Chapter 8

Diffusion Equation

This chapter describes different methods to discretize the diffusion equation

\[
\frac{\partial f}{\partial t} - \alpha \left( \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \right) = 0
\]

which represents a combined boundary and initial value problem, i.e., requires to prescribe boundary \( f_{\text{boundary}}(t) \) and initial conditions \( f(x,y,z,t=0) \).

8.1 Explicit methods

8.1.1 Forward time centered space scheme

In one dimension and using finite difference the FTCS scheme is

\[
f_j^{n+1} = f_j^n + s \Delta x^2 L_{xx} f_j^n
\]

\[s = \frac{\alpha \Delta t}{\Delta x^2}\]

The truncation error is given by

\[
E_j^n = \alpha \frac{\Delta x^2}{2} \left( s - \frac{1}{6} \right) \frac{\partial^4 f}{\partial x^4} \bigg|_j + O(\Delta x^4)
\]

The amplification factor is

\[
g = 1 - 4s \sin^2 \left( \frac{k \Delta x}{2} \right)
\]

which yields \( s < 1/2 \) for stability.
Figure 8.1: Representation of the FTCS scheme.

In two dimensions the same approach can be used to yield

\[
f_{jk}^{n+1} = f_{jk}^n + s_x \Delta x^2 L_{xx} f_{jk}^n + s_y \Delta y^2 L_{yy} f_{jk}^n
\]

with \( s_x = \alpha_x \Delta t / \Delta x^2, s_y = \alpha_y \Delta t / \Delta y^2 \)

Stability requires \( s_x + s_y \leq 1/2 \). An improved nine point scheme can be found if \( \alpha_x = \alpha_y = \alpha \) and \( \Delta x = \Delta y \) which for

\[
\frac{\Delta f_{jk}^{n+1}}{\Delta t} = \alpha L_{xx} f_{jk}^n + \alpha L_{yy} f_{jk}^n + \alpha^2 \Delta t^2 L_{xx} L_{yy} f_{jk}^n
\]

yields stability for \( s \leq 1/2 \). This scheme can efficiently be implemented with

\[
f_{jk}^* = (1 + \alpha \Delta t L_{yy}) f_{jk}^n
\]

\[
f_{jk}^{n+1} = (1 + \alpha \Delta t L_{xx}) f_{jk}^*
\]

### 8.1.2 Richardson and DuFort-Frankel schemes

One could have the idea that is is more accurate to employ a centered difference for the temporal derivative which give the Richardson scheme

\[
\frac{f_{j}^{n+1} - f_{j}^{n-1}}{2\Delta t} = \alpha L_{xx} f_{j}^n = \frac{\alpha}{\Delta x^2} \left( f_{j-1}^n - 2f_{j}^n + f_{j+1}^n \right)
\]

and is second order accurate for the time derivative. However the stability analysis shows that this scheme is unconditionally unstable. A small modification of this scheme where the term \( 2f_{j}^n \) is
split into two time levels according to \(2f_j^n = f_j^{n-1} + f_j^{n+1}\) leads to the so-called Dufort-Frankel scheme:

\[
\frac{f_j^{n+1} - f_j^{n-1}}{2\Delta t} = \alpha L_{xx} f_j^n = \frac{\alpha}{\Delta x^2} \left(f_j^{n-1} - f_j^{n+1} + f_j^n\right)
\]

Although it appears as if this were implicit it is straightforward to re-arrange terms to yield

\[
f_j^{n+1} = \frac{2s}{1+2s} \left(f_j^{n-1} + f_j^{n+1}\right) + \frac{1-2s}{1+2s} f_j^{n-1}
\]

This scheme is actually unconditionally stable. However, carrying out the consistency test, i.e., expanding terms in the corresponding Taylor series yields

\[
\left[ \frac{\partial f}{\partial t} - \alpha \frac{\partial^2 f}{\partial x^2} + \alpha \left(\frac{\Delta t}{\Delta x}\right)^2 \frac{\partial^2 f}{\partial t^2} \right]_j^n + O(\Delta t^2, \Delta x^2) = 0
\]

Thus it is not sufficient to conduct the limit of \(\Delta t, \Delta x \to 0\) to achieve consistency as long as \(\Delta t / \Delta x\) remains finite. Using the relation

\[
\alpha \left(\frac{\Delta t}{\Delta x}\right)^2 = s \Delta t
\]

demonstrates that consistency can be achieved if \(s\) is kept constant and \(\Delta t \to 0\) because \(\Delta t \sim \Delta x^2\) for constant \(s\).

