Homework 2 of Computational Mathematics

Chang, Yung-Hsuan
111652004
Department of Applied Mathematics

April 1, 2024

Problem 1. Use a fixed-point iteration method to determine a solution accurate to within 10^{-2} for $x^3 - x - 1 = 0$ on $[1, 2]$. Use $p_0 = 1$.

Solution. By Algebra, we have

$$
\begin{aligned}
x^3 - x - 1 &= 0 \\
x^2 &= 1 + \frac{1}{x} \\
x &= \sqrt{1 + \frac{1}{x}}
\end{aligned}
$$

for $x > 0$. We want to find the solution to $f(x) = x$ with $f(x) = \sqrt{1 + \frac{1}{x}}$. Then, by calculator,

$$
\begin{aligned}
p_1 &= \sqrt{2} = 1.414, \\
p_2 &= 1.307, \\
p_3 &= 1.329, \\
p_4 &= 1.324,
\end{aligned}
$$

which is accurate to within 10^{-2} for $x^3 - x - 1 = 0$ on $[1, 2]$. □

Problem 2. Use Theorem 2.1 to find a bound for the number of iterations needed to achieve an approximation with accuracy 10^{-3} to the solution of $x^3 + x - 4 = 0$ lying in the interval $[1, 4]$. Find an approximation to the root with this degree of accuracy.

Solution. Let $f(x) = x^3 + x - 4$. Since $f \in C[1, 4]$ and $f(1) \cdot f(4) = (-2) \cdot 64 < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approaches to a zero p of f with

$$|p_n - p| \leq \frac{4-1}{2^n}, \quad \text{when } n \geq 1.$$

Then,

$$\frac{3}{2^n} \leq 10^{-3} \implies n \geq \log_2(3000) > 11.$$

111652004_CM_HW2.tex U   111652004_CM_HW2_P2.py U

HW2 > 111652004_CM_HW2_P2.py > ...

```python
end_point_1 = 1
end_point_2 = 4
tolerance = 0.001
maximum_number_of_iterations = 50
p_0 = 2.5
i = 1

def f(x):
    return x**3 + x - 4

FA = f(end_point_1)

while i <= maximum_number_of_iterations:
    p = end_point_1 + (end_point_2 - end_point_1) / 2
    FP = f(p)
    if (p == 0 or (end_point_2 - end_point_1) / 2 < tolerance):
        print(f"p = {p} with {i} iterations.")
        exit(0)
    i = i + 1
    if FA * FP > 0:
        end_point_1 = p
        FA = FP
    else:
        end_point_2 = p

print(f"Method failed after {maximum_number_of_iterations}.")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   COMMENTS

