P. Koumoutsakos
ETH Zentrum, CLT E 13
CH-8092 Zürich

Fall semester 2018

Set 11 - 2D Diffusion, ADI, Thomas algorithm

Issued: December 07, 2018
Hand in (optional): **December 17**, 2018 23:59

Question 1: Diffusion in 2D using ADI scheme

40 points total

Heat flow in a medium can be described by the diffusion equation of the form

$$\frac{\partial \rho(x,y,t)}{\partial t} = D\nabla^2 \rho(x,y,t) \tag{1}$$

where $\rho(x,y,t)$ is a measure for the amount of heat at position $\boldsymbol{r}$ and time t and the diffusion coefficient D is constant. Lets define the domain Ω in two dimensions as $x, y \in [-1,1]$. We will use the boundary condition:

$$\rho(x,y,t) = 0 \quad \forall\, t \geq 0 \,\text{and}\, (x,y) \notin \Omega \tag{2}$$

and an initial distribution:

$$\rho(x,y,0) = \begin{cases} 1 & |x,y| < 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

a) Discretize equation (1) using the Alternating Direction Implicit (ADI) scheme. Write down the resulting system in matrix form. What do you observe? Comment on your choice of method/algorithm for the solution of the resulting implicit scheme and explain why this choice is justified.

Solving parabolic PDEs with implicit schemes requires the solution of a large linear system, resulting from the numerical discretization of the PDEs. The solution of a linear system is in general costly compared to explicit methods. With the ADI scheme the problem reduces to a sequence of 1D problems, one in each dimension, greatly reducing the complexity of the problem. We consider the diffusion equation in 2D, with explicit discretization in time and central differences for the spatial derivatives.

Step 1:

$$\frac{\rho_{i,j}^{(n+1/2)} - \rho_{i,j}^{(n)}}{\Delta t/2} = D\left[\frac{\rho_{i-1,j}^{(n+1/2)} - 2\rho_{i,j}^{(n+1/2)} + \rho_{i+1,j}^{(n+1/2)}}{\Delta x^2} + \frac{\rho_{i,j-1}^{(n)} - 2\rho_{i,j}^{(n)} + \rho_{i,j+1}^{(n)}}{\Delta y^2}\right] \tag{4}$$

Step 2:

$$\frac{\rho_{i,j}^{(n+1)} - \rho_{i,j}^{(n+1/2)}}{\Delta t/2} = D\left[\frac{\rho_{i-1,j}^{(n+1/2)} - 2\rho_{i,j}^{(n+1/2)} + \rho_{i+1,j}^{(n+1/2)}}{\Delta x^2} + \frac{\rho_{i,j-1}^{(n+1)} - 2\rho_{i,j}^{(n+1)} + \rho_{i,j+1}^{(n+1)}}{\Delta y^2}\right] \quad (5)$$

We assume that $\Delta x = \Delta y = \Delta s$ and define a constant $R = \Delta t D/(2\Delta s^2)$. We rearrange the equations such that all terms of ρ at time $t + \Delta t/2$ in step 1 and at time $t + \Delta t$ in step 1 are on the left hand side.

Step 1:

$$-R\rho_{i-1,j}^{(n+1/2)} + (1+2R)\rho_{i,j}^{(n+1/2)} - R\rho_{i+1,j}^{(n+1/2)} = R\rho_{i,j-1}^{(n)} + (1-2R)\rho_{i,j}^{(n)} + R\rho_{i,j+1}^{(n)} \quad (6)$$

Step 2:

$$-R\rho_{i,j-1}^{(n+1)} + (1+2R)\rho_{i,j}^{(n+1)} - R\rho_{i,j+1}^{(n+1)} = R\rho_{i-1,j}^{(n+1/2)} + (1-2R)\rho_{i,j}^{(n+1/2)} + R\rho_{i+1,j}^{(n+1/2)} \quad (7)$$

and denote the right hand side of equation 6 as $\mathrm{RHS}_{(i,j)}$. The boundary conditions are $\rho(x, y, t) = 0 \quad \forall\, t \geq 0 \,\text{and}\, (x, y) \notin \Omega$. Therefore in matrix form, the full system (including the boundary/ghost cells) of step 1 can be written as follows:

$$\begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ -R & (1+2R) & -R & 0 & \dots \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & -R & (1+2R) & -R \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \rho_{0,j}^{(n+1/2)} \\ \rho_{1,j}^{(n+1/2)} \\ \vdots \\ \rho_{N,j}^{(n+1/2)} \\ \rho_{N+1,j}^{(n+1/2)} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathrm{RHS}_{(1,j)} \\ \vdots \\ \mathrm{RHS}_{(N,j)} \\ 0 \end{bmatrix}$$

This is a tridiagonal system. A similar system is generated for step 2. Tridiagonal systems are solved efficiently with Thomas algorithm. This reduces the complexity for the solution of the linear system from $\mathcal{O}(N^3)$ required for Gaussian elimination, to $\mathcal{O}(N)$.

