week_3

January 21, 2025

1 CS2201

1.1 Week 2

2 Set 2

2.0.1 Solution by Abhisek Sarkar

as20ms091@iiserkol.ac.in Q1. Using Tabulation method find a better approximate interval in the interval [0, 1] and then on this better interval use Bisection to find the root of the equation $sin(x) + x^2 - 1 = 0$. You may use the $sin()$ function in the math module.

[1]:
```
import math

def f(x):
    return math.sin(x) + x**2 - 1

def tabulation_method(start, end, step):
    """
    Tabulation method to find a smaller interval where the root exists.
    Returns a tuple with the narrowed interval.
    """
    x = start
    while x < end:
        if f(x) * f(x + step) < 0:
            return x, x + step
        x += step
    raise ValueError("No root found in the given interval using tabulation.")

def bisection_method(a, b, tolerance):
    """
    Bisection method to find the root in the given interval [a, b].
    Returns the approximate root.
    """
    while (b - a) / 2 > tolerance:
        c = (a + b) / 2
        if f(c) == 0:  # Exact root found
            return c
        elif f(a) * f(c) < 0:
```

```
            b = c
        else:
            a = c
    return (a + b) / 2
# Main Execution
start_interval = 0
end_interval = 1
step_size = 0.1  # Step size for tabulation
tolerance = 1e-6  # Tolerance for bisection

# Step 1: Use Tabulation Method
try:
    better_interval = tabulation_method(start_interval, end_interval, step_size)
    print(f"Better interval found using tabulation: {better_interval}")

    # Step 2: Use Bisection Method
    root = bisection_method(better_interval[0], better_interval[1], tolerance)
    print(f"Root found using bisection: {root:.6f}")
except ValueError as e:
    print(e)
```

```
Better interval found using tabulation: (0.6, 0.7)
Root found using bisection: 0.636733
```

Q2. Using Newton-Raphson method find as many roots as you can of the equation $f(x) = x^3 + x^2 - x = 0$ in the interval $[0, 1]$. Note that the first order derivative of $f(x)$ is $3x^2 + 2x - 1$. You may plot f using a web application like Desmos to first identify the roots visually and choose appropriate initial approximations for the Newton-Raphson method.

[2]:
```
import math

def f(x):
    return x**3 + x**2 - x

def df(x):
    return 3*x**2 + 2*x - 1

def newton_raphson(x0, tolerance=1e-6, max_iterations=1000):
    """
    Newton-Raphson method to find a root of f(x).
    :param x0: Initial approximation
    :param tolerance: Tolerance for stopping condition
    :param max_iterations: Maximum number of iterations
    :return: Root if found, None otherwise
    """
    for i in range(max_iterations):
        f_x0 = f(x0)
        df_x0 = df(x0)
```

```
        if df_x0 == 0:
            raise ValueError("Derivative is zero. Newton-Raphson method fails.")

        x1 = x0 - f_x0 / df_x0

        if abs(x1 - x0) < tolerance:
            return x1

        x0 = x1

    raise ValueError("Newton-Raphson method did not converge.")

# Main execution
initial_guesses = [0, 0.5, 1]  # Initial guesses based on visual inspection
roots = []
tolerance = 1e-6

for guess in initial_guesses:
    try:
        root = newton_raphson(guess, tolerance)
        root = round(root, 6)  # Round to avoid duplicates due to precision␣
↪errors

        if root not in roots:  # Avoid duplicate roots
            roots.append(root)
    except ValueError as e:
        print(f"Initial guess {guess} failed: {e}")

print(f"Roots found: {roots}")
```

```
Roots found: [0.0, 0.618034]
```

Above is the Python code for solving $ f(x) = x^3 + x^2 - x = 0 $ using the Newton-Raphson method. To identify the roots visually, you can plot the function in Desmos or another tool.

2.0.2 Steps in the Code:

1. **Function Definition**:
   - $ f(x) = x^3 + x^2 - x $
   - $ f'(x) = 3x^2 + 2x - 1 $
2. **Newton-Raphson Method**:
   - Iterates starting from an initial guess $ x_0 $.
   - Updates the guess using $ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} $. Stops when the difference between successive guesses is less than the to
3. **Main Execution**:
   - Takes a list of initial guesses based on visual inspection.
   - Finds roots while avoiding duplicates due to numerical precision issues.
4. **Output**:

- Lists all distinct roots found in the interval $[0, 1]$.

This code can compute the roots numerically. The roots can be confirmed by plotting (f(x)) and observing its intersections with the x-axis.

Q3. Given a list L with repeating elements, remove the duplicates. You should not use any temporary list.

