8

Alternating Direction Implicit Method

The following Section deals with the so-called Alternating Direction Implicit Method (ADI). Its basics purpose is to address the increase of the computational complexity when one transitions from a one-dimensional to a two-dimensional beam propagation problem. We have already experienced this difficulty in a couple of contexts; for example direct application of the Crank-Nicolson approach in two transverse dimension that utilized a direct linear solver turned out to be significantly slower in comparison to simulations based on method of lines. Natural question then is if the favorable properties of the Crank-Nicolson approach can be somehow “salvaged,” or transferred to situations with two dimensional diffraction. The ADI method is one way to achieve this.

8.1 Basic formulation of ADI for free space diffraction

For simplicity, we only consider the Laplacian part of paraxial propagation equation, but this time in two transverse dimensions:

$$\partial_z E = \frac{i}{2k_0}[\partial_{xx} + \partial_{yy}]E \tag{8.1}$$

ADI consists in processing each of the two dimensions of the above Laplacian in turns, by splitting one complete integration step into two stages as follows:

$$\frac{E^{n+1/2} - E^n}{\Delta z/2} = \frac{i}{2k_0 \Delta x^2}\Delta_{xx}E^{n+1/2} + \frac{i}{2k_0 \Delta y^2}\Delta_{yy}E^n \tag{8.2}$$

$$\frac{E^{n+1} - E^{n+1/2}}{\Delta z/2} = \frac{i}{2k_0 \Delta x^2}\Delta_{xx}E^{n+1/2} + \frac{i}{2k_0 \Delta y^2}\Delta_{yy}E^{n+1} \tag{8.3}$$

Note than in each, one direction is treated implicitly, i.e. a second-order derivative acts on the new field sample to be calculated. Similarity with the C-N method may not be very obvious at this stage, because the characteristic C-N feature of averaging between two integration-step levels does not appear here. However once same-step field samples are collected together, and the update scheme is written with the help of the same matrices we used before to formulate the C-N method, the connection becomes obvious with

$$(1 - \frac{i\Delta z}{4k_0 \Delta x^2}\Delta_{xx})E^{n+1/2} = (1 + \frac{i\Delta z}{4k_0 \Delta y^2}\Delta_{yy})E^n$$

$$(1 - \frac{i\Delta z}{4k_0\Delta y^2}\Delta_{yy})E^{n+1} = (1 + \frac{i\Delta z}{4k_0\Delta x^2}\Delta_{xx})E^{n+1/2} \tag{8.4}$$

Notation can be further simplified, by specializing our familiar delta-parameter to two directions

$$\delta_x = \frac{i\Delta z}{4k_0\Delta x^2} \quad \delta_x = \frac{i\Delta z}{4k_0\Delta y^2} \tag{8.5}$$

and $L^{\pm}$ matrices to

$$L_x^{\pm} = 1 \pm \delta_x\Delta_{xx} \quad L_y^{\pm} = 1 \pm \delta_y\Delta_{yy} \tag{8.6}$$

Then the update can be written in a compact form

$$\begin{aligned} L_x^- E^{n+1/2} &= L_y^+ E^n \\ L_y^- E^{n+1} &= L_x^+ E^{n+1/2} \end{aligned} \tag{8.7}$$

For these expressions, one has to keep in mind that the above matrix multiplications are symbolic in the sense that if x matrices act on column vectors, y matrices must act on rows or the other way around. Of course, one can also express the whole update as a "single" matrix-based operation in Matlab-like expression

$$E^{n+1} = (L_y^-)^{-1}[L_x^+((L_x^-)^{-1}\{L_y^+E^n\}^T)]^T \tag{8.8}$$

However, this would not be a good approach for practical implementation. There is no need to create any matrices and/or their inverses explicitly.

A much more efficient way to program ADI is essentially a sequence of two Crank-Nicolson type steps, utilizing the same tri-diagonal solver technique. One minor modification consists in how the right-hand-side for each linear system is obtained, and another not so minor change is that what may seem as a single C-N step application here is actually a series of mutually independent actions, each working with a different row or column of our matrix representation of the beam.

8.2 ADI relation to Crank-Nicolson method

The scheme just derived reduces the basic operation to one that deals with single-dimensional arrays and only tri-diagonal matrices. This is indeed a great improvement when considering that the original problem seemed to require a sparse-matrix solver with dimensions $(N_xN_y)\times(N_xN_y)$. This seems too good to be true, so the question is what price in terms of accuracy, or perhaps (in)stability one has to pay?

To understand this better, it is a useful exercise to show that ADI is almost "equal" to the two-dimensional Crank-Nicolson. There appears a correction that is of second order in propagation step Δz and depends on higher-order spatial derivatives, but the overall truncation error turns out to be the same as in the C-N method.

Subtracting the two equations

$$\begin{aligned} (1 - i\delta_x\Delta_{xx})E^{n+1/2} &= (1 + i\delta_y\Delta_{yy})E^n \\ (1 - i\delta_y\Delta_{yy})E^{n+1} &= (1 + i\delta_x\Delta_{xx})E^{n+1/2} \end{aligned}, \tag{8.9}$$

one gets the expression for the intermediate step amplitude

$$2E^{n+1/2} = (1 + i\delta_y\Delta_{yy})E^n - (1 - i\delta_y\Delta_{yy})E^{n+1} \quad . \tag{8.10}$$

Inserting this into first equation results in the following

$$\frac{1}{2}(\frac{i}{2k_0\Delta x^2}\Delta_{xx}+\frac{i}{2k_0\Delta y^2}\Delta_{yy})(E^{n+1}+E^n)=\frac{E^{n+1}-E^n}{\Delta z}+\frac{\Delta z}{4}\Delta_{xx}\Delta_{yy}(E^{n+1}-E^n) \quad (8.11)$$

Obviously, the first two terms constitute the standard two-dimensional Crank-Nicolson method. The last term on the right is the correction which scales in the continuum limit as the high-order derivative $\partial_{xxyyz}E$, and is proportional to $\delta_x\delta_y$ on a discrete grid.

