3

Basics of finite-difference beam propagation methods

3.1 Paraxial beam propagation equations

The beam propagation solutions were obtained in the previous sections without explicit reference to propagation equations. The sole guiding principle was the linearity of Maxwell equations and from it following superposition principle. That made it possible to construct any solution from the basic modal fields, such as plane waves or conical Bessel beams. However, most of beam propagation methods depart from some sort of propagation equations derived from the Maxwell system. This section concentrates on the simplest case of paraxial equations in a homogeneous medium. Generalizations to weakly nonlinear and/or weakly guiding structures will also be discussed.

3.1.1 Spatial-Spectral Representation

The most general wave equation for a dispersive medium characterized by a frequency dependent permittivity, and response embodied in current density $\mathbf{J}$ and in polarization $\mathbf{P}$ can be written as

$$\nabla^2 \mathbf{E} - \nabla(\nabla \cdot \mathbf{E}) - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\int_{-\infty}^{t} \epsilon(t-t')\mathbf{E}(\mathbf{r},t',z)dt' = \mu_0 \left(\frac{\partial \mathbf{J}}{\partial t} + \frac{\partial^2 \mathbf{P}}{\partial t^2} \right) \; . \tag{3.1}$$

Linear properties of the medium are usually measured in terms of the frequency dependent, complex-valued permittivity $\epsilon(\omega)$. The electric induction then relates to the electric intensity through the constitutive relation

$$\hat{\mathbf{D}}(\mathbf{r},\omega,z) = \epsilon_0\epsilon(\omega)\hat{\mathbf{E}}(\mathbf{r},\omega,z) + \hat{\mathbf{P}}(\mathbf{r},\omega,z), \tag{3.2}$$

where the vector fields are partial Fourier transforms with respect to time. In such a spectral representation, the wave equation attains a more friendly look:

$$\nabla^2 \hat{\mathbf{E}} - \nabla(\nabla \cdot \hat{\mathbf{E}}) + \frac{\omega^2 n^2(\omega)}{c^2}\hat{\mathbf{E}} = \mu_0 \left(-i\omega\hat{\mathbf{J}} - \omega^2 \hat{\mathbf{P}} \right) \; . \tag{3.3}$$

This is in fact a version of Helmholtz equation with the driving terms on the right hand side tacitly understood as being driven by the electric field. This is a frequent departure point for derivation of beam propagation equations.

3.1.2 Paraxial Beam Propagation Equation in a Homogeneous Medium

We start our gradual building of various finite-difference BPM techniques with the simplest possible case. Assuming

- homogeneous medium;
- single frequency (Continuous Wave regime);
- no other material response then the medium‘s own refractive index;

the wave (Helmholtz) equation, (3.3) reduces to

$$\nabla^2 \hat{\mathbf{E}} + \frac{\omega^2 n^2(\omega)}{c^2} \hat{\mathbf{E}} = 0 \ . \tag{3.4}$$

The divergence term disappeared because $\nabla \cdot \hat{\mathbf{E}}$ can only be non-vanishing if the medium generates a spatially inhomogeneous polarization response which we assumed to be absent in this case. We thus have a true Helmholtz equation to solve.

The next standard step is to write the forward (i.e. positive z) propagating wave in the envelope form

$$E = A(\boldsymbol{r}) e^{-i\omega t + ik_0 z} \quad \text{with} \quad k_0 = \frac{\omega n(\omega)}{c} \ . \tag{3.5}$$

Let it be noted immediately that nothing prevents the above ansatz to represent a waveform that actually propagates in the backward direction. It is only when $A(\boldsymbol{r})$ changes slowly with respect to z that the carrier wave can alone determine in which direction the propagation occurs. This is what will be assumed shortly, but let us remember that such an assumption cannot be self-consistently checked within the BPM framework.