[1]:
```
def remove_duplicates(L):
    """
    Removes duplicates from the list L in place without using a temporary list.
    :param L: List with potentially duplicate elements
    """
    i = 0
    while i < len(L):
        j = i + 1
        while j < len(L):
            if L[j] == L[i]:
                del L[j]
            else:
                j += 1
        i += 1

# Example usage
L = [1, 2, 3, 2, 1, 4, 5, 3]
remove_duplicates(L)
print("List after removing duplicates:", L)
```

```
List after removing duplicates: [1, 2, 3, 4, 5]
```

Q4. Given a list L = [1, 2, 3, 4, 5], use list comprehension to generate another list L1 containing the even numbers in L

[1]:
```
L = [1, 2, 3, 4, 5]
L1 = [x for x in L if x % 2 == 0]
print("List of even numbers:", L1)
```

```
List of even numbers: [2, 4]
```

Q5. Given two lists L1 = [1, 2, 3, 4, 5] and L2 = [5, 4, 10, 12], use list comprehension to generate another list L3 containing the sum of the odd elements in L1 and L2.

[2]:
```
L1 = [1, 2, 3, 4, 5]
L2 = [5, 4, 10, 12]

L3 = [x + y for x in L1 if x % 2 != 0 for y in L2 if y % 2 != 0]
print("List containing the sums of odd elements in L1 and L2:", L3)
```

```
List containing the sums of odd elements in L1 and L2: [6, 8, 10]
```

2.0.3 Explanation:

- The comprehension iterates over all elements in (L1) and (L2).
- It filters the odd elements from both lists using the condition `x % 2 != 0` and `y % 2 != 0`.
- For each pair of odd elements, it calculates their sum and adds it to (L3).

Q6. Use a while loop to take a number as input from the user and continue the loop until the user inputs an even number.

[5]:
```
while True:
    number = int(input("Enter a number: "))
    if number % 2 == 0:
        print("You entered an even number:", number)
        #break
    else:
        print("That is an odd number, try again.", number)
        break
```

```
You entered an even number: 2
You entered an even number: 4
You entered an even number: 66
You entered an even number: 22
You entered an even number: 68
That is an odd number, try again. 55
```

Q7. Use while loop to reverse an integer stored in a variable n and store the reversed number in a variable rev. You can't use a list, string operations or any Python functions for reversal.

[6]:
```
n = 12345  # Example integer
rev = 0    # Variable to store the reversed number

while n > 0:
    rev = rev * 10 + n % 10  # Add the last digit of n to rev
    n = n // 10               # Remove the last digit from n

print("Reversed number:", rev)
```

```
Reversed number: 54321
```

Q8. Take a number as input (use input()) and using the dictionary d = {1 : 'ONE', 2 : 'TWO', 3 : 'THREE', 4 : 'FOUR', 5 : 'FIVE', 6 : 'SIX', 7 : 'SEVEN', 8 : 'EIGHT', 9 : 'NINE'} only print the number as words. If the input is 1234, the output should be ONE TWO THREE FOUR. Note that, you have to use only this dictionary (and no other source) to convert each digit of the input number to the corresponding word equivalent.

[13]:
```
d = {1: 'ONE', 2: 'TWO', 3: 'THREE', 4: 'FOUR', 5: 'FIVE',
     6: 'SIX', 7: 'SEVEN', 8: 'EIGHT', 9: 'NINE'}

# Take input number
number = input("Enter a number: ")
```

```
print(f"You have input {number}")

# Loop through each digit in the input number
for digit in number:
    if digit == '0':  # Handle the case for 0 separately
        print("ZERO", end=" ")
    else:
        # Convert each digit to its integer form and use the dictionary to print␣
 ↪the corresponding word
        print(d[int(digit)], end=" ")
```

```
You have input 5678
FIVE SIX SEVEN EIGHT
```

Q9. Write a Python program to determine the direction ('increasing' or 'decreasing' or 'not monotonic') of monotonic sequence (consistently increasing or consistently decreasing) numbers stored as a list. E.g. [1, 2, 3, 4, 5, 6] contains a monotonically increasing sequence, while [25, 24, 23, 22] contains a monotonically decreasing sequence; [25, 24, 10, 23, 22] is not monotonic.

[15]:
```
def check_monotonic(sequence):
    increasing = decreasing = True

    # Iterate through the list to check for increasing or decreasing sequence
    for i in range(1, len(sequence)):
        if sequence[i] < sequence[i - 1]:
            increasing = False  # It is not increasing
        if sequence[i] > sequence[i - 1]:
            decreasing = False  # It is not decreasing

    # Determine the type of sequence based on flags
    if increasing:
        return "increasing"
    elif decreasing:
        return "decreasing"
    else:
        return "not monotonic"

# Example usage
numbers = [1,2,3,4,5,6]
result = check_monotonic(numbers)
print(f"The sequence is {result}.")
```

```
The sequence is increasing.
```

Q10. Design an automatic Quizzing System (choose an apt name like "Proshnobaan") that asks science questions. Use a Python dictionary (KB) to store the QAs with correct and wrong options and scores for the correct option. Each element in KB is a key:value pair, key being the question (e.g. "What is the smallest known organism?") and the value is a list of tuples containing both correct and wrong options. Each element of the list is a tuple in the format (option, correct/wrong,

score), e.g. (“Mycoplasma gallicepticum”, 1, 2), where “Mycoplasma gallicepticum” is an option for the question, ‘1’ indicates that this is the CORRECT option for the question and ‘2’ is the score that the participant will get if s/he chooses this option. On the other hand, the tuple for a wrong option is (“Valonia ventricosa”, 0, 0). An example key:value pair for one Question with the options is “What is the smallest known organism?”: [(“Mycoplasma gallicepticum”, 1, 2), (“Valonia ventricosa”, 0, 0)]

For each question, the system should ask the question, show the options and ask for the answer. After the participant enters the answer, the system will print an appropriate message like “Correct answer” or “Wrong answer”.