Point distribution:

- **3 points for correct coefficients for step 1.**
- **3 points for correct coefficients for step 2.**
- **3 points for including correct boundary conditions in matrix.**
- **3 points for usage of Thomas algorithm to solve the resulting tridiagonal systems.**

b) Use the provided skeleton code (`diffusion2d_adi_openmp.cpp`) to solve the 2D diffusion problem on a uniform grid. Implement the missing code parts in all sections marked by `TODO`. Use Thomas algorithm for the solution of the implicit systems resulting from the ADI scheme.

The solution code can be found in `solution_code/diffusion2d_adi_openmp.cpp`. The routine for Thomas algorithm is placed in a separate function to enhance readability. An example of the solution over time is presented in Figure 1. Each plot represents the solution at different time, as denoted on the top of the plots. It is shown that the maximum heat reduces over time as heat diffuses from the center towards the edges of the domain. This

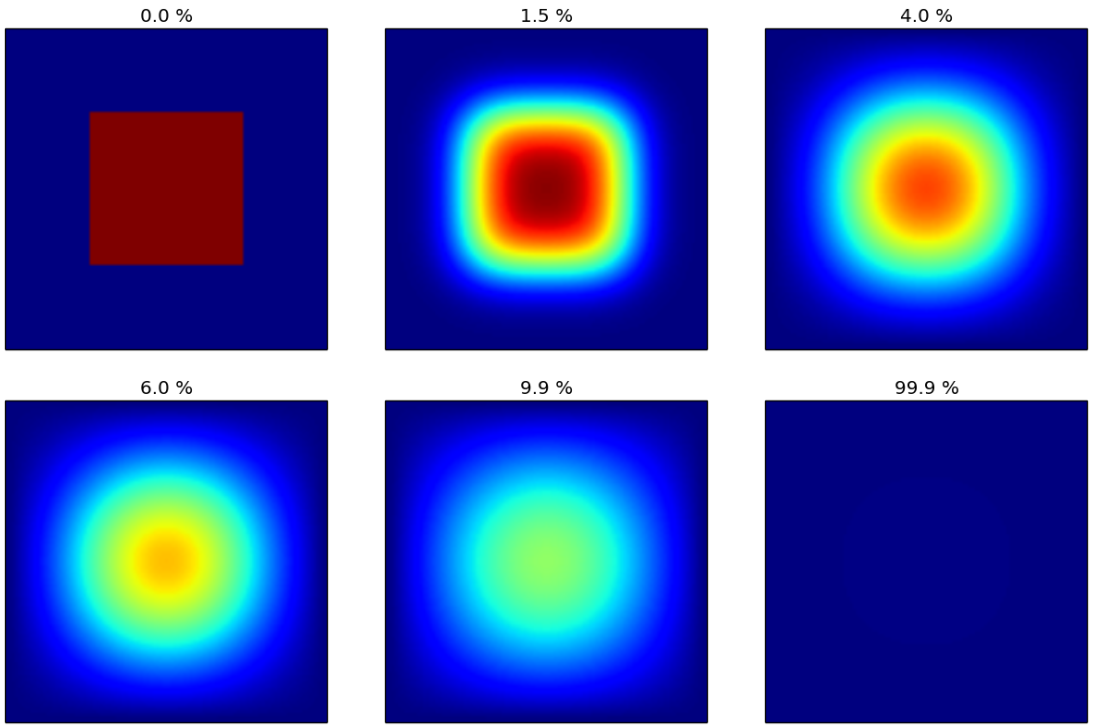

Figure 1: Demonstration of solution over time. The number in percentage shown on top of each subplot denotes the fraction of time required for the total heat in the domain to reach 1% of the initial value.

is caused by the boundary condition which absorbs the heat that reaches the boundary and thus removes it from the system.