The Helmholtz equation (3.4) implies the following for the complex-valued amplitude A:

$$\partial_{zz} A + 2ik_0 \partial_z A + (-1) k_0^2 A + \partial_{xx} A + \partial_{yy} A + k_0^2 A = 0 \ . \tag{3.6}$$

This is where approximations must be adopted. In particular, in the **slowly varying envelope approximation** (SVEA), the first term is neglected,

$$\partial_{zz} A + 2ik_0 \partial_z A \to 2ik_0 \partial_z A \ . \tag{3.7}$$

In other words, one assumes that the envelope changes little in comparison to the change experienced by the carrier wave:

$$|\partial_{zz} A| << k_0 |\partial_z A| \ . \tag{3.8}$$

This allows to neglect the second partial derivative with respect to z, and leads to the paraxial beam propagation equation

$$\partial_z A = \frac{i}{2k_0} (\partial_{xx} A + \partial_{yy} A) \ . \tag{3.9}$$

This is the system for which we want to developed some numerical techniques. We will see that the algorithms will be applied in a number of more general situations.

This equation has been characterized as *paraxial*, but in the course of its derivation paraxiality was not explicitly required. From the approaximation made it may not be clear that the field this equation describes must have a narrow spatial spectrum and only encompass small propagation angles. To see what wave propagation properties this equation implies it helps to transform it into (spatial) spectral domain. Fourier transforming (3.9) gives

$$\partial_z \hat{A}(k_x, k_y, z) = -\frac{i}{2k_0} (k_x^2 + k_y^2) \hat{A} \ , \tag{3.10}$$

which is easy to solve for an arbitrary initial condition $\hat{A}(k_x, k_y, z = 0)$.

$$A(k_x, k_y, z) = \exp\left[\frac{-iz}{2k_0}(k_x^2 + k_y^2)\right] A(k_x, k_y, 0) \ , \tag{3.11}$$

and here one can recognize the paraxial propagator we have encountered in connection with spectral methods. This means, in particular, that plane wave solutions propagate with propagation constants that are paraxial approximations of their exact counterparts. Hence the equation is rightly termed as paraxial.

It is interesting to note that paraxiality emerged as a consequence of the slowly evolving envelope approximation. The latter also assumes that the beam propagation is one-way. It may not seem related, but recall that the wave equation does not have any preferred direction. The assumption that all solutions propagating in any of the "forbidden" directions are somehow eliminated must result in a severe approximation. It should become more obvious in the latter stages of this course that between the slowly evolving envelope approximation and the one-way propagation assumption, it is the latter that implies paraxial nature of the resulting beam propagation equations.

The paraxial beam propagation equation admits several families of exact solutions. These are very useful, particularly for our purposes. They can be utilized to test various algorithms and implementations of beam propagation. Gaussian and Airy beams are two kinds of exact solutions we will use frequently in our tests and mock simulations.

Exercise: This problem deals with so-called Airy beams. Consider a scaled paraxial beam propagation equation in one transverse dimension,

$$i\partial_z A + \frac{1}{2}\partial_{xx} A(x, z)$$

and show that the following Airy beam

$$A(x, z) = Ai[x - (z/2)^2] \exp[ixz/2 - iz^3/12]$$

is an exact solution. Also:
a) plot the intensity of this solution in a range of (x, z) — you should observe that intensity maxima propagate along parabolic trajectories;
b) write down the solution for the non-scaled version of the propagation equation; and
c) generalize this solution to two transverse dimensions.

3.1.3 Paraxial Beam Propagation Equation in a Weakly Inhomogeneous Medium

The simplest generalization of the homogeneous paraxial propagation equation derived in the previous subsection includes weak effects of medium response, or small inhomogeneities in the refractive index. Both cases can be treated jointly by representing them as contributions to the polarization in Eqn.(3.3)

$$P = \epsilon_0 \Delta\chi(\boldsymbol{r}) E \ . \tag{3.12}$$

Here, the modification of the local medium susceptibility $\Delta\chi$ can be either a spatial variation of the refractive index, or it can be a result of the medium's nonlinear response, for example due to optical Kerr effect. The paraxial propagation equation is derived the same way as above while keeping this polarization term on the right-hand-side. Similar to the electric field expressed in terms of its envelope and a carrier wave,

$$E = \mathcal{E}(\boldsymbol{r})e^{-i\omega t + ik_0 z} \tag{3.13}$$

polarization envelope is defined through

$$P = \mathcal{P}(\boldsymbol{r})e^{-i\omega t + ik_0 z} \ , \tag{3.14}$$

i.e. sharing the same carrier with the electric field. The resulting propagation equation reads

$$\partial_z \mathcal{E} = \frac{i}{2k_0}(\partial_{xx}\mathcal{E} + \partial_{yy}\mathcal{E}) + \frac{i\omega}{2c\epsilon_0 n(\omega)}\mathcal{P} \ . \tag{3.15}$$

In the above derivation, it seems that no approximation related to the nature of the polarization term was needed. However, we can show easily that the polarization term must be a small perturbation.