```
PS E:\Eiken\Visual Studio Code Git Sync\CM_HW> & C:/Users/yungh/AppData/Local/Microsoft/WindowsApps/python3.11.exe "
p = 1.378662109375 with 12 iterations.
```

Using the bisection method, by Python, $p = 1.37866$. $\square$

Problem 3. The following four methods are preposed to compute $21^{1/3}$. Rank them in order, based on their apparent speed of convergence, assuming $p_0 = 1$.

a. $p_n = \dfrac{20p_{n-1} + \frac{21}{{p_{n-1}}^2}}{21}$

b. $p_n = p_{n-1} - \dfrac{{p_{n-1}}^3 - 21}{3{p_{n-1}}^2}$

c. $p_n = p_{n-1} - \dfrac{{p_{n-1}}^4 - 21p_{n-1}}{{p_{n-1}}^2 - 21}$

d. $p_n = \sqrt{\dfrac{21}{p_{n-1}}}$

Solution. For a, choose $g_1(x) = \dfrac{20x + \frac{21}{x^2}}{21} = \dfrac{20x}{21} + \dfrac{1}{x^2}$. Then ${g_1}'(x) = \dfrac{20}{21} - \dfrac{2}{x^3}$. Hence

$$ {g_1}'(\sqrt[3]{21}) = \frac{20}{21} - \frac{2}{21} \approx 0.86. $$

For b, choose $g_2(x) = x - \dfrac{x^3 - 21}{3x^2} = \dfrac{2x}{3} + \dfrac{7}{x^2}$. Then ${g_2}'(x) = \dfrac{2}{3} - \dfrac{14}{x^3}$. Hence

$$ {g_2}'(\sqrt[3]{21}) = \frac{2}{3} - \frac{1}{3} \approx 0.33. $$

For c, choose $g_3(x) = x - \dfrac{x^4 - 21x}{x^2 - 21} = \dfrac{x^3 - x^4}{x^2 - 21}$. Then ${g_3}'(x) = \dfrac{-2x^5 + x^4 + 84x^3 - 63x^2}{x^4 - 42x^2 + 441}$. Hence

$$ {g_3}'(\sqrt[3]{21}) \approx 5.7. $$

For d, choose $g_4(x) = \sqrt{\dfrac{21}{x}}$. Then ${g_4}'(x) = \dfrac{-\sqrt{21}}{2x^{\frac{3}{2}}}$. Hence

$$ {g_4}'(\sqrt[3]{21}) = \frac{1}{2}. $$

To sum up, the apparent speed of convergence in order is b, d, a, and c does not converge (the derivative at $\sqrt[3]{21}$ is greater than 1.) $\square$

Problem 4. Use Theorem 2.3 to show that $g(x) = 2^{-x}$ has a unique fixed point on $\left[\frac{1}{3}, 1\right]$. Use fixed-point iteration to find an approximation to the fixed point accurate to within 10^{-4}. Use corollary 2.5 to estimate the number of iterations required to achieve 10^{-4} accuracy, and compare this theoratical estimate the the number actually needed.

Solution. We know that $g \in C\left[\frac{1}{3}, 1\right]$ and $g(x) \in [0.5, 0.9637] \subseteq \left[\frac{1}{3}, 1\right]$ for all $x \in \left[\frac{1}{3}, 1\right]$. Then g has at least a fixed point in $[a, b]$. Moreover, $g'(x)$ exists on $\left(\frac{1}{3}, 1\right)$. Choose $k = 0.7$. Then

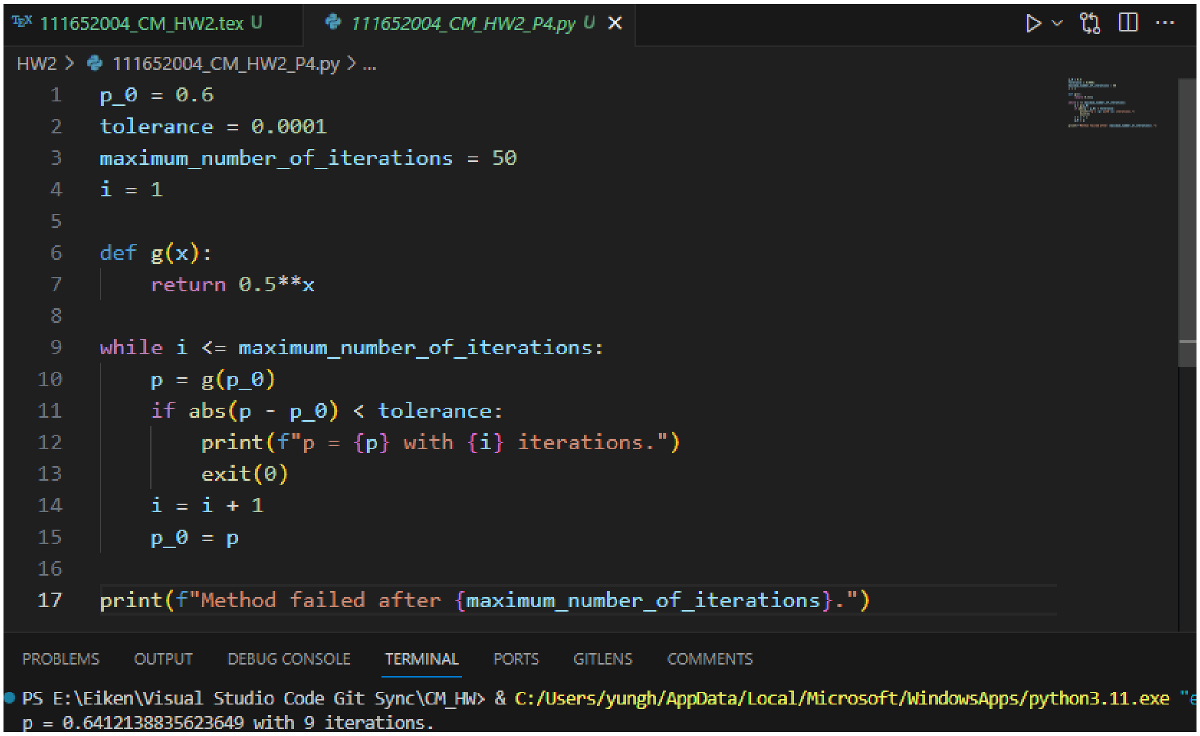
$$\begin{aligned}
\left|\frac{\mathrm{d}}{\mathrm{d}x} 2^{-x}\right| &= \ln 2 \cdot 2^{-x} \\
&< \ln 2 \cdot 2^{-0} \\
&= \ln 2 \\
&< k
\end{aligned}$$

for all $x \in (0, \infty)$. Hence, $g'(x)$ exists on $\left(\frac{1}{3}, 1\right)$ and a positive $0 < k < 1$ exsits with $|g'(x)| \leq k$ for all $x \in \left(\frac{1}{3}, 1\right)$. Then there exists exactly one fixed point in $\left[\frac{1}{3}, 1\right]$. It is known the assumption of Theorem 2.4 holds, i.e., $g'(x)$ exists on $\left(\frac{1}{3}, 1\right)$ and a positive $0 < k < 1$ exsits with $|g'(x)| \leq k$ for all $x \in \left(\frac{1}{3}, 1\right)$. By Corollary 2.5,

$$|p_n - p| \leq 0.7^n \max\{0.6 - \frac{1}{3}, 1 - 0.6\}.$$

Then,

$$0.7^n \max\{0.6 - \frac{1}{3}, 1 - 0.6\} \leq 10^{-4} \implies n > 23.$$