The system will ask a mix of easy-to-tough questions (the score will be higher for the tougher questions) one after the other and add up the scores obtained by the participant. If the total score at the end of these questions is less than or equal to a threshold, the system will announce a consolation prize. If this total score is more than the threshold, the system will ask a JACKPOT question (an unusual question like “What is the smallest resolvable unit of distance by a given computer mouse pointing device called?”) WITHOUT any option. If the participant answers the jackpot question correctly, the system should print “Congratulations” with the announcement of a mega prize; otherwise, it should print an appropriate message.

[1]:
```
import random

def proshnobaan_quiz():
    # Knowledge Base (KB)
    kb = {
        "What is the smallest known organism?": [
            ("Mycoplasma gallicepticum", 1, 2),
            ("Valonia ventricosa", 0, 0),
            ("Escherichia coli", 0, 0),
            ("Pandoravirus", 0, 0)
        ],
        "What is the speed of light in vacuum?": [
            ("300,000 km/s", 1, 5),
            ("150,000 km/s", 0, 0),
            ("100,000 km/s", 0, 0),
            ("3,000 km/s", 0, 0)
        ],
        "What is the chemical symbol for water?": [
            ("H2O", 1, 1),
            ("H2", 0, 0),
            ("O2", 0, 0),
            ("HO", 0, 0)
        ],
        "What planet is known as the Red Planet?": [
            ("Mars", 1, 2),
            ("Earth", 0, 0),
            ("Jupiter", 0, 0),
            ("Saturn", 0, 0)
```

```
        ],
        "What is the powerhouse of the cell?": [
            ("Mitochondria", 1, 3),
            ("Nucleus", 0, 0),
            ("Ribosome", 0, 0),
            ("Chloroplast", 0, 0)
        ]
    }

    # Jackpot Question
    jackpot_question = "What is the smallest resolvable unit of distance by a␣
↪given computer mouse pointing device called?"
    jackpot_answer = "mickey"

    # Threshold score for jackpot
    threshold = 7

    # Initialize total score
    total_score = 0

    print("Welcome to Proshnobaan - The Ultimate Science Quiz!")
    print("Answer the questions and earn points.")

    # Shuffle and ask questions
    questions = list(kb.keys())
    random.shuffle(questions)

    for question in questions:
        print(f"\nQuestion: {question}")
        options = kb[question]
        random.shuffle(options)

        for i, (option, _, _) in enumerate(options, start=1):
            print(f"{i}. {option}")

        # Input answer from user
        try:
            answer = int(input("Enter the option number: ")) - 1
            if 0 <= answer < len(options):
                selected_option = options[answer]
                if selected_option[1] == 1:
                    print("Correct answer!")
                    total_score += selected_option[2]
                else:
                    print("Wrong answer!")
            else:
                print("Invalid option. Moving to the next question.")
```

```
        except ValueError:
            print("Invalid input. Moving to the next question.")

    print(f"\nYour total score is: {total_score}")

    # Determine prize and jackpot eligibility
    if total_score <= threshold:
        print("Better luck next time! You win a consolation prize!")
    else:
        print("You qualify for the JACKPOT question!")
        print(f"\nJACKPOT Question: {jackpot_question}")
        jackpot_answer_input = input("Your answer: ").strip().lower()

        if jackpot_answer_input == jackpot_answer:
            print("Congratulations! You've won the mega prize!")
        else:
            print("Sorry, that's incorrect. Better luck next time!")

# Run the quiz
proshnobaan_quiz()
```

```
Welcome to Proshnobaan - The Ultimate Science Quiz!
Answer the questions and earn points.

Question: What is the chemical symbol for water?
1. O2
2. H2
3. HO
4. H2O

Enter the option number:  4

Correct answer!

Question: What is the powerhouse of the cell?
1. Chloroplast
2. Ribosome
3. Mitochondria
4. Nucleus

Enter the option number:  3

Correct answer!

Question: What planet is known as the Red Planet?
1. Earth
2. Mars
3. Jupiter
4. Saturn
```

```
Enter the option number:  2

Correct answer!

Question: What is the speed of light in vacuum?
1. 300,000 km/s
2. 100,000 km/s
3. 3,000 km/s
4. 150,000 km/s

Enter the option number:  1

Correct answer!

Question: What is the smallest known organism?
1. Valonia ventricosa
2. Mycoplasma gallicepticum
3. Pandoravirus
4. Escherichia coli

Enter the option number:  2

Correct answer!

Your total score is: 13
You qualify for the JACKPOT question!

JACKPOT Question: What is the smallest resolvable unit of distance by a given
computer mouse pointing device called?

Your answer:  Muggeseggele

Sorry, that's incorrect. Better luck next time!
```