Point distribution:

- **2 points for correct implementation of initial condition.**
- **5 points for correct implementation of step 1.**
- **5 points for correct implementation of step 2.**
- **3 points for correct implementation of Thomas algorithm.**
- **Subtract 1 point for each mistake. The total amount of subtracted points cannot be larger than 15.**

c) Parallelize your code using OpenMP. Comment on any complexity that would arise if you chose to parallelize the ADI scheme with MPI.

The solution code can be found in `solution_code/diffusion2d_adi_openmp.cpp`. In the solution the threads are spawned and rejoined at every time iteration. The parallelization of the main part of the algorithm, the method `advance()`, is done over the rows of the domain for step 1 of ADI, and then over the columns for step 2. A `pragma omp parallel for` is added in front of the corresponding for loops. In the same manner we parallelize the loop that sets the initial conditions. Note that this is needed on Non-Uniform Memory Architecture (NUMA) nodes such as Euler's [1], where the access time to a NUMA node from another is different than the access time within the same NUMA node. Parallelizing the data

[1] The 24 core Euler nodes have 2 NUMA nodes.

initialization as mentioned above, is a simple - but quite effective - way to make the code "NUMA-aware": each thread creates the data it will work on in its own physical memory so that we reduce the need for threads to fetch data that others have. The principle of initializing the data owned by the thread is called "first-touch" policy. Additionally, on Euler, to get good and stable performance using all 24 cores of a node we set the environment variable `OMP_PROC_BIND` so that each thread is mapped onto a single core.

Point distribution:

- **1 point for including OpenMP header.**
- **2 points for correct parallelization of the for loop in method `compute_diagnostics()`.**
- **3 points for correct parallelization of the for loop in method `initialize_rho()`.**
- **4 points for parallelizing method `advance()`: 1 point for each of the four `parallel for` directives.**
- **Subtract 1 point for each mistake. The total amount of subtracted points cannot be larger than 10.**

d) Compute an approximation to the integral of ρ over the entire domain in `compute_diagnostics` and plot the result as a function of time using the parameters $D = 1$, $L = 2$ and $N = 256$.

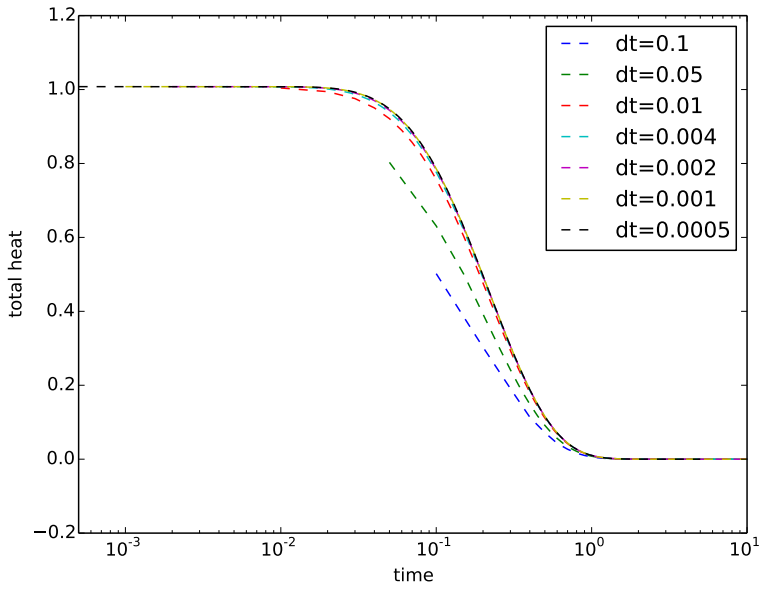

Figure 2: Total heat in the domain as a function of time for different timesteps.

The total heat in the domain reduces over time, as a result of the boundary condition (Fig. 2). Notice that as the timestep increases, the solution does not become unstable (since the ADI method is unconditionally stable), but diverges from the correct solution. The solution converges for $\Delta t \leq 0.004$. If Eq. 1 was discretized with a central finite difference scheme in space and explicit Euler in time, the stability condition on the timestep using the same parameters D and N, would yield $\Delta t \leq \Delta x^2/(4D) = 0.000015$.

Point distribution:

- **2 points for a plot showing the total heat decreasing over time.**
- **1 point for correct axis labels.**