The amplification factor for the Dufort-Frankel scheme is

\[
g = \frac{2s \cos k \Delta}{1+2s} \pm \frac{1}{1+2s} \sqrt{1 - 4s^2 + 4s^2 \cos k \Delta}
\]

### 8.1.3 Three-level scheme

The general method suggest an approach such as

\[
a f_j^{n+1} + b f_j^n + c f_j^{n-1} - \left(dL_{xx} f_j^n + eL_{xx} f_j^{n-1}\right) = 0
\]
Using this method and applying consistency yields the equation

\[
\frac{1}{\Delta t} (1 + \gamma) \left( f_j^{n+1} - f_j^n \right) - \frac{1}{\Delta t} \gamma \left( f_j^n - f_j^{n-1} \right) = \alpha \left[ (1 - \beta) L_{xx} f_j^n + \beta L_{xx} f_j^{n-1} \right]
\]

The error resulting from this scheme is

\[
E_j^n = \alpha s \Delta x^2 \left( \frac{1}{2} + \gamma + \beta - \frac{1}{12s} \right) \frac{\partial^4 f}{\partial x^4}\bigg|_j + O(\Delta x^4)
\]

such that this scheme becomes 4th order accurate for \( \beta = -0.5 - \gamma + 1/12s \). The equation for the amplification factor is

\[(1 + \gamma) g^2 - [1 + 2\gamma + 2s(1 - \beta)(\cos k\Delta - 1)] + [\gamma - 2s(\cos k\Delta - 1)] = 0\]

The discussion of the parameter space is somewhat complicated but the general result is that there is a stability limit with values of \( s \) increasing from about 0.35 to 5 if \( \gamma \) is raised from 0 to about 6.

Figure 8.2: Illustration of the stability space for the three level scheme.

The truncation error for the previous methods for different grid resolution is shown in the following table. It illustrates that for specific values of \( s \) the simple FTCS and DuFort-Frankel methods can achieve 4th order accuracy.

### 8.1.4 Hopscotch method

This is a fairly original method which uses a two stage FTCS algorithm. Here we describe the two-dimensional variant.

In the **first stage** which is carried out for \( j + k + n = \) even the usual FTCS scheme is applied
Table 8.1: RMS errors for different explicit schemes and varying parameters as a function of grid resolution.

<table>
<thead>
<tr>
<th>Case</th>
<th>$s$</th>
<th>$\gamma$</th>
<th>RMS $\Delta x = 0.2$</th>
<th>RMS $\Delta x = 0.1$</th>
<th>RMS $\Delta x = 0.05$</th>
<th>approx. conv. rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTCS</td>
<td>1/6</td>
<td>0.007266</td>
<td>0.00049</td>
<td>0.000033</td>
<td>3.9</td>
<td></td>
</tr>
<tr>
<td>FTCS</td>
<td>0.3</td>
<td>0.6437</td>
<td>0.1634</td>
<td>0.0413</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>FTCS</td>
<td>0.41</td>
<td>1.2440</td>
<td>0.3023</td>
<td>0.0755</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>DuF-F</td>
<td>0.289</td>
<td>0.0498</td>
<td>0.00233</td>
<td>0.00012</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td>DuF-F</td>
<td>0.3</td>
<td>0.0244</td>
<td>0.0136</td>
<td>0.00395</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>DuF-F</td>
<td>0.41</td>
<td>0.8481</td>
<td>0.2085</td>
<td>0.0525</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>3L-4th</td>
<td>0.3</td>
<td>0.0711</td>
<td>0.00416</td>
<td>0.00022</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>3L-4th</td>
<td>0.3</td>
<td>0.1372</td>
<td>0.00665</td>
<td>0.00029</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>3L-4th</td>
<td>0.3</td>
<td>0.2332</td>
<td>0.00916</td>
<td>0.00054</td>
<td>4.1</td>
<td></td>
</tr>
<tr>
<td>3L-4th</td>
<td>0.41</td>
<td>0.7347</td>
<td>0.0229</td>
<td>0.00140</td>
<td>4.0</td>
<td></td>
</tr>
</tbody>
</table>

\[
\frac{\Delta f_{jk}^{n+1}}{\Delta t} = \alpha L_{xx} f_{jk}^n + \alpha L_{yy} f_{jk}^n, \quad j+k+n = \text{even}
\]

