15 The Heat equation in 2 and 3 spatial dimensions

In this Lecture, which concludes our treatment of parabolic equations, we will develop numerical methods for the Heat equation in 2 and 3 dimensions in space. We will present the details of these developments for the 2-dimensional case, while for the 3-dimensional case, we will mention only those aspects which cannot be straightforwardly generalized from 2 to 3 spatial dimensions.

Since this Lecture is quite long, here we give a brief preview of its results. First, we will explain how the solution vector can be set up on a 3-dimensional grid (two dimensions in space and one in time). We will discuss both the conceptual part of this setup and its implementation in Matlab. Then we will present the simple explicit scheme for the 2D Heat equation and will show that it is even more time-inefficient than it was for the Heat equation in one dimension. In search of a time-efficient substitute, we will analyze the naive version of the Crank-Nicolson scheme for the 2D Heat equation, and will discover that that scheme is not time-efficient either! We will then show how a number of *time-efficient* generalizations of the Crank-Nicolson scheme to 2 and 3 dimensions can be constructed. These generalizations are known under the common name of *Alternating Direction* methods, and are a particular case of an even more general class of so-called *operator-splitting* methods. In Appendix 1 we will point out a relation between these methods and the IMEX methods mentioned in Lecture 14, as well as with the predictor-corrector methods considered in Lecture 3. Finally, we will also describe that prescribing boundary conditions (even the Dirichlet ones) for those time-efficient schemes is not always a trivial matter, and demonstrate *how* they can be prescribed.

15.1 Setting up the solution vector on a three-dimensional grid

In this Lecture, we study the following IBVP:

$$u_t = u_{xx} + u_{yy} \qquad 0 < x < 1, \quad 0 < y < Y \quad t > 0\,; \tag{15.1}$$

$$u(x, y, t = 0) = u_0(x, y) \qquad 0 \le x \le 1 \quad 0 \le y \le Y\,; \tag{15.2}$$

$$u(0, y, t) = g_0(y, t), \quad u(1, y, t) = g_1(y, t), \qquad 0 \le y \le Y, \quad t \ge 0\,; \tag{15.3}$$

$$u(x, 0, t) = g_2(x, t), \quad u(x, Y, t) = g_3(x, t), \qquad 0 \le x \le 1, \quad t \ge 0\,. \tag{15.4}$$

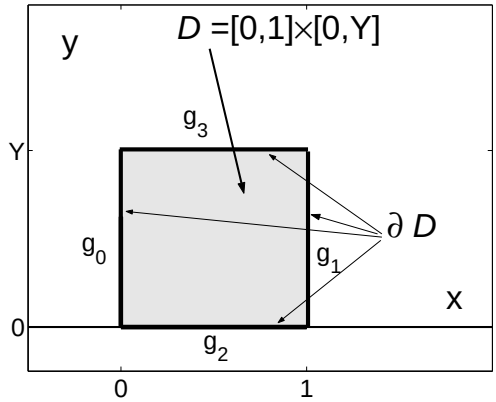
We will always assume that the boundary conditions are consistent with the initial condition:

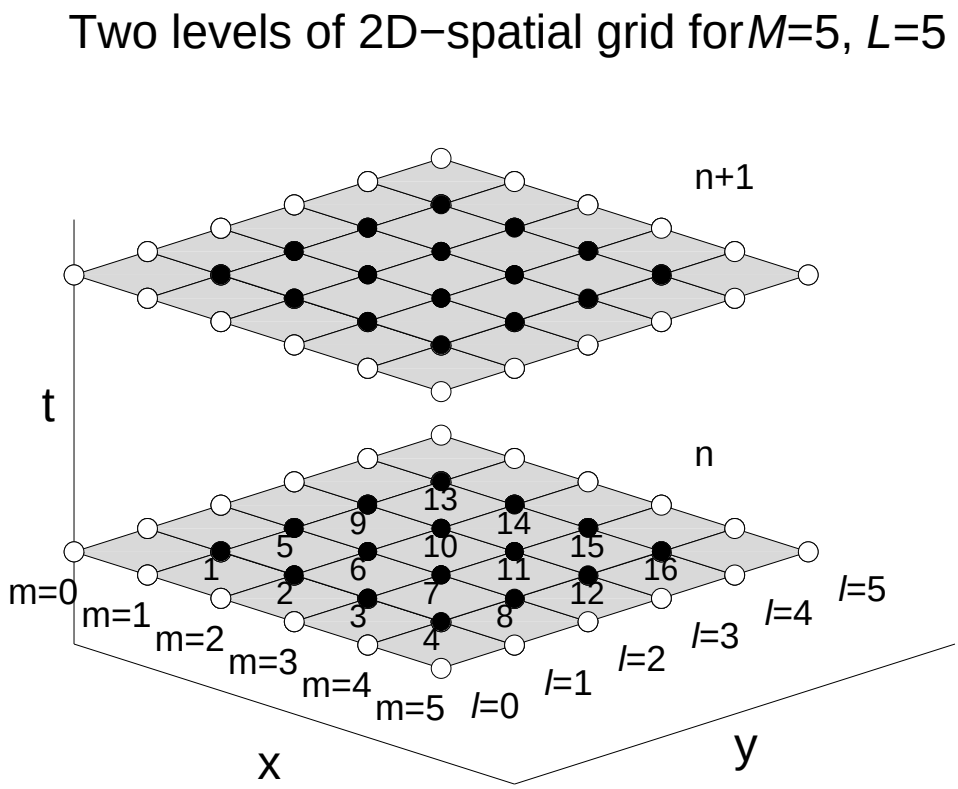
$$g_0(y, 0) = u_0(0, y), \quad g_1(y, 0) = u_0(1, y), \quad g_2(x, 0) = u_0(x, 0), \quad g_3(x, 0) = u_0(x, Y), \tag{15.5}$$

and, at the corners of the domain, with each other:

$$g_0(0, t) = g_2(0, t), \quad g_0(Y, t) = g_3(0, t), \quad g_3(1, t) = g_1(Y, t), \quad g_1(0, t) = g_2(1, t), \quad \text{for } t > 0. \tag{15.6}$$

The figure on the right shows the two-dimensional spatial domain, where the Heat equation (15.1) holds, as well as the domain's boundary, where the boundary conditions (15.3), (15.4) are specified. Note that we have allowed the lengths of the domain in the x and y directions to be different (if $Y \neq 1$). Although, in principle, one can always make $Y = 1$ by a suitable scaling of the spatial coordinates, we prefer not to do so in order to allow, later on, the step sizes in x and y to be the same. The latter is simply the matter of convenience.





To discretize the Heat equation (15.1), we cover domain D with a two-dimensional grid. As we have just noted above, in what follows we will assume that the step sizes in the x and y directions are the same and equal h. We also discretize the time variable with a step size κ. Then the three-dimensional grid for the 2D Heat equation consists of points ($x = mh$, $y = lh$, $t = n\kappa$), $0 \leq m \leq M = 1/h$, $0 \leq l \leq L = Y/h$, and $0 \leq n \leq N = t_{\max}/\kappa$. Two time levels of such a grid for the case $M = 5$ and $L = 5$ are shown in the figure above.

We will denote the solution on the above grid as

$$U^n_{ml} = u(mh, lh, n\kappa)\,, \qquad 0 \leq m \leq M, \quad 0 \leq l \leq L, \quad 0 \leq n \leq N. \tag{15.7}$$

We expect that any numerical scheme that we will design will give some recurrence relation

between U_{ml}^{n+1} and U_{ml}^{n} (and, possibly, U_{ml}^{n-1} etc.). As long as our grid is rectangular, the array of values U_{ml}^{n} at each given n can be conveniently represented as an $(M+1)\times(L+1)$ matrix. In this Lecture, we will consider only this case of a rectangular grid. Then, to step from level n to level $(n+1)$, we just apply the recurrence formula to each element of the matrix U_{ml}^{n}. For example:

```
for m = 2 : mmax-1
    for ell = 2 : ellmax-1
        Unew(m,ell) = a*U(m,ell) + b*U(m+1,ell-1);
    end
end
U(2:mmax-1, 2:ellmax-1) = Unew(2:mmax-1, 2:ellmax-1);
```

If we want to record and keep the value of the solution at each time level, we can instead use: `U(m,ell,n+1) = a*U(m,ell,n) + b*U(m+1,ell-1,n);` .

On the other hand, if the spatial domain in *not* rectangular, as occurs in many practical problems, then defining U_{ml}^{n} as a matrix is not possible, or at least not straightforward. In this case, one needs to *reshape* the two-dimensional array U_{ml}^{n} into a one-dimensional vector. Even though it will *not* be needed for the purposes of this lecture or homework, we will still illustrate the idea and implementation behind this reshaping. For simplicity, we will consider the case where the grid is rectangular. This reshaping can be done in more than one way. Here we will consider only the so-called *lexicographic ordering*. In this ordering, the first $(M-1)$ components of the solution vector $\vec{\mathbf{U}}$ will be the values $U_{m,1}$ with $m=1,2,\ldots,\ M-1$. (Here, for brevity, we have omitted the superscript n, and also inserted a comma between the subscripts pertaining to the x and y axes for visual convenience.) The next $M-1$ components will be $U_{m,2}$ with $m=1,2,\ldots,\ M-1$, and so on. The resulting vector is:

$$\vec{\mathbf{U}} = \begin{pmatrix} U_{1,1} \\ U_{2,1} \\ \cdot \\ \cdot \\ U_{M-1,1} \\ U_{1,2} \\ U_{2,2} \\ \cdot \\ \cdot \\ U_{M-1,2} \\ \cdot \\ \cdot \\ \cdot \\ U_{1,L-1} \\ U_{2,L-1} \\ \cdot \\ \cdot \\ U_{M-1,L-1} \end{pmatrix} \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \\ \end{array}\right\} \text{1st row of 2D level (along } y=1\cdot h \text{ (i.e. } l=1)) \\ \left.\begin{array}{l} \\ \\ \\ \\ \\ \end{array}\right\} \text{2nd row of 2D level (along } y=2\cdot h \text{ (i.e. } l=2)) \\ \\ \\ \\ \left.\begin{array}{l} \\ \\ \\ \\ \\ \end{array}\right\} (L-1)\text{th row of 2D level (along } y=(L-1)\cdot h \text{ (i.e. } l=L-1)) \end{array} \tag{15.8}$$

An example of lexicographic ordering of the nodes of one time level is shown in the figure on the previous page for $M=5$ and $L=5$ (see the numbers next to the filled circles on the lower level).

Let us now show how one can set up one time level of the grid and construct a vector of the form (15.8), using built-in commands in Matlab. In order to avoid possible confusion, we will define the domain D slightly differently than was done above. Namely, we let $x \in [0, 2]$ and $y \in [3, 4]$. Next, let us discretize the x coordinate as

```
>> x=[0 1 2]
x =
     0     1     2
```

and the y coordinate as

```
>> y=[3 4]
y =
     3     4
```

(Such a coarse discretization is quite sufficient for the demonstration of how Matlab commands can be used.) Now, let us construct a two-dimensional grid as follows:

```
>> [X,Y]=meshgrid(x,y)
X =
     0     1     2
     0     1     2
Y =
     3     3     3
     4     4     4
```

Thus, entries of matrix `X` along the rows equal the values of `x`, and these entries do not change along the columns. Similarly, entries of matrix `Y` along the columns equal the values of `y`, and these entries do not change along the rows.

Here is a function Z of two variables x and y, constructed with the help of the above matrices `X` and `Y`:

```
>> Z=100*Y+X
Z =
   300   301   302
   400   401   402
```

Now, if we need to reshape matrix `Z` into a vector, we simply say:

```
>> Zr=reshape(Z,prod(size(Z)),1)
Zr =
   300
   400
   301
   401
   302
   402
```

(Note that

```
>> size(Z)
ans =
     2     3
```

and command `prod` simply computes the product of all entries of its argument.) If we want to go back and forth between using `Z` and `Zr`, we can use the reversibility of command `reshape`:

```
>> Zrr=reshape(Zr,size(X,1),size(X,2))
Zrr =
   300    301    302
   400    401    402
```

which, of course, gives you back the `Z`. Finally, if you want to plot the two-dimensional function $Z(x, y)$, you can type either `mesh(x,y,Z)` or `mesh(X,Y,Z)`. You may always look up the `help` for any of the above (or any other) commands if you have questions about them.

