GENERIC ELECTIVE (GE – 8):

NUMERICAL ANALYSIS AND COMPUTATIONAL PHYSICS

Course Title & Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course
		Lecture	Tutorial	Practical		
NUMERICAL ANALYSIS AND COMPUTATIONAL PHYSICS GE – 8	4	2	0	2	Passed 12th Class	Differential calculus, integration and ordinary differential calculus at the class 12 level.

LEARNING OBJECTIVES

The emphasis of course is to equip students with the mathematical tools required in solving problem of interest to physicists. To expose students to fundamental computational physics skills and hence enable them to solve a wide range of physics problems.

LEARNING OUTCOMES

After completing this course, student will be able to,

- Develop numerical methods to understand errors and solution of Algebraic and Transcendental equations.
- Understand interpolation, least square fitting, Numerical differentiation, Numerical integration and solution of ordinary differential equations.

In the laboratory course, the students will learn to,

- apply appropriate numerical method to solve selected physics problems using user defined and inbuilt functions
- solve non-linear equations
- perform least square fitting of the data taken in physics lab by user defined functions
- Interpolate a data by polynomial approximations
- numerically integrate a function and
- solve first order initial value problems numerically

SYLLABUS OF GE - 8

THEORY COMPONENT

Unit – I **(8 Hours)**

Errors and iterative Methods: Truncation and Round-off Errors. Floating Point Computation, Overflow and underflow. Single and Double Precision Arithmetic, Iterative Methods. Review of Taylor's Theorem and Mean value Theorem (No proofs).

Solutions of Algebraic and Transcendental Equations: Bisection method, Secant Method, Newton Raphson method. Comparison and error estimation

Unit – II **(10 Hours)**

Interpolation: Concept of Interpolation, Lagrange Form of interpolating polynomial, Newton's Forward and Backward Differences, Newton's Forward and Backward Interpolation Formulas.

Regression: Algorithm for Least square fitting of a straight line, Fitting a Power function, and Exponential Function using conversion to linear relation by transforming the variables.

Unit – III **(7 Hours)**

Numerical Differentiation: Approximating the derivative of a function given in the form of discrete data, Numerical Computation of First and second order derivative of a function given in closed form (using Taylor's expansion) , errors in Numerical Differentiation.

Numerical Integration: Newton Cotes Quadrature methods for evaluation of definite integrals numerically, Trapezoidal Rule, Simpson's 1/3 and 3/8 Rules. Derivation of composite formulae for these methods and discussion of error estimation

Unit – IV **(5 Hours)**

Solution of Ordinary Differential Equations: First Order ODE's: solution of Initial Value problems: (1) Euler's Method and (2) Runge Kutta methods

References:

Essential Readings:

1) Elementary Numerical Analysis, K. E. Atkinson, 3rd Edition, Wiley India Edition, 2007
2) Introduction to Numerical Analysis, S. S. Sastry, 5th Edition, PHI Learning Pvt. Ltd, 2012
3) Computational Physics, Darren Walker, 1st Edition, Scientific International Pvt. Ltd, 2015
4) Applied numerical analysis, Cutis F. Gerald and P. O. Wheatley, Pearson Education, 2007

Additional Readings:

1) An Introduction to Computational Physics, T. Pang, Cambridge University Press, 2010
2) Numerical Recipes: The art of scientific computing, William H. Press, Saul A. Teukolsky and William Vetterling, Cambridge University Press, 3rd Edition, 2007
3) Computational Problems for Physics, R. H. Landau and M. J. Páez, CRC Press, 2018

PRACTICAL COMPONENT

(15 Weeks with 4 hours of laboratory session per week)

The aim of this lab is not just to teach computer programming and numerical analysis but to emphasize its role in solving problems in Physics.

- The course will consist of practical sessions and lectures on Python.
- Assessment is to be done not only on the programming but also on the basis of formulating the problem.
- The list of recommended programs is suggestive only. Students should be encouraged to do more physics applications. Emphasis should be given to formulate a physics problem as mathematical one and solve by computational methods.
- At least 6 programs must be attempted (taking at least one from each unit).

