INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

MA635P-Scientific Programming Laboratory

Lab Exercise-4 (30 Marks) Deadline: 27 January 2026, 4:00 PM

---

1. Write a Python program to define a function

   `roundoff(x, n)`

   that returns the value of a real number x rounded to n decimal places. [3]

   The input x is provided as a **list of digits**, where the decimal point position is specified separately. Your function should:

   - Construct the real number from the given digit list and decimal position,
   - Perform rounding to n decimal places *without using built-in rounding functions*,
   - Return the rounded value as a floating-point number.

2. **Machine Epsilon Experiment:** Write a Python program with `eps=1` and divide `eps` by 2, until `1.0+eps==1.0`. Print the last value of `eps`. [1]

3. Write a Python code to compute and plot the first 6 Taylor Polynomials with $f(x)$ [3]

   (a) about $x_0 = 1$ for $f(x) = 1/x$

   (b) about $x_0 = 0$ for $f(x) = e^x$

   (c) about $x_0 = 1$ for $f(x) = \ln x$

4. Approximate the value of $e^{0.5}$ using the above Taylor's polynomials up to $P_1, P_2, \cdots, P_6$. Compute the exact value using `math.exp`. Compute the truncation error for each P_i. How does the truncation error change as terms increase? Is truncation error related to machine epsilon? Repeat it for e. [3]

5. The exact value of $\ln(e)$ is 1. In practical computation, however, the number e is available only through a finite-precision approximation, as we calculated in the above examples. For example, `math.e` gives the value as `2.718281828459045`

   The Taylor series for the natural logarithm is given by

   $$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots, \quad |x| < 1.$$

   Let $\tilde{e}$ denote an approximation of e rounded to n decimal digits obtained by combining Problem 4(b) and Problem 1, and write

   $$\ln(e) = \ln\big(1 + (\tilde{e} - 1)\big).$$

   Let P_k denote the Taylor polynomial obtained by truncating the series after k terms.

(a) For a fixed number of digits n, compute the approximations $P_1, P_2, \ldots, P_6$ to $\ln(e)$ using $\tilde{e}$.

(b) Compute the exact value using `math.log(math.e)`.

(c) Compute the total numerical error for each P_k.

(d) Repeat the experiment by increasing the number of digits n used to approximate e.

(e) Describe how the total numerical error changes as both the number of Taylor terms and the precision of e are increased. [5 ×1 =5]

6. Write a Python code for Newton's interpolating polynomial, including ε_t, and test all examples discussed in the class. [5]

7. Using Newton's interpolating polynomial, compute $f(2.5)$, $f(1.25)$ and $f(3.25)$ using $P_3(x)$ and then compute $f(2.5)$ using $P_4(x)$. [5]

x	1	1.5	2	3	3.5	4
$f(x)$	0.0	0.17609	0.30103	0.47712	0.54407	0.60206

8. Based on an experiment on a heated plate, temperatures are measured at various points as given in below table. Estimate the temperature at $(x, y) = (4, 3.2)$ and $(x, y) = (4.3, 2.7)$ using Newton's interpolating polynomial $P_4(x)$. [5]

y	$x = 0$	$x = 2$	$x = 4$	$x = 6$	$x = 8$
0	100	90	80	70	60
2	85	64.49	53.5	48.15	50
4	70	48.9	38.43	35.03	40
6	55	38.78	30.39	27.07	30
8	40	35	30	25	20