Chapter 12

SPLIT AND FACTORED FORMS

In the next two chapters, we present and analyze split and factored algorithms. This gives the reader a feel for some of the modifications which can be made to the basic algorithms in order to obtain efficient solvers for practical multidimensional applications, and a means for analyzing such modified forms.

12.1 The Concept

Factored forms of numerical operators are used extensively in constructing and applying numerical methods to problems in fluid mechanics. They are the basis for a wide variety of methods variously known by the labels "hybrid", "time split", and "fractional step". Factored forms are especially useful for the derivation of practical algorithms that use implicit methods. When we approach numerical analysis in the light of matrix derivative operators, the concept of factoring is quite simple to present and grasp. Let us start with the following observations:

1. Matrices can be split in quite arbitrary ways.
2. Advancing to the next time level always requires some reference to a previous one.
3. Time marching methods are valid only to some order of accuracy in the step size, h.

Now recall the generic ODE's produced by the semi-discrete approach

$$\frac{d\vec{u}}{dt} = A\vec{u} - \vec{f} \tag{12.1}$$

and consider the above observations. From observation 1 (arbitrary splitting of A) :

$$\frac{d\vec{u}}{dt} = [\mathrm{A}_1 + \mathrm{A}_2]\vec{u} - \vec{f} \tag{12.2}$$

where $A = [\mathrm{A}_1 + \mathrm{A}_2]$ but A_1 and A_2 are not unique. For the time march let us choose the simple, first-order,[1] explicit Euler method. Then, from observation 2 (new data $\vec{u}_{n+1}$ in terms of old $\vec{u}_n$):

$$\vec{u}_{n+1} = [\, I \ + h\mathrm{A}_1 + h\mathrm{A}_2]\vec{u}_n - h\vec{f} + O(h^2) \tag{12.3}$$

or its equivalent

$$\vec{u}_{n+1} = \Big[[\, I \ + h\mathrm{A}_1][\, I \ + h\mathrm{A}_2] - h^2\mathrm{A}_1\mathrm{A}_2\Big]\vec{u}_n - h\vec{f} + O(h^2)$$

Finally, from observation 3 (allowing us to drop higher order terms $-h^2\mathrm{A}_1\mathrm{A}_2\vec{u}_n$):

$$\vec{u}_{n+1} = [\, I \ + h\mathrm{A}_1][\, I \ + h\mathrm{A}_2]\vec{u}_n - h\vec{f} + O(h^2) \tag{12.4}$$

Notice that Eqs. 12.3 and 12.4 have the same formal order of accuracy and, in this sense, neither one is to be preferred over the other. However, their numerical stability can be quite different, and techniques to carry out their numerical evaluation can have arithmetic operation counts that vary by orders of magnitude. Both of these considerations are investigated later. Here we seek only to apply to some simple cases the concept of factoring.

12.2 Factoring Physical Representations — Time Splitting

Suppose we have a PDE that represents both the processes of convection and dissipation. The semi-discrete approach to its solution might be put in the form

$$\frac{d\vec{u}}{dt} = \mathrm{A}_c\vec{u} + \mathrm{A}_d\vec{u} + (\vec{bc}) \tag{12.5}$$

where A_c and A_d are matrices representing the convection and dissipation terms, respectively; and their sum forms the A matrix we have considered in the previous sections. Choose again the explicit Euler time march so that

$$\vec{u}_{n+1} = [\, I \ + h\mathrm{A}_d + h\mathrm{A}_c]\vec{u}_n + h(\vec{bc}) + O(h^2) \tag{12.6}$$

[1] Second-order time-marching methods are considered later.

Now consider the factored form

$$\begin{aligned}\vec{u}_{n+1} &= [\, I \;+ h\mathrm{A}_d]\Big([\, I \;+ h\mathrm{A}_c]\vec{u}_n + h(\vec{bc})\Big) \\ &= \underbrace{[\, I \;+ h\mathrm{A}_d + h\mathrm{A}_c]\vec{u}_n + h(\vec{bc})}_{\text{Original Unfactored Terms}} + \underbrace{h^2\mathrm{A}_d\Big(\mathrm{A}_c\vec{u}_n + (\vec{bc})\Big)}_{\text{Higher Order Terms}} + O(h^2) \qquad (12.7)\end{aligned}$$

and we see that Eq. 12.7 and the original unfactored form Eq. 12.6 have identical orders of accuracy in the time approximation. Therefore, on this basis, their selection is arbitrary. In practical applications[2] equations such as 12.7 are often applied in a predictor-corrector sequence. In this case one could write

$$\begin{aligned}\tilde{u}_{n+1} &= [\, I \;+ h\mathrm{A}_c]\vec{u}_n + h(\vec{bc}) \\ \vec{u}_{n+1} &= [\, I \;+ h\mathrm{A}_d]\tilde{u}_{n+1} \qquad (12.8)\end{aligned}$$

