_i) + \frac{h^2}{2}y''(t_i) - \frac{h^3}{6}y'''(\xi_{i2})\right)}{h} \\
&\quad - \frac{5}{2}y'(t_i) - \frac{1}{2}\left(y'(t_i) - hy''(t_i) + \frac{h^2}{2}y'''(\xi_{i3})\right) \\
&= \frac{hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(\xi_{i1}) + 2hy'(t_i) - h^2y''(t_i) + \frac{h^3}{3}y'''(\xi_{i2})}{h} \\
&\quad - \frac{5}{2}y'(t_i) - \frac{1}{2}y'(t_i) + \frac{1}{2}hy''(t_i) - \frac{h^2}{4}y'''(\xi_{i3}) \\
&= y'(t_i) + \frac{h}{2}y''(t_i) + \frac{h^2}{6}y'''(\xi_{i1}) + 2y'(t_i) - hy''(t_i) + \frac{h^2}{3}y'''(\xi_{i2}) - \frac{5}{2}y'(t_i) - \frac{1}{2}y'(t_i) + \frac{1}{2}hy''(t_i) - \frac{h^2}{4}y'''(\xi_{i3}) \\
&= 0y'(t_i) + 0hy''(t_i) + \frac{h^2}{6}y'''(\xi_{i1}) + \frac{h^2}{3}y'''(\xi_{i2}) - \frac{h^2}{4}y'''(\xi_{i3}) \\
&= \frac{h^2}{6}y'''(\xi_{i1}) + \frac{h^2}{3}y'''(\xi_{i2}) - \frac{h^2}{4}y'''(\xi_{i3}) \\
&= \frac{1}{4}h^2y'''(\xi_i)
\end{aligned}$$

for some $\xi_i \in (t_{i-1}, t_{i+1})$, where we invoke the intermediate value theorem to obtain ξ_i. As such, the local truncation error for this multistep method is given by $\tau_{i+1}(h) = \frac{1}{4}h^2y'''(\xi_i)$ for $\xi_i \in (t_{i-1}, t_{i+1})$.

Question 6

Suppose we want to solve the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A}$ is an $n \times n$ nonsingular matrix. We can rewrite this linear system equivalently as $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$, which is simply compact notation for

$$\boldsymbol{F}(\boldsymbol{x}) := \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

We observe that because $f_i(x_1, \ldots, x_n) = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - b_i$ for each $i = 1, \ldots, n$, we have that $\frac{\partial f_i}{\partial x_j} = a_{ij}$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, n$. As such, it follows that

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \boldsymbol{A}.$$

Now, recall that each iterative update of Newton's Method for systems is given by

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - \boldsymbol{J}\big(\boldsymbol{x}^{(k-1)}\big)^{-1}\boldsymbol{F}\big(\boldsymbol{x}^{(k-1)}\big)$$

for $k \geq 1$. As such, for any initial approximation $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ to $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ as defined above, we have that

$$\begin{aligned} \boldsymbol{x}^{(1)} &= \boldsymbol{x}^{(0)} - \boldsymbol{J}\big(\boldsymbol{x}^{(0)}\big)^{-1}\boldsymbol{F}\big(\boldsymbol{x}^{(0)}\big) \\ &= \boldsymbol{x}^{(0)} - \boldsymbol{A}^{-1}\big(\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\big) \\ &= \boldsymbol{x}^{(0)} - \boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x}^{(0)} + \boldsymbol{A}^{-1}\boldsymbol{b} \\ &= \boldsymbol{x}^{(0)} - \boldsymbol{I}\boldsymbol{x}^{(0)} + \boldsymbol{A}^{-1}\boldsymbol{b} \\ &= \boldsymbol{x}^{(0)} - \boldsymbol{x}^{(0)} + \boldsymbol{A}^{-1}\boldsymbol{b} \\ &= \boldsymbol{A}^{-1}\boldsymbol{b} \end{aligned}$$

where we used the fact that $\boldsymbol{A}$ is nonsingular. Hence, for any initial approximation $\boldsymbol{x}^{(0)}$, Newton's Method for systems reduces to $x^{(1)} = \boldsymbol{A}^{-1}\boldsymbol{b}$ for this linear system. In other words, for any $\boldsymbol{x}^{(0)}$, the method converges to the exact solution $\boldsymbol{x}^{(1)}$ of the system in just one iteration.