15.2 Simple explicit method for the 2D Heat equation

Construction of the simple explicit scheme for the 2D Heat equation is a fairly straightforward matter. Namely, we discretize the terms in (15.1) in the standard way:

$$\begin{aligned} u_t &\to \frac{U_{m,l}^{n+1} - U_{m,l}^n}{\kappa} \equiv \frac{\delta_t U_{m,l}^n}{\kappa}, \\ u_{xx} &\to \frac{U_{m+1,l}^n - 2U_{m,l}^n + U_{m-1,l}^n}{h^2} \equiv \frac{\delta_x^2 U_{m,l}^n}{h^2}, \\ u_{yy} &\to \frac{U_{m,l+1}^n - 2U_{m,l}^n + U_{m,l-1}^n}{h^2} \equiv \frac{\delta_y^2 U_{m,l}^n}{h^2}, \end{aligned} \tag{15.9}$$

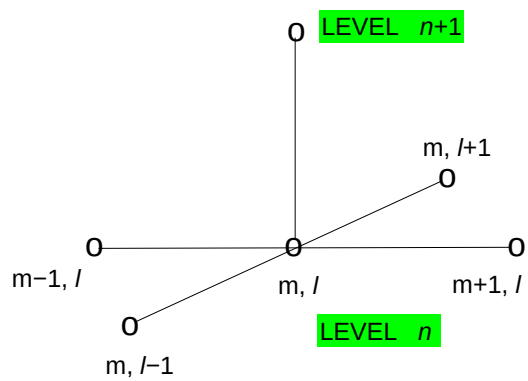
and substitute these expressions into (15.1) to obtain:

$$\begin{aligned} U_{ml}^{n+1} &= U_{ml}^n + r\left(\delta_x^2 U_{ml}^n + \delta_y^2 U_{ml}^n\right) \\ &\equiv \left(1 + r\delta_x^2 + r\delta_y^2\right) U_{ml}^n . \end{aligned} \tag{15.10}$$

Three remarks about notations in (15.9) and (15.10) are in order. First,

$$r = \frac{\kappa}{h^2},$$

as before. Second, we will use the notations $U_{m,l}$ and U_{ml} (i.e. with and without a comma between m and l) interchangeably; i.e., they denote the same thing. Third, the operators δ_x^2 and δ_y^2 will be used extensively in this Lecture.



The stencil for the simple explicit scheme (15.10) is shown on the right. Implementation of this scheme is discussed in a homework problem.

Next, we perform the von Neumann stability analysis of scheme (15.10). To this end, we use the fact that the solution of this constant-coefficient difference equation is satisfied by the Fourier harmonics

$$U_{ml}^n = \rho^n e^{i\beta mh} e^{i\gamma lh}, \tag{15.11}$$

which we substitute into (15.10) to find the amplification factor ρ. In this calculation, as well as in many other calculations in the remainder of this Lecture, we will use the following formulae:

$$\delta_x^2 \left[e^{i\beta mh}\, e^{i\gamma lh}\right] = -4\sin^2\left(\frac{\beta h}{2}\right)\left[e^{i\beta mh}\, e^{i\gamma lh}\right], \tag{15.12}$$

$$\delta_y^2 \left[e^{i\beta mh}\, e^{i\gamma lh}\right] = -4\sin^2\left(\frac{\gamma h}{2}\right)\left[e^{i\beta mh}\, e^{i\gamma lh}\right]. \tag{15.13}$$

(You will be asked to confirm the validity of these formulae in a homework problem.) Substituting (15.11) into (15.10) and using (15.12) and (15.13), one finds

$$\rho = 1 - 4r\left(\sin^2\frac{\beta h}{2} + \sin^2\frac{\gamma h}{2}\right). \tag{15.14}$$

The harmonics most prone to instability are, as for the one-dimensional Heat equation, those with the highest spatial frequency, and for which

$$\sin^2\frac{\beta h}{2} = \sin^2\frac{\gamma h}{2} = 1\,.$$

For these harmonics, the stability condition $|\rho| \le 1$ implies

$$r \le \frac{1}{4} \qquad \text{or, equivalently,} \qquad \kappa \le \frac{h^2}{4}. \tag{15.15}$$

Thus, in order to ensure the stability of the simple explicit scheme (15.10), one has to impose a restriction on the time step κ that is twice as strong as the analogous restriction in the case of the one-dimensional Heat equation. Therefore, the simple explicit scheme is computationally inefficient, and our next step is, of course, to look for a computationally efficient scheme. As the first candidate for that position, we will analyze the Crank-Nicolson scheme.

15.3 Naive generalization of Crank-Nicolson scheme for the 2D Heat equation

Our main finding in this subsection will be that a naive generalization of the CN method (13.6) is also computationally inefficient. The underlying analysis will allow us to formulate specific properties that a computationally efficient scheme must possess.

The naive generalization to two dimensions of the CN scheme, (13.5) or (13.6), is:

$$U_{ml}^{n+1} = U_{ml}^n + \frac{r}{2}\left(\delta_x^2 + \delta_y^2\right)\left(U_{ml}^n + U_{ml}^{n+1}\right), \tag{15.16}$$

or, equivalently,

$$\left(1 - \frac{r}{2}\delta_x^2 - \frac{r}{2}\delta_y^2\right) U_{ml}^{n+1} = \left(1 + \frac{r}{2}\delta_x^2 + \frac{r}{2}\delta_y^2\right) U_{ml}^n\,. \tag{15.17}$$

Following the lines of Lecture 13, one can show that the accuracy of this scheme is $O(\kappa^2 + h^2)$. Also, the von Neumann analysis yields the following expression for the error amplification factor:

$$\rho = \frac{1 - 2r\left(\sin^2\frac{\beta h}{2} + \sin^2\frac{\gamma h}{2}\right)}{1 + 2r\left(\sin^2\frac{\beta h}{2} + \sin^2\frac{\gamma h}{2}\right)}, \tag{15.18}$$

so that $|\rho| \le 1$ for any r and hence the CN scheme (15.17) is unconditionally stable.

We will now demonstrate that scheme (15.16) / (15.17) is computationally inefficient. To that end, we need to exhibit the explicit matrix form of that scheme. We begin by rewriting (15.16) in the form[26]:

$$\begin{aligned} & (1+2r)U_{m,l}^{n+1} - \frac{r}{2}\left(U_{m+1,l}^{n+1} + U_{m-1,l}^{n+1}\right) - \frac{r}{2}\left(U_{m,l+1}^{n+1} + U_{m,l-1}^{n+1}\right) \\ = \; & (1-2r)U_{m,l}^{n} + \frac{r}{2}\left(U_{m+1,l}^{n} + U_{m-1,l}^{n}\right) + \frac{r}{2}\left(U_{m,l+1}^{n} + U_{m,l-1}^{n}\right) . \end{aligned} \tag{15.19}$$

To write down Eqs. (15.19) for all m and l in a compact form, we will need the following notations:

$$A = \begin{pmatrix} 2r & -r/2 & 0 & \cdot & \cdot & 0 \\ -r/2 & 2r & -r/2 & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & -r/2 & 2r & -r/2 \\ 0 & \cdot & \cdot & 0 & -r/2 & 2r \end{pmatrix}, \quad \vec{\mathbf{U}}_{;l} = \begin{pmatrix} U_{1,l} \\ U_{2,l} \\ \cdot \\ U_{M-2,l} \\ U_{M-1,l} \end{pmatrix}, \tag{15.20}$$

and

$$\vec{\mathbf{B}}_k = \begin{pmatrix} (g_k)_1 \\ (g_k)_2 \\ \cdot \\ \cdot \\ (g_k)_{M-1} \end{pmatrix}, \quad \text{for } k = 2, 3; \qquad \vec{\mathbf{b}}_l^n = \begin{pmatrix} (g_0)_l^n + (g_0)_l^{n+1} \\ 0 \\ \cdot \\ 0 \\ (g_1)_l^n + (g_1)_l^{n+1} \end{pmatrix}. \tag{15.21}$$

Using these notations, one can recast Eq. (15.19) in a matrix form. Namely, for $l = 2, \ldots, L-2$ (i.e. for layers with constant y and which are *not* adjacent to the boundaries), Eq. (15.19) becomes:

$$(I+A)\vec{\mathbf{U}}_{;l}^{n+1} - \frac{r}{2} I\, \vec{\mathbf{U}}_{;l+1}^{n+1} - \frac{r}{2} I\, \vec{\mathbf{U}}_{;l-1}^{n+1} = (I-A)\vec{\mathbf{U}}_{;l}^{n} + \frac{r}{2} I\, \vec{\mathbf{U}}_{;l+1}^{n} + \frac{r}{2} I\, \vec{\mathbf{U}}_{;l-1}^{n} + \frac{r}{2}\vec{\mathbf{b}}_l^n , \tag{15.22}$$

where I is the $(M-1)\times(M-1)$ identity matrix. Note that Eq. (15.22) is analogous to Eq. (13.9), although the meanings of notation A is *different* in these two equations. Continuing, for the layer with $l = 1$ one obtains:

$$(I+A)\vec{\mathbf{U}}_{;l}^{n+1} - \frac{r}{2} I\, \vec{\mathbf{U}}_{;l+1}^{n+1} - \frac{r}{2}\vec{\mathbf{B}}_2^{n+1} = (I-A)\vec{\mathbf{U}}_{;l}^{n} + \frac{r}{2} I\, \vec{\mathbf{U}}_{;l+1}^{n} + \frac{r}{2}\vec{\mathbf{B}}_2^{n} + \frac{r}{2}\vec{\mathbf{b}}_l^n . \tag{15.23}$$

The equation for $l = L-1$ has a similar form. Combining now all these equations into one, we obtain:

$$(\mathcal{I}+\mathcal{A})\vec{\mathbf{U}}^{n+1} = (\mathcal{I}-\mathcal{A})\vec{\mathbf{U}}^{n} + \mathcal{B}^n, \tag{15.24}$$

where $\vec{\mathbf{U}}$ has been defined in (15.8), $\mathcal{I}$ is the $[(M-1)(L-1)]\times[(M-1)(L-1)]$ identity matrix, and

$$\mathcal{A} = \begin{pmatrix} A & -\frac{r}{2}I & O & \cdot & \cdot & O \\ -\frac{r}{2}I & A & -\frac{r}{2}I & O & \cdot & O \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ O & \cdot & O & -\frac{r}{2}I & A & -\frac{r}{2}I \\ O & \cdot & \cdot & O & -\frac{r}{2}I & A \end{pmatrix}, \qquad \mathcal{B}^n = \frac{r}{2}\begin{pmatrix} \vec{\mathbf{B}}_2^n + \vec{\mathbf{B}}_2^{n+1} + \vec{\mathbf{b}}_1^n \\ \vec{\mathbf{b}}_2^n \\ \cdot \\ \vec{\mathbf{b}}_{L-2}^n \\ \vec{\mathbf{B}}_3^n + \vec{\mathbf{B}}_3^{n+1} + \vec{\mathbf{b}}_{L-1}^n \end{pmatrix}. \tag{15.25}$$

[26]Recall our convention to use notations U_{ml} and $U_{m,l}$ interchangeably.

In (15.25), O stands for the $(M-1)\times(M-1)$ zero matrix; hopefully, the use of the same character here and in the O-symbol (e.g., $O(h^2)$) will not cause any confusion.

Now, the $[(M-1)(L-1)]\times[(M-1)(L-1)]$ matrix $\mathcal{A}$ in (15.25) is *block*-tridiagonal, but *not* tridiagonal. Namely, it has only 5 nonzero diagonals or subdiagonals, but the outer subdiagonals are not located next to the inner subdiagonals but separated from them by a band of zeros, with the band's width being $(M-2)$. Thus, the total width of the central nonzero band in matrix $\mathcal{A}$ is $2(M-2)+3$. Inverting such a matrix is not a computationally efficient process in the sense that it will require not $O(ML)$, but $O(ML)^2$ or $O(ML)^3$ operations. In other words, the number of operations required to solve Eq. (15.25) is *much greater* than the number of unknowns.[27]

Let us summarize what we have established about the CN method (15.17) for the 2D Heat equation. The method: (i) has accuracy $O(\kappa^2+h^2)$, (ii) is unconditionally stable, but (iii) requires much more operations per time step than the number of unknown variables. We are satisfied with features (i) and (ii), but not with (iii). In the remainder of this Lecture, we will be concerned with constructing methods that do not have the deficiency stated in (iii). For reference purposes, we will now repeat the properties that we want our "dream scheme" to have.

In order to be considered computationally efficient, the scheme:

(i) must have accuracy $O(\kappa^2+h^2)$ (or better);

(ii) must be unconditionally stable;

(iii) must require the number of operations per time step that is proportional to the number of the unknowns. (15.26)

In the next subsection, we will set the ground for obtaining such schemes.

