# Programming Techniques 2025-2026

## Course exercise 1: basic dynamical simulation

Hannu Parviainen

Universidad de la Laguna

September 29, 2025

# Exercise 1: basic dynamical simulation

**Exercise:** Direct numerical integration of a system consisting of gravitationally interacting particles using the leapfrog integration method.

- See TAP p. 32-48 and p. 74-85

**To do**

1. Create module `geometry` stored in `geometry.f90` file that contains:
   - types `vector3d` and `point3d`, both with (64-bit) real components `x`, `y`, and `z`.
   - functions `sumvp`, `sumpv`, `subvp`, `subpv`, `mulrv`, `mulvr`, `divvr` for adding and subtracting points and vectors, and for multiplying and dividing vectors with reals.
   - operators matching these functions.
   - function `distance` that calculates the distance between two 3d points.
   - function `angle` that calculates the angle between two vectors a and b (in radians).
   - function `normalize` that takes a vector a and returns it divided by its length.
   - function `cross_product` that takes two 3d vectors (a and b) and returns their cross product.

# Exercise 1: basic dynamical simulation

2. Create a module `particle` stored in `particle.f90` that uses the `geometry` module and contains a type `particle3d`. This type should have components: a `point3d` variable `p` storing the particle's position, a `vector3d` variable `v` storing the particle's velocity, and a real variable `m` storing the particle's mass.
3. Take the old `leapfrog.f90` code and modify it to use the `geometry` and `particle` modules. Store the program in `ex1.f90` file.
4. Modify the program to read the simulation setup from a file given as a command line argument.
5. Modify the program to write the simulation data into a file "output.dat", where each line contains the simulation state as "time p1x p1y p1z p2x p2y p2z ... pnx pny pnz".
6. Write a Makefile for the project

# Exercise 1: basic dynamical simulation

7. Create a pull request (PR) with your code and add all the course attendees as reviewers.
8. Review the code in the other student's PRs and write at least one comment to each.
9. (Optional): revise your code based on the comment's you've received.
10. Write a `README.md` for the exercise where you explain the code.
11. In the PR, tag the teacher and say that you're finished with the exercise.

# Exercise 1: basic dynamical simulation

**Scoring** is based on

- Code (80%)
  1. Functionality: the code should work as described.
  2. Understandability: the code should be clear to read and logical.
  3. Comments: all the functionality in both the module and the program should be documented using comments.
  4. Use of guidelines: the code should use the type and method names exactly as written in the exercise.
- Collaboration (20%): GitHub PRs, PR comments, and participation.

**Bonus points** from anything that goes beyond to what I've taught at the classes, frequent push requests, interaction (asking questions, answering questions, and so on). If in doubt, ask!

**Notes:** All the source files should be saved in your `ex1` folder. Add and commit your changes often and feel free to try working with different branches. All the real variables should be 64-bit floats (double precision).