In the second stage the same scheme is applied but (a) now on all grid points with $j+k+n = \text{odd}$ and (b) the second order derivative is using the newly computed time level $n + 1$. This is possible now in an explicit method because all grid points adjacent to the one to be updated have been updated in the first stage.

\[
\frac{\Delta f_{jk}^{n+1}}{\Delta t} = \alpha L_{xx} f_{jk}^{n+1} + \alpha L_{yy} f_{jk}^{n+1}, \quad j+k+n = \text{odd}
\]

The Figure above shows a schematic of the time update. The grid topology is shown in the following figure. Here the first stage updates for instance all grid points indicated in blue. The 2nd stage then uses the spatial derivative from those grid points to update the orange points. The resulting pattern looks like a chess board.

Note that this method is not only very efficient and simple but also unconditionally stable (in 2 dimensions?). The error associated with the scheme is $O(\Delta t, \Delta x^2, \Delta y^2)$.

### 8.2 Implicit methods

#### 8.2.1 Fully implicit scheme

This method is equivalent to the FTCS method, however, with the 2nd derivative operator evaluated at the new time level.
CHAPTER 8. DIFFUSION EQUATION

Figure 8.3: Representation of the grid topology used for the Hopscotch method.

\[
\frac{\Delta f_j^{n+1}}{\Delta t} = \alpha L_{xx} f_j^{n+1}
\]

Consistency yields

\[
E_j^n = -\frac{\Delta t}{2} \left( 1 + \frac{1}{6s} \right) \frac{\partial^4 f^n_j}{\partial x^4} + O(\Delta x^4)
\]

and the amplification factor is

\[
g = \left( 1 + 4s \sin^2 \left( \frac{k\Delta x}{2} \right) \right)^{-1}
\]

which demonstrates that the scheme is unconditionally stable. The solution to this method can be found by solving

\[
\begin{pmatrix}
1 + 2s & -s & 0 & 0 & 0 \\
-s & 1 + 2s & -s & 0 & 0 \\
0 & -s & 1 + 2s & -s & 0 \\
0 & 0 & -s & 1 + 2s & -s \\
0 & 0 & 0 & -s & 1 + 2s \\
\end{pmatrix}
\begin{pmatrix}
f_1^{n+1} \\
f_2^{n+1} \\
f_3^{n+1} \\
f_4^{n+1} \\
f_5^{n+1} \\
\end{pmatrix}
= \begin{pmatrix}
d_1 \\
d_2 \\
d_3 \\
d_4 \\
d_5 \\
\end{pmatrix}
\]

of rank \(N\) corresponding to the number of grid points. The tridiagonal system is easily solved using the Thomas algorithm.

8.2.2 Crank-Nicholson scheme

This method uses a mixture of spatial derivative using time levels \(n\) and \(n+1\).
\[ \frac{\Delta f_j^{n+1}}{\Delta t} = \frac{\alpha}{2} L_{xx} \left( f_j^n + f_j^{n+1} \right) \]

which generates an error of order \( E_j^n = O(\Delta t^2, \Delta x^2) \). Note that the scaling of \( \Delta t^2 \) in the error is caused by the centered time derivative. The equation for the amplification factor is

\[
g - 1 + 2s \sin^2 \frac{k\Delta}{2} + 2s \sin^2 \frac{k\Delta}{2} = 0
\]

or

\[
g = \frac{1 - 2s \sin^2 \frac{k\Delta}{2}}{1 + 2s \sin^2 \frac{k\Delta}{2}}
\]

which implies unconditional stability.

**Finite element Crank-Nicholson**

Note that this is easily expanded to the finite element Crank-Nicholson scheme by applying the corresponding mass operators to the time derivative term

\[
\frac{1}{\Delta t} M_x \Delta f_j^{n+1} = \alpha \left( \frac{1}{2} L_{xx} f_j^n + \frac{1}{2} L_{xx} f_j^{n+1} \right)
\]

The Crank-Nicholson scheme can also be generalized in substituting the factors of \( 1/2 \) by a variable parameter in the following manner

\[
\frac{\Delta f_j^{n+1}}{\Delta t} = \alpha \left[ (1 - \beta) L_{xx} f_j^n + \beta L_{xx} f_j^{n+1} \right]
\]

