HIGH PERFORMANCE COMPUTING for SCIENCE & ENGINEERING (HPCSE) I

HS 2021

EXERCISE 04: DIFFUSION & PARTICLE STRENGTH EXCHANGE (PSE)

Daniel Waelchli (wadaniel@ethz.ch)

Computational **S**cience and **E**ngineering **Lab**
ETH Zürich

12.11.2021

Outline

I. Diffusion
II. Exercise 4, Question 1 (Diffusion)
III. Exercise 4, Question 2 (PSE)
IV. Euler Cluster Usage

Diffusion

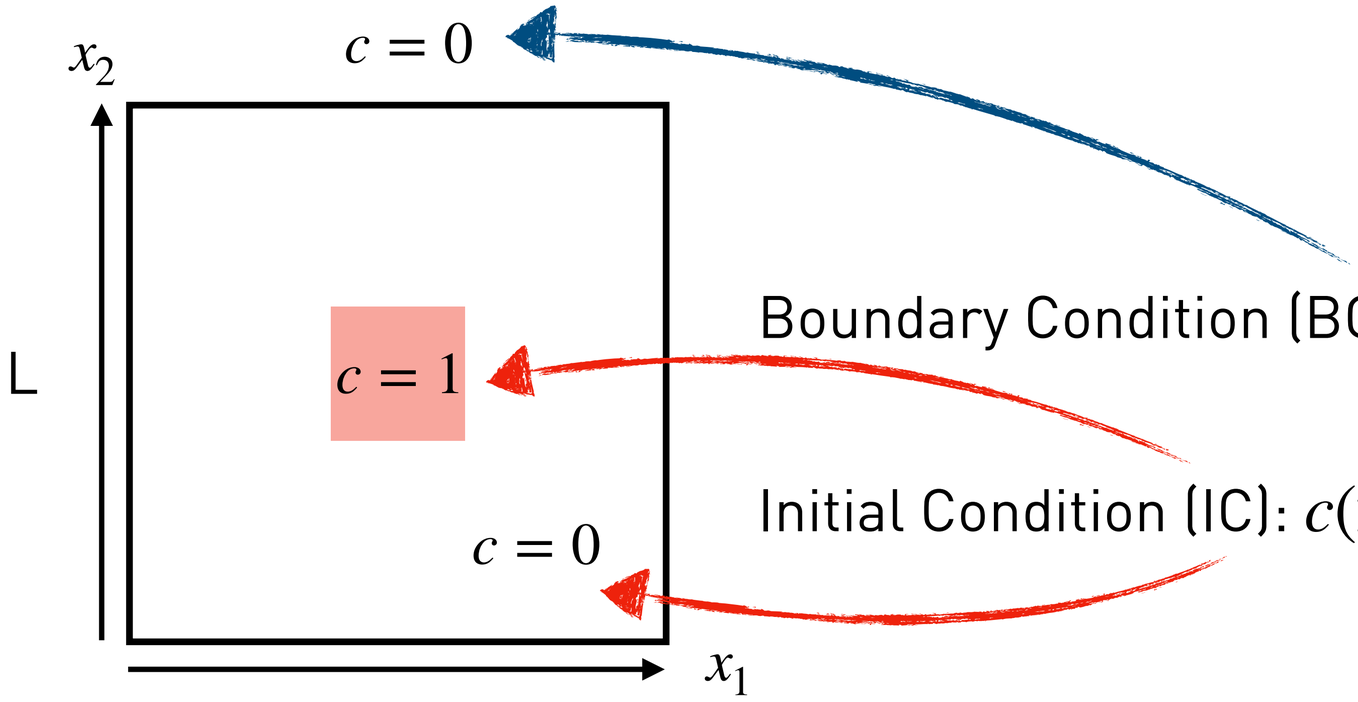
Diffusion is a physical process, which leads du a uniform distribution of particles.