15.4 Derivation of a computationally efficient scheme

In this section, we will derive a scheme which we will use later on to obtain methods that satisfy all the three conditions (15.26). Specifically, we pose the problem as follows: *Find a scheme that (a) reduces to the Crank-Nicolson scheme (13.6) in the case of the one-dimensional Heat equation and (b) has the same order of truncation error, i.e. $O(\kappa^2+h^2)$; or, in other words, satisfies property (i) of (15.26).* Of course, there are many (probably, infinitely many) such schemes. A significant contribution by computational scientists in the 1950's was finding, among those schemes, the ones which are unconditionally stable (property (ii)) and could be implemented in a time-efficient manner (property (iii)). In the remainder of this section, we

[27] One might have reasoned that, since $\mathcal{A}$ in (15.25) is block-tridiagonal, then one could solve Eq. (15.24) by the block-Thomas algorithm. This well-known generalization of the Thomas algorithm presented in Lecture 8 assumes that the coefficients a_k, b_k, c_k and α_k, β_k in (8.18) and (8.19) are $(M-1)\times(M-1)$ square matrices. Then formulae (8.21)–(8.23) of the Thomas algorithm are straightforwardly generalized by assigning the matrix sense to all the operations in those formulae.

However, this naive idea of being able to solve (15.24) by the block-Thomas algorithm does not work. Indeed, consider the defining equation for α_2 in (8.21). It involves β_1^{-1}. While matrix $\beta_1=b_1$ is tridiagonal, its inverse β_1^{-1} is full. Hence α_2 is also a full matrix. Then by the last equation in (8.21), all subsequent β_k's are also full matrices. But then finding the inverse of each β_k in (8.21)–(8.23) would require $O(M^3)$ operations, and this would have to be repeated $O(L)$ times. Thus, the total operation count in this naive approach is $O(M^3L)$, which renders the approach computationally inefficient.

will concentrate on the derivation of a scheme, alternative to (15.17), that has property (i). We postpone the discussion of implementation of that scheme, as well as demonstration of the unconditional stability of such implementations, until the next section.

Since we want to obtain a scheme that reduces to the Crank-Nicolson method for the one-dimensional Heat equation, it is natural to start with its naive 2D generalization, scheme (15.17). Now, note the following: When applied to the solution of the discretized equation, *operators* $\frac{1}{\kappa}\delta_t$, $\frac{1}{h^2}\delta_x^2$, and $\frac{1}{h^2}\delta_y^2$ (see (15.9)) produce quantities of order $O(1)$ (that is, not $O(\kappa)$, $O(\kappa^{-1})$, or anything else):

$$\frac{\delta_x^2}{h^2}\,U_{ml}^n = O(1), \qquad \frac{\delta_t}{\kappa}\,U_{ml}^n = O(1), \qquad \frac{\delta_t}{\kappa}\,\frac{\delta_x^2}{h^2}\,U_{ml}^n = O(1), \qquad \text{etc.} \tag{15.27}$$

Before we proceed with the derivation, we will pause and make a number of comments about handling operators in equations. Note that the operators mentioned before (15.27) are simply the discrete analogues of the continuous operators $\partial/\partial t$, $\partial^2/\partial x^2$, and $\partial^2/\partial y^2$, respectively. In the discrete case, the latter two operators become *matrices*; for example, *in the one-dimensional case*, operator δ_x^2 coincides with matrix A in (13.10)[28]. Therefore, when reading about, or writing yourself, formulae involving operators (which you will have to do extensively in the remainder of this Lecture), think of the latter as matrices. From this simple observation there follows an important practical conclusion: **If a formula involves a product of two operators, the order of the operators in the product must not be arbitrarily changed, because different operators, in general, do not commute.** This is completely analogous to the fact that for two matrices A and B,

$$AB \neq BA \quad \text{in general.}$$

(But, of course,

$$A + B = B + A,$$

and the same is true about any two operators.)

We conclude this detour about operator notations with two remarks.

Remark 1 One can show that operators δ_x^2 and δ_y^2 actually *do* commute, as do their continuous prototypes. However, we will *not* use this fact in our derivation, so that the latter remains valid for more general operators that do not necessarily commute.

Remark 2 Any two operators, which are (arbitrary) functions of *the same primordial operator*, commute. That is, if $\mathcal{O}$ is any operator and $f(\cdot)$ and $g(\cdot)$ are any two functions, then

$$f(\mathcal{O})\,g(\mathcal{O}) \;=\; g(\mathcal{O})\,f(\mathcal{O})\,. \tag{15.28}$$

For example,

$$\left(a + b\delta_x^2\right)\,\left(c + d\delta_x^2\right) \;=\; \left(c + d\delta_x^2\right)\,\left(a + b\delta_x^2\right) \tag{15.29}$$

for any scalars a, b, c, d.

We now return to the derivation of a suitable modification of (15.17). From (15.27) it follows that

$$\frac{\delta_x^2}{h^2}\,\frac{\delta_y^2}{h^2}\,\frac{\delta_t}{\kappa}\,U_{ml}^n = O(1), \quad \text{and so, for instance,} \quad \frac{\kappa^2}{4}\,\frac{\delta_x^2}{h^2}\,\frac{\delta_y^2}{h^2}\,\frac{U_{ml}^{n+1} - U_{ml}^n}{\kappa} = O(\kappa^2)\,. \tag{15.30}$$

[28] In the two-dimensional case, the matrices for δ_x^2 and δ_y^2 are more complicated and depend on the order in which the grid points are arranged into the vector $\vec{\mathbf{U}}$. Fortunately for us, we will *not* require the corresponding explicit forms of δ_x^2 and δ_y^2.

The accuracy of scheme (15.17) is $O(\kappa^2 + h^2)$, and therefore we can add to it any term of the same order without changing the accuracy of the scheme. Let us use this observation and add the term appearing on the l.h.s. of the second equation in (15.30) to the l.h.s. of scheme (15.17), whose both sides are divided by κ. The result is:

$$\frac{U_{ml}^{n+1} - U_{ml}^n}{\kappa} + \frac{\kappa^2}{4}\frac{\delta_x^2}{h^2}\frac{\delta_y^2}{h^2}\frac{U_{ml}^{n+1} - U_{ml}^n}{\kappa} = \frac{1}{2h^2}\left(\delta_x^2 + \delta_y^2\right)\left(U_{ml}^{n+1} + U_{ml}^n\right) . \tag{15.31}$$

Note that scheme (15.31) still has the accuracy $O(\kappa^2 + h^2)$.

Next, we rewrite the last equation in the equivalent form:

$$\left(1 - \frac{r}{2}\delta_x^2 - \frac{r}{2}\delta_y^2 + \frac{r}{2}\delta_x^2\frac{r}{2}\delta_y^2\right) U_{ml}^{n+1} = \left(1 + \frac{r}{2}\delta_x^2 + \frac{r}{2}\delta_y^2 + \frac{r}{2}\delta_x^2\frac{r}{2}\delta_y^2\right) U_{ml}^n . \tag{15.32}$$

The operator expressions on both sides of the above equation can be factored, resulting in

$$\left(1 - \frac{r}{2}\delta_x^2\right)\left(1 - \frac{r}{2}\delta_y^2\right) U_{ml}^{n+1} = \left(1 + \frac{r}{2}\delta_x^2\right)\left(1 + \frac{r}{2}\delta_y^2\right) U_{ml}^n . \tag{15.33}$$

Note that when factoring the operator expressions, we did *not* change the order of operators in their product.

Scheme (15.33) is the main result of this section. In the next section, we will show how this scheme can be implemented in a time-efficient manner. The methods that do so are called the *Alternating Direction Implicit (ADI)* methods. Here we preview the basic idea common to all of them. Namely, the computations are split in 2 (for the 2D case, and 3, for the 3D case) steps. In the first step, one applies an implicit method in the x-direction and an explicit method in the y-direction, producing an intermediate solution. The operations count for this step is as follows: One needs to solve $(L-1)$ tridiagonal $(M-1)\times(M-1)$ systems; this can be done with $O(ML)$ operations. In the second step, one applies an implicit method in the y-direction and an explicit method in the x-direction, which can also be implemented with $O(ML)$ operations. Hence the total operations count is also $O(ML)$.

15.5 Alternating Direction Implicit methods

Peaceman–Rachford method

For this ADI method, the two steps mentioned at the end of the previous section are implemented as follows:

$$\begin{aligned} &\text{(a)}: \quad \left(1 - \frac{r}{2}\delta_x^2\right) \overset{*}{U}_{ml} = \left(1 + \frac{r}{2}\delta_y^2\right) U_{ml}^n , \\ &\text{(b)}: \quad \left(1 - \frac{r}{2}\delta_y^2\right) U_{ml}^{n+1} = \left(1 + \frac{r}{2}\delta_x^2\right) \overset{*}{U}_{ml} . \end{aligned} \tag{15.34}$$

Let us first show that this method is equivalent to (15.33). This will imply that it satisfies property (i) of the "dream scheme" conditions (15.26). Indeed, let us apply the operator

$$\left(1 - \frac{r}{2}\delta_x^2\right)$$

to both sides of (15.34b). Then we obtain the following sequence of equations:

$$\begin{aligned}
\left(1-\frac{r}{2}\delta_x^2\right)\left(1-\frac{r}{2}\delta_y^2\right) U_{ml}^{n+1} &= \\
\left(1-\frac{r}{2}\delta_x^2\right)\left(1+\frac{r}{2}\delta_x^2\right) \overset{*}{U}_{ml} &\overset{(15.29)}{=} \\
\left(1+\frac{r}{2}\delta_x^2\right)\left(1-\frac{r}{2}\delta_x^2\right) \overset{*}{U}_{ml} &\overset{(15.34a)}{=} \\
\left(1+\frac{r}{2}\delta_x^2\right)\left(1+\frac{r}{2}\delta_y^2\right) U_{ml}^{n}, &
\end{aligned} \tag{15.35}$$

which proves that (15.34) is equivalent to (15.33).

It is easy to see that the Peaceman–Rachford method (15.34) possesses property (iii) of (15.26), i.e. is computationally efficient. Indeed, in order to compute each of the $L-1$ sub-vectors

$$\vec{\overset{*}{\mathbf{U}}}_{;l} = \left[\overset{*}{U}_{1,l},\ \overset{*}{U}_{2,l}\ \ldots,\ \overset{*}{U}_{M-2,l},\ \overset{*}{U}_{M-1,l}\right]^T, \tag{15.36}$$

of the intermediate solution $\overset{*}{U}_{ml}$, one needs to solve a tridiagonal $(M-1)\times(M-1)$ system given by Eq. (15.34a) for each l. Thus, the step described by (15.34a) requires $O(ML)$ operations. Specifically, for $l = 2, \ldots, L-2$ (i.e. away from the boundaries), such a system has the form

$$\left(1-\frac{r}{2}\delta_x^2\right)\vec{\overset{*}{\mathbf{U}}}_{;l} = \vec{\mathbf{U}}_{;l}^n + \frac{r}{2}\left[\vec{\mathbf{U}}_{;l+1}^n - 2\vec{\mathbf{U}}_{;l}^n + \vec{\mathbf{U}}_{;l-1}^n\right], \qquad 2 \le l \le L-2, \tag{15.37}$$

where $\vec{\mathbf{U}}_{;l}^n$ is defined in (15.20). The counterpart of (15.37) for the boundary rows (with $l=1$ and $l=L-1$) will be given in the next section. Continuing, the operator δ_x^2 in (15.37) is an $(M-1)\times(M-1)$ tridiagonal matrix, whose specific form depends on the boundary conditions and will be discussed in the next section. Note that the operator δ_y^2 on the r.h.s. of (15.34a) is *not* a matrix. Indeed, if it were a matrix, it would have been $(L-1)\times(L-1)$, because the discretization along the y-direction contains $L-1$ inner (i.e. non-boundary) points. However, it would then have been impossible to multiply such a matrix with the $(M-1)$-component vectors $\vec{\mathbf{U}}_{;l}^n$. Therefore, *in (15.34a)*, δ_y^2 is interpreted not as a matrix but as the operation of addition and subtraction of vectors $\vec{\mathbf{U}}_{;l}^n$, as shown on the r.h.s. of (15.37).

