Programming Techniques 2024-2025

Course exercise 1: basic dynamical simulation

Hannu Parviainen

Universidad de la Laguna

October 9, 2024

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 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.
    ▶ function orthv that takes two vectors (a and b) and returns a vector orthogonal to them.

Exercise 1: basic dynamical simulation

2. Create a module `particle` stored in `particle.f90` that uses the geometry module and contains a type `particle`. This type should have components: a `point3d` variable `p` storing the particle's position, a `vector2d` variable `v` storing the particle's velocity, and a real variable `m` storing the particle's mass.

3. Take the code from TAP page 34 and modify it to use the `geometry` and `particle` modules. Store the program in `ex1.f90` file.

4. Write a Makefile

5. Create a pull request with your final code.

Note: 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. Also feel free to create pull requests as often you like, but remember that other students can see the code you add to the main repository (this is perfectly ok, though).