Factoring can also be useful to form split combinations of implicit and explicit techniques. For example, another way to approximate Eq. 12.6 with the same order of accuracy is given by the expression

$$\begin{aligned}\vec{u}_{n+1} &= [\, I \;- h\mathrm{A}_d]^{-1}\Big([\, I \;+ h\mathrm{A}_c]\vec{u}_n + h(\vec{bc})\Big) \\ &= \underbrace{[\, I \;+ h\mathrm{A}_d + h\mathrm{A}_c]\vec{u}_n + h(\vec{bc})}_{\text{Original Unfactored Terms}} + O(h^2) \qquad (12.9)\end{aligned}$$

where in this approximation we have used the fact that

$$[\, I \;- h\mathrm{A}_d]^{-1} = I + h\mathrm{A}_d + h^2\mathrm{A}_d^2 + \cdots$$

if $h \cdot ||\mathrm{A}_d|| < 1$, where $||\mathrm{A}_d||$ is some norm of $[\mathrm{A}_d]$. This time a predictor-corrector interpretation leads to the sequence

$$\begin{aligned}\tilde{u}_{n+1} &= [\, I \;+ h\mathrm{A}_c]\vec{u}_n + h(\vec{bc}) \\ [\, I \;- h\mathrm{A}_d]\vec{u}_{n+1} &= \tilde{u}_{n+1} \qquad (12.10)\end{aligned}$$

The convection operator is applied explicitly, as before, but the diffusion operator is now implicit, requiring a tridiagonal solver if the diffusion term is central differenced. Since numerical stiffness is generally much more severe for the diffusion process, this factored form would appear to be superior to that provided by Eq. 12.8. However, the important aspect of stability has yet to be discussed.

[2]We do not suggest that this particular method is suitable for use. We have yet to determine its stability, and a first-order time-march method is usually unsatisfactory.

We should mention here that Eq. 12.9 can be derived for a different point of view by writing Eq. 12.6 in the form

$$\frac{u_{n+1} - u_n}{h} = \mathrm{A}_c u_n + \mathrm{A}_d u_{n+1} + (\vec{bc}) + O(h^2)$$

Then

$$[\, I \; - h\mathrm{A}_d] u_{n+1} = [\, I \; + h\mathrm{A}_c] u_n + h(\vec{bc})$$

which is identical to Eq. 12.10.

12.3 Factoring Space Matrix Operators in 2–D

12.3.1 Mesh Indexing Convention

Factoring is widely used in codes designed for the numerical solution of equations governing unsteady two- and three-dimensional flows. Let us study the basic concept of factoring by inspecting its use on the linear 2-D scalar PDE that models diffusion:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{12.11}$$

We begin by reducing this PDE to a coupled set of ODE's by differencing the space derivatives and inspecting the resulting matrix operator.

A clear description of a matrix finite-difference operator in 2- and 3-D requires some reference to a mesh. We choose the 3×4 point mesh[3] shown in the Sketch 12.12. In this example M_x, the number of (interior) x points, is 4 and M_y, the number of (interior) y points is 3. The numbers $11\,,\, 12\,,\, \cdots\,,\, 43$ represent the location in the mesh of the dependent variable bearing that index. Thus u_{32} represents the value of u at $j = 3$ and $k = 2$.

$$\begin{array}{ccccccc}
 & & \odot & \odot & \odot & \odot & \\
M_y & \odot & 13 & 23 & 33 & 43 & \odot \\
k & \odot & 12 & 22 & 32 & 42 & \odot \\
1 & \odot & 11 & 21 & 31 & 41 & \odot \\
 & & \odot & \odot & \odot & \odot & \\
 & & 1 & j & \cdots & M_x &
\end{array} \tag{12.12}$$

Mesh indexing in 2-D.

[3]This could also be called a 5×6 point mesh if the boundary points (labeled $\odot$ in the sketch) were included, but in these notes we describe the size of a mesh by the number of *interior* points.