Question 7

Consider the Implicit Trapezoidal method, which is given by

$$w_{i+1} = w_i + \frac{h}{2}\left[f(t_{i+1}, w_{i+1}) + f(t_i, w_i)\right]$$

for $0 \le i \le N-1$ and an initial value $w_0 = \alpha$. Applying the method to the test equation $y' = \lambda y$, we get

$$\begin{aligned}
w_{i+1} &= w_i + \frac{h}{2}\left[f(t_{i+1}, w_{i+1}) + f(t_i, w_i)\right] \\
&= w_i + \frac{h}{2}\left[\lambda w_{i+1} + \lambda w_i\right] \\
&= w_i + \frac{h\lambda}{2} w_{i+1} + \frac{h\lambda}{2} w_i \\
&= \left(1 + \frac{h\lambda}{2}\right) w_i + \frac{h\lambda}{2} w_{i+1}
\end{aligned}$$

from which we can rearrange to obtain

$$w_{i+1} - \frac{h\lambda}{2} w_{i+1} = \left(1 - \frac{h\lambda}{2}\right) w_{i+1} = \left(1 + \frac{h\lambda}{2}\right) w_i.$$

So we then have that

$$w_{i+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} w_i = \frac{2 + h\lambda}{2 - h\lambda} w_i = Q(h\lambda) w_i$$

where $Q(h\lambda) = \frac{2+h\lambda}{2-h\lambda}$. Now, we want to show that this method is A-stable. Consider the region of absolute stability R, which is defined to be

$$R = \left\{h\lambda \in \mathbb{C} : |Q(h\lambda)| < 1\right\}.$$

A method is said to be A-stable if the region of absolute stability R contains the entire left half-plane. We observe that for $\mathrm{Re}(h\lambda) < 0$, $|Q(h\lambda)| < 1$. Therefore, the region of absolute stability R for the Implicit Trapezoidal method contains the entire left half-plane, and the method is A-stable.

Question 8

Consider the multistep method

$$w_{i+1} = -3w_{i-1} + 4w_i - 2hf(t_{i-1}, w_{i-1})$$

for $i = 1, \ldots, N-1$ that can be used to approximate the solution to general IVPs of the form

$$y' = f(t, y),\ a \le t \le b,\ y(a) = \alpha.$$

We want to determine if this method is consistent, so we must first find the local truncation error. Using a definition in the textbook, the local truncation error of this method is

$$\tau_{i+1}(h) = \frac{y(t_{i+1}) - 4y(t_i) + 3y(t_{i-1})}{h} + 2f\big(t_{i-1}, y(t_{i-1})\big).$$

Assume that $y \in \mathcal{C}^3$. We now find the Taylor expansion of $y(t_{i+1})$ about t_i to be

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(\xi_{i1})$$

for some $\xi_{i1} \in (t_i, t_{i+1})$. Next, we find the Taylor expansion of $y(t_{i-1})$ about t_i to be

$$y(t_{i-1}) = y(t_i) - hy'(t_i) + \frac{h^2}{2}y''(t_i) - \frac{h^3}{6}y'''(\xi_{i2})$$

for some $\xi_{i2} \in (t_{i-1}, t_i)$. Finally, the Taylor expansion of $f\big(t_{i-1}, y(t_{i-1})\big)$ is

$$f\big(t_{i-1}, y(t_{i-1})\big) = y'(t_i) = y'(t_i) - hy''(t_i) + \frac{h^2}{2}y'''(\xi_{i3})$$

for some $\xi_{i3} \in (t_{i-1}, t_i)$. Substituting these expansions into the expression for $\tau_{i+1}(h)$, we find that