```python
p_0 = 0.6
tolerance = 0.0001
maximum_number_of_iterations = 50
i = 1

def g(x):
    return 0.5**x

while i <= maximum_number_of_iterations:
    p = g(p_0)
    if abs(p - p_0) < tolerance:
        print(f"p = {p} with {i} iterations.")
        exit(0)
    i = i + 1
    p_0 = p

print(f"Method failed after {maximum_number_of_iterations}.")
```

```
PS E:\Eiken\Visual Studio Code Git Sync\CM_HW> & C:/Users/yungh/AppData/Local/Microsoft/WindowsApps/python3.11.exe "e
p = 0.6412138835623649 with 9 iterations.
```

By Python, the number of iteration actually needed is 9, which is smaller due to my overestimation for error. □

Problem 5. Let A be a given positive constant and $g(x) = 2x - Ax^2$.

a. Show that if fixed-point iteration converges to a nonzero limit, then the limit is $p = \frac{1}{A}$, so the inverse of a number can be found using only multiplications and subtractions.

b. Find an interval about $\frac{1}{A}$ for which fixed-point iteration converges, provided p_0 is in that interval.

Solution.

a. Suppose that fixed-point iteration converges to a nonzero limit, say p. We want to show that $p = \frac{1}{A}$. For the sake of contradiction, suppose $p \neq \frac{1}{A}$. Then, by the definition of convergence, we have

$$p = 2p - Ap^2.$$

By the fact that p is nonzero, we further have

$$1 = 2 - Ap,$$

which implies $Ap = 1$, a contradiction.

b. We have $g(x) = 2x - Ax^2$ with $g \in C\left[\frac{2}{3A}, \frac{4}{3A}\right]$ and then

$$\begin{aligned} g(x) &= -A\left(x + \frac{1}{A}\right)^2 + \frac{1}{A} \\ &\leq -A \cdot \left(\frac{1}{3A}\right)^2 + \frac{1}{A} \\ &= \frac{1}{A} - \frac{1}{9A} \\ &= \frac{8}{9A}. \end{aligned}$$

Hence $g(x) \in \left[\frac{2}{3A}, \frac{4}{3A}\right]$ for all $x \in \left[\frac{2}{3A}, \frac{4}{3A}\right]$. In addition, $g'(x) = 2 - 2Ax$ exists on $\left(\frac{2}{3A}, \frac{4}{3A}\right)$. Then $|g'(x)| < \frac{2}{3}$ for all $x \in \left(\frac{2}{3A}, \frac{4}{3A}\right)$. By the fix-point theorem, the sequence defined by

$$p_n = g(g_{n-1}), \quad n \geq 1$$

converges to the unique fixed point $\frac{1}{A}$ in $\left[\frac{2}{3A}, \frac{4}{3A}\right]$. $\square$

Problem 6. Show that if A is any positive number, then the sequence defined by

$$x_n = \frac{1}{2}x_{n-1} + \frac{A}{2x_{n-1}}, \quad \text{for } n \geq 1,$$

converges to $\sqrt{A}$ whenever $x_0 > 0$.

Solution. Let $A > 0$. Let $x_0 > 0$. Let $k \in \mathbb{N}$. Suppose $x_k > 0$. Then $x_{k+1} = \dfrac{1}{2}x_k + \dfrac{A}{2x_k} > 0$. Thus $x_k > 0$ for all $k \in \mathbb{N}$. By the AM-GM inequality, we have

$$x_n = \frac{1}{2}x_{n-1} + \frac{A}{2x_{n-1}} \geq \sqrt{A} \tag{6.1}$$

for all $n \in \mathbb{N}$, and the equation does not hold provided that $x_{n-1} = \sqrt{A}$. We now separate three cases for the initial x_0. First, suppose that $x_0 = \sqrt{A}$. Then it is obvious that $x_k = \sqrt{A}$ for all $k \in \mathbb{N}$. Hence $x_k \to \sqrt{A}$ as $k \to \infty$. We now deal with $x_0 \neq \sqrt{A}$. Suppose $x_0 \neq \sqrt{A}$. Then by (6.1), $x_k > \sqrt{A}$ for all $k \in \mathbb{N}$. Thus

$$\begin{aligned}