In this case the method is unconditionally stable for

\[
0 \leq \beta \leq \frac{1}{2}
\]

and has the restriction

\[
s \leq \frac{1}{2(1 - 2\beta)}
\]

if \( \beta > 0.5 \).
8.2.3 Generalized three level schemes

Another generalization is to consider a weighted time differencing over three time levels:

\[\frac{(1 + \gamma) \Delta f_{j+1}^n}{\Delta t} - \frac{\gamma \Delta f_j^n}{\Delta t} = \alpha \left[ (1 - \beta) L_{xx} f_j^n + \beta L_{xx} f_j^{n+1} \right]\]

where the particular choice

\[\gamma = \frac{1}{2} \quad \beta = 1\]

yields an error \(E = O(\Delta t^2, \Delta x^2)\) with unconditional stability.

Finally we can apply a generalized mass operator \(M_x = (\delta, 1 - 2\delta, \delta)\) which yields

\[\frac{(1 + \gamma) M_x \Delta f_{j+1}^n}{\Delta t} - \gamma M_x \Delta f_j^n = \alpha \left[ (1 - \beta) L_{xx} f_j^n + \beta L_{xx} f_j^{n+1} \right]\]

Note that the Crank-Nicholson schemes are recovered using \(\gamma = 0, \beta = 1/2\). The FEM Crank-Nicholson scheme is recovered in this case with \(\delta = 1/6\).

The error for this scheme is

\[E_j^n = \alpha s \Delta x^2 \left( \frac{1}{2} + \gamma - \beta + \frac{s - 1/12}{s} \right) \left. \frac{\partial^4 f^n}{\partial x^4} \right|_{x_j} + O(\Delta x^4)\]

such that the scheme becomes fourth order accurate for

\[\beta = \frac{1}{2} + \gamma + \frac{s - 1/12}{s}\]

The algebraic equations for this scheme are

\[a_j f_{j-1}^{n+1} + b_j f_j^{n+1} + c_j f_{j+1}^{n+1} = (1 + 2\gamma) M_x f_j^n - \gamma M_x f_j^{n-1} + (1 - \beta) s L_{xx} f_j^n\]

\[a_i = c_i = (1 + \gamma) \delta - s \beta\]

\[b_j = (1 + \gamma) (1 - 2\delta) + 2s \beta\]

\[\hat{L}_{xx} f_j^n = f_{j-1}^n - 2f_j^n + f_{j+1}^n\]

The specific choice of \(\delta = 1/12\) yields fourth order accuracy for \(\beta = \frac{1}{2} + \gamma\).

A summary on the methods implemented in the program Diffim.f of implicit schemes is given in the next table.

The following table is a summary of the RMS error obtained for implicit methods with the program Diffim.f. All results use \(s = 1.0\)
### Table 8.2: Overview of implicit methods implemented for the one-dimensional diffusion equation.

<table>
<thead>
<tr>
<th>Method</th>
<th>$M_x$</th>
<th>$\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - FDM-2nd order</td>
<td>(0, 1, 0)</td>
<td>0.5 + $\gamma$</td>
</tr>
<tr>
<td>2 - FEM-2nd order</td>
<td>(1/6, 2/3, 1/6)</td>
<td>0.5 + $\gamma$</td>
</tr>
<tr>
<td>3 - FDM-4th order</td>
<td>(0, 1, 0)</td>
<td>0.5 + $\gamma - \frac{1}{12}\Delta x$</td>
</tr>
<tr>
<td>4 - FEM-4th order</td>
<td>(1/6, 2/3, 1/6)</td>
<td>0.5 + $\gamma + \frac{1}{12}\Delta x$</td>
</tr>
<tr>
<td>5 - Composite</td>
<td>(1/12, 5/6, 1/12)</td>
<td>0.5 + $\gamma$</td>
</tr>
</tbody>
</table>