Similarly, after all components of the intermediate solution have been determined, it remains to solve $(M-1)$ equations (15.34b) for the unknown vectors

$$\vec{\mathbf{U}}_{m;} = [U_{m,1},\ U_{m,2}\ \ldots,\ U_{m,L-2},\ U_{m,L-1}]^T, \qquad m = 1, \ldots, M-1. \tag{15.38}$$

Each of these equations is an $(L-1)\times(L-1)$ tridiagonal system of the form

$$\left(1-\frac{r}{2}\delta_y^2\right)\vec{\mathbf{U}}_{m;}^{n+1} = \vec{\overset{*}{\mathbf{U}}}_{m;} + \frac{r}{2}\left[\vec{\overset{*}{\mathbf{U}}}_{m+1;} - 2\vec{\overset{*}{\mathbf{U}}}_{m;} + \vec{\overset{*}{\mathbf{U}}}_{m-1;}\right], \qquad 1 \le m \le M-1, \tag{15.39}$$

where $\vec{\overset{*}{\mathbf{U}}}_{m;}$ are defined similarly to $\vec{\mathbf{U}}_{m;}$. Note that now the interpretations of operators δ_x^2 and δ_y^2 have interchanged. Namely, the δ_y^2 on the l.h.s. of (15.39) is an $(L-1)\times(L-1)$ matrix, while the δ_x^2 has to be interpreted as an operation of addition and subtraction of $(L-1)$-component vectors $\vec{\overset{*}{\mathbf{U}}}_{m;}$. The solution of $M-1$ tridiagonal systems (15.39), and hence the implementation of step (15.34b), requires $O(ML)$ operations, and thus the total operations count for the Peaceman–Rachford method is $O(ML)$.

Finally, it remains to show that the Peaceman–Rachford method is unconditionally stable, i.e. has property (ii) of (15.26). This can be done as follows. Equations (15.34) have constant (in x and y) coefficients and hence their solution can be sought in the form:

$$U^n_{ml} = \rho^n\, e^{i\beta mh}\, e^{i\gamma lh}\,, \qquad \overset{*}{U}_{ml} = \overset{*}{\rho}\, \rho^n\, e^{i\beta mh}\, e^{i\gamma lh}\,. \tag{15.40}$$

Substituting (15.40) into (15.34) and using (15.12) and (15.13), one obtains:

$$\begin{aligned} \overset{*}{\rho} &= \frac{1-Y}{1+X}\,, \\ \rho &= \overset{*}{\rho}\cdot\frac{1-X}{1+Y} = \frac{1-X}{1+X}\cdot\frac{1-Y}{1+Y}\,, \end{aligned} \tag{15.41}$$

where we have introduced two more shorthand notations:

$$X = 2r\sin^2\frac{\beta h}{2}\,, \qquad Y = 2r\sin^2\frac{\gamma h}{2}\,. \tag{15.42}$$

From the second of Eqs. (15.41) it follows that $|\rho| \le 1$ for all harmonics (i.e., for all β and γ), because

$$\left|\frac{1-X}{1+X}\right| \le 1 \qquad \text{for all } X \ge 0. \tag{15.43}$$

This shows that the Peaceman–Rachford method for the 2D Heat equation is unconditionally stable. Altogether, the above has shown that this method satisfies all the three conditions (15.26) of a "dream scheme".

A *drawback* of the Peaceman–Rachford method is that its generalization to 3 spatial dimensions is no longer unconditionally stable. Below we provide a sketch of proof of this statement.

For the 3D Heat equation

$$u_t = u_{xx} + u_{yy} + u_{zz}, \tag{15.44}$$

the generalization of the Peaceman–Rachford method is:

$$\begin{aligned} &\text{(a)}: \quad \left(1-\frac{r}{3}\delta_x^2\right)\overset{*}{U}_{mlj} = \left(1+\frac{r}{3}\delta_y^2+\frac{r}{3}\delta_z^2\right)U^n_{mlj}\,, \\ &\text{(b)}: \quad \left(1-\frac{r}{3}\delta_y^2\right)\overset{**}{U}_{mlj} = \left(1+\frac{r}{3}\delta_x^2+\frac{r}{3}\delta_z^2\right)\overset{*}{U}_{mlj}\,, \\ &\text{(c)}: \quad \left(1-\frac{r}{3}\delta_z^2\right)U^{n+1}_{mlj} = \left(1+\frac{r}{3}\delta_x^2+\frac{r}{3}\delta_y^2\right)\overset{**}{U}_{mlj}\,, \end{aligned} \tag{15.45}$$

where δ_z^2 is defined similarly to δ_x^2 and δ_y^2. Substituting into (15.45) the *ansätze*

$$U^n_{mlj} = \rho^n\, e^{i\beta mh}\, e^{i\gamma lh}\, e^{i\xi jh}\,, \qquad \overset{*}{U}_{mlj} = \overset{*}{\rho}\, \rho^n\, e^{i\beta mh}\, e^{i\gamma lh}\, e^{i\xi jh}\,, \qquad \overset{**}{U}_{mlj} = \overset{**}{\rho}\, \rho^n\, e^{i\beta mh}\, e^{i\gamma lh}\, e^{i\xi jh}\,, \tag{15.46}$$

one obtains, similarly to (15.41):

$$\rho = \frac{\left(1-\frac{2}{3}(Y+Z)\right)\left(1-\frac{2}{3}(X+Z)\right)\left(1-\frac{2}{3}(X+Y)\right)}{\left(1+\frac{2}{3}X\right)\left(1+\frac{2}{3}Y\right)\left(1+\frac{2}{3}Z\right)}\,, \tag{15.47}$$

where X and Y have been defined in (15.42) and Z is defined similarly. The amplification factor (15.47) is not always less than 1 in magnitude. For example, when X, Y, and Z are all large numbers (and hence r is large), the value of the amplification factor is $\rho \approx -8$ (you will

be asked to verify this in a QSA), and hence the 3D Peaceman–Rachford method (15.45) is *not* unconditionally stable.

An alternative ADI method that has an unconditionally stable generalization to 3 spatial dimensions is described next.

Douglas method, a.k.a. Douglas–Gunn method[29]

The equations of this method are:

$$\begin{aligned} &(\text{a}): \quad \left(1-\frac{r}{2}\delta_x^2\right)\overset{*}{U}_{ml} = \left(1+\frac{r}{2}\delta_x^2 + r\delta_y^2\right)U_{ml}^n\,, \\ &(\text{b}): \quad \left(1-\frac{r}{2}\delta_y^2\right)U_{ml}^{n+1} = \overset{*}{U}_{ml} - \frac{r}{2}\delta_y^2\, U_{ml}^n\,. \end{aligned} \tag{15.48}$$

Let us now demonstrate that all the three properties (15.26) hold for the Douglas method.

To demonstrate property (i), it is sufficient to show that (15.48) is equivalent to scheme (15.33). One can do so following the idea(s) of (15.35); you will be asked to provide the details in a homework problem.

To demonstrate property (ii), one proceeds similarly to the lines of (15.40) and (15.41). Namely, substituting (15.40) into (15.48) and using Eqs. (15.12), (15.13), and (15.42), one finds:

$$\begin{aligned} &\overset{*}{\rho} = \frac{1-X-2Y}{1+X}\,, \\ &\rho = \frac{\overset{*}{\rho}+Y}{1+Y} = \frac{1-X}{1+X}\cdot\frac{1-Y}{1+Y}\,. \end{aligned} \tag{15.49}$$

Thus, the amplification factor for the Douglas method in 2D is the same as that factor of the Peaceman–Rachford method, and hence the Douglas method is unconditionally stable in 2D.

Finally, property (iii) for the Douglas method is established in complete analogy with how that was done for the Peaceman–Rachford method (see the text around Eqs. (15.36)–(15.39)).

Let us now show that the generalization of the Douglas method to 3D is also unconditionally stable. The corresponding equations have the form:

$$\begin{aligned} &(\text{a}): \quad \left(1-\frac{r}{2}\delta_x^2\right)\overset{*}{U}_{mlj} = \left(1+\frac{r}{2}\delta_x^2 + r\delta_y^2 + r\delta_z^2\right)U_{mlj}^n\,, \\ &(\text{b}): \quad \left(1-\frac{r}{2}\delta_y^2\right)\overset{**}{U}_{mlj} = \overset{*}{U}_{mlj} - \frac{r}{2}\delta_y^2\, U_{mlj}^n\,, \\ &(\text{c}): \quad \left(1-\frac{r}{2}\delta_z^2\right)U_{mlj}^{n+1} = \overset{**}{U}_{mlj} - \frac{r}{2}\delta_z^2\, U_{mlj}^n\,,\ . \end{aligned} \tag{15.50}$$

Using the von Neumann analysis, one can show that amplification factor for (15.50) is

$$\rho = 1 - \frac{2(X+Y+Z)}{(1+X)(1+Y)(1+Z)}, \tag{15.51}$$

so that, clearly, $\rho \le 1$. Using techniques from multivariable Calculus, it is easy to show that also $\rho \ge -1$, and hence the 3D Douglas method is unconditionally stable.

[29] This method was proposed by J. Douglas for the two- and three-dimensional Heat equation in ["On the numerical integration of $u_{xx}+u_{yy}=u_t$ by implicit methods," J. Soc. Indust. Appl. Math. **3** 42–65 (1955)] and in ["Alternating direction methods for three space variables," Numerische Mathematik **4** 41–63 (1962)]. A general form of such methods was discussed by J. Douglas and J. Gunn in ["A general formulation of alternating direction methods, I. Parabolic and hyperbolic problems," Numerische Mathematik **6** 428–453 (1964)].

To conclude this subsection, we mention two more methods for the 2D Heat equation.

D'yakonov method

The equations of this method are

$$\begin{aligned} &(\mathrm{a}): \quad \left(1-\frac{r}{2}\delta_x^2\right)\overset{*}{U}_{ml} = \left(1+\frac{r}{2}\delta_x^2\right)\left(1+\frac{r}{2}\delta_y^2\right)U_{ml}^n\,, \\ &(\mathrm{b}): \quad \left(1-\frac{r}{2}\delta_y^2\right)U_{ml}^{n+1} = \overset{*}{U}_{ml}\ . \end{aligned} \tag{15.52}$$

One can show, similarly to how that was done for the Peaceman–Rachford and Douglas methods, that the D'yakonov method possesses all the three properties (15.26).

Fairweather-Mitchell scheme

This scheme is

$$\begin{gathered} \left(1-\theta_0\delta_x^2\right)\left(1-\theta_0\delta_y^2\right)U_{ml}^{n+1} = \left(1+(1-\theta_0)\delta_x^2\right)\left(1+(1-\theta_0)\delta_y^2\right)U_{ml}^n\,, \\ \theta_0 = \frac{r}{2}-\frac{1}{12}\,. \end{gathered} \tag{15.53}$$

This scheme improves scheme (15.33) in the same manner in which the Crandall method "(13.17)+(13.19)" for the 1D Heat equations improves the Crank-Nicolson method. Consequently, its accuracy is $O(\kappa^2+h^4)$, and the scheme is stable. As far as implementing this scheme in a time-efficient manner, this can be done straighforwardly by using suitable modifications of the Peaceman–Rachford or D'yakonov methods.

Generalizations

In Appendix 1 we will present two important generalizations.

First, we will take an another look at the Douglas method (15.48) and thereby observe its relation to the predictor-corrector methods considered in Lecture 3 and to the IMEX methods mentioned in Lecture 14.

Second, we will show how one can construct an *unconditionally stable* method whose global error is of the order $O(\kappa^2+h^2)$, for a parabolic-type equation with a *mixed derivative* term, e.g.:

$$u_t = a^{(xx)}u_{xx} + \underline{a^{(xy)}u_{xy}} + a^{(yy)}u_{yy}\,; \tag{15.54}$$

here $a^{(xx)}$ etc. are coefficients, and the term with the mixed derivatives is underlined. Our construction will utilize the first generalization considered in Appendix 1. It is worth pointing out that construction of a scheme with aforementioned properties for (15.54) was not a trivial problem. This is attested by the fact that it was solved more than 30 years after the pioneering works by Peaceman, Rachford, Douglas, and others on the Heat equation (15.1). The paper[30] where this problem was solved, is posted on the course website.

A good reference on finite difference methods in two and three spatial dimensions is a book by A.R. Mitchell and G.F. Griffiths, "The Finite Difference Method in Partial Differential Equations" (Wiley, 1980).

[30] I.J.D. Craig and A.D. Sneyd, "An alternating-direction implicit scheme for parabolic equations with mixed derivatives," Computers and Mathematics with Applications **16**(4), 341–350 (1988).

