0.1 Practice track: BPM with a tri-diagonal linear solver

Summary: This practice-track package departs from the 1D method implemented in the previous section, and illustrates the following:

- Modification for the radially symmetric beam solution.
- Validation against exact solutions, comparison with the Discrete Hankel Transform method.
- Solution with the help of the tri-diagonal matrix solver (Thomas algorithm).
- Packaging of the one-dimensional C-N solver for usage in more complex algorithms.

0.1.1 Thomas algorithm and radially symmetric, finite-difference BPM

Task 1:
Implement a solver for a linear system of equations with the left-hand-side given by a matrix that only has non-zero elements on the main diagonal and on the upper and lower sub-diagonal. Assume that the solver will only be used for problems that arise in the BPM context, with the implication that the Gauss elimination forward step does not need to check for non-zero pivot. Your goal is a fast, though specialized solver.

Solution:

The very first choice to make when implementing the solver is to decide how the input values will be passed. The most practical way is to pass to the solver function the size of the matrix, plus arrays (as pointers in C) that carry the lower, main, and upper diagonal. Of course, anther array is needed for the right-hand-side vector, and one more to store the solution.

The algorithm is a simplified version of the Gauss elimination. An example of how it could be written (in C) is shown next:

Listing 1: Thomas algorithm

```
1  void TDMsolver(int n,complex *a,complex *b,complex *c,complex *v,complex *x)
2  {
3     /*
4      * n - number of equations
5      * a - lower-diagonal -- indices used = 1...n-1
6      * b - the main diagonal
7      * c - upper-diagonal -- indices used = 0...n-2
8      * v - right hand side vector
9      * x - solution holder
10     * NOTE: array b will be DESTROYED!!!
11     */
12
13     /* Gauss elimination of the sub-diagonal elements */
```

```
14      for (int i = 1; i < n; i++)
15        {
16         comple m = a[i]/b[i-1]; // no need to guard against b[i-1]=0
17         b[i] = b[i] - m*c[i-1];
18         v[i] = v[i] - m*v[i-1];
19        }
20
21        /* Backward substitution */
22        x[n-1] = v[n-1]/b[n-1];
23
24        for (int i = n - 2; i >= 0; i--)
25        x[i]=(v[i]-c[i]*x[i+1])/b[i];
26  }
```

This listing show a simple program to realize what is the Thomas algorithm to solve a tri-diagonal system of linear algebraic equations. There are two properties that are worthwhile of a note. The first is about the non-vanishing matrix element, $b[i-1]$ in line 16 used to eliminate the sub-diagonal value in line i. Normally, whenever one divides by a value, a check that it is not zero is in order. Fortunately, here we do not need to do such a check. The result is a faster running code, which the compiler optimizer has a better chance to improve in terms of performance. This is of utmost importance, because the algorithm shown above may be called hundreds of times during each propagation step in a beam propagation method, such as alternate direction implicit method.

But how do we actually know that there will be no division by zero in line 16? Clearly, it can not be true for a general tri-diagonal system. However, for a BPM application we can 'predict' this from what we know about physics of diffraction. Let us assume that a zero occurs at $b[i]$. It happens during the elimination of the sub-diagonal element in the same matrix row. As a result there is only one non-zero value in this particular row after this step, and that will imply that $x[i] = 0$. From here, one could start the backward substitution and evaluate all $x[j], j < i$. It would also mean that the solution vector splits at position i into two sections that "do not talk to each other," and can be calculated independently. This is, however, unphysical because in a homogeneous material diffraction *must* communicate information across the point i from left to right and in the opposite direction. Thus, for a diffraction problem type, one does not expect to encounter a situation in which $b[i-1]$ vanishes.

Another important property of the Thomas algorithm is its stability. It means that numerical noise does not grow during the solution even if the size of the problem, i.e. length of the vector x is large. I other words, this simple algorithm can be applied in situations requiring large computational domains.

The favorable properties of this algorithm can be shown shown rigorously for the so-called diagonal dominant systems, in which

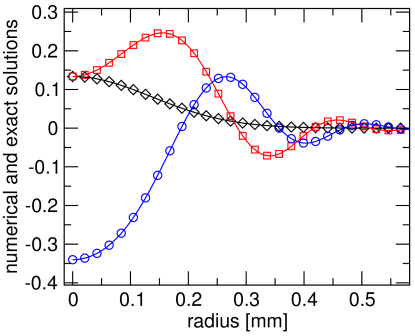
$$|b[i]| \geq |a[i]| + |c[i]| \ .$$

This constraint is just satisfied in a one-dimensional diffraction problem.

Task 2:
Implement a CN-based BPM simulator for radially symmetric solutions. Try to come up with a re-usable design, producing an object that will be utilized in the subsequent practice sessions of this course.

Solution:
The instructor's solution, written in C++, is placed in file *cn-bpm-radial.cc*. The reader is encouraged to utilize this program for inspiration, but write her/his own, perhaps in a different language. Since this and the one-dimensional Crank-Nicolson algorithm will be re-used multiple times during this course, the reader should attempt to design the CN solver as an object that encapsulates/hides all auxiliary variables and can be used as a building block for more complex algorithms.

Task 3: Validate your simulator in a comparison with the exact solution of a Gaussian beam. The solution to this task is completely analogous to a similar task for one-dimensional beam propagation practice session. The only difference lies in using the analytic formula for a Gaussian beam in two transverse dimensions. This should be a straightforward exercise and is left to the reader. The following figure can be used as a guide for what this exercise should produce in terms of numerical-vs-exact solution comparison



Testing the finite-difference BPM implementation. Initially collimated Gaussian beam was numerically propagated over a distance corresponding to several Rayleigh ranges — a length sufficient to produce significant change in the beam profile, so that the comparison is non-trivial. The numerical (symbols) solution is then plotted against the analytic (lines). One should plot both the real and imaginary parts of the amplitude together with its modulus. The two kinds of solution should appear essentially identical on the scale of the figure.