INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

MA635P-Scientific Programming Laboratory

Lab Exercise-8 (30 Marks) Deadline: 3 March 2026, 4:00 PM

---

1. Create an algorithm for the Secant method. [2.5]

2. Create an algorithm for Müller's method. Refer to Slide 15 of Lecture 20. However, use the Newton-Divided difference formula to compute the coefficients a, b and c [5]

3. Write a Python code for the developed for Newton-Raphson method and find the roots of the following equations. Use sympy to compute the derivative. [5]

   (a) $x + 1 - 2\sin(\pi x) = 0, x_0 \in [0, 0.5], x_0 \in [0.5, 1]$

   (b) $x - \frac{1}{2^x} = 0, x_0 \in [0, 1]$

   (c) $e^x - x^2 + 3x - 2 = 0, x_0 \in [0, 1]$

   (d) $e^x - 2 = \cos(e^x - 2), x_0 \in [0.5, 1.5]$

4. Write a Python code for the developed Secant method and find the roots of the following equations **Caution**: Polynomials may have imaginary roots. [5]

   (a) $x^3 - 7x^2 + 14x - 6 = 0, x_0, x_1 \in [0, 1], [1, 3.2], [3.2, 4]$

   (b) $x^4 - 2x^3 - 4x^2 + 4x + 4 = 0, x_0, x_1 \in [-2, 1], [0, 2], [2, 3], [-1, 0]$

   (c) $x - \frac{1}{2^x} = 0, x_0, x_1 \in [0, 1]$

   (d) $e^x - x^2 + 3x - 2 = 0, x_0, x_1 \in [0, 1]$

   (e) $e^x - 2 = \cos(e^x - 2), x_0, x_1 \in [0.5, 1.5]$

5. Write a Python code for Müllers method and the following Horner's Method. Find all zeros (accurate within 10^{-5}) of $f(x) = x^4 - 18x^3 + 111x^2 - 278x + 240$ by using the Müllers method and Horner's Method. [12.5]

1. Implement the following Horner's Method Algorithm

Pseudocode 1: Horner's Method

```
Function HornersMethod(x_0, a_0, a_1, ···, a_n)
    sum = 0
    diff = 0
    for i = n, n − 1, ···, 0 do
        diff = diff * x_0 + sum
        sum = sum * x_0 + a_i
    return sum, diff
```

Pseudocode 2: Polynomial Deflation

Function `PolynomialDeflation`$(x_r, a_0, a_1, \cdots, a_n)$
  $b_n = a_n$
  for $i = n, n-1, \cdots, 0$ **do**
    $b_i = a_i + b_{i+1} x_r$
  return $b_1, b_2, \cdots b_n$

Pseudocode 3: Newton-Raphson Method with Horner's Method

Function `HornerNewtonRaphson`$(n, a_0, a_1, a_2, \cdots a_n, x_0, M, \epsilon_s, \epsilon)$
  if $n == 2$ **then**
    $x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}, x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$
  else
    for $i = 1, 2, ..., M$ **do**
      $sum, diff = HornersMethod(x_0, a_0, a_1, \cdots, a_n)$
      $x_1 = x_0 - \frac{sum}{diff}$
      if $x_1 \neq 0$ **then**
        $\epsilon_a \leftarrow |\frac{x_1 - x_0}{x_1}|$
      if $|\epsilon_a| < \epsilon_s \ or \ |f(x_1)| < \epsilon$ **then**
        break the for loop
      else
        $x_0 = x_1$
    return x_0

Pseudocode 4: Müller's Method

Function `Müller`$(x_0, x_1, x_2, h, M, \epsilon, \delta)$
  for $i = 1, 2, ..., M$ **do**
    $h_0 = x_1 - x_0, h_1 = x_2 - x_1, d_0 = \frac{f(x_1)-f(x_0)}{h_0}, d_1 = \frac{f(x_2)-f(x_1)}{h_1}$
    $a = \frac{d_1 - d_0}{h_1 + h_0}, b = ah_1 + d_1, c = f(x_2), D = \sqrt{b^2 - 4ac}$
  if $|\frac{b+D}{b-D}| > 1$ **then**
    $den = b + D$
  else
    $den = b - D$
  $x_3 = x_2 + \frac{-2c}{den}$
  if $|\epsilon_a| < \delta \ or \ |f(x_3)| < \epsilon$ **then**
    break the for loop
  else
    $x_0 = x_1, \quad x_1 = x_2, \quad x_2 = x_3$