# High Performance Computing for Science and Engineering I.

## Exercise Set 6

Florian Mahlknecht

2019-12-20

# 1 Diffusion with ADI

## 1.1 ADI Scheme Derivation

The partial differential equation to treat is the diffusion equation in 2 dimension:

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

The idea of Peaceman-Rachford is to combine the unconditional stability of implicit forward schemes with the computational convenience of the Thomas algorithm from the 1D case in the 2 dimensional case. The key step is to treat the x and y directions independently and insert a half step in between, where implicitly and explicitly calculated directions are swapped for the next half step. In this way the method gets unconditionally stable.

In the first step we use the explicit scheme in the x direction and the implicit (or backward) euler scheme in the y direction. From eq. (1) in this way we obtain:

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

In the second step we swap the directions for the explicit and implicit approach:

$$\frac{\phi_{i,j}^{(n+1)} - \phi_{i,j}^{(n+1/2)}}{\frac{\Delta t}{2}} = D\left(\frac{\phi_{i-1,j}^{(n+1)} - 2\phi_{i,j}^{(n+1)} + \phi_{i+1,j}^{(n+1)}}{\Delta x^2} + \frac{\phi_{i,j-1}^{(n+1/2)} - 2\phi_{i,j}^{(n+1/2)} + \phi_{i,j+1}^{(n+1/2)}}{\Delta y^2}\right) \tag{3}$$

By assuming an equispaced grid, i.e. $\Delta x = \Delta y = \Delta s$ and multiplying both equations by $\frac{\Delta t}{2}$, we can conveniently introduce a new constant $C = \frac{\Delta t\, D}{2\,\Delta s^2}$, yielding:

$$\text{Step 1:} \quad \phi_{i,j}^{(n+1/2)} - \phi_{i,j}^{(n)} = C\left(\phi_{i-1,j}^{(n)} - 2\phi_{i,j}^{(n)} + \phi_{i+1,j}^{(n)} + \phi_{i,j-1}^{(n+1/2)} - 2\phi_{i,j}^{(n+1/2)} + \phi_{i,j+1}^{(n+1/2)}\right)$$

$$\text{Step 2:} \quad \phi_{i,j}^{(n+1)} - \phi_{i,j}^{(n+1/2)} = C\left(\phi_{i-1,j}^{(n+1)} - 2\phi_{i,j}^{(n+1)} + \phi_{i+1,j}^{(n+1)} + \phi_{i,j-1}^{(n+1/2)} - 2\phi_{i,j}^{(n+1/2)} + \phi_{i,j+1}^{(n+1/2)}\right)$$

By rearranging the terms such that the "newer" terms are all on the left hand side, i.e. in step one the $(n + 1/2)$ and the $n + 1$ terms in step two respectively, we get:


fmahlknecht@student.ethz.ch
19-945-351

$$\text{Step 1:} \quad -C\,\phi_{i,j-1}^{(n+1/2)} + (1+2C)\,\phi_{i,j}^{(n+1/2)} - C\,\phi_{i,j+1}^{(n+1/2)} = C\,\phi_{i-1,j}^{(n)} + (1-2C)\,\phi_{i,j}^{(n)} + C\,\phi_{i+1,j}^{(n)}$$
$$\text{Step 2:} \quad -C\,\phi_{i-1,j}^{(n+1)} + (1+2C)\,\phi_{i,j}^{(n+1)} - C\,\phi_{i+1,j}^{(n+1)} = C\,\phi_{i,j-1}^{(n+1/2)} + (1-2C)\,\phi_{i,j}^{(n+1/2)} + C\,\phi_{i,j+1}^{(n+1/2)}$$

For the right hand sides we introduce:

$$b_{i,j}^{(n)} = C\,\phi_{i-1,j}^{(n)} + (1-2C)\,\phi_{i,j}^{(n)} + C\,\phi_{i+1,j}^{(n)}$$
$$b_{i,j}^{(n+1/2)} = C\,\phi_{i,j-1}^{(n+1/2)} + (1-2C)\,\phi_{i,j}^{(n+1/2)} + C\,\phi_{i,j+1}^{(n+1/2)}$$

By including boundary cells 0 and $M+1$ respectively as well as their boundary condition we get:

$$\begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ -C & (1+2C) & -C & 0 & \dots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \dots & 0 & -C & (1+2C) & -C \\ 0 & \dots & \dots & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_{i,0}^{(n+1/2)} \\ \phi_{i,1}^{(n+1/2)} \\ \vdots \\ \phi_{i,M}^{(n+1/2)} \\ \phi_{i,M+1}^{(n+1/2)} \end{bmatrix} = \begin{bmatrix} 0 \\ b_{i,1}^{(n)} \\ \vdots \\ b_{i,M}^{(n)} \\ 0 \end{bmatrix} \tag{4}$$