### Table 8.3: RMS errors for different explicit schemes and varying parameters as a function of grid resolution.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\gamma$</th>
<th>RMS $\Delta x = 0.2$</th>
<th>RMS $\Delta x = 0.1$</th>
<th>RMS $\Delta x = 0.05$</th>
<th>approx. conv. rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.3895</td>
<td>0.1466</td>
<td>0.03993</td>
<td>1.9</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>0.8938</td>
<td>0.1787</td>
<td>0.04185</td>
<td>2.1</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.2393</td>
<td>0.01526</td>
<td>0.00105</td>
<td>3.9</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>0.2393</td>
<td>0.01522</td>
<td>0.00090</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.2393</td>
<td>0.01525</td>
<td>0.00103</td>
<td>3.9</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>2.090</td>
<td>0.03003</td>
<td>0.03245</td>
<td>3.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.760</td>
<td>0.2475</td>
<td>0.04668</td>
<td>2.4</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>2.367</td>
<td>0.1246</td>
<td>0.00813</td>
<td>3.9</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>1.395</td>
<td>0.0927</td>
<td>0.00591</td>
<td>4.0</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>1.867</td>
<td>0.1087</td>
<td>0.00710</td>
<td>3.9</td>
</tr>
</tbody>
</table>

### 8.2.4 Boundary conditions

Thus far we have mostly implied Dirichlet boundary conditions which are straightforward to implement. Von Neumann conditions provide more of a challenge. The most straightforward implementation is a one sided difference (here for the boundary at $x_{min}$)

$$\frac{f_{2}^{n+1} - f_{1}^{n+1}}{\Delta x} = c^{n+1}$$

which yields an equation for the boundary value $f_{1}^{n+1}$. However, this gives only first order accuracy while the overall methods usually give at least 2nd order accuracy.

Better approach: Introduce artificial (mathematical) boundary with $j = 0$ and

$$\frac{f_{2}^{n+1} - f_{0}^{n+1}}{\Delta x} = c^{n+1}$$

where $c^{n+1}$ is the gradient of $f$ at the boundary for time $t^{n+1}$. For the FTCS scheme this gives
\[ f_1^{n+1} = (1 - 2s) f_1^n + s (f_2^n + f_0^n) = (1 - 2s) f_1^n + 2s (f_2^n - c^{n+1} \Delta x) \]

Similar for the fully implicit method this yields

\[ (1 + 2s) f_1^{n+1} - s (f_2^{n+1} + f_0^{n+1}) = f_1^n \]

or

\[ (1 + 2s) f_1^{n+1} - 2s f_2^{n+1} = f_1^n - 2s c^{n+1} \Delta x \]
### 8.2.5 Summary on methods for the one-dimensional diffusion equation

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Algebraic equation</th>
<th>Truncation error</th>
<th>Ampl. factor</th>
<th>Stability</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTCS</td>
<td>( \frac{\Delta f_{j}^{n+1}}{\Delta x} = \alpha L_{xx} f_{j}^{n} )</td>
<td>( E_{n}^{\Delta t} = \alpha \Delta t \left( \begin{array}{c} \frac{2}{3} \frac{n^{3}}{\Delta x^{3}} \end{array} \right) )</td>
<td>( s \leq 1/2 )</td>
<td>( s \leq 1/2 )</td>
<td>( s \leq 1/2 )</td>
</tr>
<tr>
<td>DuFort-Frankel</td>
<td>( \frac{\Delta f_{j}^{n+1}}{\Delta x} = \frac{2}{3} \left( f_{j-1}^{n} + f_{j}^{n} + f_{j+1}^{n} \right) )</td>
<td>( E_{n}^{\Delta t} = \alpha \Delta t \left( \begin{array}{c} \frac{2}{3} \frac{n^{3}}{\Delta x^{3}} \end{array} \right) )</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Crank-Nicholson</td>
<td>( \frac{\Delta f_{j}^{n+1}}{\Delta x} = \frac{2}{3} \left( f_{j-1}^{n} + f_{j}^{n} + f_{j+1}^{n} \right) )</td>
<td>( E_{n}^{\Delta t} = \alpha \Delta t \left( \begin{array}{c} \frac{2}{3} \frac{n^{3}}{\Delta x^{3}} \end{array} \right) )</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>3-level implicit</td>
<td>( \frac{\Delta f_{j}^{n+1}}{\Delta x} = \frac{2}{3} \left( f_{j-1}^{n} + f_{j}^{n} + f_{j+1}^{n} \right) )</td>
<td>( E_{n}^{\Delta t} = \alpha \Delta t \left( \begin{array}{c} \frac{2}{3} \frac{n^{3}}{\Delta x^{3}} \end{array} \right) )</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>FEM Crank-Nicholson</td>
<td>( \frac{\Delta f_{j}^{n+1}}{\Delta x} = \frac{2}{3} \left( f_{j-1}^{n} + f_{j}^{n} + f_{j+1}^{n} \right) )</td>
<td>( E_{n}^{\Delta t} = \alpha \Delta t \left( \begin{array}{c} \frac{2}{3} \frac{n^{3}}{\Delta x^{3}} \end{array} \right) )</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