Having a particular beam-propagation equation, or even a laser-pulse propagation equation, it is not really necessary to know what assumptions underlined its derivation. One way to read these equations, and understand what approximations they impose on wave propagation, is to transform them into spectral representation. That usually allows to construct dispersion relations.

For equation (3.15), it is instructive to consider a special situation in which the real medium is homogeneous, but we view its refractive index as a "sum" of the reference index $n(\omega)$, plus a modification that is represented by the polarization term through a susceptibility modification $\Delta\chi$. Then, the constant (in space) polarization that is a result of a modified refractive index is

$$P = \epsilon_0 \Delta\chi E \tag{3.16}$$

so that the "total" physical refractive index of the medium

$$n_{phys}^2 = n(\omega)^2 + \Delta\chi \tag{3.17}$$

is related to the "reference" refractive index n that was assumed for $k_0 = \omega n/c$ in the carrier wave. n thus determines what is the relation between the physical field and its envelope, but it is an artificial quantity that we could choose arbitrarily, and that is why the propagation properties of waves should not depend on it — this is because the wave equation only "knows" about the true n_{phys}. So, the equation now reads

$$\partial_z \mathcal{E} = \frac{i}{2k_0}(\partial_{xx}\mathcal{E} + \partial_{yy}\mathcal{E}) + \frac{i\omega}{2cn(\omega)}\Delta\chi\mathcal{E}. \tag{3.18}$$

Transforming to the spectral representation (i.e. taking its Fourier transform w.r.t. x, y) leads to

$$\partial_z \hat{\mathcal{E}} = i\left[\frac{-1}{2k_0}(k_x^2 + k_y^2) + \frac{i\omega}{2cn(\omega)}\Delta\chi\right]\hat{\mathcal{E}} \tag{3.19}$$

which has a solution

$$E = e^{-i\omega t + ik_0 z}\exp\left[i\left(\frac{-1}{2k_0}(k_x^2 + k_y^2) + \frac{i\omega}{2cn(\omega)}\Delta\chi\right)z\right] \tag{3.20}$$

that says that the propagation constant of a plane wave with the transverse wave-vector $k_{x,y}$ is

$$k_0 + \frac{-1}{2k_0}(k_x^2 + k_y^2) + \frac{\omega}{2cn(\omega)}\Delta\chi = \frac{\omega}{c}\left[n(\omega) + \frac{\Delta\chi}{2n(\omega)}\right] - \frac{1}{2k_0}(k_x^2 + k_y^2) \ . \tag{3.21}$$

However, for the homogeneous medium with the refractive index of $n_{phys} = \sqrt{n^2 + \Delta\chi}$, the exact propagation constant should rather be

$$\sqrt{\frac{\omega^2[n(\omega)^2 + \Delta\chi]}{c^2} + \left[k_x^2 + k_y^2\right]^2} \ . \tag{3.22}$$

It is clear that the propagation constant "produced" by the equation is an approximation, and namely a Taylor expansion in both the transverse wavenumbers *and* in $\Delta\chi$. Thus, the latter must be small in comparison to n^2 for the equation to have accurate dispersion properties.

3.2 Finite-Difference basics

Our Finite-Difference-based BPM treatments will start with the simplest possible equation, namely

$$\frac{\partial \mathcal{E}}{\partial z} = \frac{i}{2k_0} \Delta_\perp \mathcal{E} \tag{3.23}$$

which only describes effects of diffraction on a beam in a homogeneous medium. The Laplacian operator $\Delta_\perp$ appears as a building block in many partial differential equations. It is present, in one form or the other, in the optical beam and also in laser-pulse propagation equations. Hence, this is a good simple problem to start developing our "numerical toolkit" — complexity will be added later as we build up more realistic propagation models and include other physically important effects.

