Assignments and Guidelines for Mathematical Physics III Lab

Paper: PHSACOR08P

25% of the hours, on an average, allotted for this lab component per week, should be devoted regularly to the detailed discussions on the underlying theory of the following numerical methods including efficiency of the methods in each case and to the discussion on the results of the assignments performed in the previous class. It is recommended to use an IDE, preferably Spyder, to write programs by the students in Python. For documentation and sharing purpose Jupyter may also be used.

---

1. ODE initial value problems by RK2 & RK4:
   (a) Find numerical solution of the equation $\dot{x} = \cos(\pi t)$ with initial value of $x = 0$ at $t = 0$ within $[0, 10]$ using –
      i. Euler method,
      ii. RK2 method,
      iii. RK4 method;

      taking a small step size and plot all the solutions on same on the same graph using matplotlib.pyplot for comparison.
   (b) Reduce the step size to one fifth of the previous one and repeat the work.
   (c) Solve the Hamilton's equations of one dimensional motion of a particle with unit mass attached with a spring (of spring constant π^2) fixed with a rigid support at the other end and draw its phase space plot applying Euler method and RK4 method on same graph paper along with the plot from analytical solution.

> **Guideline:** For subpart (a) and (b) take initial condition as $t = 0, x = 0$ and find the solution up to $t = 2$. For part (a) Δt may be taken as 0.1 which has to be reduced for part (b). Different line-types are to be used for plotting multiple plot on the same graph paper. For the part (c) the plot should be done for at least one complete cycle using $\Delta t = 0.02$.

2. Use of numpy ndarray objects for various matrix operations.
   (a) Matrix generation including special matrices, reshaping, addition, subtraction, transposition, matrix multiplication, element-wise product, trace calculation, slicing and concatenation using numpy ndarray objects.
   (b) Checking Hermiticity and Unitarity of matrices.
   (c) Row operations and column operations within matrices using ndarray objects.

> **Guideline:** All the operations should involve ndarray objects only (without any use of numpy matrix objects to avoid any possible confusion during initiation). Codes are to be written for matrices of any arbitrary dimension. In part (b) generation of Hermitian matrices may be performed using the methods introduced in part (a) and numpy.linalg may be used to generate unitary matrices or to calculate norms, if required.

3. Solution of Linear system of equations by Gauss elimination method and by Gauss Jordan method.
   (a) Solution of linear system of equations with three unknowns using Gauss elimination method. Calculation of determinant of the coefficient matrix from it.
   (b) Solution of linear system of equations with three unknowns using Gauss-Jordan method. Calculation of determinant and inverse of coefficient matrix from it.
   (c) Generalisation of the above codes for arbitrary numbers of variables and equations.

Guideline: In part (a) the same code may determine the determinant first and if the determinant is found to be non-zero it will proceed to find the solution using back substitution. In part (b) too, the code should find the determinant first and if it is found to be non-zero, then only should proceed to find the inverse and solution. In either case pivoting technique need not be implemented, but students should be aware of limitations of the methods without using pivoting technique.

4. Solution of linear system of equations by Gauss-Seidal iterative method.
   (a) Solution of linear system of equations with three unknowns using Gauss-Seidal iterative method.
   (b) Choosing suitable set of constants in inhomogeneous part of linear equations to find the inverse of the coefficient matrix.
   (c) Use the code for determinant calculation through Gauss elimination method to show that the determinant of inverse matrix is the reciprocal of the same of the original matrix.

Guideline: In part (b) the solutions of the linear system has to be found for three times using only one non-vanishing constant (unity) as the inhomogeneous term of the system of equations in each case. The student should be aware of the limitations of Gauss-Seidal method.

5. Gram-Schmidt orthogonalisation method with 3 vectors.
   (a) Apply the method for two arbitrarily chosen linearly independent vectors ($\mathcal{R}^n$) and verify the result explicitly with inner products.
   (b) Apply the same for three arbitrarily chosen linearly independent vectors ($\mathcal{R}^n$) using file input and output methods.

Guideline: The programs should run for any arbitrary value of n. At the end of the program in each case, the orthogonality of the vectors hence obtained should be checked.