x_{k+1} - x_k &= \frac{1}{2}x_k + \frac{A}{2x_k} - x_k \\
&= \frac{A}{2x_k} - \frac{1}{2}x_k \\
&= \frac{1}{2x_k}\left(A - {x_k}^2\right) \\
&< 0
\end{aligned}$$

for all $k \in \mathbb{N}$, which implies that $\{x_n\}$ is decreasing. Since $\{x_n\}$ is bounded and monotone, by the monotone convergence theorem, $\{x_n\}$ converges. Say the limit is L. Then by the recursive relation, $L = \dfrac{1}{2}L + \dfrac{A}{2L}$, which implies $L = \pm\sqrt{A}$. Since $\{x_n\} \subseteq [\sqrt{A}, \infty)$, the limit is $\sqrt{A}$. $\square$

Problem 7. Let $f(x) = -x^3 - \cos x$. With $p_0 = -1$ and $p_1 = 0$, find p_3.

a. Use the Secant method.

b. Use the method of False Position.

Solution.

a. By the algorithm of the secant method, we have the formula

$$p_n = p_{n-1} - f(p_{n-1}) \cdot \frac{p_{n-1} - p_{n-2}}{f(p_{n-1}) - f(p_{n-2})}.$$

Thus, by calculator,

$$\begin{aligned} p_2 &= 0 - f(0) \cdot \frac{0 - (-1)}{f(0) - f(-1)} \\ &= \frac{1}{-1 - (1 - \cos(-1))} \\ &= \frac{1}{-2 + \cos(-1)} \\ &\approx -0.685, \end{aligned}$$

and

$$\begin{aligned} p_3 &= -0.685 - f(-0.685) \cdot \frac{-0.685 - 0}{f(-0.685) - f(0)} \\ &\approx -1.252. \end{aligned}$$

b. We first check that $f(-1)$ and $f(0)$ has different sign. By calculator,

$$f(-1) \approx 0.45$$

and

$$f(0) = -1.$$

By the algorithm of the false position, we have the formula

$$p_n = p_{n-1} - f(p_{n-1}) \cdot \frac{p_{n-1} - p_{n-2}}{f(p_{n-1}) - f(p_{n-2})}.$$

Thus, by calculator,

$$\begin{aligned}
p_2 &= 0 - f(0) \cdot \frac{0 - (-1)}{f(0) - f(-1)} \\
&= \frac{1}{-1 - (1 - \cos(-1))} \\
&= \frac{1}{-2 + \cos(-1)} \\
&\approx -0.685.
\end{aligned}$$

By calculator, $f(p_2) \approx -0.453$. Hence the endpoints become -0.685 and 0. By calculator,

$$\begin{aligned}
p_3 &= 0 - f(0) \cdot \frac{0 - (-0.685)}{f(0) - f(-0.685)} \\
&\approx -0.841.
\end{aligned}$$

$\square$

Problem 8. Problems involving the amount of money required to pay off a mortgage over a fixed period of time involve the formula

$$A = \frac{P}{i}\left(1 - (1+i)^{-n}\right),$$

known as an *ordinary annuity equation.* In this equation, A is the amount of the mortgage, P is the amount of each payment, and i is the interest rate per period for the n payment periods. Suppose that a 30-year home mortgage in the amount of \$135,000 is needed and that the borrower can afford house payments of at most \$1000 per month. What is the maximal interest rate the borrower can afford to pay?

Solution. Using the information, we have

$$A = 135000, \qquad P = 1000, \qquad \text{and} \quad n = 360,$$

and we aim to solve for i in the following equation

$$f(i) = 0 \tag{8.1}$$

with $f(i) = 135000i - 1000\left(1 - (1+i)^{-360}\right)$. We use the bisection method with Python to solve (8.1). By calculator, we have

$$f(0.001) \approx -167.2$$

and

$$f(0.01) \approx 377.8.$$

Since f is continuous, by the intermediate value theroem, there is a solution to $f(i) = 0$ in $(0.001, 0.01)$. By Python, we have that the solution (month interest) to (8.1) is approximately 0.675%.

111652004_CM_HW2.tex M   111652004_CM_HW2_P8.py U

HW2 > 111652004_CM_HW2_P8.py > ...