12.3.2 Data Bases and Space Vectors

The dimensioned array in a computer code that allots the storage locations of the dependent variable(s) is referred to as a *data-base*. There are many ways to lay out a data-base. Of these, we consider only two: (1), consecutively along rows that are themselves consecutive from $k = 1$ to M_y, and (2), consecutively along columns that are consecutive from $j = 1$ to M_x. We refer to each row or column group as a *space vector* (they represent data along lines that are continuous in space) and label their sum with the symbol U. In particular, (1) and (2) above are referred to as x-vectors and y-vectors, respectively. The symbol U by itself is not enough to identify the structure of the data-base and is used only when the structure is immaterial or understood.

To be specific about the structure, we label a data–base composed of x-vectors with $U^{(x)}$, and one composed of y-vectors with $U^{(y)}$. Examples of the order of indexing for these space vectors are given in Eq. 12.16 part a and b.

12.3.3 Data Base Permutations

The two vectors (arrays) are related by a permutation matrix P such that

$$U^{(x)} = \mathrm{P}_{xy} U^{(y)} \qquad \text{and} \qquad U^{(y)} = \mathrm{P}_{yx} U^{(x)} \tag{12.13}$$

where

$$\mathrm{P}_{yx} = \mathrm{P}_{xy}^T = \mathrm{P}_{xy}^{-1}$$

Now consider the structure of a matrix finite-difference operator representing 3-point central-differencing schemes for both space derivatives in two dimensions. When the matrix is multiplying a space vector U, the usual (but ambiguous) representation is given by A_{x+y}. In this notation the ODE form of Eq. 12.11 can be written [4]

$$\frac{dU}{dt} = A_{x+y} U + (\vec{bc}) \tag{12.14}$$

If it is important to be specific about the data-base structure, we use the notation $A_{x+y}^{(x)}$ or $A_{x+y}^{(y)}$, depending on the data–base chosen for the U it multiplies. Examples are in Eq. 12.16 part a and b. Notice that the matrices are not the same although they represent the same derivative operation. Their structures are similar, however, and they are related by the same permutation matrix that relates $U^{(x)}$ to $U^{(y)}$. Thus

$$A_{x+y}^{(x)} = \mathrm{P}_{xy} \cdot A_{x+y}^{(y)} \cdot \mathrm{P}_{yx} \tag{12.15}$$

[4]Notice that A_{x+y} and U, which are notations used in the special case of space vectors, are subsets of A and $\vec{u}$, used in the previous sections.

$$A^{(x)}_{x+y} \cdot U^{(x)} = \left[\begin{array}{cccc|cccc|cccc}
\bullet & x & & & o & & & & & & & \\
x & \bullet & x & & & o & & & & & & \\
 & x & \bullet & x & & & o & & & & & \\
 & & x & \bullet & & & & o & & & & \\
\hline
o & & & & \bullet & x & & & o & & & \\
 & o & & & x & \bullet & x & & & o & & \\
 & & o & & & x & \bullet & x & & & o & \\
 & & & o & & & x & \bullet & & & & o \\
\hline
 & & & & o & & & & \bullet & x & & \\
 & & & & & o & & & x & \bullet & x & \\
 & & & & & & o & & & x & \bullet & x \\
 & & & & & & & o & & & x & \bullet
\end{array}\right] \cdot \begin{array}{c} 11\\ 21\\ 31\\ 41\\ --\\ 12\\ 22\\ 32\\ 42\\ --\\ 13\\ 23\\ 33\\ 43 \end{array}$$

a:Elements in 2-dimensional, central-difference, matrix operator, A_{x+y}, for 3×4 mesh shown in Sketch 12.12. Data base composed of M_y x–vectors stored in $U^{(x)}$. Entries for $x \to x$, for $y \to o$, for both $\to \bullet$.