In this exercise we consider the heat-flow in a 2D medium that can be described by:

$$\frac{\partial c(\mathbf{x},t)}{\partial t} = D\,\nabla^2 c(\mathbf{x},t)$$

$c(\mathbf{x},t)$ is the concentration of heat at position $\mathbf{x} = (x_1, x_2)$

D is a constant diffusion coefficient $[\frac{\partial x^2}{\partial t}]$



Boundary Condition (BC): $c(\mathbf{x},t) = 0$ for $t \geq 0$

Initial Condition (IC): $c(\mathbf{x},0) = \begin{cases} 1, & \text{if } ||\mathbf{x}||_1 \leq L/4 \\ 0, & \text{else} \end{cases}$

Numerical Integration

- Explicit Euler (**this exercise**): $\frac{c_{i,j}^{(n+1)} - c_{i,j}^{(n)}}{\Delta t} = D\left[\frac{c_{i-1,j}^{(n)} - 2c_{i,j}^{(n)} + c_{i+1,j}^{(n)}}{\Delta x^2} + \frac{c_{i,j-1}^{(n)} - 2c_{i,j}^{(n)} + c_{i,j+1}^{(n)}}{\Delta y^2}\right]$
  - Easy to implement
  - Requires few computations and often has acceptable accuracy.
  - Drawback: Instability, local errors are small but solution diverges exponentially over time.
    Condition on step size for stability is very small for 2D: for $\delta x = \delta y = h$ we get $\delta t \sim \mathcal{O}(h^2)$

von Neumann Stability Analysis

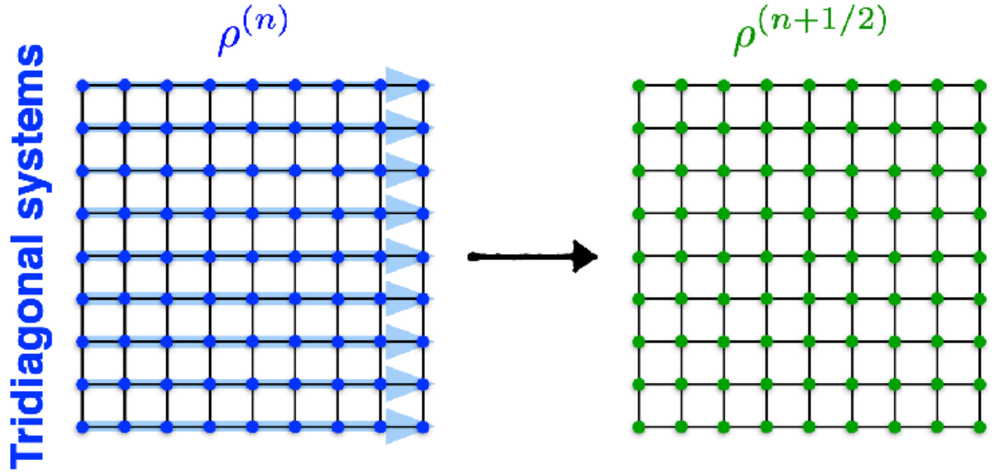
- Implicit Euler: $\frac{c_{i,j}^{(n+1)} - c_{i,j}^{(n)}}{\Delta t} = D\left[\frac{c_{i-1,j}^{(n+1)} - 2c_{i,j}^{(n+1)} + c_{i+1,j}^{(n+1)}}{\Delta x^2} + \frac{c_{i,j-1}^{(n+1)} - 2c_{i,j}^{(n+1)} + c_{i,j+1}^{(n+1)}}{\Delta y^2}\right]$
  - Always stable
  - Drawback: must solve a system of linear equations with sparse matrix

- Alternating Direction Implicit (ADI):
  - Split one iteration in two steps to separate backward and forward Euler in x and y direction
  - Stable method, 2nd order of accuracy in time
  - Easy to solve: Solve multiple independent 1D systems with tridiagonal matrices