**Remarks**
- \( s = \frac{\Delta f}{\Delta x} \)
- \( L_{xx} = \frac{1}{3} \left( 1, -2, 1 \right) \)
- \( M_{xx} = \frac{1}{6} \left( 1, 6, 3, 1 \right) \)
8.3 Splitting schemes

8.3.1 ADI method

In two-dimensions implicit methods are usually computationally very expensive. Consider the two-dimensional version of the fully implicit method.

\[(1 + 2s_x + 2s_y) f_{jk}^{n+1} - s_x (f_{j-1,k}^{n+1} + f_{j+1,k}^{n+1}) - s_y (f_{j,k-1}^{n+1} + f_{j,k+1}^{n+1}) = f_{jk}^{n}\]

The problem with this equation is the inversion of the matrix defined by the rhs of the equation. In the one-dimensional case the resulting matrix was a tridiagonal banded matrix. In the two-dimensional case the matrix is not anymore banded but has elements that are far offset off the diagonal even though the matrix is sparse. While there are some techniques to deal with sparse matrices Gauss elimination is still rather and often prohibitively expensive.

The alternative to a fully implicit solution is the ADI method which is illustrated using the following basic equations.

\[
\begin{align*}
\frac{f_{jk}^* - f_{jk}^n}{\Delta t/2} - \alpha_x L_{xx} f_{jk}^* - \alpha_y L_{yy} f_{jk}^n &= 0 \\
\frac{f_{jk}^{n+1} - f_{jk}^*}{\Delta t/2} - \alpha_x L_{xx} f_{jk}^{n+1} - \alpha_y L_{yy} f_{jk}^* &= 0
\end{align*}
\]

Here the \( * \) in the first equation is interpreted as an auxiliary intermediate time level \( n + 1/2 \). The corresponding algebraic equations are cast in the form

\[
\begin{align*}
-\frac{1}{2} s_x f_{j-1,k}^* + (1 + s_x) f_{jk}^* - \frac{1}{2} s_x f_{j+1,k}^* &= \frac{1}{2} s_y f_{j,k-1}^n + (1 - s_y) f_{jk}^n + \frac{1}{2} s_y f_{j,k+1}^n \\
-\frac{1}{2} s_y f_{j,k-1}^{n+1} + (1 + s_y) f_{jk}^{n+1} - \frac{1}{2} s_y f_{j,k+1}^{n+1} &= \frac{1}{2} s_x f_{j-1,k}^* + (1 - s_x) f_{jk}^* + \frac{1}{2} s_x f_{j+1,k}^*
\end{align*}
\]

Note that these equations are almost identical to the ones used for the corresponding elliptic equation solver using the ADI scheme ((7.4) and (7.5)). These equations are used in two stages to evolve the system from time level \( n \) to time level \( n + 1 \). Note that - as in the prior introduction of the ADI scheme - each step requires only the solution of an implicit equation in one dimension. In the first step (equation) the system is solved for \( k \) considered fixed. Since the solution is sought only for the \( x \) grid, i.e., one-dimensional, the resulting matrices are banded tridiagonal and easy to solve with the Thomas algorithm.

Similarly the second step is conducted only for the \( y \) grid (with \( x \) or \( j \) fixed) such that the second step also involves only one-dimension and thus the solution of a banded tridiagonal matrix.

The von Neumann stability analysis is used to determine an amplification factor for each half step. The product of the resulting amplification factors yields
which implies \(|g| \leq 1\) and therefore unconditional stability. The scheme has an error of \(O(\Delta t^2, \Delta x^2, \Delta y^2)\).