6. Explicit calculation of largest eigenvalue calculation by power iterative method for real symmetric matrix and corresponding eigenvector.
   (a) Use power method for a 3×3 real symmetric matrix to find the largest eigenvalue by magnitude and corresponding eigenvector.
   (b) Generalise the previous code for the real symmetric matrices of arbitrary dimension with file I/O.

Guideline: For part (b) the input matrices should be read from an input file and the program should check the dimension of an input matrix to ensure it as a square matrix, before processing it. Also show that the eigenvalue and eigenvector so found, satisfies the eigenvalue equation of the matrix itself.

7. Boundary value problems (by finite difference method with fixed grid size): (any one to be performed in the class)
   (a) Laplace equation in **2D** with Dirichlet boundary condition.
   (b) **1D** Fourier heat equation with Dirichlet boundary condition.
   (c) Poisson equation in **2D** with Dirichlet boundary condition.
   (d) Wave equation in **1D** on a string fixed at two ends with a given initial configuration.

Guideline: Numpy array operations should be used instead of explicit loops. The final solution should be plotted in 3-d using matplotlib.pyplot. The grid size should be small enough to minimize discretization effect and hence it has to be optimized acording to the memory and speed of the computing machine. Care should be taken to choose optimum relation between grid sizes of two independent varaibles to ensure convergence, where required. It is encouraged to allot different equation (from the above four) to different group of students in a class and the final results are shared among all the students of the class, so that every student in the class gets aware of the characteristics of all the solutions.

8. Find square roots, cube roots of a complex number using two dimensional Newton-Raphson method.
   (a) Finding square root of a complex number using Newton-Raphson method in two dimension.
   (b) Finding cube roots of a complex number using Newton-Raphson method in two dimension.
   (c) Comparison of the above results with the values of the square root and cube roots of the complex numbers using Euler's formula. Verify the results also by finding square and cube respectively.
   (d) Find complex roots of an arbitrary algebraic equation.

> **Guideline:** For finding nth root of a complex number z_0, first construct the complex equation of the form $z^n - z_0 = 0$. Separate the complex equation into two real equations equating real and imaginary parts separately from either side, to find their solution using Newton-Raphson method in 2-dimension. Newton-Raphson method in D-dimension should be discussed before implementing it in two dimension. Verify the result of part (a) using library function cmath.sqrt. For part (b) one of the complex roots is to be found by Newton-Raphson method and the other complex root may be calculated from it; also verify the solution of the separated real equations using library function fsolve from scipy.optimize. For part (c), use newton from scipy.optimize to verify the solution directly.

9. Integrals for different mathematical expressions.
   (a) Integral transform: Fourier Transform of $\exp(-\alpha x^2)$ and $x\exp(-\alpha x^2)$ and plot real part and imaginary part of the transform in each case. Show that in each case either of the real and imaginary part vanishes everywhere.
   (b) Dirac Delta Function: Evaluate $\frac{1}{\sqrt{2\pi}\sigma}\int \exp\left(-\frac{(x-2)^2}{2\sigma^2}\right) f(x+3)\mathrm{d}x$, (integration to be performed within $2 \pm 5\sigma$) for $\sigma = 1, 0.1, 0.01$ and show it tends to $f(5)$.

> **Guideline:** Use quad from scipy.integrate to carry out the integrations separately for real and imaginary part of the integrals. For part (b) the function $f(x)$ may be chosen as x^n, $\sin x$, $\cos x$, $\exp x$, $\ln x$ etc. and the dependence on σ should be demonstrated graphically (using intermediate values of σ as well). Note that it is independent on σ for $f(x)$ being linear.

10. Use of mathematical packages for solving
    - differential equations (up to second order) and
    - matrix eigenvalue problems.

    (Any one of the following to be performed in the class):
    (a) Introduction of Octave with its basic features and use its library functions to carry out the tasks mentioned above.
    (b) Introduction of scipy.integrate and numpy.linalg with its detailed features and use those to carry out the tasks mentioned above.

> **Guideline:** Use of gnu octave as an IDE, may be preferred to introduce the basic syntaxes of octave. Plotting of solutions of differential equations under different physical perspectives and file handling for input/output for matrix eigenvalue problems in different physical problems are to be performed.