```python
def f(i):
    return 135000*i-1000*(1-(1+i)**(-360))

# Set endpoints
a = 0.001
b = 0.01

# Set tolerance to 10^-9
TOL = 0.000000001

# Set the maximum number of iterations
N_0 = 500

FA = f(a)
for n in range(N_0):
    i = a + (b - a) / 2
    FI = f(i)
    if FI == 0 or (b - a) / 2 < TOL:
        print(f"The solution is approximately i_{n + 1} = {i}.")
        exit(1)
    if FA * FI > 0:
        a = i
        FA = FI
    else:
        b = i
        FB = FI
print(f"The bisection method failed after {N_0} times.")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   COMMENTS

```
PS E:\Eiken\Visual Studio Code Git Sync\CM_HW> & C:/Users/yungh/AppData/Local/Microsoft/WindowsApps/python3.11.exe
The solution is approximately i_24 = 0.006749917447566986.
```

□

Problem 9.

a. Show that for any positive integer k, the sequence defined by $p_n = \dfrac{1}{n^k}$ converges linearly to $p = 0$.

b. Show that the sequence $p_n = 10^{-2^n}$ converges quadratically to 0.

Solution.

a. Let $k \in \mathbb{N}$. It is clear that $\dfrac{1}{n^k} \to 0$ as $n \to \infty$. Choose $\alpha = 1$. Then

$$\begin{aligned}\lim_{n\to\infty} \frac{\left|\frac{1}{(n+1)^k} - 0\right|}{\left|\frac{1}{n^k} - 0\right|^1} &= \left(\lim_{n\to\infty} \frac{n}{n+1}\right)^k \\ &= 1.\end{aligned}$$

Hence the sequence is lienarly convergent.

b. It is clear that $10^{-2^n} \to 0$ as $n \to \infty$. Choose $\alpha = 2$. Then

$$\begin{aligned}\lim_{n\to\infty} \frac{\left|10^{-2^{n+1}} - 0\right|}{\left|10^{-2^n} - 0\right|^2} &= \lim_{n\to\infty} 10^{-2^{n+1}+2^{n+1}} \\ &= 1.\end{aligned}$$

Since the asymptotic error constant is $\lambda = 1$, the sequence is quadratically convergent. $\square$

Problem 10.

a. The following sequences are linearly convergent. Generate the first five terms of the sequence $\{\hat{p_n}\}$ using Aitken's Δ^2 method.

$$p_0 = 0.5, \quad p_n = \cos(p_{n-1}), \quad n \geq 1$$

b. Use Steffensen's method to find, to an accuracy of 10^{-4}, the root of $x^3 - x - 1 = 0$ that lies in $[1, 2]$.

