DISCIPLINE SPECIFIC ELECTIVE COURSE – PHYSICS DSE 14a: INTRODUCTION TO NUMERICAL METHODS

Course Title & Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course
		Lecture	Tutorial	Practical		
Introduction to Numerical Methods PHYSICS DSE 14a	**4**	**2**	**0**	**2**	Class XII Pass with Science	Elementary calculus

LEARNING OBJECTIVES

The main objective of this course is to introduce the students to the field of numerical analysis enabling them to solve a wide range of physics problems. The skills developed during the course will prepare them not only for doing fundamental and applied research but also for a wide variety of careers.

LEARNING OUTCOMES

After completing this course, student will be able to,

- Analyse a physics problem, establish the mathematical model and determine the appropriate numerical techniques to solve it.
- Derive numerical methods for various mathematical tasks such as root finding, interpolation, least squares fitting, numerical differentiation, numerical integration, and solution of initial value problems.
- Analyse and evaluate the accuracy of the numerical methods learned.

In the laboratory course, the students will learn to implement these numerical methods in Python and develop codes to solve various physics problems and interpret the results.

SYLLABUS OF PHYSICS DSE – 14a
THEORY COMPONENT

Unit – I **(7 Hours)**

Approximation and errors in computing: Introduction to numerical computation, Taylor's expansion and mean value theorem; Floating point computation, overflow and underflow; IEEE single and double precision format; Rounding and truncation error, absolute and relative error, error propagation.

Solutions of algebraic and transcendental equations: Basic idea of iteration method, Bisection method, Secant method, Newton Raphson method; comparison of order of convergence.

Unit – II **(7 hours)**

Interpolation: Interpolation and Lagrange polynomial, divided differences, Newton divided-difference form of the interpolating polynomial with equally spaced nodes. Theoretical error in interpolation.

Least Squares Approximation: Least squares linear regression, Least squares regression for exponential and power functions by taking logarithm.

Unit - III **(8 Hours)**

Numerical Differentiation: Using finite difference to approximate derivatives of first and second order using Taylor series and error in this approximation.

Numerical Integration: Newton Cotes quadrature methods; derivation of Trapezoidal and Simpson (1/3 and 3/8) rules from Lagrange interpolating polynomial; error and degree of precision of a quadrature formula; composite formulae for trapezoidal and Simpson methods; Gauss Legendre quadrature method.

Unit - IV **(8 Hours)**

Initial Value Problems: Solution of initial value problems by Euler, modified Euler and Runge Kutta (RK2, RK4) methods; local and global errors, comparison of errors in the Euler and RK methods, system of first order differential equations. Solving higher order initial value problems by converting them into a system of first order equations.

References:

Essential Readings:

1) Introduction to Numerical Analysis, S. S. Sastry, 5th edition, 2012, PHI Learning Pvt. Ltd.
2) Elementary Numerical Analysis, K. E. Atkinson, 3rd edition, 2007, Wiley India Edition.
3) Numerical methods for scientific and engineering computation, M. K. Jain, S. R. K. Iyenger and R. K. Jain, 2012, New Age Publishers
4) A Friendly Introduction to Numerical Analysis, B. Bradie, 2007, Pearson India

Additional Readings:

1) Numerical Recipes: The art of scientific computing, W. H. Press, S. A. Teukolsky and W. Vetterling, 3rd edition, 2007, Cambridge University Press
2) Numerical Methods for Scientists and Engineers, R. W. Hamming, 1987, Dover Publications
3) Applied numerical analysis, C. F. Gerald and P. O. Wheatley, 2007, Pearson Education
4) Numerical Analysis, R. L. Burden and J. D. Faires, 2011, Brooks/Cole, Cengage Learning
5) Numerical Methods, V. N. Vedamurthy and N. Ch. S.N. Iyengar, 2011, Vikas Publishing House

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.

- 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.
- The students should be encouraged to develop and present an independent project.
- **At least 12 programs must be attempted (taking two from each unit). The implementation is to be done in Python. Use of scipy inbuilt functions may be encouraged**

Unit 1

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, ***if, 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. Use of inbuilt functions to generate 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) Write functions to convert Cartesian coordinates of a given point to cylindrical and spherical polar coordinates or vice versa.
d) Solve quadratic equations for the three cases of distinct real, double real and complex conjugate roots.

Unit 2

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) To generate data for coordinates of a projectile and plot the trajectory. Determine the range, maximum height and time of flight for a projectile motion.
(b) To plot the displacement-time and velocity-time graph for the undamped, under damped critically damped and over damped oscillator using *matplotlib* (using given formulae).
(c) To generate array of N random numbers drawn from a given distribution (uniform, binomial, poisson and gaussian) and draw histogram using *matplotlib* for increasing N to verify the distribution.
(d) To approximate the elementary functions (e.g. $\exp(x)$, $\sin(x)$, $\cos(x)$, $\ln(1+x)$, etc.) by a finite number of terms of Taylor's series and discuss the truncation error. To plot the function as well the nth partial sum of its series for various values of *n* on the same graph and visualise the convergence of series.

Unit 3

Root Finding: Implement the algorithms for Bisection, Secant and Newton Raphson methods or their combinations to,

(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) Approximate nth root of a number up to a given number of significant digits.

Unit 4

Interpolation and Least Square Fitting:

a) Given a dataset (x, y) with equidistant x values, prepare the Newton's divided difference table. Generate a tabulated data for an elementary function, approximate it by a polynomial and compare with the true function.
b) Given a dataset (x, y) corresponding to a physics problem, use Lagrange and Newton's forms of interpolating polynomials and compare. Determine the value of y at an intermediate value of x not included in the data set. This may be done with equally spaced and non-equally spaced x-values.
c) Make Python 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 :

i. Linear $(y = ax + b)$
ii. Power law $(y = ax^b)$ and
iii. Exponential $(y = ae^{bx})$

The real data taken in physics lab may be used here.

d) Compare the interpolating polynomial for a given dataset (following a known form e.g. exponential) with the approximation obtained by least square fitting.

Unit 5

Differentiation and Integration:

a) To compute the left, right and central approximations for derivative of a function given in closed form. Plot both the function and derivative on the same graph. Plot (using *matplotlib)* 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.
b) Use integral definition of error function to compute and plot erf(x) in a given range. Use Trapezoidal, Simpson and Gauss Legendre methods and compare the results for small and large values of x.
c) Verify the degree of precision of each quadrature rule.
d) Approximate the value of π by evaluating the integral $\int_0^\infty \frac{1}{x^2+1} dx$ using Simpson and Gauss Legendre method. More integrals may be evaluated.

Unit 6

Initial Value Problems (IVP):

a) Compare the errors in Euler, RK2 and RK4 by solving a first order IVP with known solution. Reduce the step size to a point where the round off errors takes over.
b) Radioactive decay: With a given number of initial nuclei and decay constant plot the number of nuclei left as a function of time and determine the half life
c) Solve a system of two first order differential equations by Euler, RK2 and RK4 methods. Use it to solve an nth order IVP. Solve a damped free and forced harmonic oscillator problem using this.
d) Solve a physics problem like free fall with air drag or parachute problem using RK method.
e) Obtain the current flowing in a series LCR circuit with constant voltage for a given set of initial conditions.

References for laboratory work:

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, D. Walker, 1st edition, 2015, Scientific International Pvt. Ltd
4) An Introduction to Computational Physics, T. Pang, 2010, Cambridge University Press
5) Python Programming and Numerical Methods - A Guide for Engineers and Scientists, Q. Kong, T. Siauw, A. M. Bayen, 2021, Academic Press