UNIVERSITÀ
DEGLI STUDI DI MILANO-BICOCCA

SYLLABUS DEL CORSO

Physical Modelling and Simulations

2526-1-E3004Q004

Aims

This course aims to provide students with the fundamentals of programming and computational methods for physics. Students will learn how to model and simulate simple physical problems, make predictions, and analyze the results.

objectives according to Dublin Descriptors:
2. Applying knowledge and understanding
3. Making judgements
5. Learning skills

Contents

• Introduction to Bash and Bash scripts.
• Introduction programming, compiled vs inteprted languages. Pyhton and fundamental libraries: NumPy, SciPy, Matplotlib. Mention to symbolic calculus with SimPy.
• Basic notion about object-oriented programming and parallel computation
• Numerical methods: interpolation, approximation, root finding, differentiation, and integration.
• Linear algebra: basic matrix operations, linear systems, eigenvalue equations.
• Ordinary differential equations.
• Modelling simple physical problems: examples from classical mechanics.
• Gravitational dynamics of multi-body systems.
• Monte Carlo methods for simulations.

Detailed program

Part 1: Basics of Programming and Numerical Computing

Introduction to Bash: shell, Unix commands, creating and managing files and directories, environment variables, and aliases.
Creating and executing a Bash script, first steps in coding: variables, loops, conditions, functions.

Introduction to programming languages. Python, brief mention to C++; create and execute a code. Variable types and fundamental operations. Reading and writing files, defining functions, importing modules. Lists and list manipulations.
Specific Modules
NumPy: fundamental functions, pseudorandom numbers, multidimensional arrays.
Matplotlib: creating and manipulating images, 2D and 3D plotting.

Introduction to object-oriented programming in Python: classes, members, methods, inheritance.

Numerical methods for finding zeros: bisection, Newton's method and extensions, secant method. Implementations in Python with SciPy.

Interpolation in one and two dimensions. Implementation with SciPy.

First and second derivatives using finite differences. Numerical implementation, np.gradient

Numerical methods for integration: rectangle method, trapezoidal rule, Cavalieri-Simpson's formula, Gauss quadrature. Implementation with SciPy.integrate.

Operations with arrays and matrices: sums, differences, row-by-column multiplication. Solving linear systems, computing the determinant and inverse of a matrix. Diagonalization, computing eigenvalues and eigenvectors with NumPy.

Solving ordinary differential equations: Euler and Runge-Kutta methods. Boundary conditions and the Shooting method.

Part 2: Modelling and Simulation
Solving elementary physics problems using numerical methods:
• **Kinematics of a material point**: numerical derivation of trajectories in one, two, and three dimensions.
• **Mechanics**: equations of motion for the simple and damped pendulum.
• Double pendulum.
• Mechanical systems with friction.
• Rolling motion.

Modelling complex problems:
• Many-particle collisional systems (e.g., gas in a box and computation of physical observables such as temperature and pressure).
• Many-particle non-collisional systems: gravitational potential of a self-gravitating system.
• Computational time. introduction to parallel computing.

I**ntroduction to Monte Carlo methods**: historical background, Metropolis-Hastings algorithm. Numerical integration using Monte Carlo methods.

Prerequisites

Knowledge of basic Mathematics is required, elementary functions and algebraic calculus: equations, inequalities,

trigonometry.

Teaching form

Lectures will be delivered both remotely, in asynchronous mode, for 30 hours, and in person for 18 hours. Each lesson includes a theoretical lecture component (30 hours) as well as a guided coding session (18 hours).

Textbook and teaching resource

Lecture notes and codes examples provided gradually.
Suggested Textbook for Python programming: “Introduction to Python Programming by OpenStack”, reperibile liberamente online https://openstax.org/details/books/introduction-python-programming (CC license)

Semester

First semester

Assessment method

The assessment relies on presenting a group project with a final report, to be discussed during an oral exam, in order to evaluate the programming and modeling skills acquired during the course. No partial exams are planned, but weekly exercises will be assigned. If completed and discussed correctly during the exam, these exercises will grant an additional 1 to 3 points on the final grade.

Office hours

Every day with preferred requested appointment via email

Sustainable Development Goals

QUALITY EDUCATION