2-Year Master's Program in Mathematics (NEP-2020)
COMPUTATIONAL MATHEMATICS

Masters Mathematics (1st Semester) Total Credits: 04
Course Code: MMTHDCM126 (15 hours per credit) Total Marks: 100
Continuous Assessment: Marks 30, Theory: Marks 70 Time Duration: 2½ hrs

Course Learning Outcomes(CLO's): After the completion of this course, students shall be able to:

CLO1. Apply basic Python programming constructs such as variables, loops, conditionals, functions, and libraries to solve mathematical problems.

CLO2. Use Python libraries (such as NumPy, SymPy, and Matplotlib) to perform symbolic and numerical computation in calculus and visualize mathematical functions.

CLO3. Implement numerical methods and solution for linear algebra problems such as root-finding, numerical integration, and numerical linear algebra using Python-based tools and libraries.

CLO4. Formulate and solve ordinary and partial differential equations using Python libraries and numerical techniques, and interpret the solutions graphically and analytically.

Unit I:

Python Essentials for Mathematical Thinking: Variables, data types, control flow (conditionals, loops), Functions, scope, and exceptions, Using Jupyter Notebook / Google Colab, Introduction to the math and numpy libraries, Basic vector and matrix operations, Plotting with matplotlib.

Key Activities: Write functions to compute quadratic roots, factorials, etc. Perform vector arithmetic and matrix multiplication in numpy, Plot polynomials, trigonometric, and exponential functions

Unit II:

Calculus and Symbolic Math with Python: Symbolic computation with sympy: expressions, limits, derivatives, integrals, Partial derivatives and gradient vectors, Numerical differentiation and integration with scipy.integrate, Overlay plots of functions, their derivatives, and integrals.

Key Activities: Symbolically derive and plot $\boldsymbol{f(x), f'(x)}$ and $\int \boldsymbol{f(x)dx}$, Compute definite integrals both symbolically and numerically; compare results, Use gradients to analyze functions of two variables

Unit III:

Linear Algebra & Numerical Methods: Matrix operations in numpy.linalg: inverse, determinant, transpose, Solving $\boldsymbol{Ax = B}$ and interpreting solutions, Eigenvalues, eigenvectors, and diagonalization, LU and QR factorizations (scipy.linalg), Root finding: bisection, Newton–Raphson, and secant methods

Key Activities: Solve circuit-modeled linear systems and interpret physical meaning, Implement and compare convergence of Newton's method vs. bisection, Decompose matrices and visualize the effect on linear transformations

Unit IV:
Modeling & Differential Equations: Formulating first and second order ODEs (e.g., logistic growth, pendulum), Solving ODEs numerically with scipy.integrate.odeint and solve_ivp, Introduction to finite-difference schemes for PDEs, Simulating heat and wave equations on simple domains.
Key Activities: Code and visualize ODE solutions; compare to analytical forms, Build a finite-difference solver for the 1D heat equation, Animate solutions over time and interpret behaviour

CLO-PLO Mapping Matrix (Strength version)

CLO ↓ PLO →	PLO1	PLO 2	PLO3	PLO4	PLO 5	PLO6	PLO7	PLO8	Average CLO
MMTHDCM12 6.1	3	3	2	2	3	3	2	2	**2.5**
MMTHDCM12 6.2	3	2	1	1	2	3	2	3	**2.13**
MMTHDCM12 6.3	3	2	2	3	1	3	2	3	**2.38**
MMTHDCM12 6.4	3	2	3	2	3	3	2	3	**2.63**
Average PLO	**3**	**2.25**	**2**	**2**	**2.25**	**3**	**2**	**2.75**	**2.41**

Recommended Books:

1. Christian Hill, Learning Scientific Programming with Python, Cambridge University Press, 2nd Edition, 2020.
2. Jaan Kiusalaas, Numerical Methods in Engineering with Python 3, Cambridge University Press, 3rd Edition, 2013.
3. Svein Linge & Hans Petter Langtangen, Programming for Computations – A Gentle Introduction to Numerical Simulations with Python, Springer, 2nd Edition, 2020.
4. Gilbert Strang, Linear Algebra and Learning from Data, Wellesley-Cambridge Press, 1st Edition, 2019.