Test exam problems

Fundamentals of Simulation Methods

WS 2020/21

Lecturers: Prof. Dr. Frauke Gräter, Prof. Dr. Friedrich Röpke
Tutors: Benedikt Rennekamp, Fabian Kutzki, Giovanni Leidi, Siddhant Deshmukh

16. Februar 2021

1	2	3	4	Total

Name: _____ *Matrikel-Nr.:* _____

1) Short questions

a) Full multi-grid cycles: State whether the following statements are right or wrong, and discuss in 1-2 sentences why this is the case. Note: There might be any number of correct statements.

1) The computational cost of one V-cycle is $\mathcal{O}(N_{\text{grid}})$, where N_{grid} is the number of grid cells of the finest mesh.
2) For a three-dimensional grid, the computational cost for a full multi-grid cycle is twice as large as one V-cycle.
3) Multigrid methods are generally preferred over particle-mesh methods because of their favorable linear scaling with the grid size.
4) A full multi-grid starts from the finest grid.

b) Which of the following statements are advantages of a normalized representation of floating point numbers? Note: There might be any number of correct statements.

1) higher precision
2) less storage required
3) representation of a larger range of values
4) calculations always follow the law of associativity

c) 1000 particles of a monatomic gas shall be simulated using the Lennard Jones potential and a simple cut-off. Which of the following statements are correct. Note: There might be any number of correct statements.

1) Monte Carlo simulations of this system show a better scaling than Molecular Dynamics simulations of the same system.
2) One Monte Carlo step of this system is less computationally demanding compared to one Molecular Dynamics simulation step.

3) Both type of simulations require expensive pairwise force calculations between the particles.

4) A correct canonical ensemble can only be obtained with Molecular Dynamics simulations.

d) Euler and Lagrangian schemes

1) Consider now the system of Euler equations for solving compressible gas dynamics. Is the limit on the time step for Eulerian and Lagrangian codes the same? Why? What is the numerical interpretation of such limit?

2) Using explicit time stepping in Eulerian codes to study flows at low Mach numbers is a computationally inefficient method. Why?
Hint: compare the number of time steps required to follow the evolution of the flow moving across a length-scale L for two different (maximum) Mach numbers: $M_{max} = 10^{-4}$ and $M_{max} = 1$. Assume that the system is isothermal and that the grid has spatial resolution Δx.

2) Sampling with nonuniform probability distribution functions

Suppose you have a large set of random numbers x drawn from the interval $[0, 1)$ with a uniform probability distribution.

a) Calculate the transformation $y(x)$ such that you get random numbers y with the probability density function

$$p(y) = N \cdot y^2 \quad \text{for } y \in [0, 3) \tag{1}$$

Find also an appropiate value for N.

b) How many more random numbers with uniform probability distribution would you have to generate to obtain equally many random numbers according to $p(y)$ using instead the rejection method with a constant envelope function?

3) Discretization of the advection equation

The one dimensional advection equation for a field $\rho = \rho(x, t)$ is given by

$$\frac{\partial \rho}{\partial t} + v\frac{\partial \rho}{\partial x} = 0, \qquad \text{where } v = \text{const. and } v > 0. \tag{2}$$

a) Show that an upwind discretization (using cell i and $i-1$ to estimate the gradient) of the spatial part is mathematically equivalent to a central differentiation (using cells $i+1$ and $i-1$ to estimate the gradient) and an additional diffusion term

$$D\,\frac{\partial^2 \rho}{\partial x^2} \tag{3}$$

and calculate the numerical diffusion coefficient D.

b) Explain why the upwind discretization is preferred over the central difference discretization here, even though it adds a diffusion term.

c) We now approximate the time-derivative using an explicit Euler discretization:

$$\frac{\partial \rho}{\partial t} = \frac{\rho_{n+1} - \rho_n}{\Delta t}, \tag{4}$$

using all values for the spatial derivative from the old time step. State and motivate the Courant-Friedrichs-Levy (CFL) timestep criterion.

d) Write a *pseudo-code* to solve the following advection equation on a 3D Cartesian grid:

$$\frac{\partial \rho}{\partial t} + u(x)\frac{\partial \rho}{\partial x} + v(y)\frac{\partial \rho}{\partial y} + w(z)\frac{\partial \rho}{\partial z} = 0 \qquad \text{where } x, y, z \in [0,1] \times [0,1] \times [0,1] \tag{5}$$

Assume to have the same resolution on each axis and periodic boundary conditions. Use the *dimensional splitting* technique to integrate the equation in time. Start from the grid generation until the end of the time loop.

4) Code example

The following two functions are part of a code which adds particles to a discrete 1D density field (first function), solves Poisson's equation in some way to calculate a discretised acceleration field (not shown here) and maps the force field back to the particles (second function).

a) Assuming that all parts of the code which are not shown work perfectly fine, what problem will occur when running the code with the two functions shown below?

b) Modify the second function such that the above problem is solved. It's sufficient if you write pseudo code.

c) Do the functions work for periodic boundary conditions? Why?

```
void add_particle_to_density_field(double rho_field[N],/* the density field
    discretised on a grid of length N */
              int N,
              double cellsize, /* size of a single cell of the grid */
              double particle_pos, /* the particle position */
              double particle_mass) /* the particle mass */
{
  double xx = particle_pos / cellsize;
  int i = floor(xx); /* floor(x) truncates all decimal digits of the floating point
      number x, similar to ((int) x) in C (or: it does a strict rounding to the next
      lower integer number; floor(5.9) = 5) */

  double u = xx - i;

  int ii = i + 1;
  if(ii >= N)
    ii = 0;

  rho_field[i] += (1 - u) * particle_mass / cellsize;
  rho_field[ii] += u * particle_mass / cellsize;
```

```
}

double interpolate_force_field_to_particle_position(double acc_field[N],/* the
    acceleration field discretised on a grid of length N */
                    int N,
                    double cellsize, /* size of a single cell of the grid */
                    double particle_pos, /* the particle position */
                    double particle_mass) /* the particle mass */
{
  double xx = particle_pos / cellsize;

  int i = floor(xx + 0.5);

  double acceleration = acc_field[i];

  return acceleration * particle_mass;
}
```