And in a completely analogous fashion for step two:

$$\begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ -C & (1+2C) & -C & 0 & \dots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \dots & 0 & -C & (1+2C) & -C \\ 0 & \dots & \dots & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_{0,j}^{(n+1} \\ \phi_{1,j}^{(n+1} \\ \vdots \\ \phi_{N,j}^{(n+1} \\ \phi_{N+1,j}^{(n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ b_{1,j}^{(n)} \\ \vdots \\ b_{N,j}^{(n)} \\ 0 \end{bmatrix} \tag{5}$$

Those two tri-diagonal systems can be efficiently solved with the Thomas algorithm in $\Theta(N)$ and $\Theta(M)$ respectively.

## 1.2 Implementation

Once the solver and the midpoint derivative are implemented, the solution of `advance()` becomes straightforward:

```
1 midpointDerivative(Direction::Y);
2 #pragma omp parallel for
3 for (size_t i=1; i<m_realN-1; ++i)
4   thomasSolver(Direction::X, i);
5
6 midpointDerivative(Direction::X);
7 #pragma omp parallel for
8 for (size_t j=1; j<m_realN-1; ++j)
9   thomasSolver(Direction::Y, j);
```

Listing 1: Serial particles iterator implementation

All details are in the files attached.


Florian Mahlknecht
fmahlknecht@student.ethz.ch
19-945-351

### 1.3 Parallelization

The openMP implementation can again be found in the files attached. Note that by using already parallelized access during the initialization, the implementation should perform well on NUMA architectures.

Regarding a possible MPI implementation, care needs to be taken on how to divide the grid. The boundary conditions could be exchanged in a row and column wise fashion, while leaving the computation involving them to the very end (allowing asynchronous communication, i.e. hiding the communication 'behind' the calculations). A possible issue with that strategy is the Thomas Algorithm involved, given that it substitutes back in a recursive fashion starting from the boundary condition.

### 1.4 Total heat comparison

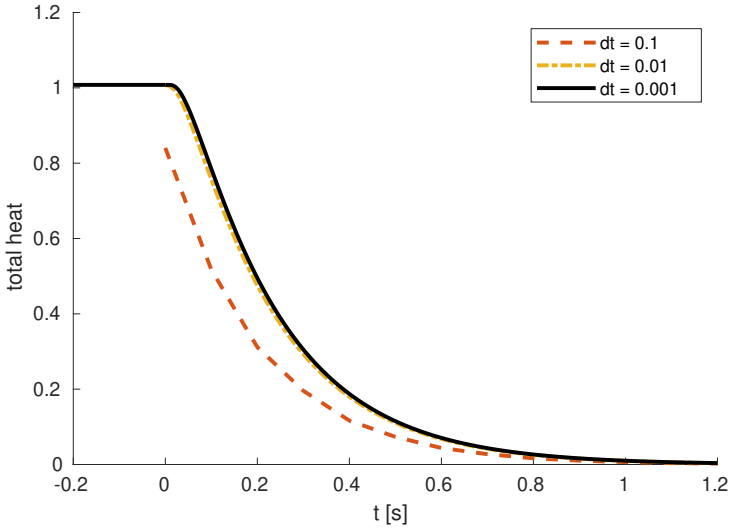


Figure 1: Total heat comparison

Figure 1 compares different $\Delta t$s. No conditional stability is observed anymore, i.e. the solution does not blow up for large values of $\Delta t$. However, to converge to the right solution, a smaller value is needed, $\Delta t = 0.001$ works out well as the plot shows.

## 2 Diffusion with PSE

### 2.1 Derivation

A derivation is provided in our lecture notes, see fig. 2.


fmahlknecht@student.ethz.ch
19-945-351

### 3 Particle Strength Exchange

Particle strength exchange (PSE) is a method that aims at high order approximations for the diffusion operator in particle systems. Assume $q = q(\mathbf{x}, t)$ is the density of some quantity $q$ (e.g. concentration density) with spatial coordinates $\mathbf{x}$ and $t$ time. We express the integral of the density $q$ over the domain $\Omega$ as

$$Q(t) = \int_{\Omega} q(\mathbf{x}, t)\, \mathrm{d}V. \tag{22}$$

Assume we discretize the domain $\Omega$ using $N$ particles. Equation (22) is then rewritten in the form

$$Q(t) = \sum_{p=1}^{N} \int_{\Omega_p} q(\mathbf{x}, t)\, \mathrm{d}V = \sum_{p=1}^{N} Q_p(t), \tag{23}$$

where $\Omega_p \subseteq \Omega$ is the domain occupied by particle $p$ and $Q_p(t)$ the "strength" of the quantity of interest carried by particle $p$. Note that $V_p = \int_{\Omega_p} \mathrm{d}V$ is the volume of particle $p$.