15.6 Boundary conditions for the ADI methods

Here we will show how to prescribe boundary conditions for the intermediate solution $\overset{*}{U}_{ml}$ appearing in the ADI methods considered above. We will do so for the Dirichlet boundary conditions (15.3) and (15.4) and for the Neumann boundary conditions

$$u_x(0,y,t) = g_0(y,t), \quad u_x(1,y,t) = g_1(y,t), \qquad 0 \le y \le Y, \quad t \ge 0\,; \tag{15.55}$$

$$u_y(x,0,t) = g_2(x,t), \quad u_y(x,Y,t) = g_3(x,t), \qquad 0 \le x \le 1, \quad t \ge 0\,. \tag{15.56}$$

The corresponding generalizations for the mixed boundary conditions (14.3) can be obtained straightforwardly. Note that the counterpart of the matching conditions (15.5) between the boundary conditions on one hand and the initial condition one the other, for the Neumann boundary conditions has the form:

$$\begin{aligned} g_0(y,0) &= (u_0)_x(0,y), \quad g_1(y,0) = (u_0)_x(1,y), \\ g_2(x,0) &= (u_0)_y(x,0), \quad g_3(x,0) = (u_0)_y(x,Y)\,. \end{aligned} \tag{15.57}$$

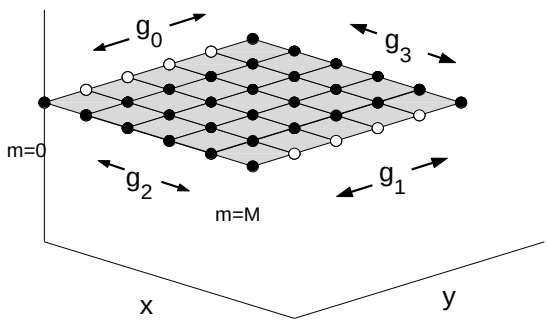
The counterpart of the requirement (15.6) that the boundary conditions match at the corners of the domain follows from the relation $u_{xy}(x,y) = u_{yx}(x,y)$ and has the form:

$$\begin{aligned} (g_0)_y(0,t) &= (g_2)_x(0,t), \quad (g_0)_y(Y,t) = (g_3)_x(0,t), \\ (g_3)_x(1,t) &= (g_1)_y(Y,t), \quad (g_1)_y(0,t) = (g_2)_x(1,t), \quad \text{for } t > 0. \end{aligned} \tag{15.58}$$

Peaceman–Rachford method

Dirichlet boundary conditions

Note that in order to solve Eq. (15.34a) for the $\overset{*}{U}_{ml}$ with $1 \le \{m,l\} \le \{(M-1),(L-1)\}$, one requires the values of $\overset{*}{U}_{0,l}$ and $\overset{*}{U}_{M,l}$ with $1 \le l \le L-1$. The corresponding nodes are shown as open circles in the figure on the right. Note that one does *not* need the other boundary values, $\overset{*}{U}_{m,0}$ and $\overset{*}{U}_{m,L}$, to solve (15.34b), because the l.h.s. of the latter equation is only defined for $1 \le l \le L-1$. Hence, one does not need (and cannot determine) the values $\overset{*}{U}_{m,0}$ and $\overset{*}{U}_{m,L}$ in the Peaceman–Rachford method.



Thus, how does one find the required boundary values $\overset{*}{U}_{0,l}$ and $\overset{*}{U}_{M,l}$? To answer this question, note that the term in (15.34a) that produces $\overset{*}{U}_{0,l}$ and $\overset{*}{U}_{M,l}$ is: $\frac{r}{2}\delta_x^2\,\overset{*}{U}_{ml}$ (with $m=1$ and $m=M-1$). Let us then eliminate this term using *both* Eqs. (15.34). The most convenient way to do so is to rewrite these equations in an equivalent form:

$$\begin{aligned} \left(1 - \frac{r}{2}\delta_x^2\right)\overset{*}{U}_{ml} &= \left(1 + \frac{r}{2}\delta_y^2\right)U_{ml}^n\,, \\ \left(1 + \frac{r}{2}\delta_x^2\right)\overset{*}{U}_{ml} &= \left(1 - \frac{r}{2}\delta_y^2\right)U_{ml}^{n+1}\,, \end{aligned}$$

and then add them. The result is:

$$\overset{*}{U}_{ml} = \frac{1}{2}\left(U^n_{ml} + U^{n+1}_{ml}\right) + \frac{r}{4}\,\delta^2_y\,\left(U^n_{ml} - U^{n+1}_{ml}\right)\,. \tag{15.59}$$

Now, this equation, unlike (15.34a), *can* be evaluated for $m = 0$ and $m = M$, yielding

$$\begin{aligned}
\overset{*}{U}_{\{0,M\},l} &= \frac{1}{2}\left(U^n_{\{0,M\},l} + U^{n+1}_{\{0,M\},l}\right) + \frac{r}{4}\,\delta^2_y\,\left(U^n_{\{0,M\},l} - U^{n+1}_{\{0,M\},l}\right) \\
&= \frac{1}{2}\left((g_{\{0,1\}})^n_l + (g_{\{0,1\}})^{n+1}_l\right) + \frac{r}{4}\,\delta^2_y\,\left((g_{\{0,1\}})^n_l - (g_{\{0,1\}})^{n+1}_l\right) \\
&= \frac{1}{2}\left((g_{\{0,1\}})^n_l + (g_{\{0,1\}})^{n+1}_l\right) + \frac{r}{4}\,\cdot \\
&\qquad \left(\left[(g_{\{0,1\}})^n_{l+1} - 2(g_{\{0,1\}})^n_l + (g_{\{0,1\}})^n_{l-1}\right] - \left[(g_{\{0,1\}})^{n+1}_{l+1} - 2(g_{\{0,1\}})^{n+1}_l + (g_{\{0,1\}})^{n+1}_{l-1}\right]\right) \\
&\equiv (\mathcal{G}_{\{0,1\}})_l\,, \\
&\quad 1 \le l \le L-1\,.
\end{aligned} \tag{15.60}$$

It is now time to complete the discussion about the implementations of operators $\frac{r}{2}\delta^2_x$ and $\frac{r}{2}\delta^2_y$ in each of the equations (15.34). Recall that we started this discussion after Eq. (15.36), but were unable to complete it then because we did not have the information about boundary conditions. Let us begin with Eq. (15.34a). There, operator $\frac{r}{2}\delta^2_x$ is the following $(M-1)\times(M-1)$ matrix:

<u>On the r.h.s. of (15.34a):</u>

$$\frac{r}{2}\delta^2_x = \begin{pmatrix} -r & \frac{r}{2} & 0 & \cdot & \cdot & 0 \\ \frac{r}{2} & -r & \frac{r}{2} & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & \frac{r}{2} & -r & \frac{r}{2} \\ 0 & \cdot & \cdot & 0 & \frac{r}{2} & -r \end{pmatrix}, \tag{15.61}$$

which has been obtained in analogy with matrix A in (13.10). Operator δ^2_y should be interpreted not as a matrix but as an operation of adding and subtracting $(M-1)$-component vectors, as was shown on the r.h.s. of (15.37). Below we present the generalization of (15.37) for the boundary rows $l = 1$ and $l = L-1$:

<u>On the r.h.s. of (15.34a):</u>

$$\begin{aligned}
&\left(1 + \tfrac{r}{2}\delta^2_y\right)\,\vec{\mathbf{U}}^n_{;l} = \vec{\mathbf{U}}^n_{;l} + \tfrac{r}{2}\left[\vec{\mathbf{U}}^n_{;l+1} - 2\vec{\mathbf{U}}^n_{;l} + \vec{\mathbf{U}}^n_{;l-1}\right], \qquad 2 \le l \le L-2\,; \\
&\left(1 + \tfrac{r}{2}\delta^2_y\right)\,\vec{\mathbf{U}}^n_{;1} = \vec{\mathbf{U}}^n_{;1} + \tfrac{r}{2}\left[\vec{\mathbf{U}}^n_{;2} - 2\vec{\mathbf{U}}^n_{;1} + \vec{\mathbf{B}}^n_2\right], \\
&\left(1 + \tfrac{r}{2}\delta^2_y\right)\,\vec{\mathbf{U}}^n_{;L-1} = \vec{\mathbf{U}}^n_{;L-1} + \tfrac{r}{2}\left[\vec{\mathbf{B}}^n_3 - 2\vec{\mathbf{U}}^n_{;L-1} + \vec{\mathbf{U}}^n_{;L-2}\right],
\end{aligned} \tag{15.62}$$

where $\vec{\mathbf{B}}_2$ and $\vec{\mathbf{B}}_3$ have been defined in (15.21). Note that the r.h.s. of (15.34a) also contains terms contributed by $\overset{*}{U}_{\{0,M\},l}$, as we will explicitly show shortly.

In Eq. (15.34b), as has been mentioned earlier, the interpretations of δ^2_x and δ^2_y are reversed. Namely, now $\frac{r}{2}\delta^2_y$ has the form given by the r.h.s. of (15.61); the dimension of this matrix is $(L-1)\times(L-1)$. Operator δ^2_x is implemented as was shown on the r.h.s. of (15.39); below we show its form for completeness of the presentation:

<u>On the r.h.s. of (15.34b):</u>

$$\left(1 + \tfrac{r}{2}\delta^2_x\right)\,\vec{\overset{*}{\mathbf{U}}}_{m;} = \vec{\overset{*}{\mathbf{U}}}_{m;} + \frac{r}{2}\left[\vec{\overset{*}{\mathbf{U}}}_{m+1;} - 2\vec{\overset{*}{\mathbf{U}}}_{m;} + \vec{\overset{*}{\mathbf{U}}}_{m-1;}\right]. \qquad 1 \le m \le M-1\,; \tag{15.63}$$

Let us now summarize the above steps in the form of an algorithm.

Algorithm of solving the 2D Heat equation with Dirichlet boundary conditions by the Peaceman–Rachford method:

The following steps need to be performed inside the loop over n (i.e., advancing in time). Suppose that the solution has been computed at the nth time level.

Step 1 (set up boundary conditions):
Define the boundary conditions at the $(n+1)$th time level:

$$U^{(n+1)}_{\{0,M\},l} = \left(g_{\{0,1\}}\right)^{(n+1)}_l, \quad 0 \le l \le L; \qquad U^{(n+1)}_{m,\{0,L\}} = \left(g_{\{2,3\}}\right)^{(n+1)}_m, \quad 1 \le m \le M-1\,. \tag{15.64}$$

(Recall that the boundary conditions match at the corners; see (15.6).)

Next, determine the necessary boundary values of the intermediate solution:

$$\overset{*}{U}_{\{0,M\},l} = (\mathcal{G}_{\{0,1\}})_l\,, \qquad 1 \le l \le L-1\,, \tag{15.65}$$

where $(\mathcal{G}_{\{0,1\}})_l$ are defined in (15.60).

Step 2:
For each $l = 1, \ldots, L-1$, solve the tridiagonal system

$$\left(1 - \frac{r}{2}\delta_x^2\right)\vec{\overset{*}{\mathbf{U}}}_{;l} = \vec{\mathbf{U}}^n_{;l} + \frac{r}{2}\left[\vec{\mathbf{U}}^n_{;l+1} - 2\vec{\mathbf{U}}^n_{;l} + \vec{\mathbf{U}}^n_{;l-1}\right] + \frac{r}{2}\vec{\overset{*}{\mathbf{b}}}_{;l}\,, \qquad 1 \le l \le L-1, \tag{15.66}$$

where $\frac{r}{2}\delta_x^2$ is an $(M-1)\times(M-1)$ matrix of the form (15.61),

$$\vec{\overset{*}{\mathbf{U}}}_{;l} = \begin{pmatrix} \overset{*}{U}_{1,l} \\ \overset{*}{U}_{2,l} \\ \cdot \\ \overset{*}{U}_{M-2,l} \\ \overset{*}{U}_{M-1,l} \end{pmatrix}, \quad \vec{\mathbf{U}}^n_{;l} = \begin{pmatrix} U^n_{1,l} \\ U^n_{2,l} \\ \cdot \\ U^n_{M-2,l} \\ U^n_{M-1,l} \end{pmatrix}, \quad \vec{\overset{*}{\mathbf{b}}}_{;l} = \begin{pmatrix} (\mathcal{G}_0)_l \\ 0 \\ \cdot \\ 0 \\ (\mathcal{G}_1)_l \end{pmatrix}. \qquad \begin{pmatrix} \text{see} \\ (15.20) \\ \text{and} \\ (15.36) \end{pmatrix}$$

Note that $\vec{\mathbf{U}}^n_{;0} \equiv \vec{\mathbf{B}}^n_2$ and $\vec{\mathbf{U}}^n_{;L} \equiv \vec{\mathbf{B}}^n_3$ are determined from the boundary conditions on the nth time level.