The two most frequent version of $\Delta_\perp$ encountered in this course are the one-dimensional Laplacian,

$$\Delta_\perp \equiv \frac{\partial^2}{\partial x^2} \tag{3.24}$$

and a radial operator for functions with axial symmetry,

$$\Delta_\perp \equiv \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} \ . \tag{3.25}$$

These two cases can be treated together, with the only substantial difference between them being the treatment of the boundary conditions, when the point $r = 0$ (i.e. on-axis) needs special treatment. It may seem that the radial differential operator is singular on the optical axis, $r = 0$, but this is only a so-called coordinate singularity — it arises because in polar coordinates all points $(r = 0, \phi)$ represent a single point $(x = 0, y = 0)$ in Cartesian coordinates. If there is no real singularity in the physical field, and this is indeed what we assume about our beams, the occurrence of $1/r$ should not, and will not cause any serious problem with the discretization.

The following notation is used in what follows:

- propagation direction discretization: $z_n = n\Delta z$, $n = 0 \dots N_z$
- transverse grid in one-dimensional case: $x_j = x_{\min} + j\Delta x$, $j = 0 \dots N-1$
- radial grid: $r_j = j\Delta r$, $j = 0 \dots N-1$
- field sample at grid the point and the step labeled j and n, respectively: $E^n_j = \mathcal{E}(r = r_j, z = z_n)$.

The differential operators in Eq.(3.23), will be approximated by the so-called finite-difference expression we have first encountered in our excursion into basics of direct Maxwell solvers. For example, the evolution operator ∂_z can be discretized as:

$$\frac{\partial \mathcal{E}}{\partial z}(r = r_j, z = z_n) \simeq \frac{E^{n+1}_j - E^n_j}{\Delta z} + O(\Delta z), \tag{3.26}$$

which is the simplest possible approximation, and is only first-order accurate in Δz. However, the same formula is second-order accurate if used to estimate the derivative at a point half-way between z_n and z_{n+1}. To see how this happens, consider the above expression in which the electric field function values are represented in term of Taylor expansion in the vicinity of a chosen point z_c:

$$\begin{aligned}\frac{E^{n+1}_j - E^n_j}{\Delta z} = \frac{1}{\Delta z}[\ &\big(E(z_c) + E'(z_c)(z_{n+1} - z_c) + \tfrac{1}{2}E''(z_c)(z_{n+1} - z_c)^2\big) - \\ &\big(E(z_c) + E'(z_c)(z_n \quad - z_c) + \tfrac{1}{2}E''(z_c)(z_n \quad - z_c)^2\big) + \dots]\end{aligned} \tag{3.27}$$

where ... stand for higher-order Taylor terms, and primes indicate derivatives with respect to z. Clearly, if $z_c = (z_{n+1} + z_n)/2$, i.e. if we choose to evaluate the derivative in the center between the discrete field samples, the second-order terms cancel each other, and we obtain

$$\frac{E_j^{n+1} - E_j^n}{\Delta z} = E'(z_c) + O(\Delta z^2) \ , \tag{3.28}$$

namely a second-order accurate derivative estimate. This property will be used shortly in the derivation of the Crank-Nicolson method.

A second-order accurate discretization of the second-order spatial derivative ∂_{xx} reads:

$$\frac{\partial^2 \mathcal{E}}{\partial x^2}(x = x_j, z = z_n) \simeq \frac{E_{j+1}^n - 2E_j^n + E_{j-1}^n}{\Delta x^2} + O(\Delta x^2). \tag{3.29}$$

Recall that the second order accuracy in (3.29) originates in a) symmetry of the discretization stencil w.r.t. the point at which the derivative is calculated, and b) the constant constant grid spacing — violation of either decreases accuracy order to one. This can be shown the same way as above, using the Taylor expanded function to express its values at the discrete grid points.

