Department of Mathematics, IIT Madras

MA-5105-Scientific Computing & Advanced Programming Lab

Practice Problems 3

August 23, 2026

Lecturer: Saurav Samantaray *TA: Rohit Kumar Mani*

Q. 1 Generalising the Vector and Matrix Classes The `TBTVEC` and `TBTMAT` classes covered in lectures only work for fixed 2×1 vectors and 2×2 matrices. In this problem you will generalise these classes so that their size n is decided by the user at run time, and add several standard mathematical operations to each class.

Task 1: Generalised vector class `NVEC`. Design a class `NVEC` that represents a vector of any length n, where n is supplied by the user (e.g. via `input()` or as a constructor argument), along with the n entries themselves. Your class should implement:

(a) `__init__(self, entries)`: stores the entries as a list, and records $n =$ `len(entries)`.

(b) `__str__(self)`: displays the vector in a readable form, regardless of its length.

(c) `norm(self)`: returns the Euclidean (2-)norm,

$$\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

(d) `dot(self, other)`: returns the dot product of `self` with another `NVEC` of the same length, raising a suitable error if the lengths do not match.

(e) `__add__(self, other)` and `__sub__(self, other)` to return a new `NVEC`.

Task 2: Generalised matrix class `NMAT`. Design a class `NMAT` that represents an $n \times n$ matrix, where n and all n^2 entries are supplied by the user. Store the entries as a list of lists (i.e. a list of rows). Your class should implement:

(a) `__init__(self, entries)`: stores the matrix and records its size n.

(b) `__str__(self)`: displays the matrix row by row. cofactor expansion along the first row. did) before attempting to invert.

(c) `transpose(self)`: returns a new `NMAT` that is the transpose of `self`.

(d) `multiply(self, other)`: returns the product of `self` with another `NMAT`, *or* with an `NVEC` (matrix–vector product), detecting which case applies.

Task 3: Taking user input. Write a short driver routine that:

(a) asks the user how many entries their vector (or what size their matrix) should have,

(b) loops to collect each entry from the user (validating that numeric input has been given),

(c) constructs an `NVEC` or `NMAT` from the collected entries, and

(d) prints the object, along with its norm (for a vector)

Worked check. Test `NVEC` with the entries $(3, 4, 12)$ and confirm the norm is 13. Test `NMAT` with the identity matrix of size 3.

Q. 2 Lagrange Interpolation Class

In many industrial applications, data is only available at a discrete set of points (x_0, y_0), (x_1, y_1), $\ldots$, (x_n, y_n); for example, readings from a sensor, values from a steam table, or calibration data for an instrument. To estimate the value of y at a point x that lies between the given data points, we can use **Lagrange interpolation**.

Background. The Lagrange interpolating polynomial of degree n passing through $n + 1$ points is given by

$$P(x) = \sum_{i=0}^{n} y_i \, L_i(x),$$

where $L_i(x)$ is the i-th Lagrange basis polynomial, defined as

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Task. Design a Python class `LAGRANGE`

(a) that stores a set of data points and can interpolate a y-value at any given x

Design Ideas:

(a) `__init__(self, x_points, y_points)`: stores the given lists of x and y coordinates.

(b) `basis(self, i, x)`: computes and returns the value of the basis polynomial $L_i(x)$ for a given index i and evaluation point x.

(c) `interpolate(self, x)`: uses `basis()` to compute and return the interpolated value $P(x)$.

Worked check. Test your class using the data points

$$(0, 1), \quad (1, 2), \quad (2, 9), \quad (3, 28),$$

and confirm that your class returns a sensible interpolated value at $x = 1.5$.

Extension (optional). Add a method `derivative_estimate(self, x, h)` that uses your `interpolate()` method to approximate $P'(x)$ via a central difference,

$$P'(x) \approx \frac{P(x+h) - P(x-h)}{2h}.$$

Discuss briefly how the choice of h affects the accuracy of this estimate.

Q. 3 Projectile Motion Simulation Using Classes

A projectile is launched from the ground with an initial velocity v_0 at an angle θ above the horizontal. Neglect air resistance and assume that the only force acting on the projectile is gravity.

The position $(x(t), y(t))$ and velocity $(v_x(t), v_y(t))$ of the projectile satisfy

$$\frac{dx}{dt} = v_x, \qquad \frac{dy}{dt} = v_y,$$

$$\frac{dv_x}{dt} = 0, \qquad \frac{dv_y}{dt} = -g,$$

where

$$g = 9.81 \text{ m/s}^2.$$

Your task is to design a Python program that simulates the motion of the projectile using **three interacting classes**.

(a) **Design a `Particle` class.**
The class should store the current position and velocity of the projectile:

$$(x, y, v_x, v_y).$$

Include appropriate methods to update and access the state of the particle.

(b) **Design a `Force` class.**
Represent the gravitational force acting on the particle. The force should be determined by the particle mass and the gravitational acceleration g.

(c) **Design a `Simulation` class.**
The simulation should contain a `Particle` object and a `Force` object. Advance the particle in time using the **Forward Euler method**:

$$x_{n+1} = x_n + v_{x,n}\Delta t,$$
$$y_{n+1} = y_n + v_{y,n}\Delta t,$$
$$v_{x,n+1} = v_{x,n} + \frac{F_x}{m}\Delta t,$$
$$v_{y,n+1} = v_{y,n} + \frac{F_y}{m}\Delta t.$$

(d) Simulate a projectile with

$$v_0 = 20\ \mathrm{m/s}, \qquad \theta = 60^\circ,$$

starting from

$$(x_0, y_0) = (0, 0).$$

The initial velocity components are

$$v_x(0) = v_0 \cos\theta, \qquad v_y(0) = v_0 \sin\theta.$$

(e) Run the simulation until the projectile returns to the ground, i.e. until

$$y < 0.$$

(f) Determine from your simulation:

- the approximate time of flight,
- the maximum height,
- the horizontal range.

(g) Compare your numerical results with the analytical solution. The exact values are given by

$$T = \frac{2v_0 \sin\theta}{g},$$
$$H_{\max} = \frac{v_0^2 \sin^2\theta}{2g},$$

and

$$R = \frac{v_0^2 \sin(2\theta)}{g}.$$

(h) Repeat the simulation using different values of Δt. Investigate how the accuracy of the numerical results changes as Δt is decreased.

Additional question: Explain the role of each of the three classes in the program. Why is it useful to represent the particle, the force, and the simulation as separate objects?