Thus, combining the results of (15.64), (15.65), and (15.66), one has the following values of the intermediate solution:

$$\overset{*}{U}_{m,l} \qquad \text{for} \quad 0 \le m \le M, \quad 1 \le l \le L-1\,.$$

Step 3:
The solution $U^{n+1}_{m,l}$ with $1 \le \{m,l\} \le \{M-1, L-1\}$ is then determined from

$$\left(1 - \frac{r}{2}\delta_y^2\right)\vec{\mathbf{U}}^{n+1}_{m;} = \vec{\overset{*}{\mathbf{U}}}_{m;} + \frac{r}{2}\left[\vec{\overset{*}{\mathbf{U}}}^{\,n}_{m+1\,;} - 2\vec{\overset{*}{\mathbf{U}}}_{m;} + \vec{\overset{*}{\mathbf{U}}}_{m-1\,;}\right] + \frac{r}{2}\vec{\mathbf{b}}^{n+1}_{m;}\,, \qquad 1 \le m \le M-1. \tag{15.67}$$

Here $\frac{r}{2}\delta_y^2$ is the $(L-1)\times(L-1)$ matrix of the form (15.61), and

$$\vec{\overset{*}{\mathbf{U}}}_{m;} = \begin{pmatrix} \overset{*}{U}_{m,1} \\ \overset{*}{U}_{m,2} \\ \cdot \\ \overset{*}{U}_{m,L-2} \\ \overset{*}{U}_{m,L-1} \end{pmatrix}, \quad \vec{\mathbf{U}}_{m;}^{n+1} = \begin{pmatrix} U_{m,1}^{n+1} \\ U_{m,2}^{n+1} \\ \cdot \\ U_{m,L-2}^{n+1} \\ U_{m,L-1}^{n+1} \end{pmatrix}, \quad \vec{\mathbf{b}}_{m;}^{n+1} = \begin{pmatrix} (g_2)_m^{n+1} \\ 0 \\ \cdot \\ 0 \\ (g_3)_m^{n+1} \end{pmatrix}. \tag{15.68}$$

This completes the process of advancing the solution by one step in time.

Neumann boundary conditions

This case is technically more involved than the case of Dirichlet boundary conditions. Therefore, here we only list the steps of the algorithm of advancing the solution from the nth to the $(n+1)$st time level, while relegating the detailed derivation of these steps to Appendix 2. Also, note that you will *not* need to use this algorithm in any of the homework problems. It is presented here so that you would be able to use it whenever you have to solve a problem of this kind in your future career.

Algorithm of solving the 2D Heat equation with Neumann boundary conditions by the Peaceman–Rachford method:

Step 1 (set up boundary conditions):
Given the solution $U_{m,l}^n$ with $0 \le \{m,l\} \le \{M,L\}$, find the values at the virtual nodes, $U_{-1,l}^n$ and $U_{M+1,l}^n$ with $-1 \le l \le L+1$ and $U_{m,-1}^n$ and $U_{m,L+1}^n$ with $0 \le m \le M$, from (15.94) and (15.95) of Appendix 2:

$$\begin{aligned} &U_{-1,l}^n = U_{1,l}^n - 2h(g_0)_l^n\,, \qquad U_{M+1,l}^n = U_{M-1,l}^n + 2h(g_1)_l^n\,, \quad 0 \le l \le L\,; \\ &U_{m,-1}^n = U_{m,1}^n - 2h(g_2)_m^n\,, \quad U_{m,L+1}^n = U_{m,L-1}^n + 2h(g_3)_m^n\,, \quad 0 \le m \le M\,; \end{aligned} \tag{15.94}$$

$$\begin{aligned} &U_{-1,-1}^n = U_{1,-1}^n - 2h(g_0)_{-1}^n\,, \qquad U_{M+1,-1}^n = U_{M-1,-1}^n + 2h(g_1)_{-1}^n\,, \\ &U_{-1,L+1}^n = U_{1,L+1}^n - 2h(g_0)_{L+1}^n\,, \quad U_{M+1,L+1}^n = U_{M-1,L+1}^n + 2h(g_1)_{L+1}^n\,. \end{aligned} \tag{15.95}$$

Define auxiliary functions given by (15.97) and (15.98) of Appendix 2, which will later be used to compute the boundary values of the intermediate solution $\overset{*}{U}$:

For $0 \le l \le L$:

$$(\mathcal{G}_0)_l = \frac{1}{2}\left((g_0)_l^n + (g_0)_l^{n+1}\right) + \frac{r}{4}\left(\left[(g_0)_{l+1}^n - 2(g_0)_l^n + (g_0)_{l-1}^n\right] - \left[(g_0)_{l+1}^{n+1} - 2(g_0)_l^{n+1} + (g_0)_{l-1}^{n+1}\right]\right), \tag{15.97}$$

$$(\mathcal{G}_1)_l = \frac{1}{2}\left((g_1)_l^n + (g_1)_l^{n+1}\right) + \frac{r}{4}\left(\left[(g_1)_{l+1}^n - 2(g_1)_l^n + (g_1)_{l-1}^n\right] - \left[(g_1)_{l+1}^{n+1} - 2(g_1)_l^{n+1} + (g_1)_{l-1}^{n+1}\right]\right), \tag{15.98}$$

(Note that the form of $\mathcal{G}_0$ and $\mathcal{G}_1$ above is the same as in the case of the Dirichlet boundary conditions — see (15.60), — although the meanings of g_0 and g_1 are different in these two cases.)

Step 2:
For each $0 \le l \le L$, solve the linear system, whose form follows from (15.92) and (15.99) of Appendix 2:

$$\left(1 - \frac{r}{2}\delta_x^2\right) \vec{\overset{*}{\mathbf{U}}}_{;\,l} = \vec{\mathbf{U}}^n_{;\,l} + \frac{r}{2}\left[\vec{\mathbf{U}}^n_{;\,l+1} - 2\vec{\mathbf{U}}^n_{;\,l} + \vec{\mathbf{U}}^n_{;\,l-1}\right] + \frac{r}{2}\vec{\overset{*}{\mathbf{b}}}_{;\,l}\,, \qquad 0 \le l \le L, \tag{15.69}$$

where $\frac{r}{2}\delta_x^2$ is an $(M+1)\times(M+1)$ matrix of the form

$$\begin{pmatrix} -r & \boxed{r} & 0 & \cdot & \cdot & 0 \\ \frac{r}{2} & -r & \frac{r}{2} & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & \frac{r}{2} & -r & \frac{r}{2} \\ 0 & \cdot & \cdot & 0 & \boxed{r} & -r \end{pmatrix}, \tag{15.70}$$

(here the terms that differ from the corresponding matrix for Dirichlet boundary conditions are included in the box),

$$\vec{\overset{*}{\mathbf{U}}}_{;\,l} = \begin{pmatrix} \overset{*}{U}_{0,l} \\ \overset{*}{U}_{1,l} \\ \cdot \\ \overset{*}{U}_{M-1,l} \\ \overset{*}{U}_{M,l} \end{pmatrix}, \quad \vec{\mathbf{U}}^n_{;\,l} = \begin{pmatrix} U^n_{0,l} \\ U^n_{1,l} \\ \cdot \\ U^n_{M-1,l} \\ U^n_{M,l} \end{pmatrix}, \quad \vec{\overset{*}{\mathbf{b}}}_{;\,l} = \begin{pmatrix} -2h(\mathcal{G}_0)_l \\ 0 \\ \cdot \\ 0 \\ 2h(\mathcal{G}_1)_l \end{pmatrix}, \tag{15.71}$$

and $\mathcal{G}_0$ and $\mathcal{G}_1$ are defined in (15.97) and (15.98) (see above and in Appendix 2).

Having thus determined the following values of the intermediate solution:

$$\overset{*}{U}_{m,l} \quad \text{for } 0 \le \{m,l\} \le \{M,L\},$$

find the values

$$\overset{*}{U}_{-1,l} \quad \text{and} \quad \overset{*}{U}_{M+1,l} \quad \text{for } 0 \le l \le L$$

from (15.97) and (15.98) of Appendix 2:

$$\overset{*}{U}_{-1,l} = \overset{*}{U}_{1,l} - 2h(\mathcal{G}_0)_l, \qquad 0 \le l \le L, \tag{15.97$'$}$$

$$\overset{*}{U}_{M+1,l} = \overset{*}{U}_{M-1,l} + 2h(\mathcal{G}_1)_l, \qquad 0 \le l \le L. \tag{15.98$'$}$$

Thus, upon completing Step 2, one has the following values of the intermediate solution

$$\overset{*}{U}_{m,l} \quad \text{for } -1 \le m \le M+1 \text{ and } 0 \le l \le L,$$

which are shown in Appendix 2 to be necessary and sufficient to find the solution on the $(n+1)$st time level.

Step 3:
The solution at the new time level, $U^{n+1}_{m,l}$ with $0 \le \{m,l\} \le \{M,L\}$, is determined from (15.85), (15.88), and (15.89) of Appendix 2, which constitute the following $(L+1)\times(L+1)$ linear systems for each of the $m = 0, \ldots, M$:

$$\left(1 - \frac{r}{2}\delta_y^2\right) \vec{\mathbf{U}}^{n+1}_{m;} = \vec{\overset{*}{\mathbf{U}}}_{m;} + \frac{r}{2}\left[\vec{\overset{*}{\mathbf{U}}}^{\,n}_{m+1\,;} - 2\vec{\overset{*}{\mathbf{U}}}_{m;} + \vec{\overset{*}{\mathbf{U}}}_{m-1\,;}\right] + \frac{r}{2}\vec{\mathbf{b}}^{n+1}_{m;}\,, \qquad 0 \le m \le M. \tag{15.72}$$

Here $\frac{r}{2}\delta_y^2$ is the $(L+1)\times(L+1)$ matrix of the form (15.70), and

$$\vec{\overset{*}{\mathbf{U}}}_{m;} = \begin{pmatrix} \overset{*}{U}_{m,0} \\ \overset{*}{U}_{m,1} \\ \cdot \\ \overset{*}{U}_{m,L-1} \\ \overset{*}{U}_{m,L} \end{pmatrix}, \quad \vec{\mathbf{U}}_{m;}^{n+1} = \begin{pmatrix} U_{m,0}^{n+1} \\ U_{m,1}^{n+1} \\ \cdot \\ U_{m,L-1}^{n+1} \\ U_{m,L}^{n+1} \end{pmatrix}, \quad \vec{\mathbf{b}}_{m;}^{n+1} = \begin{pmatrix} -2h(g_2)_m^{n+1} \\ 0 \\ \cdot \\ 0 \\ 2h(g_3)_m^{n+1} \end{pmatrix}. \tag{15.73}$$

This completes the process of advancing the solution by one step in time.

We conclude this subsection with the counterparts of Eq. (15.60) for the Douglas and D'yakonov methods. You will be asked to derive these results in a homework problem. We will *not* state any results for Neumann boundary conditions for the Douglas and D'yakonov methods.

Douglas method

The Dirichlet boundary conditions for the intermediate solution $\overset{*}{U}$ have the form:

$$\overset{*}{U}_{\{0,M\},l} = U_{\{0,M\},l}^{n+1} + \frac{r}{2}\delta_y^2\left(U_{\{0,M\},l}^{n} - U_{\{0,M\},l}^{n+1}\right), \qquad 1 \le l \le L-1\,. \tag{15.74}$$

D'yakonov method

The Dirichlet boundary conditions for the intermediate solution $\overset{*}{U}$ have the form:

$$\overset{*}{U}_{\{0,M\},l} = \left(1 - \frac{r}{2}\delta_y^2\right) U_{\{0,M\},l}^{n+1}\,, \qquad 1 \le l \le L-1\,. \tag{15.75}$$

15.7 Appendix 1: A generalized form of the ADI methods, and a second-order ADI method for the parabolic equation with mixed derivatives, Eq. (15.54)

A brief preview of this section was done at the end of Section 15.5. The presentation below is based on the papers by K.J. in 't Hout and B.D. Welfert, "Stability of ADI schemes applied to convection-diffusion equations with mixed derivative terms," Applied Numerical Mathematics **57**, 19–35 (2007) and by I.J.D. Craig and A.D. Sneyd, "An alternating-direction implicit scheme for parabolic equations with mixed derivatives," Computers and Mathematics with Applications **16**(4), 341–350 (1988). Both papers are posted on the course website.