8.3 Dispersion properties of ADI method

As always, a most important issue is that of stability. Dispersion properties (i.e. accuracy and stability) of the ADI method also turn out to be closely similar to those of Crank-Nicolson approach.

The derivation method is standard, based on plane-wave anstaz. Start from the two-step scheme written in the compact form

$$L_x^- E^{n+1/2}=L_y^+ E^n \qquad L_y^- E^{n+1}=L_x^+ E^{n+1/2} \quad (8.12)$$

and, as is usual in derivation of dispersion relation, consider translationally symmetric solutions. Such plane-waves are characterized by two transverse wavenumbers k_x, k_y:

$$E^n=e^{in\Delta z\beta}e^{ik_x x+ik_y y} \quad (8.13)$$

for a whole n representing a finished integration step. At the "half step", the solution must also have the form of a plane wave, but possibly with a different amplitude

$$E^{n+1/2}=Ae^{ik_x x+ik_y y}\,. \quad (8.14)$$

With this parametrization, elimination of the unknown amplitude A gives an expression for the single step propagator in the spectral space

$$e^{i\Delta z\beta(k_x,k_y)}=\frac{1+2i\delta_x(\cos k_x\Delta x-1)}{1-2i\delta_x(\cos k_x\Delta x-1)}\frac{1+2i\delta_y(\cos k_y\Delta y-1)}{1-2i\delta_y(\cos k_y\Delta y-1)} \quad (8.15)$$

This formula is a pure phase on the right hand side, and that means that the numerical-wave propagation constant $\beta(k_x, k_y)$ is real, and the method is consequently unconditionally stable. Dispersion properties appear to receive contributions from each transverse direction

$$\Delta z\beta(k_x,k_y)=Arg\left(\frac{1+2i\delta_x(\cos k_x\Delta x-1)}{1-2i\delta_x(\cos k_x\Delta x-1)}\right)+Arg\left(\frac{1+2i\delta_y(\cos k_y\Delta y-1)}{1-2i\delta_y(\cos k_y\Delta y-1)}\right)\,, \quad (8.16)$$

where both terms are familiar C-N expressions. Taylor expansion shows that in the continuum limit of small k_x, k_y this reduces to

$$\beta(k_x,k_y)=-\frac{k_x^2}{2k_0}-\frac{k_y^2}{2k_0}+\frac{\Delta x^2k_x^4}{24k_0}+\frac{\Delta y^2k_y^4}{24k_0} \quad (8.17)$$

So, at order four there are no mixed terms which implies that ADI starts to differ from C-N only in order six. This shows that ADI indeed preserves most of the desired properties built in the C-N algorithm. It is also evident that ADI shares the unpleasant properties, an in particular the anisotropy of the grid shows up in the numerical wave propagation at the same level.

8.4 ADI implementation

Let us assume that the current solution array $E^n_{x,y}$ has been calculated at step n along the propagation direction. Here, indices x, y stand for matrix array indices in the corresponding directions, and asterisk $*$ will indicate a wild card index running over all components of a vector in the corresponding dimension. It is a way to indicate what would be a vector operation in Matlab-like notation.

The following is a summary of the ADI algorithm written in a pseudo-code

1. Calculate the first-stage right hand side $R = L^+_y E^n$:
   Operator L_y does not depend on index x, and that means that all operations can be performed simultaneously with whole column vectors:
   $$\text{loop}_y \ : \ R_{*,y} = E^n_{*,y} + \delta_y (E^n_{*,y-1} - 2E^n_{*,y} + E^n_{*,y+1})$$
   Note that this is the same type of calculation we have done for a one-dimensional C-N method, but executed in a loop for each row independently.
2. Next, look at matrix R as a collection of rows. For each index y solve a tri-diagonal C-N-like system with each column of R playing the role of right-hand-side vector in the linear system
   $$\text{loop}_y \ : \ \text{TRIDIAG}(A^-_x, B^-_x, C^-_x, R_{*,y}, E^{n+1/2}_{*,y}) \ ,$$
   where A, B, C stand for the three diagonals of the system matrix, and the result goes in $E^{n+1/2}$.
3. Calculate the right hand side for the second-stage linear problem $U = L^+_x E^{n+1/2}$:
   $$\text{loop}_x \ : \ U_{x,*} = E^{n+1/2}_{x,*} + \delta_x (E^{n+1/2}_{x-1,*} - 2E^{n+1/2}_{x,*} + E^{n+1/2}_{x+1,*}) \ .$$
   In comparison to stage 1., dimensions x and y have exchanged their roles.
4. For each index x solve a tri-diagonal C-N-like system with rows of U fed as right-hand-sides of linear-system solves:
   $$\text{loop}_x \ : \ \text{TRIDIAG}(A^-_y, B^-_y, C^-_y, U_{x,*}, E^{n+1}_{x,*})$$

This constitutes on complete step in ADI method. There are two favorable properties of this algorithm. First, it can be efficiently parallelized, because all operations work with vectors (either rows or columns of matrices) and are mutually independent. Second, the tri-diagonal solver remains the only linear-algebra routine needed for the implementation.