(12.16)

$$A^{(y)}_{x+y} \cdot U^{(y)} = \left[\begin{array}{ccc|ccc|ccc|ccc}
\bullet & o & & x & & & & & & & & \\
o & \bullet & o & & x & & & & & & & \\
 & o & \bullet & & & x & & & & & & \\
\hline
x & & & \bullet & o & & x & & & & & \\
 & x & & o & \bullet & o & & x & & & & \\
 & & x & & o & \bullet & & & x & & & \\
\hline
 & & & x & & & \bullet & o & & x & & \\
 & & & & x & & o & \bullet & o & & x & \\
 & & & & & x & & o & \bullet & & & x \\
\hline
 & & & & & & x & & & \bullet & o & \\
 & & & & & & & x & & o & \bullet & o \\
 & & & & & & & & x & & o & \bullet
\end{array}\right] \cdot \begin{array}{c} 11\\ 12\\ 13\\ --\\ 21\\ 22\\ 23\\ --\\ 31\\ 32\\ 33\\ --\\ 41\\ 42\\ 43 \end{array}$$

b: Elements in 2-dimensional, central-difference, matrix operator, A_{x+y}, for 3×4 mesh shown in Sketch 12.12. Data base composed of M_x y–vectors stored in $U^{(y)}$. Entries for $x \to x$, for $y \to o$, for both $\to \bullet$.

12.3.4 Space Splitting and Factoring

We are now prepared to discuss splitting in two dimensions. It should be clear that the matrix $A^{(x)}_{x+y}$ can be split into two matrices such that

$$A^{(x)}_{x+y} = A^{(x)}_x + A^{(x)}_y \tag{12.17}$$

where $A^{(x)}_x$ and $A^{(x)}_y$ are shown in Eq. 12.22. Similarily

$$A^{(y)}_{x+y} = A^{(y)}_x + A^{(y)}_y \tag{12.18}$$

where the split matrices are shown in Eq. 12.23.

The permutation relation also holds for the split matrices so

$$A^{(x)}_y = \ \mathrm{P}_{xy} A^{(y)}_y \ \mathrm{P}_{yx}$$

and

$$A^{(x)}_x = \ \mathrm{P}_{xy} A^{(y)}_x \ \mathrm{P}_{yx}$$

The splittings in Eqs. 12.17 and 12.18 can be combined with factoring in the manner described in Section 12.2. As an example (first-order in time), applying the implicit Euler method to Eq. 12.14 gives

$$U^{(x)}_{n+1} = U^{(x)}_n + h\Big[A^{(x)}_x + A^{(x)}_y\Big]U^{(x)}_{n+1} + h(\vec{bc})$$

or

$$\Big[I - hA^{(x)}_x - hA^{(x)}_y\Big]U^{(x)}_{n+1} = U^{(x)}_n + h(\vec{bc}) + O(h^2) \tag{12.19}$$

As in Section 12.2, we retain the same first order accuracy with the alternative

$$\Big[I - hA^{(x)}_x\Big]\Big[I - hA^{(x)}_y\Big]U^{(x)}_{n+1} = U^{(x)}_n + h(\vec{bc}) + O(h^2) \tag{12.20}$$

Write this in predictor-corrector form and permute the data base of the second row. There results

$$\begin{aligned} \Big[I - hA^{(x)}_x\Big]\tilde{U}^{(x)} &= U^{(x)}_n + h(\vec{bc}) \\ \Big[I - hA^{(y)}_y\Big]U^{(y)}_{n+1} &= \tilde{U}^{(y)} \end{aligned} \tag{12.21}$$

$$
\begin{gathered}
A_x^{(x)} \cdot U^{(x)} = \left[\begin{array}{cccc|cccc|cccc}
x & x & & & & & & & & & & \\
x & x & x & & & & & & & & & \\
 & x & x & x & & & & & & & & \\
 & & x & x & & & & & & & & \\
\hline
 & & & & x & x & & & & & & \\
 & & & & x & x & x & & & & & \\
 & & & & & x & x & x & & & & \\
 & & & & & & x & x & & & & \\
\hline
 & & & & & & & & x & x & & \\
 & & & & & & & & x & x & x & \\
 & & & & & & & & & x & x & x \\
 & & & & & & & & & & x & x
\end{array}\right] \cdot U^{(x)} \\
A_y^{(x)} \cdot U^{(x)} = \left[\begin{array}{cccc|cccc|cccc}
o & & & & o & & & & & & & \\
 & o & & & & o & & & & & & \\
 & & o & & & & o & & & & & \\
 & & & o & & & & o & & & & \\
\hline
o & & & & o & & & & o & & & \\
 & o & & & & o & & & & o & & \\
 & & o & & & & o & & & & o & \\
 & & & o & & & & o & & & & o \\
\hline
 & & & & o & & & & o & & & \\
 & & & & & o & & & & o & & \\
 & & & & & & o & & & & o & \\
 & & & & & & & o & & & & o
\end{array}\right] \cdot U^{(x)}
\end{gathered}
\tag{12.22}
$$