Let us begin by writing a general form of the equation that included the Heat equation (15.1) as a special case:

$$u_t = F \equiv F^{(0)} + F^{(1)} + F^{(2)}\,, \tag{15.76}$$

where $F^{(1)}$ and $F^{(2)}$ are terms that contain only the derivatives of u with respect to x and y, respectively, and $F^{(0)}$ contains all other terms (e.g., nonlinear or with mixed derivatives). For example, in (15.54),

$$F^{(0)} = a^{(xy)}(x,y)u_{xy}, \qquad F^{(1)} = a^{(xx)}(x,y)u_{xx}, \qquad F^{(2)} = a^{(yy)}(x,y)u_{yy}\,.$$

Next, note that the Douglas method (15.48), which we repeat here for the reader's convenience:

$$\begin{aligned} \left(1-\frac{r}{2}\delta_x^2\right)\overset{*}{U}_{ml} &= \left(1+\frac{r}{2}\delta_x^2+r\delta_y^2\right)U_{ml}^n\,, \\ \left(1-\frac{r}{2}\delta_y^2\right)U_{ml}^{n+1} &= \overset{*}{U}_{ml}-\frac{r}{2}\delta_y^2\,U_{ml}^n\,, \end{aligned} \tag{15.48}$$

can be written in an equivalent, but different form:

$$\begin{aligned} W^{(0)} &= U^n + r\delta_x^2 U^n + r\delta_y^2 U^n, \\ W^{(1)} &= W^{(0)} + \frac{1}{2}(r\delta_x^2 W^{(1)} - r\delta_x^2 U^n), \\ W^{(2)} &= W^{(1)} + \frac{1}{2}(r\delta_y^2 W^{(2)} - r\delta_y^2 U^n), \\ U^{(n+1)} &= W^{(2)}\,. \end{aligned} \tag{15.77}$$

Here, for brevity, we have omitted the subscripts $\{m,l\}$ in $U_{m,l}^n$ etc. The correspondence of notations of (15.48) and (15.77) is:

$$W^{(1)}_{(15.77)} = \overset{*}{U}_{(15.48)}\ .$$

In the notations introduced in (15.76), this can be written as

$$\begin{aligned} W^{(0)} &= U^n + \kappa F(U^n), \\ W^{(k)} &= W^{(k-1)} + \frac{1}{2}\kappa\Big(F^{(k)}(W^{(k)}) - F^{(k)}(U^n)\Big), \qquad k=1,2; \\ U^{(n+1)} &= W^{(2)}\,. \end{aligned} \tag{15.78}$$

Recall that F used in the first equation above is defined in (15.76).

Let us make two observations about scheme (15.78). First, it can be interpreted as a predictor-corrector method, which we considered in Lecture 3. Indeed, the first equation in (15.78) predicts the value of the solution at the next time level by the simple Euler method. The purpose of each of the subsequent steps is to stabilize the predictor step by employing an implicit modified Euler step in one particular direction (i.e., along either x or y). Indeed, if we set $F^{(0)} = F^{(2)} = 0$ in (15.76), then (15.78) reduces to the implicit modified Euler method. You will be asked to verify this in a QSA.

Second, (15.78) is seen to be closely related to the IMEX family of methods; see scheme (14.51) in Lecture 14.

For $F^{(0)} \neq 0$, method (15.78) has accuracy $O(\kappa+h^2)$ (for $F^{(0)} = 0$, its accuracy is $O(\kappa^2+h^2)$, as we know from the discussion of the Douglas method in Section 15.5). It is of interest and of considerable practical significance to construct an extension of this scheme that would have accuracy $O(\kappa^2+h^2)$ even when $F^{(0)} \neq 0$. Two such schemes were presented in the paper by in 't Hout and Welfert, who generalized schemes presented earlier by other researchers. The first scheme is:

$$\begin{aligned} W^{(0)} &= U^n + \kappa F(U^n), \\ W^{(k)} &= W^{(k-1)} + \frac{1}{2}\kappa\Big(F^{(k)}(W^{(k)}) - F^{(k)}(U^n)\Big), \qquad k=1,2; \\ V^{(0)} &= W^{(0)} + \frac{1}{2}\kappa\Big(F^{(0)}(W^{(2)}) - F^{(0)}(U^n)\Big), \\ V^{(k)} &= V^{(k-1)} + \frac{1}{2}\kappa\Big(F^{(k)}(V^{(k)}) - F^{(k)}(U^n)\Big), \qquad k=1,2; \\ U^{(n+1)} &= V^{(2)}\,. \end{aligned} \tag{15.79}$$

After one round of prediction and correction, accomplished by the first two lines of this scheme, it proceeds to do another round of prediction and correction, given by the third and fourth lines. It appears that it is this second round that brings the accuracy of the scheme up to the order $O(\kappa^2)$. Intuitively, the reason why this is so can be understood by making an analogy of these two rounds with the two steps of the modified *explicit* Euler method. Specifically, in a QSA you will be asked to show that the scheme (15.79) with $F^{(1)} = F^{(2)} = 0$ reduces to the modified explicit Euler method.

The second scheme proposed by in 't Hout and Welfert is obtained from (15.79) by replacing $F^{(0)}$ in its third line by F.

Stability of the above schemes, as well as of the generalized Douglas scheme (15.78), has been investigated by in 't Hout and Welfert. In particular, they showed that schemes (15.78) and (15.79) are unconditionally stable for the so-called convection-diffusion equation

$$u_t = c^{(x)}u_x + c^{(y)}u_y + a^{(xx)}u_{xx} + a^{(xy)}u_{xy} + a^{(yy)}u_{yy}, \tag{15.80}$$

where all the coefficients may depend on x and y, and the quadratic form $a^{(xx)}x^2 + a^{(xy)}xy + a^{(yy)}y^2$ is positive definite. The third scheme, mentioned after (15.79), can also be made unconditionally stable upon replacing the coefficient 1/2 in the second and fourth lines by any number $\theta \geq 3/4$.

Below we will specify scheme (15.79) for the case of equation (15.80) with $c^{(x)} = c^{(y)} = 0$:

$$\begin{aligned}
W^{(0)} &= U^n + r\Big(a^{(xx)}\delta_x^2 + a^{(xy)}\delta_{xy} + a^{(yy)}\delta_y^2\Big)U^n, \\
\Big(1 - \frac{1}{2}ra^{(xx)}\delta_x^2\Big)W^{(1)} &= W^{(0)} - \frac{1}{2}ra^{(xx)}\delta_x^2 U^n, \\
\Big(1 - \frac{1}{2}ra^{(yy)}\delta_y^2\Big)W^{(2)} &= W^{(1)} - \frac{1}{2}ra^{(yy)}\delta_y^2 U^n, \\
V^{(0)} &= W^{(0)} + \frac{1}{2}r\Big(a^{(xy)}\delta_{xy}W^{(2)} - a^{(xy)}\delta_{xy}U^n\Big), \\
\Big(1 - \frac{1}{2}ra^{(xx)}\delta_x^2\Big)V^{(1)} &= V^{(0)} - \frac{1}{2}ra^{(xx)}\delta_x^2 U^n, \\
\Big(1 - \frac{1}{2}ra^{(yy)}\delta_y^2\Big)U^{(n+1)} &= V^{(1)} - \frac{1}{2}ra^{(yy)}\delta_y^2 U^n\,.
\end{aligned} \tag{15.81}$$

Here all the coefficients are evaluated at node (m, l), and the mixed-derivative operator is:

$$\delta_{xy}U_{m,l} = \frac{1}{4h^2}\Big(U_{m+1,l+1} + U_{m-1,l-1} - U_{m-1,l+1} - U_{m+1,l-1}\Big). \tag{15.82}$$

Scheme (15.81) was originally proposed by Craig and Sneyd in their paper cited above. There, it was given by their Eq. (7) in more condensed notations, which we will write as:

$$\begin{aligned}
\Big(1 - \frac{1}{2}ra^{(xx)}\delta_x^2\Big)\Big(1 - \frac{1}{2}ra^{(yy)}\delta_y^2\Big)W^{(2)} &= \Big(1 + \frac{1}{2}ra^{(xx)}\delta_x^2\Big)\Big(1 + \frac{1}{2}ra^{(yy)}\delta_y^2\Big)U^n + ra^{(xy)}\delta_{xy}U^n, \\
\Big(1 - \frac{1}{2}ra^{(xx)}\delta_x^2\Big)\Big(1 - \frac{1}{2}ra^{(yy)}\delta_y^2\Big)U^{n+1} &= \Big(1 + \frac{1}{2}ra^{(xx)}\delta_x^2\Big)\Big(1 + \frac{1}{2}ra^{(yy)}\delta_y^2\Big)U^n \\
&\quad + \frac{1}{2}ra^{(xy)}\delta_{xy}\Big(W^{(2)} + U^n\Big)\,.
\end{aligned} \tag{15.83}$$

In order to turn the Graig–Sneyd scheme (15.81) into a practical algorithm, one needs to specify what boundary conditions for its auxiliary variables are needed and how those can

be found. I have been unable to find an answer to this question in the published literature; therefore below I present my own answer. Let us start at the first line of (15.81). Its left-hand side can be computed only at the interior grid points, i.e., for $1 \le m \le M-1$ and $1 \le l \le L-1$, since the calculation of the terms on the right-hand side requires boundary values of U^n along the entire perimeter of the computational domain. Next, to determine $W^{(1)}$ in the second line, we need its boundary values $W^{(1)}_{\{0,M\},l}$ for $1 \le l \le L-1$. Those can be found from the third line with $m = 0$ and $m = M$ if one knows the boundary values $W^{(2)}_{\{0,M\},l}$ for *all* l, i.e., for $0 \le l \le L$. Thus, we focus on specifying or finding these latter boundary values. It turns out that one *cannot find* them. Indeed, the fourth line of (15.81) does not provide any information about $W^{(2)}_{\{0,M\},l}$ but rather, to aggravate the matters, requires the values $W^{(2)}_{m,\{0,L\}}$ at the other boundary in order to compute $\delta^2_{xy}W^{(2)}$ for all interior points $1 \le m \le M-1$, $1 \le l \le L-1$. One can also verify that none of these boundary values can be computed if we start from the *last* line of (15.81), either. Thus, the only option remains to *specify* the values of $W^{(2)}$ *along the entire perimeter* of the computational domain.

This option is consistent with the form (15.83) of the Craig–Sneyd scheme. Indeed, then the first line of that scheme is nothing but the inhomogeneous version of (15.33) (with the inhomogeneity being the $\delta_{xy}U^n$-term), and we know that to solve it, the boundary values of the variable on the left-hand side must be specified.

The way to specify the boundary values of $W^{(2)}$ appears to be to let them equal those of U^{n+1}:

$$\begin{aligned} W^{(2)}_{\{0,M\},l} &= U^{n+1}_{\{0,M\},l}\,, \qquad 0 \le l \le L; \\ W^{(2)}_{m,\{0,L\}} &= U^{n+1}_{m,\{0,L\}}\,, \qquad 1 \le m \le M-1. \end{aligned} \tag{15.84}$$

To see that, let the mixed-derivative term in (15.81) vanish: $a^{(xy)} = 0$. Then from the fourth line of that scheme, $V^{(0)} = W^{(0)}$, and then $W^{(2)}$ simply coincides with U^{n+1} at all nodes.

To summarize, we list the steps of implementing the algorithm (15.81), (15.84) into a code.

<u>*Step 1*</u>: Define the boundary values $U^{n+1}_{\{0,M\},l}$ and $W^{(2)}_{\{0,M\},l}$ for $0 \le l \le L$. Next, compute the boundary values $V^{(1)}_{\{0,M\},l}$ and $W^{(1)}_{\{0,M\},l}$ for $1 \le l \le L-1$ from the last and third lines of (15.81), respectively.

<u>*Step 2*</u>: Define the boundary values $U^{n+1}_{m,\{0,L\}}$ and $W^{(2)}_{m,\{0,L\}}$ for $1 \le m \le M-1$.

<u>*Step 3*</u>: Find the variables on the left-hand sides of the first two lines of (15.81). This can be done because the required boundary values of $W^{(1)}$ have been computed in Step 1.

<u>*Step 4*</u>: Find $W^{(2)}$ at all the interior points from the third line of (15.81). This can be done because the required boundary values of $W^{(2)}$ have been defined in Step 2.

<u>*Step 5*</u>: Find the variables on the left-hand sides of the fifth and sixth lines of (15.81). This can be done because the required boundary values of $V^{(1)}$ have been computed in Step 1 and the required boundary values of $W^{(2)}$ have been defined in Steps 1 and 2.

<u>*Step 6*</u>: Find U^{n+1} at all the interior points from the last line of (15.81). This can be done because the required boundary values have been defined in Step 2.

Generalization of the Craig–Sneyd scheme (15.81) to three spatial dimensions is straightforward. Craig and Sneyd also generalized it to a *system of coupled equations* of the form (15.80); see "An alternating direction implicit scheme for parabolic systems of partial differential equations," Computers and Mathematics with Applications **20**(3), 53–62 (1990). Also, in 't

Hout and Welfert published a follow-up paper to their paper cited above; it is: "Unconditional stability of second-order ADI schemes applied to multi-dimensional diffusion equations with mixed derivative terms," Applied Numerical Mathematics **59**, 677–692 (2009). There, they consider a more restricted class of equations than (15.80) (diffusion only, no convection), but in exchange are able to prove unconditional stability of a certain scheme in any number of spatial dimensions. This has applications in, e.g., financial mathematics.