Alternating Direction Implicit

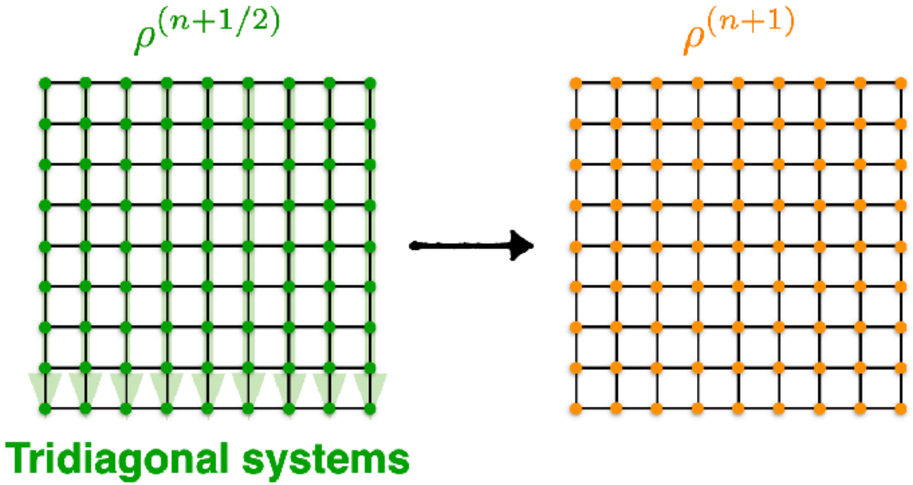
Step 1:

$$\frac{c_{i,j}^{(n+1/2)} - c_{i,j}^{(n)}}{\Delta t} = D\left[\frac{c_{i-1,j}^{(n+1/2)} - 2c_{i,j}^{(n+1/2)} + c_{i+1,j}^{(n+1/2)}}{\Delta x^2} + \frac{c_{i,j-1}^{(n)} - 2c_{i,j}^{(n)} + c_{i,j+1}^{(n)}}{\Delta y^2}\right]$$



Step 2:

$$\frac{c_{i,j}^{(n+1)} - c_{i,j}^{(n+1/2)}}{\Delta t} = D\left[\frac{c_{i-1,j}^{(n+1/2)} - 2c_{i,j}^{(n+1/2)} + c_{i+1,j}^{(n+1/2)}}{\Delta x^2} + \frac{c_{i,j-1}^{(n+1)} - 2c_{i,j}^{(n+1)} + c_{i,j+1}^{(n+1)}}{\Delta y^2}\right]$$



..solve tridiagonal systems with Thomas algorithm $\mathcal{O}(n)$

Question 1: Diffusion

Question 1: Diffusion (25 points)

The diffusion of a substance can be described by the equation

$$\frac{\partial c(x,y,t)}{\partial t} = D\left(\frac{\partial^2 c(x,y,t)}{\partial x^2} + \frac{\partial^2 c(x,y,t)}{\partial y^2}\right), \tag{1}$$

where c is the concentration of the substance at position $(x,\ y)$ and at time t, and D is the diffusion constant. The diffusion process happens in the domain $|x| < L/2$ and $|y| < L/2$. The concentration is zero on the boundaries of the domain for $t \geq 0$. The initial concentration is

$$c(x,y,0) = \begin{cases} 1, & \text{if } |x| < L/4 \text{ and } |y| < L/4, \\ 0, & \text{otherwise.} \end{cases}$$

a) Write down the 2-dimensional discretized diffusion process for eq. (1). Assume a uniform grid with spacing h and a central finite difference scheme in space and forward Euler time integration. For the forward Euler integration assume time intervals of size dt. Make sure that you annotate all variables.

b) Based on the discretization found in the previous subquestion, find the maximal timestep dt using the von Neumann stability analysis. Replace the hardcoded timestep ($t = 0.0001$) in the code with your solution.

c) Based on the discretization found in the first subquestion, provide a cache-friendly implementation of the diffusion equation in the method `advance`. I.e. avoid copying of memory if possible and mind the access patterns of the memory. Blocking must not be implemented.

explain your notation

$$\frac{\partial c(x,y,t)}{\partial t} \simeq \frac{c_{i,j}^{t+1} - c_{i,j}^{t}}{dt}$$

...