$$\begin{aligned}
\tau_{i+1}(h) &= \frac{y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(\xi_{i1}) - 4y(t_i) + 3\left(y(t_i) - hy'(t_i) + \frac{h^2}{2}y''(t_i) - \frac{h^3}{6}y'''(\xi_{i2})\right)}{h} \\
&\quad + 2\left(y'(t_i) - hy''(t_i) + \frac{h^2}{2}y'''(\xi_{i3})\right) \\
&= \frac{hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(\xi_{i1}) - 3hy'(t_i) + \frac{3h^2}{2}y''(t_i) - \frac{h^3}{2}y'''(\xi_{i2})}{h} \\
&\quad + 2y'(t_i) - 2hy''(t_i) + h^2y'''(\xi_{i3}) \\
&= y'(t_i) + \frac{h}{2}y''(t_i) + \frac{h^2}{6}y'''(\xi_{i1}) - 3y'(t_i) + \frac{3h}{2}y''(t_i) - \frac{h^2}{2}y'''(\xi_{i2}) + 2y'(t_i) - 2hy''(t_i) + h^2y'''(\xi_{i3}) \\
&= 0y'(t_i) + 0y''(t_i) + \frac{h^2}{6}y'''(\xi_{i1}) - \frac{h^2}{2}y'''(\xi_{i2}) + h^2y'''(\xi_{i3}) \\
&= \frac{2}{3}h^2y'''(\xi_i)
\end{aligned}$$

for some $\xi_i \in (t_{i-1}, t_{i+1}$ where we use the intermediate value theorem to obtain ξ_i. Hence, the local truncation error for this multistep method is $\tau_{i+1}(h) = \frac{2}{3}h^2y'''(\xi_i)$ for some $\xi_i \in (t_{i-1}, t_{i+1})$. We can now determine if this method is consistent. The method is said to be consistent if $|\tau_{i+1}(h)| \to 0$ as $h \to 0$. Assuming that the one-step method used to generate the starting value w_1 is consistent, we have that

$$\lim_{h\to 0}|\tau_{i+1}(h)| = \lim_{h\to 0}\left|\frac{2}{3}y'''(\xi_i)h^2\right| = \lim_{h\to 0}|h^2|\left|\frac{2}{3}y'''(\xi_i)\right| = 0$$

so the multistep method is consistent. Let's now investigate the stability and convergence of this method. By Definition 5.22, the characteristic polynomial associated with a general multistep method of the form

$$\begin{aligned}
&w_0 = \alpha, w_1 = \alpha_1, \ldots, w_{m-1} = \alpha_{m-1} \\
&w_{i+1} = a_{m-1}w_i + a_{m-2}w_{i-1} + \cdots + a_0w_{i+1-m} + hF(t_i, h, w_{i+1}, w_i, \ldots, w_{i+1-m})
\end{aligned}$$

is given by

$$P(\lambda) = \lambda^m - a_{m-1}\lambda^{m-1} - \cdots - a_1\lambda - a_0.$$

In this case, we see that $m = 2$, $a_1 = 4$, and $a_0 = 3$. The characteristic polynomial is then

$$P(\lambda) = \lambda^2 - a_1\lambda - a_0 = \lambda^2 - 4\lambda + 3.$$

We find the roots of $P(\lambda)$ to be

$$\lambda = \frac{4 \pm \sqrt{(-4)^2 - 4(1)(3)}}{2(1)} = \frac{4 \pm \sqrt{16-12}}{2} = \frac{4 \pm \sqrt{4}}{2} = \frac{4 \pm 2}{2} \Rightarrow \lambda_1 = 1, \lambda_2 = 3.$$

This multistep method is said to satisfy the root condition if $|\lambda_i| \leq 1$ for $i \in \{1,2\}$. However, we have that $|\lambda_2| = 3 > 1$, so the method does not satisfy the root condition. Thus, by Definition 5.23, this method is unstable. Since it's unstable, it's also not convergent. To summarize, this multistep method is consistent but neither stable nor convergent.