15.8 Appendix 2: Derivation of the Peaceman–Rachford algorithm for the 2D Heat equation with Neumann boundary conditions

In order for you to understand the details of this derivation better, it is recommended that you first review the corresponding derivation for the Crank-Nicolson method in Sec. 14.1, since it is that derivation on which the present one is based. It should also help you to draw a single time level and refer to that drawing throughout the derivation.

We begin by determining which boundary values of the intermediate solution $\overset{*}{U}$ are required to compute the solution U_{ml}^{n+1}, $0 \le \{m,l\} \le \{M,L\}$ at the new time level. To that end, let us write down Eq. (15.34b) in a detailed form:

$$U_{m,l}^{n+1} - \frac{r}{2}\left[U_{m,l+1}^{n+1} - 2U_{m,l}^{n+1} + U_{m,l-1}^{n+1}\right] = \overset{*}{U}_{m,l} + \frac{r}{2}\left[\overset{*}{U}_{m+1,l} - 2\,\overset{*}{U}_{m,l} + \overset{*}{U}_{m-1,l}\right]. \tag{15.85}$$

We need to determine the solution on the l.h.s. for $0 \le \{m,l\} \le \{M,L\}$. First, we note that we *can* set $l = 0$ in (15.85) despite the fact that $U_{m,-1}^{n+1}$ will then appear in the last term on the l.h.s., and that value is *not* part of the solution. To eliminate that value, we use the boundary condition at $y = 0$:

$$\frac{U_{m,1}^{n+1} - U_{m,-1}^{n+1}}{2h} = (g_2)_m^{n+1}, \quad \Rightarrow \quad U_{m,-1}^{n+1} = U_{m,1}^{n+1} - 2h(g_2)_m^{n+1}, \qquad 0 \le m \le M\,. \tag{15.86}$$

Similarly, at $y = Y$, we have

$$\frac{U_{m,L+1}^{n+1} - U_{m,L-1}^{n+1}}{2h} = (g_3)_m^{n+1}, \quad \Rightarrow \quad U_{m,L+1}^{n+1} = U_{m,L-1}^{n+1} + 2h(g_3)_m^{n+1}, \qquad 0 \le m \le M\,. \tag{15.87}$$

Therefore, Eq. (15.85) for $l = 0$ and $l = L$ is replaced by the respective equations:

$$U_{m,0}^{n+1} - \frac{r}{2}\left[2U_{m,1}^{n+1} - 2U_{m,0}^{n+1} - 2h(g_2)_m^{n+1}\right] = \overset{*}{U}_{m,0} + \frac{r}{2}\left[\overset{*}{U}_{m+1,0} - 2\,\overset{*}{U}_{m,0} + \overset{*}{U}_{m-1,0}\right]; \tag{15.88}$$

$$U_{m,L}^{n+1} - \frac{r}{2}\left[2h(g_3)_m^{n+1} - 2U_{m,L}^{n+1} + 2U_{m,L-1}^{n+1}\right] = \overset{*}{U}_{m,L} + \frac{r}{2}\left[\overset{*}{U}_{m+1,L} - 2\,\overset{*}{U}_{m,L} + \overset{*}{U}_{m-1,L}\right]. \tag{15.89}$$

Thus, in order to determine from (15.85), (15.88), and (15.89) the solution $U_{m,l}^{n+1}$ for all $0 \le \{m,l\} \le \{M,L\}$, we will need to know

$$\overset{*}{U}_{m,l} \quad \text{for } -1 \le m \le M+1 \text{ and } 0 \le l \le L. \tag{15.90}$$

The difficulty that we need to overcome is the determination of the boundary values

$$\overset{*}{U}_{-1,l} \quad \text{and} \quad \overset{*}{U}_{M+1,l} \quad \text{for } 0 \le l \le L. \tag{15.91}$$

Now let us see which of the values $\overset{*}{U}_{m,l}$ we can determine directly from (15.34a). To that end, let us write down that equation in a detailed form, similarly to (15.85):

$$\overset{*}{U}_{m,l} - \frac{r}{2}\left[\overset{*}{U}_{m+1,l} - 2\,\overset{*}{U}_{m,l} + \overset{*}{U}_{m-1,l}\right] = U^n_{m,l} + \frac{r}{2}\left[U^n_{m,l+1} - 2U^n_{m,l} + U^n_{m,l-1}\right]. \tag{15.92}$$

From this equation, we see that in order to determine the values required in (15.90), we need to know

$$U^n_{ml} \quad \text{for } -1 \le m \le M+1 \text{ and } -1 \le l \le L+1. \tag{15.93}$$

While those values for $0 \le \{m,l\} \le \{M,L\}$ are known from the solution on the nth time level, the values $U^n_{m,l}$ for $\{m,l\} = -1$ and $\{m,l\} = \{M+1, L+1\}$ are not, and hence they need to be found from the boundary conditions. This is done similarly to (15.86) and (15.87):

$$\begin{aligned} &U^n_{-1,l} = U^n_{1,l} - 2h(g_0)^n_l\,, \qquad U^n_{M+1,l} = U^n_{M-1,l} + 2h(g_1)^n_l\,, \quad 0 \le l \le L\,; \\ &U^n_{m,-1} = U^n_{m,1} - 2h(g_2)^n_m\,, \quad U^n_{m,L+1} = U^n_{m,L-1} + 2h(g_3)^n_m\,, \quad 0 \le m \le M\,. \end{aligned} \tag{15.94}$$

Once the values in (15.94) have been found, we determine the values at the corners:

$$\begin{aligned} &U^n_{-1,-1} = U^n_{1,-1} - 2h(g_0)^n_{-1}\,, \qquad U^n_{M+1,-1} = U^n_{M-1,-1} + 2h(g_1)^n_{-1}\,, \\ &U^n_{-1,L+1} = U^n_{1,L+1} - 2h(g_0)^n_{L+1}\,, \quad U^n_{M+1,L+1} = U^n_{M-1,L+1} + 2h(g_1)^n_{L+1}\,. \end{aligned} \tag{15.95}$$

Note that the smoothness of the solution at the corners is ensured by the matching conditions (15.58). Thus, with (15.94) and (15.95), we have all the values required in (15.93).

We now turn back to (15.92). From it, we see that with all the values (15.93) being known, the l.h.s. of (15.92) can be determined from an $(M+1)\times(M+1)$ system of equations only if we know the values in (15.91). To determine these values, we use Eq. (15.59) in the following way: we subtract that equation with $m \to (m-1)$ from the same equation with $m \to (m+1)$ and divide the result by $(2h)$. For $m = 0$, this yields:

$$\begin{aligned} \frac{\overset{*}{U}_{1,l} - \overset{*}{U}_{-1,l}}{2h} = \; & \frac{1}{2}\left(\frac{U^n_{1,l} - U^n_{-1,l}}{2h} + \frac{U^{n+1}_{1,l} - U^{n+1}_{-1,l}}{2h}\right) + \frac{r}{4}\cdot \\ & \left(\left[\frac{U^n_{1,l+1} - U^n_{-1,l+1}}{2h} - 2\,\frac{U^n_{1,l} - U^n_{-1,l}}{2h} + \frac{U^n_{1,l-1} - U^n_{-1,l-1}}{2h}\right]\right. \\ & \left. + \left[\frac{U^n_{1,l+1} - U^n_{-1,l+1}}{2h} - 2\,\frac{U^n_{1,l} - U^n_{-1,l}}{2h} + \frac{U^n_{1,l-1} - U^n_{-1,l-1}}{2h}\right]\right), \\ & \text{for } 0 \le l \le L. \end{aligned} \tag{15.96}$$

If we can find each term on the r.h.s. of this equation, then we know the value of the term on the l.h.s. and hence can determine $\overset{*}{U}_{-1,l}$. But each of these terms can be found from the boundary conditions! Upon this observation, (15.96) can be rewritten as follows:

$$\begin{aligned} \frac{\overset{*}{U}_{1,l} - \overset{*}{U}_{-1,l}}{2h} = \; & \frac{1}{2}\left((g_0)^n_l + (g_0)^{n+1}_l\right) + \frac{r}{4}\cdot \\ & \left(\left[(g_0)^n_{l+1} - 2(g_0)^n_l + (g_0)^n_{l-1}\right]\right. \\ & \left. + \left[(g_0)^{n+1}_{l+1} - 2(g_0)^{n+1}_l + (g_0)^{n+1}_{l-1}\right]\right), \\ \equiv \; & (\mathcal{G}_0)_l\,, \\ & \text{for } 0 \le l \le L. \end{aligned} \tag{15.97}$$

Similarly,

$$\begin{aligned}\frac{\overset{*}{U}_{M+1,l} - \overset{*}{U}_{M-1,l}}{2h} &= \frac{1}{2}\left((g_1)_l^n + (g_1)_l^{n+1}\right) + \frac{r}{4}\cdot \\ &\quad \Big(\big[(g_1)_{l+1}^n - 2(g_1)_l^n + (g_1)_{l-1}^n\big] \\ &\quad + \big[(g_1)_{l+1}^{n+1} - 2(g_1)_l^{n+1} + (g_1)_{l-1}^{n+1}\big]\Big)\,, \\ &\equiv (\mathcal{G}_1)_l\,, \\ &\quad \text{for } 0 \le l \le L.\end{aligned} \tag{15.98}$$

Using Eqs. (15.97) and (15.98), the linear systems given, for each $l = 0, \ldots, L$, by Eqs. (15.92) with $1 \le m \le M-1$, can be supplemented by the following equations for $m = 0$ and $m = M$:

$$\begin{aligned}&\overset{*}{U}_{0,l} - \frac{r}{2}\left[2\,\overset{*}{U}_{1,l} - 2\,\overset{*}{U}_{0,l} - 2h(\mathcal{G}_0)_l\right] = U_{0,l}^n + \frac{r}{2}\left[U_{0,l+1}^n - 2U_{0,l}^n + U_{0,l-1}^n\right], \\ &\overset{*}{U}_{M,l} - \frac{r}{2}\left[2h(\mathcal{G}_1)_l - 2\,\overset{*}{U}_{M,l} + 2\,\overset{*}{U}_{M-1,l}\right] = U_{M,l}^n + \frac{r}{2}\left[U_{M,l+1}^n - 2U_{M,l}^n + U_{M,l-1}^n\right], \\ &0 \le l \le L\,.\end{aligned} \tag{15.99}$$

From (15.99) and the remaining equations in (15.92) one can determine

$$\overset{*}{U}_{m,l} \quad \text{for } 0 \le \{m, l\} \le \{M, L\}, \tag{15.100}$$

and the remaining values (15.91) are determined from (15.97) and (15.98).

15.9 Questions for self-assessment

1. According to the lexicographic ordering, which quantity appears earlier in the vector $\vec{\mathbf{U}}$ in Eq. (15.8): $U_{2,5}$ or $U_{5,2}$?
2. Verify Eq. (15.14).
3. Verify Eq. (15.15).
4. Verify Eq. (15.19).
5. What is the length of vector $\vec{\mathbf{b}}_l^n$ in Eq. (15.21)?
6. Write down Eq. (15.19) for $l = 2$. Then verify that it is equivalent to Eq. (15.22) for the same value of l.
7. Obtain the analog of Eq. (15.23) for $l = L - 1$.
8. What is the length of vector $\mathcal{B}^n$ in Eq. (15.25)?
9. Why is the CN scheme (15.16) computationally inefficient?
10. State the three properties that a computationally efficient scheme for the 2D Heat equation must have.
11. What is the order of the truncation error of scheme (15.31)?

12. Verify that (15.33) is equivalent to (15.32).
13. What is the order of the truncation error of scheme (15.33)?
14. Make sure you can justify each step in (15.35).
15. What is the order of the truncation error of scheme (15.34)?
16. Explain in detail why (15.34) is computationally efficient (that is, which systems need to be solved at each step).
17. Obtain both equations in (15.41).
18. Why does one want to look for alternatives to the Peaceman–Rachford method?
19. Make sure you can obtain (15.47).
20. Produce an example of r, β, γ, and ξ such that the corresponding amplification factor (15.47) is greater than 1 in magnitude.
21. Explain in detail why (15.48) is computationally efficient (that is, which systems need to be solved at each step).
22. Consider the Peaceman–Rachford method for the Heat equation with Dirichlet boundary conditions. Explain which boundary values of the intermediate solution one requires, and why one does not need other boundary values.
23. Verify that the scheme (15.78) with $F^{(0)} = F^{(2)} = 0$ reduces to the modified implicit Euler method.
24. Verify that the scheme (15.79) with $F^{(1)} = F^{(2)} = 0$ reduces to the modified explicit Euler method.
25. Make sure you can follow the argument made around (15.84).