Unit I

Basic Elements of Python: The Python interpreter, the print statement, comments, Python as simple calculator, objects and expressions, variables (numeric, character and sequence types) and assignments, mathematical operators. Strings, Lists, Tuples and Dictionaries, type conversions, input statement, list methods. List mutability, formatting in the print statement

Control Structures: Conditional operations, i*f, if-else, if-elif-else, while* and *for* Loops, indentation, break and continue, List comprehension. Simple programs for practice like solving quadratic equations, temperature conversion etc.

Functions: Inbuilt functions, user-defined functions, local and global variables, passing functions, modules, importing modules, math module, making new modules. Writing functions to perform simple operations like finding largest of three numbers, listing prime numbers, etc. Generating pseudo random numbers

Recommended List of Programs

(a) Make a function that takes a number *N* as input and returns the value of factorial of *N*. Use this function to print the number of ways a set of *m* red and *n* blue balls can be arranged.
(b) Generate random numbers (integers and floats) in a given range and calculate area and volume of regular shapes with random dimensions.
(c) Generate data for coordinates of a projectile and plot the trajectory. Determine the range, maximum height and time of flight for a projectile motion.

Unit II

NumPy Fundamentals: Importing *Numpy*, Difference between List and NumPy array, Adding, removing and sorting elements, creating arrays using *ones(), zeros(), random(), arange(), linspace()*. Basic array operations (*sum, max, min, mean, variance*), 2-d arrays, matrix operations, reshaping and transposing arrays, savetxt() and loadtxt().

Plotting with Matplotlib: *matplotlib.pyplot* functions, Plotting of functions given in closed form as well as in the form of discrete data and making histograms.

Recommended List of Programs

(a) Given a function in closed form y=f(x), generate numpy arrays for x and y and plot y as a function of x with appropriate scale and legend.
(b) Generate data for coordinates of a projectile and plot the trajectory.
(c) Given the expressions in closed form, plot the displacement-time and velocity-time graph for the un-damped, under damped critically damped and over damped oscillator.

Unit III

Root Finding

a) Determine the depth up to which a spherical homogeneous object of given radius and density will sink into a fluid of given density.
b) Solve transcendental equations like $\alpha = \tan(\alpha)$.
c) To approximate nth root of a number up to a given number of significant digits.

Unit IV

Least Square fitting

Make function for least square fitting, use it for fitting given data (*x,y*) and estimate the parameters *a, b* as well as uncertainties in the parameters for the following cases:

a) Linear ($y = ax + b$)
b) Power law ($y = ax^b$)

c) Exponential ($y = ae^{bx}$)

Interpolation:

(a) Write program to determine the unique polynomial of a degree *n* that agrees with a given set of (*n*+1) data points (x_i, y_i) and use this polynomial to find the value of *y* at a value of *x* not included in the data.

(b) Generate a tabulated data containing a given number of values (x_i, $f(x_i)$) of a function f(x) and use it to interpolate at a value of x not used in table.

Unit V

Numerical Differentiation

a) Given displacement at equidistant time values, calculate velocity and acceleration and plot them.

b) Compute the left, right and central approximations for derivative of a function given in closed form. Plot both the function and derivative (forward, backward and central derivatives) on the same graph. Plot the error as a function of step size on a log-log graph, study the behaviour of the plot as step size decreases and hence discuss the effect of round off error.

Numerical Integration:

a) Given acceleration at equidistant time values, calculate position and velocity and plot them.

b) Use integral definition of ln(x) to compute and plot ln(x) in a given range. Use trapezoidal and Simpson methods and compare the results.

c) Verify the rate of convergence of the composite Trapezoidal and Simpson methods by approximating the value of a given definite integral.

References

1) Documentation at the Python home page (https://docs.python.org/3/) and the tutorials there (https://docs.python.org/3/tutorial/).
2) Documentation of NumPy and Matplotlib: https://numpy.org/doc/stable/user/ and https://matplotlib.org/stable/tutorials/
3) Computational Physics, Darren Walker, 1st Edition, Scientific International Pvt. Ltd, 2015
4) Introduction to Numerical Analysis, S. S. Sastry, 5th Edition, PHI Learning Pvt. Ltd, 2012
5) Elementary Numerical Analysis, K. E. Atkinson, 3rd Edition, Wiley India Edition, 2007
6) Applied numerical analysis, Cutis F. Gerald and P. O. Wheatley, Pearson Education, 2007