The splitting of $A_{x+y}^{(x)}$.

$$
A_x^{(y)} \cdot U^{(y)} = \left[\begin{array}{ccc|ccc|ccc|ccc}
x & & & x & & & & & & & & \\
 & x & & & x & & & & & & & \\
 & & x & & & x & & & & & & \\
\hline
x & & & x & & & x & & & & & \\
 & x & & & x & & & x & & & & \\
 & & x & & & x & & & x & & & \\
\hline
 & & & x & & & x & & & x & & \\
 & & & & x & & & x & & & x & \\
 & & & & & x & & & x & & & x \\
\hline
 & & & & & & x & & & x & & \\
 & & & & & & & x & & & x & \\
 & & & & & & & & x & & & x
\end{array} \right] \cdot U^{(y)}
$$

$$
A_y^{(y)} \cdot U^{(y)} = \left[\begin{array}{ccc|ccc|ccc|ccc}
o & o & & & & & & & & & & \\
o & o & o & & & & & & & & & \\
 & o & o & & & & & & & & & \\
\hline
 & & & o & o & & & & & & & \\
 & & & o & o & o & & & & & & \\
 & & & & o & o & & & & & & \\
\hline
 & & & & & & o & o & & & & \\
 & & & & & & o & o & o & & & \\
 & & & & & & & o & o & & & \\
\hline
 & & & & & & & & & o & o & \\
 & & & & & & & & & o & o & o \\
 & & & & & & & & & & o & o
\end{array} \right] \cdot U^{(y)}
\tag{12.23}
$$

The splitting of $A_{x+y}^{(y)}$.

12.4 Second-Order Factored Implicit Methods

Second-order accuracy in time can be maintained in a certain factored implicit methods. For example, apply the trapezoidal method to Eq. 12.14 where the derivative operators have been split as in Eq. 12.17 or 12.18. Let the data base be immaterial and the $(\vec{bc})$ be time invariant. There results

$$\left[I - \frac{1}{2}hA_x - \frac{1}{2}hA_y\right]U_{n+1} = \left[I + \frac{1}{2}hA_x + \frac{1}{2}hA_y\right]U_n + h(\vec{bc}) + O(h^3) \qquad (12.24)$$

Factor both sides giving

$$\begin{aligned}
&\left[\left[I - \frac{1}{2}hA_x\right]\left[I - \frac{1}{2}hA_y\right] - \frac{1}{4}h^2 A_x A_y\right]U_{n+1} \\
= &\left[\left[I + \frac{1}{2}hA_x\right]\left[I + \frac{1}{2}hA_y\right] - \frac{1}{4}h^2 A_x A_y\right]U_n + h(\vec{bc}) + O(h^3) \qquad (12.25)
\end{aligned}$$

Then notice that the combination $\frac{1}{4}h^2[A_x A_y](U_{n+1} - U_n)$ is proportional to h^3 since the leading term in the expansion of $(U_{n+1} - U_n)$ is proportional to h. Therefore, we can write

$$\left[I - \frac{1}{2}hA_x\right]\left[I - \frac{1}{2}hA_y\right]U_{n+1} = \left[I + \frac{1}{2}hA_x\right]\left[I + \frac{1}{2}hA_y\right]U_n + h(\vec{bc}) + O(h^3) \qquad (12.26)$$

and both the factored and unfactored form of the trapezoidal method are second-order accurate in the time march.

An alternative form of this kind of factorization is the classical ADI (alternating direction implicit) method[5] usually written

$$\begin{aligned}
\left[I - \frac{1}{2}hA_x\right]\tilde{U} &= \left[I + \frac{1}{2}hA_y\right]U_n + \frac{1}{2}hF_n \\
\left[I - \frac{1}{2}hA_y\right]U_{n+1} &= \left[I + \frac{1}{2}hA_x\right]\tilde{U} + \frac{1}{2}hF_{n+1} + O(h^3) \qquad (12.27)
\end{aligned}$$

For idealized commuting systems the methods given by Eqs. 12.26 and 12.27 differ only in their evaluation of a time-dependent forcing term.