Note that boundary conditions need to be considered carefully to insure that the global error indeed remains second order. Using Dirichlet conditions the evaluation of the boundary at \(x_{\text{max}} = 1\) for the intermediate step using

\[
f_{n+1/2}^* = b_k^{n+1/2}
\]

yields an error of \(O(\Delta t)\). The correct approach for this boundary should be

\[
f_{n+1/2}^* = \frac{1}{2} (b_k^n + b_k^{n+1}) - \frac{1}{4} \Delta t L_{yy} (b_k^n + b_k^{n+1})
\]

### 8.3.2 Generalized two level scheme

As in the case of the one-dimensional schemes splitting schemes can easily be generalized by introducing weights for the spatial and temporal derivatives at different time levels. For two time levels this generalization is

\[
\frac{\Delta f_{j+1,k}^{n+1}}{\Delta t} = (1 - \beta) [\alpha_x L_{xx} + \alpha_y L_{yy}] f_{j,k}^n + \beta [\alpha_x L_{xx} + \alpha_y L_{yy}] f_{j,k}^{n+1}
\]

(8.1)

again with \(\Delta f_{j,k}^{n+1} = f_{j,k}^{n+1} - f_{j,k}^n\). One can rewrite this as an equation for \(\Delta f_{j,k}^{n+1}\) by moving the \(\beta\) terms to the left side in the following manner

\[
[1 - \beta \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy})] \Delta f_{j,k}^{n+1} = \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy}) f_{j,k}^n
\]
Up to $O(\Delta t^2)$ this can be rewritten as

$$(1 - \beta \Delta t \alpha_x L_{xx}) (1 - \beta \Delta t \alpha_y L_{yy}) \Delta f_{jk}^{n+1} = \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy}) f_{jk}^n$$

where we have added a term $\beta^2 \Delta t^2 \alpha_x \alpha_y L_{xx} L_{yy} \Delta f_{jk}^{n+1}$ which however is $O(\Delta t^2)$. With the definition $\Delta f_{jk}^* = (1 - \beta \Delta t \alpha_y L_{yy}) \Delta f_{jk}^{n+1}$ one obtains the following two stage scheme:

1. $$(1 - \beta \Delta t \alpha_x L_{xx}) \Delta f_{jk}^* = \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy}) f_{jk}^n$$
2. $$(1 - \beta \Delta t \alpha_y L_{yy}) \Delta f_{jk}^{n+1} = \Delta f_{jk}^*$$

Each of these steps only involves the inversion of a tridiagonally banded matrix.

Similar to the one-dimensional equivalent the resulting scheme is unconditionally stable for $\beta \geq 0.5$ (also in 3D) and the resulting error is of the order $O(\Delta t^2, \Delta x^2, \Delta y^2)$ for $\beta = 0.5$.

Note that an extension to three time levels is straightforward as in prior examples of one-dimensional schemes

$$\frac{(1 + \gamma) \Delta f_{jk}^{n+1}}{\Delta t} - \gamma \frac{\Delta f_{jk}^n}{\Delta t} = (1 - \beta) \left[ \alpha_x L_{xx} f_{jk}^n + \alpha_y L_{yy} f_{jk}^n \right]$$

yields with some minor algebra

1. $$(1 - \frac{\beta \Delta t}{1 + \gamma} \alpha_x L_{xx}) \Delta f_{jk}^* = \frac{\Delta t}{1 + \gamma} (\alpha_x L_{xx} + \alpha_y L_{yy}) f_{jk}^n + \frac{\gamma}{1 + \gamma} \Delta f_{jk}^n$$
2. $$(1 - \frac{\beta \Delta t}{1 + \gamma} \alpha_y L_{yy}) \Delta f_{jk}^{n+1} = \Delta f_{jk}^*$$

Again this scheme requires only a one-dimensional implicit solution at each stage.

### 8.3.3 Finite element methods

Similar to one-dimensional implicit schemes it is straightforward to extend two or three-dimensional implicit schemes to finite elements. In general the inclusion of finite elements can be done using a mass operator such that the diffusion equation in operator form becomes

$$M_x \otimes M_y \frac{\partial f}{\partial t} \bigg|_{jk} = \alpha_x M_y \otimes L_{xx} f_{jk} + \alpha_y M_x \otimes L_{yy} f_{jk}$$
where this operator for linear finite elements is given by \( M_x = (1/6, 2/3, 1/6) \) and as usual the second derivative operator is \( L_{xx} = \frac{1}{\Delta x^2} (1, -2, 1) \). Another way to express the operation is

\[
M_x f_{jk} = \sum_{i=j-1}^{j+1} m_{ji} f_{ik} , \quad M_y f_{jk} = \sum_{m=k-1}^{k+1} f_{jm} m_{mk}
\]

Applying the mass operator in combinations with the second derivative operator yields for instance

\[
M_y \otimes L_{xx} f_{jk} = \frac{1}{6} L_{xx} f_{j,k-1} + \frac{2}{3} L_{xx} f_{j,k} + \frac{1}{6} L_{xx} f_{j,k+1}
\]

Note that one can further generalize the approach by using instead of the fem mass operator an operator defined as \( M_x = (\delta, 1 - 2\delta, \delta) \).