Implementation hints:

- don't mess up the BC, check initialization of the grid
- mind the order of the for-loops
- avoid copying `c_tmp` into `c`, find elegant solution

```
void advance() {
  /* Central differences in space, forward Euler in time, Dirichlet BCs */
  // TODO: 1c Implement central difference in space, forward Euler in time
  // TODO: 1e Parallelize diffusion with OpenMP
}
```

Question 1: Diffusion (cont.)

d) Plot the total concentration as a function of time for $t \in [0, 0.5]$ using $D = 1$, $L = 2$ and $N = 100$. The concentration can be read from the file `diagnostics.dat` (column 0 and 1). Qualitatively explain the behaviour of the graph in less than 3 sentences, is this result expected?

output after running code

plot total concentration c_{tot} with your favourite tool

$$c_{tot}(t) = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{i,j}^{t}$$

Plotting hints:
- full graph visible
- x & y label
- value range clear on x & y axis

e) Parallelize the diffusion process (your implementation from subquestion 1c) in the method `advance` using OpenMP.

f) Parallelize the integration of the concentration (marked with `TODO` in `compute_diagnostics`) and the calculation of the histogram (marked with `TODO` in the method `compute_histogram`) using OpenMP.

Parallelization with OpenMP topic of next week(s)

```
#pragma omp parallel for
#pragma omp parallel for reduction(…)
#pragma omp atomic
```

Question 2: PSE

Question 2: Particle Strength Exchange (20 points)

Consider the diffusion equation eq. (1) from the previous exercise. We want to utilize the particle strength exchange (PSE) method to solve this diffusion problem. Instead of discretizing the field $c(x, y, t)$ on a grid, we will use a collection of N particles. A particle represents a small "volume" of the field and is defined by its position $\mathbf{x}_i$ and field value $\phi_i = \pi_i(t)$. In this exercise, we assume the volume of each particle is equal $V_i = V_{total}/N = L^2/N$. We rewrite eq. (1) as a system of equations on particles:

$$\frac{\partial \phi_i}{\partial t} = \frac{D}{\epsilon^2} \sum_{j=1}^{N} V_j(\phi_j - \phi_i)\eta_\epsilon(x_j - x_i)\,, \tag{2}$$

where $\eta_\epsilon(r)$ is a kernel representing the Laplacian operator, and ϵ a scale constant. In this exercise we consider the following kernel:

$$\eta_\epsilon(\mathbf{r}) = \frac{4}{\epsilon^2 \pi} e^{-\frac{1}{\epsilon^2}|r|^2}. \tag{3}$$

Assume the same initial and boundary conditions as in the previous exercise.

In the following subquestions, you will work with the codes provided in `/ex04/skeleton_code/q2/`. Please have a look at the README file for further information. We recommend compiling and running the code on the Euler cluster.

a) You are given a skeleton code that initializes the particle positions and values. Get familiar with the skeleton code. Use `make run` and `make plot` to run the code and the the scripts.

b) Extend the provided skeleton code to compute the right-hand side of eq. (2) using eq. (3) for the kernel η_ϵ. Reuse pair-wise quantities and reduce the number of operations. Implement the forward Euler scheme for the integration of the eq. (2).

have a look at README.md, Makefile
get the right modules

I. compute RHS, reuse symmetry

dphi[i] $\leftarrow RHS_i$ $\rightarrow \frac{N(N-1)}{2}$ ops

II. update phi:

phi[i] $\leftarrow$ phi[i] $+ dt \times$ dphi[i]

```
/*
 * Perform a single timestep. Compute RHS of Eq. (2) and update values of phi_i.
 */
// TODO  2d: Parallelize this function with OpenMP.
void timestep(void) {

  // TODO 2b: Implement the RHS.
  //
  // store your solution in dphi

  // TODO 2b: Implement the forward Euler update of phi_i.
  //
  // update phi
}
```

Question 2: PSE (cont. I)

c) Considering the domain size and the number of particles, what is qualitatively the distance h between neighboring particles? Parameter ϵ determines the spread of the kernel. Run the code for different values of ϵ. Experiment with

- $\epsilon << h$,
- $\epsilon \approx h$,
- and $\epsilon >> h$.