12.5 Importance of Factored Forms in 2 and 3 Dimensions

When the time-march equations are stiff and implicit methods are required to permit reasonably large time steps, the use of factored forms becomes a very valuable tool

[5] A form of the Douglas or Peaceman-Rachford methods.

for realistic problems. Consider, for example, the problem of computing the time advance in the unfactored form of the trapezoidal method given by Eq. 12.24

$$\left[I - \frac{1}{2}hA_{x+y}\right]U_{n+1} = \left[I + \frac{1}{2}hA_{x+y}\right]U_n + h(\vec{bc})$$

Forming the right hand side poses no problem, but finding U_{n+1} requires the solution of a sparse, but very large, set of coupled simultaneous equations having the matrix form shown in Eq. 12.16 part a and b. Furthermore, in real cases involving the Euler or Navier-Stokes equations, each symbol $(o, x, \bullet)$ represents a 4×4 block matrix with entries that depend on the pressure, density and velocity field. Suppose we were to solve the equations directly. The forward sweep of a simple Gaussian elimination fills[6] all of the 4×4 blocks between the main and outermost diagonal[7] (e.g. between $\bullet$ and o in Eq. 12.16 part b.). This must be stored in computer memory to be used to find the final solution in the backward sweep. If N_e represents the order of the small block matrix (4 in the 2-D Euler case), the approximate memory requirement is

$$(N_e \times M_y) \cdot (N_e \times M_y) \cdot M_x$$

floating point words. Here it is assumed that $M_y < M_x$. If $M_y > M_x$, M_y and M_x would be interchanged. A moderate mesh of 60×200 points would require over 11 million words to find the solution. Actually current computer power is able to cope rather easily with storage requirements of this order of magnitude. With computing speeds of over one gigaflop,[8] direct solvers may become useful for finding steady-state solutions of practical problems in two dimensions. However, a three-dimensional solver would require a memory of approximatly

$$N_e^2 \cdot M_y^2 \cdot M_z^2 \cdot M_x$$

words and, for well resolved flow fields, this probably exceeds memory availability for some time to come.

On the other hand, consider computing a solution using the *factored* implicit equation 12.25. Again computing the right hand side poses no problem. Accumulate the result of such a computation in the array (RHS). One can then write the remaining terms in the two-step predictor-corrector form

$$\begin{aligned}\left[I - \frac{1}{2}hA_x^{(x)}\right]\tilde{U}^{(x)} &= (RHS)^{(x)} \\ \left[I - \frac{1}{2}hA_y^{(y)}\right]U_{n+1}^{(y)} &= \tilde{U}^{(y)}\end{aligned} \tag{12.28}$$

[6]For matrices as small as those shown there are many gaps in this "fill", but for meshes of practical size the fill is mostly dense.

[7]The lower band is also computed but does not have to be saved unless the solution is to be repeated for another vector.

[8]One billion floating-point operations per second.

which has the same appearance as Eq. 12.21 but is second-order time accurate. The first step would be solved using M_y uncoupled block tridiagonal solvers[9]. Inspecting the top of Eq. 12.22, we see that this is equivalent to solving M_y *one-dimensional* problems, each with M_x blocks of order N_e. The temporary solution $\tilde{U}^{(x)}$ would then be permuted to $\tilde{U}^{(y)}$ and an inspection of the bottom of Eq. 12.23 shows that the final step consists of solving M_x *one-dimensional* implicit problems each with dimension M_y.

12.6 The Delta Form

Clearly many ways can be devised to split the matrices and generate factored forms. One way that is especially useful, for ensuring a correct steady-state solution in a converged time-march, is referred to as the "delta form" and we develop it next.

Consider the unfactored form of the trapezoidal method given by Eq. 12.24, and let the $(\vec{bc})$ be time invariant:

$$\left[I - \frac{1}{2}hA_x - \frac{1}{2}hA_y\right]U_{n+1} = \left[I + \frac{1}{2}hA_x + \frac{1}{2}hA_y\right]U_n + h(\vec{bc}) + O(h^3)$$

From both sides subtract

$$\left[I - \frac{1}{2}hA_x - \frac{1}{2}hA_y\right]U_n$$

leaving the equality unchanged. Then, using the standard definition of the difference operator Δ,

$$\Delta U_n = U_{n+1} - U_n$$

one finds

$$\left[I - \frac{1}{2}hA_x - \frac{1}{2}hA_y\right]\Delta U_n = h\Big[A_{x+y}U_n + (\vec{bc})\Big] + O(h^3) \tag{12.29}$$

Notice that the right side of this equation is the product of h and a term that is identical to the right side of Eq. 12.14, our original ODE. Thus, if Eq. 12.29 converges, it is guaranteed to converge to the correct steady-state solution of the ODE. Now we can factor Eq. 12.29 and maintain $O(h^2)$ accuracy. We arrive at the expression

$$\left[I - \frac{1}{2}hA_x\right]\left[I - \frac{1}{2}hA_y\right]\Delta U_n = h\Big[A_{x+y}U_n + (\vec{bc})\Big] + O(h^3) \tag{12.30}$$

This is the delta form of a factored, 2nd-order, 2-D equation.