It is useful to realize that the above structure of the Laplacian discretization is quite general. What the formula (3.29) says is this: Take difference between the field value at the nearest neighbor grid point and the value at the point where the Laplacian is being evaluated, and add such contributions from all nearest neighbors. For example, for a cubic lattice the discrete Laplacian is the sum over the six neighbors minus six times the central value, all divided by square of lattice constant. For a square lattice with grid spacing Δx,

$$\Delta_\perp E_{i,j} \approx \frac{E_{i+1,j} + E_{i,j+1} + E_{i-1,j} + E_{i,j-1} - 4E_{i,j}}{\Delta x^2} \ . \tag{3.30}$$

Now, let us look at the discretization of the radial Laplacian,

$$\Delta_\perp E(r) \equiv \partial^2 E(r)/\partial r^2 + (1/r)\partial E(r)/\partial r \ . \tag{3.31}$$

First consider radial points away from the axis $r = 0$. Then the two terms can be approximated separately, each with a second-order accurate expression centered around the grid point r_j, $j > 0$:

$$\Delta_\perp E(r_j) \approx \frac{E_{j+1} - 2E_j + E_{j-1}}{\Delta r^2} + \frac{1}{r_j}\frac{E_{j+1} - E_{j-1}}{2\Delta r} \tag{3.32}$$

Because $r_j = j\Delta_r$, both terms scale with Δ_r^2 as they should. Also, the second term becomes less and less important with increasing j — this is because in a large distance from the axis, the radially symmetric function behaves more and more as a one-dimensional one, and its Laplacian reduces to the first term. On the other hand, the above formula can not be used in the center where $r = 0, j = 0$. A different approximation must be applied to this grid point, and that follows readily from the Cartesian expression for the Laplacian (3.30):

$$\Delta_\perp E(r_0 = 0) \approx \frac{1}{\Delta r^2}\left[E(r_1, \phi = 0) + E(r_1, \phi = \pi/2) + E(r_1, \phi = \pi) + E(r_1, \phi = 3/2\pi) - 4E(r_0)\right] \tag{3.33}$$

Since the field is assumed to be radially symmetric it does not depend on the polar angle ϕ, and the above reduces to

$$\Delta_\perp E(r_0 = 0) \approx \frac{4(E_1 - E_0)}{\Delta r^2} \ . \tag{3.34}$$

This has the same accuracy and is thus compatible with the discrete Laplacian (3.32) away from the axis.

Note that error estimates in these discrete derivative and Laplacian estimates indicate the so-called *local truncation error*. This is introduced in the numerical solution at each particular step, and its magnitude is estimated assuming the solution at the previous step was exact. In practice, it is of course not exact, unless it corresponds to the initial condition. Local errors enter the simulation at each integration step, and subsequently "propagate" through the solution, giving rise to the *global error*. The numerical algorithm must control both kinds of errors — this may require adaptive control of the integration step.

3.3 Exercise in futility: An explicit method

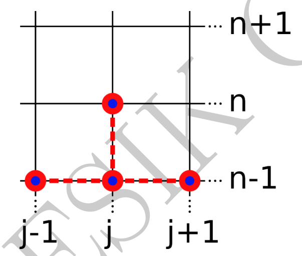
The finite-difference approximations for derivatives that appear in the paraxial beam propagation equations are here applied to construction of an *explicit* method. To keep everything simple, only the one-dimensional paraxial beam propagation equation in the form

$$\partial_z E = \frac{i}{2k_0}\partial_{xx} E \tag{3.35}$$

is considered. We will utilize this to demonstrate both the construction of the discretization, and the calculation of the corresponding dispersion relation that in turn will be used to determine the stability properties.

3.3.1 Discretization

To set up a discretized version of the propagation equation, we use the so-called stencil which marks the vertices of the grid that contribute to the discretized representation in the vicinity of a given point. Let us label our grid points with the index j and n for the transverse (x) and propagation direction (z). The simplest explicit discretization stencil looks like this:



The lower "row" represents the grid points carrying the field samples

$$E^n_{j-1} \equiv E(n\Delta z, (j-1)\Delta x) \quad , \quad E^n_j \equiv E(n\Delta z, j\Delta x) \quad , \quad E^n_{j+1} \equiv E(n\Delta z, (j+1)\Delta x) \tag{3.36}$$

that act as a source to evaluate the new, propagated value at the "upper" point labeled $(j, n+1)$

$$E^{n+1}_j \equiv E((n+1)\Delta z, j\Delta x) \ . \tag{3.37}$$

Because there is only this one new value to calculate, it can be expressed explicitly as a function of its inputs in the lower row. This is why the method is called explicit.