What do you observe for different cases? You can visualize the output of you runs with `make plot`.

Play with ϵ, compile, visualize:

```
make
make run
make plot
```

d) Parallelize the `timestep` method using OpenMP. For this, you need to modify the `Makefile` and include the library in your source code.

Parallelization with OpenMP topic of next week(s)

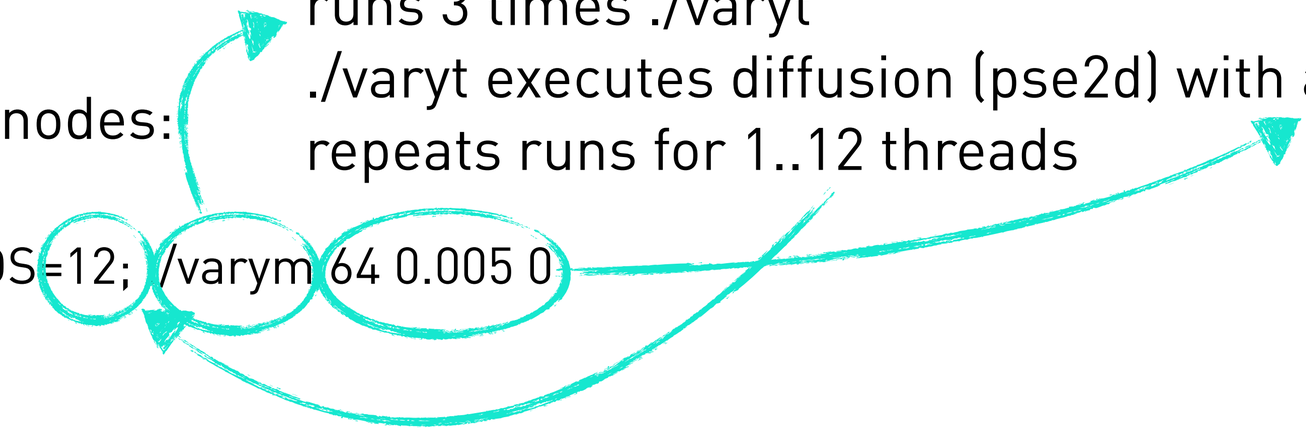
```
#pragma omp parallel for
#pragma omp critical
#pragma omp atomic
```

Question 2: PSE (cont. II)

e) Measure and plot the runtimes of your programs (diffusion and PSE) using 1...12 threads. For benchmarking on the Euler cluster, you can run `make stat` in an interactive shell or launch a job with `make statjob` (or `make statjobfull`). For plotting the runtimes run `make plotstat`. Answer the following: Which code (diffusion or PSE) scales better and how do you measure that? Do you consider strong or weak-scaling?

Benchmark code of Q1 & Q2:

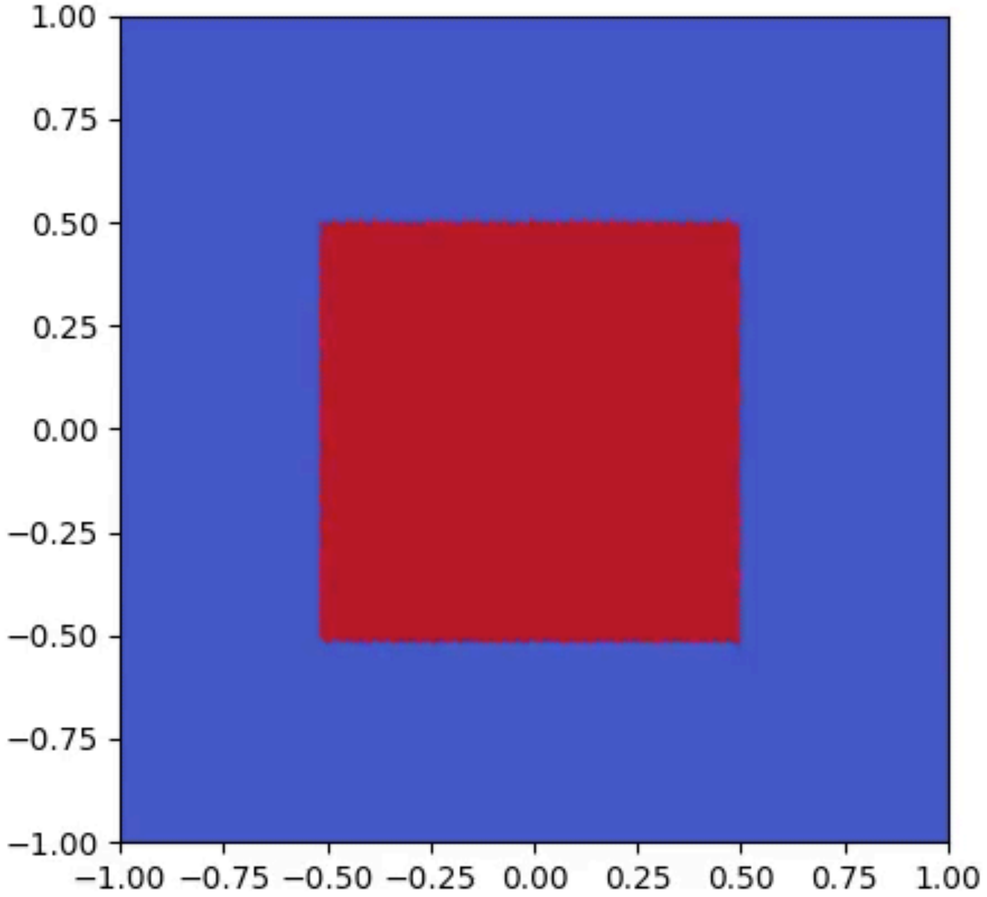
- interactive shell for 30 mins and 12 nodes:
  bsub -n 12 -W 00:30 --Is bash
  make stat // export OMP_NUM_THREADS=12; ./varym 64 0.005 0



- or launch a job with 12 nodes
  make statjob // export OMP_NUM_THREADS=12; bsub -n 12 -W 00:30 ./varym 64 0.005 0
- or launch a job with 12 nodes and allocate fullnode
  make statjobfull // export OMP_NUM_THREADS=12; bsub -n 24 -R fullnode -W 00:30 ./varym 64 0.005 0
- Plot timings with
  make plotstat // ./plotstat

Question 2: PSE (cont. III)

Expected Simulation output



Euler cluster: Login & Setup

1. Connect
   - `ssh <usename>@euler.ethz.ch`
2. Develop
   - Get the exercise with GIT:
     - `git clone https://gitlab.ethz.ch/hpcseI_hs21/lecture .`
   - Use a text editor to write your code, e.g. vi(m), emacs, nano
   - **OR mount your folder locally and edit files locally** (changes are copied to Euler)
     - `sshfs <username>@euler.ethz.ch:/cluster/home/<username>/ <local_dir>`
     - https://www.digitalocean.com/community/tutorials/how-to-use-sshfs-to-mount-remote-file-systems-over-ssh
3. Compile
   - You need the appropriate programming tools and libraries to compile your code
   - **See README.md**
     - `module load gcc/6.4.0` (compiler, get newer versions with `module load new`)
   - Use Makefiles for compilation and running!

Euler cluster: Usage

4. Run your code

- login nodes are only for development
- **The program must run on a compute node**
- To do that, you must use the bsub command
  `bsub -W 00:30 -n 12  ./diffusion 1.0 2.0 100 0.5 1`
- You can also get an *interactive shell* on a compute node
  - `bsub -W 00:30 -n 12 -Is bash`
  - `./diffusion 1.0 2.0 100 0.5 1` (after receiving int. shell)
- **`See README.md & Makefiles`**