Solution.

a. Aitken's Δ^2 method requires the first 7 terms of $\{p_n\}$. By calculator, we have the table:

n	p_n
0	0.5
1	0.877583
2	0.639012
3	0.802685
4	0.694778
5	0.768196
6	0.719165

The sequence $\{\hat{p}_n\}$ generated by Aitken's Δ^2 method is defined by the following formula:

$$\hat{p}_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}.$$

By calculator, we have

n	$\hat{p}_n$
0	0.731385
1	0.736087
2	0.737653
3	0.738469
4	0.738798

b. Using the result in Problem 1, we look for the solution to $x = \sqrt{1 + \dfrac{1}{x}}$ in $[1, 2]$ with $p_0 = 1.3$. By Python, the solution is approximately 1.32472.

111652004_CM_HW2.tex M | 111652004_CM_HW2_10b.py U

HW2 > 111652004_CM_HW2_10b.py > ...

```python
import math

def g(p):
    return math.sqrt(1 + 1 / p)

# Set initial approximation
p_0 = 1.3

# Set tolerance to 10^-4
TOL = 0.0001

# Set the maximum number of iteration
N_0 = 500

for i in range(N_0):
    p_1 = g(p_0)
    p_2 = g(p_1)
    p = p_0 - (p_1 - p_0) ** 2 / (p_2 - 2 * p_1 + p_0)
    if abs(p - p_0) < TOL:
        print(f"The solution calculated by Steffensen's method is
        approximately {p}.")
        exit(0)
    p_0 = p
print(f"Steffensen's method failed after {N_0} times.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

```
PS E:\Eiken\Visual Studio Code Git Sync\CM_HW> & C:/Users/yungh/AppData/Local/Microsoft/WindowsApps/python3.11.exe
The solution calculated by Steffensen's method is approximately 1.3247179572514045.
```

□

Problem 11. Given a polynomial $P(x) = x^3 - 5x^2 + 8x - 6$, do the following:

a. Evaluate $P(2)$, $P'(2)$, $P(4)$, and $P'(4)$ by Horner's method.

b. Find the root of $P(x)$ with error less than 0.00001 between $[2, 4]$ by using the Newton method with initial point $x_0 = 2$ and $x_0 = 4$. Determine which initial point may lead to the root.

c. Deflate $P(x)$ into a quadartic polynomial by using the results in (b) and find the complex roots of $P(x)$.

d. Perform one step of Müller's Method starting from initial $(0, P(0))$, $(1, P(1))$ and $(2, P(2))$.

e. Implement a MATLAB[a] code of Müller's Method to find the complex root within error less than 0.00001 and compare with the answer you find in (c).

[a] Prof. Wu says that we can use Python as well.

Solution.

a. We first evaluate $P(2)$ and $P'(2)$. The table appears as follows:

	Coefficient of x^3	Coefficient of x^2	Coefficient of x	Constant term
$x_0 = 2$	$a_3 = 1$	$a_2 = -5$	$a_1 = 8$	$a_0 = -6$
		$b_3 x_0 = 2$	$b_2 x_0 = -6$	$b_1 x_0 = 4$
	$b_3 = 1$	$b_2 = -3$	$b_1 = 2$	$b_0 = -2$

Hence

$$P(x) = (x - 2)(x^2 - 3x + 2) - 2.$$

We can therefore evaluate $P(2) = -2$ and $P'(2) = x^2 - 3x + 2\big|_{x=2} = 0$. We now evaluate $P(4)$ and $P'(4)$. The table appears as follows:

	Coefficient of x^3	Coefficient of x^2	Coefficient of x	Constant term
$x_0 = 4$	$a_3 = 1$	$a_2 = -5$	$a_1 = 8$	$a_0 = -6$
		$b_3 x_0 = 4$	$b_2 x_0 = -4$	$b_1 x_0 = 16$
	$b_3 = 1$	$b_2 = -1$	$b_1 = 4$	$b_0 = 10$

Hence

$$P(x) = (x - 4)(x^2 - x + 4) + 10.$$

We can therefore evaluate $P(4) = 10$ and $P'(4) = x^2 - x + 4\big|_{x=4} = 16$.

b. Since $P'(2) = 0$, $x_0 = 2$ can't lead to the root. So we use $x_0 = 4$ only. By Python, the root is approximately 3.00002.