[9]A block tridiagonal solver is similar to a scalar solver except that small block matrix operations replace the scalar ones, and matrix multiplications do not commute.

The point at which the factoring is made may not affect the order of time-accuracy, but it can have a profound effect on the stability and convergence properties of a method. For example, the unfactored form of a first-order method derived from the implicit Euler time march is given by Eq. 12.19, and if it is immediately factored, the factored form is presented in Eq. 12.20. On the other hand, the *delta form* of the unfactored Eq. 12.19 is

$$[I - hA_x - hA_y]\Delta U_n = h\Big[A_{x+y}U_n + (\vec{bc})\Big]$$

and *its* factored form becomes[10]

$$[I - hA_x][I - hA_y]\Delta U_n = h\Big[A_{x+y}U_n + (\vec{bc})\Big] \qquad (12.31)$$

In spite of the similarities in derivation, we will see in the next chapter that the convergence properties of Eq. 12.20 and Eq. 12.31 are vastly different.

12.7 Problems

1. Consider the 1-D heat equation:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} \qquad 0 \le x \le 9$$

Let $u(0,t) = 0$ and $u(9,t) = 0$, so that we can simplify the boundary conditions. Assume that second order central differencing is used, i.e.,

$$(\delta_{xx}u)_j = \frac{1}{\Delta x^2}(u_{j-1} - 2u_j + u_{j+1})$$

The uniform grid has $\Delta x = 1$ and 8 interior points.

(a) *Space vector definition*

i. What is the space vector for the natural ordering (monotonically increasing in index), $u^{(1)}$? Only include the interior points.

ii. If we reorder the points with the odd points first and then the even points, write the space vector, $u^{(2)}$?

iii. Write down the permutation matrices,(P_{12},P_{21}).

[10]Notice that the only difference between the $O(h^2)$ method given by Eq. 12.30 and the $O(h)$ method given by Eq. 12.31 is the appearance of the factor $\frac{1}{2}$ on the left side of the $O(h^2)$ method.

iv. The generic ODE representing the discrete form of the heat equation is

$$\frac{du^{(1)}}{dt} = A_1 u^{(1)} + f$$

Write down the matrix A_1. (Note $f = 0$, due to the boundary conditions) Next find the matrix A_2 such that

$$\frac{du^{(2)}}{dt} = A_2 u^{(2)}$$

Note that A_2 can be written as

$$A_2 = \left[\begin{array}{c|c} D & U^T \\ \hline U & D \end{array}\right]$$

Define D and U.

v. Applying implicit Euler time marching, write the delta form of the implicit algorithm. Comment on the form of the resulting implicit matrix operator.

(b) *System definition*

In problem 1a, we defined $u^{(1)}, u^{(2)}, A_1, A_2, P_{12}$, and P_{21} which partition the odd points from the even points. We can put such a partitioning to use. First define extraction operators

$$I^{(o)} = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c|c} I_4 & 0_4 \\ \hline 0_4 & 0_4 \end{array}\right]$$

$$I^{(e)} = \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} 0_4 & 0_4 \\ \hline 0_4 & I_4 \end{array}\right]$$

which extract the odd even points from $u^{(2)}$ as follows: $u^{(o)} = I^{(o)}u^{(2)}$ and $u^{(e)} = I^{(e)}u^{(2)}$.

i. Beginning with the ODE written in terms of $u^{(2)}$, define a splitting $A_2 = A_o + A_e$, such that A_o operates only on the odd terms, and A_e operates only on the even terms. Write out the matrices A_o and A_e. Also, write them in terms of D and U defined above.

ii. Apply implicit Euler time marching to the split ODE. Write down the delta form of the algorithm and the factored delta form. Comment on the order of the error terms.

iii. Examine the implicit operators for the factored delta form. Comment on their form. You should be able to argue that these are now trangular matrices (a lower and an upper). Comment on the solution process this gives us relative to the direct inversion of the original system.