Using the mass operator for the two level scheme (8.1) yields

\[
M_x \otimes M_y \frac{\Delta f_{jk}^{n+1}}{\Delta t} = (\alpha_x M_y \otimes L_{xx} + \alpha_y M_x \otimes L_{yy}) \left[ (1 - \beta) f_{jk}^{n} + \beta f_{jk}^{n+1} \right]
\]

Similar to the finite difference two level scheme one can re-arrange terms (and add a term of order \( O(\Delta t^2) \)) to yield

\[
(M_x - \beta \Delta t \alpha_x L_{xx}) \otimes (M_y - \beta \Delta t \alpha_y L_{yy}) f_{jk}^{n+1} = (M_x + (1 - \beta) \Delta t \alpha_x L_{xx}) \otimes (M_y + (1 - \beta) \Delta t \alpha_y L_{yy}) f_{jk}^{n} \quad \text{(8.2)}
\]

Note that the terms proportional to \( \Delta t^2 \) have been added compared to the original equation which limits the accuracy to \( O(\Delta t^2) \) which, however was anyhow the accuracy of the original equation. Similar to the finite difference approach one can now solve this equation in two stages.
\[ (1) \quad (M_x - \beta \Delta t \alpha_x L_{xx}) f_{jk}^* = (M_y + (1 - \beta) \Delta t \alpha_y L_{yy}) f_{jk}^n \quad (8.3) \]
\[ (2) \quad (M_y - \beta \Delta t \alpha_y L_{yy}) f_{jk}^{n+1} = (M_x + (1 - \beta) \Delta t \alpha_x L_{xx}) f_{jk}^n \quad (8.4) \]

Here one can easily that this set of equations is identical to (8.2) by multiplying the second stage equation with \((M_x - \beta \Delta t \alpha_x L_{xx})\) and insert \(f_{jk}^*\) from the first stage.

Note that the choice of \(\beta = 0.5\) yields the ADI finite element method. The two level finite element scheme with \(M_x = (\delta, 1 - 2\delta, \delta)\) is unconditionally stable for

\[ \beta \geq 0.5 + \frac{\delta - 0.25}{s} \]

for \(\Delta x = \Delta y\) and \(\alpha_x = \alpha_y\).

Finally note that the finite element method can also easily be applied to a linear version computing \(\Delta f_{jk}^{n+1}\) instead of the equations (8.3) and (8.4). Equation (8.2) is easily re-written as

\[
(M_x - \beta \Delta t \alpha_x L_{xx}) \otimes (M_y - \beta \Delta t \alpha_y L_{yy}) \Delta f_{jk}^{n+1} = \Delta t \left( \alpha_x M_y \otimes L_{xx} + \alpha_y M_x \otimes L_{yy} \right) f_{jk}^n \equiv R_{jk}^n
\]

which with the definition of \(\Delta f_{jk}^* = (M_y - \beta \Delta t \alpha_y L_{yy}) \Delta f_{jk}^{n+1}\) lends itself to the following two stage splitting scheme

\[
(1) \quad (M_x - \beta \Delta t \alpha_x L_{xx}) \Delta f_{jk}^* = R_{jk}^n
\]
\[
(2) \quad (M_y - \beta \Delta t \alpha_y L_{yy}) \Delta f_{jk}^{n+1} = \Delta f_{jk}^*
\]

![Figure 8.6: Schematic of the two stages of the two level finite element ADI method.](image)

The scheme is basically identical with the nonlinear formulation. The advantage of the linear formulation is usually a better accuracy. The following figure shows a schematic of the finite
element scheme. Different from the finite difference method the first stage involves all nodes in
the vicinity of \( j, k \). The overall numerical effort of the various splitting schemes in two dimensions
is very comparable. Although the finite element method involve some more nodes for the 1st stage
step the difference with finite difference methods is not significant. Thus the eventual choice is
more determined by properties of the resulting schemes.

Finally it is worth pointing out that one area of increased complexity is that of boundary conditions.
These need to be formulated in agreement with the discretization and the solution method. Thus
choice of particular boundary conditions alters the algebraic equations at the boundaries and must
be taken into account in the solution of the tridiagonal matrices for splitting schemes.