```
111652004_CM_HW2.tex M    111652004_CM_HW2_P11b.py U ×
HW2 > 111652004_CM_HW2_P11b.py > ...
def f(x):
    return x ** 3 - 5 * x ** 2 + 8 * x - 6

# Set initial approximation
x_0 = 4

# Set tolerance
TOL = 0.00001

# Set the maximum number of iterations
N_0 = 500

for i in range(N_0):
    x = x_0 - f(x_0) / 16
    if abs(x - x_0) < TOL:
        print(f"The root of P(x) calculated by Newton's method is approixmately {x}.")
        exit(0)
    x_0 = x

print(f"Newton's method failed after {N_0} iterations.")
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   COMMENTS
PS E:\Eiken\Visual Studio Code Git Sync\CM_HW> & C:/Users/yungh/AppData/Local/Microsoft/WindowsApps/python3.11.exe "
The root of P(x) calculated by Newton's method is approixmately 3.0000213667260756.
```

c. By the approximation, we first guess that $x = 3$ is a solution to $P(x) = 0$. Indeed, $P(3) = 0$. Using the synthetic division, we have

	Coefficient of x^3	Coefficient of x^2	Coefficient of x	Constant term
$x_0 = 3$	$a_3 = 1$	$a_2 = -5$	$a_1 = 8$	$a_0 = -6$
		$b_3 x_0 = 3$	$b_2 x_0 = -6$	$b_1 x_0 = 6$
	$b_3 = 1$	$b_2 = -2$	$b_1 = 2$	$b_0 = 0$

which suggests that $P(x) = (x-3)(x^2 - 2x + 2)$. By the quadratic formula, the complex solutions are $1 \pm i$.

d. We have $p_0 = 0$, $p_1 = 1$, and $p_2 = 2$. Set

$$h_1 = 1 - 0 = 1,$$

$$h_2 = 2 - 1 = 1,$$

$$\delta_1 = \frac{P(1) - P(0)}{1} = 4,$$

$$\delta_2 = \frac{P(2) - P(1)}{1} = 0,$$

$$d = \frac{0 - 4}{1 + 1} = -2.$$

Then

$$b = 0 + 1 \cdot (-2) = -2,$$

$$D = \sqrt{(-2)^2 - 4 \cdot P(2) \cdot (-2)} = 2\sqrt{3}i.$$

Since $|b - D| > |b + D|$, set $E = b - D = -2 - 2\sqrt{3}i$. Then $h = -2 \cdot \dfrac{P(2)}{E} = \dfrac{2}{-1 - \sqrt{3}i} = \dfrac{-1}{2} + \dfrac{\sqrt{3}}{2}i$. Thus $p = 2 + \dfrac{-1}{2} + \dfrac{\sqrt{3}}{2}i = \dfrac{3}{2} + \dfrac{\sqrt{3}}{2}i$.

e. By Python, the root is approximately $1.000000 + 1.000000i$.

111652004_CM_HW2.tex M   111652004_CM_HW2_P11e.py U

HW2 > 111652004_CM_HW2_P11e.py > ...

```python
def P(x):
    return x ** 3 - 5 * x ** 2 + 8 * x - 6

# Set approximations
p_0 = 0; p_1 = 1; p_2 = 2

# Set tolerance
TOL = 0.00001

# Set the maximum number of iterations
N_0 = 500

h_1 = p_1 - p_0; h_2 = p_2 - p_1
delta_1 = (P(p_1) - P(p_0)) / h_1; delta_2 = (P(p_2) - P(p_1)) / h_2
d = (delta_2 - delta_1) / (h_2 + h_1)

for i in range(N_0):
    b = delta_2 + h_2 * d
    D = (b ** 2 - 4 * P(p_2) * d) ** 0.5
    if abs(b - D) < abs(b + D):
        E = b + D
    else:
        E = b - D
    h = -2 * P(p_2) / E
    p = p_2 + h
    if abs(h) < TOL:
        print(f"The solution calculated by Muller's method is 
        approximately {p}.")
        exit(0)
    p_0 = p_1; p_1 = p_2; p_2 = p
    h_1 = p_1 - p_0; h_2 = p_2 - p_1
    delta_1 = (P(p_1) - P(p_0)) / h_1; delta_2 = (P(p_2) - P(p_1)) / h_2
    d = (delta_2 - delta_1) / (h_2 + h_1)

print(f"Muller's method failed after {N_0} iterations.")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS   COMMENTS

```
PS E:\Eiken\Visual Studio Code Git Sync\CM_HW> & C:/Users/yungh/AppData/Local/Microsoft/WindowsApps/python3.11.exe '
The solution calculated by Muller's method is approximately (1.000000000019976+0.9999999999082199j).
```

□