Our goal is to solve the diffusion equation

$$\frac{\partial f}{\partial t} = D \frac{\partial^2 f}{\partial x^2} \tag{24}$$

in a particle system for a function $f(x, t)$ and diffusion coefficient $D$. For simplicity, we assume a one dimensional system and set $d = 1$. We are interested in the integral quantity $Q(t)$ for which we define a function $f_p(t)$ as follows

$$f_p(t) = \frac{\int_{\Omega_p} q(x, t)\, \mathrm{d}x}{\int_{\Omega_p} \mathrm{d}x} = \frac{Q_p(t)}{V_p}, \tag{25}$$

which is defined at each particle location $x_p$. In Equation (21) we have found a second order accurate approximation for the second derivative, which we now use together with the function $f_p(t)$ to reduce the partial differential equation of Equation (24) into a system of ordinary differential equations for $N$ particles, that is

$$\frac{\mathrm{d}Q_i}{\mathrm{d}t} = \frac{D}{\varepsilon^2} \sum_{p=1}^{N} [Q_p V_i - Q_i V_p] \eta_\varepsilon (x_i - x_p). \tag{26}$$

Integration of the system can be carried out using a suitable numerical scheme. The scheme in Equation (26) is called particle strength exchange. Note that the PSE scheme is *conservative* for a symmetric kernel $\eta_\varepsilon$.

Figure 2: Recalls to the derivation treated in class

From equation 26, it can be seen that if we plug in $\phi$ for $Q$ and use the fact that in our exercise the volume is assumed to the same for each particle, i.e. $V_i = V_j$, we can simplify the expression and obtain:

$$\frac{d\phi_i}{dt} = \frac{D}{\varepsilon^2} \sum_j V_j \left(\phi_j - \phi_i\right) \eta_\varepsilon \left(\mathbf{x}_i - \mathbf{x}_j\right) \tag{6}$$

□ *q.e.d.*

## 2.2 Implementation

The implementation is pretty straight forward, the main part is shown in the following listing.


Florian Mahlknecht
fmahlknecht@student.ethz.ch
19-945-351

```
for (int i = 0; i < N; ++i) {
  double sum = 0.0;
  for (int j = 0; j < N; ++j) {
    auto dx = particle_dist(x[i], x[j]);
    auto dy = particle_dist(y[i], y[j]);
    sum += (phi[j] - phi[i]) * eta_epsilon(dx, dy);
  }
  dphi[i] = nu * volume / SQUARE(eps) * sum;
}

// TODO 1c: Implement the forward Euler update of phi_i, as defined in Eq. (2).
for (int i = 0; i < N; ++i)
  phi[i] += dt * dphi[i];
```

Listing 2: Diffusion with particle methods

Details are provided in the attached source code files. Figure 3 shows the output of the generated diffusion video.

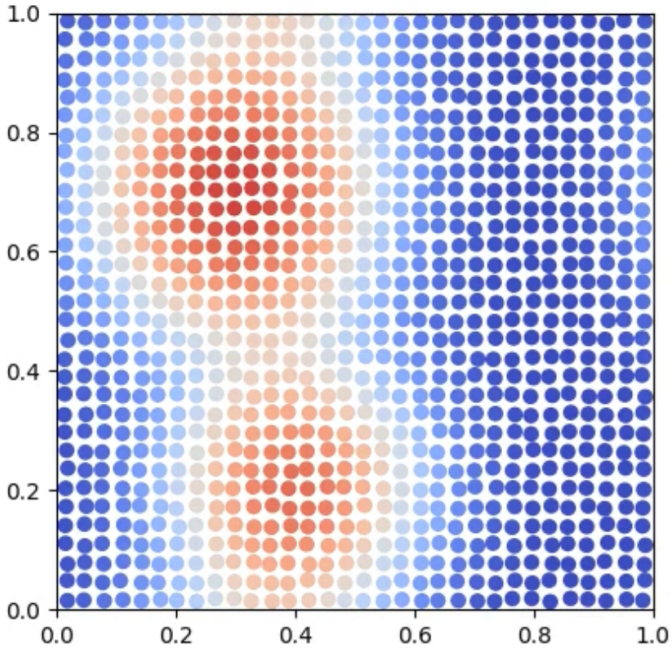


Figure 3: Diffusion with ADI video ouptut


fmahlknecht@student.ethz.ch
19-945-351