Let us use this stencil to set up finite difference representation. First, the partial derivative in the propagation direction is approximated "along the vertical line of the stencil" as:

$$\partial_z E \approx \frac{1}{\Delta z}(E^{n+1}_i - E^n_i) \ . \tag{3.38}$$

Then the second spatial derivative along x is approximated along the horizontal direction of the stencil, using the Laplacian approximation from the previous section:

$$\frac{i}{2k_0}\frac{1}{\Delta x^2}(E^n_{i-1} - 2E^n_i + E^n_{i+1}) \approx \frac{i}{2k_0}\partial_{xx}E \ . \tag{3.39}$$

Putting the two together, the new field variable can be explicitly obtained as

$$E^{n+1}_i = E^n_i + \frac{i\Delta z}{2k_0\Delta x^2}(E^n_{i-1} - 2E^n_i + E^n_{i+1}) \tag{3.40}$$

This constitutes an explicit update scheme. Its local discretization error scales as

$$\delta E = O(\Delta z) + O(\Delta x^2) \tag{3.41}$$

and the method is therefore second order in space, but only first order accurate in "time" or in the propagation direction.

The obvious attractive property of this scheme is simplicity. It only takes a few operations to update one grid point, and it is local. Locality is a rather desirable property, since it means that the updates can be performed concurrently at distant locations. If this method worked, it would be easy to parallelize thanks to its locality. Unfortunately, it does not work — for reasons discussed next.

3.3.2 Numerical dispersion and stability

To derive the dispersion properties of the explicit update proposed above, we apply the same procedure we have used for analysis of the Yee-scheme Maxwell-solver update. First, seek a solution in the form of a (one-dimensional) plane wave:

$$E^n_j \equiv E(n\Delta z, j\Delta x) = Ae^{iK_z z_n}e^{ik_x x_j} \tag{3.42}$$

Here, k_x is a freely chosen transverse wavenumber, and the resulting propagation "constant" K_z has to be calculated. As usual, this ansatz is an expression of the translational symmetry of the propagation problem.

Inserted into the discretized version of the propagation equation, the ansatz produces an equation that the amplitude A must satisfy,

$$Ae^{iK_z\Delta z} = A + i\delta A(e^{-ik_x\Delta x} - 2 + e^{ik_x\Delta x}) \quad , \quad \delta = \frac{\Delta z}{2k_0\Delta x^2} \ . \tag{3.43}$$

For a nontrivial solution A to exist, the solvability condition reads

$$e^{iK_z\Delta z} = 1 + 2i\delta(\cos k_x\Delta x - 1) \tag{3.44}$$

This is a form of a dispersion relation that determines the longitudinal propagation constant K_z as a function of a given k_x. It is easy to show that in the continuum limit $(\Delta z, \Delta x \to 0)$ one obtains the paraxial dispersion relation

$$K_z = -\frac{1}{2}\frac{k_x^2}{k_0} \ . \tag{3.45}$$

While the continuum limit is as it should be, for any finite grid spacings the propagation constant attains an imaginary part. For the ansatz to represent or approximate a physical plane-wave propagating through free space, the modulus of $e^{iK_z \Delta z}$ should be equal to one. However, the dispersion relation implies

$$|e^{iK_z \Delta z}|^2 = 1 + \frac{\Delta z^2 \sin^4 \frac{k\Delta x}{2}}{4\Delta x^4 k_0^2} > 1 \tag{3.46}$$

This means that no matter how fine the discretization, and how short the integration step, all waves (and that includes any unwanted numerical noise) with nonzero transverse wavenumber k_x are amplified. Thus this simple explicit method is **unstable**.

Exercise: Write a program to implement method (3.40). Produce a simulation to illustrate how is the method‘s instability manifested in numerical solutions.

3.4 Ensuring stability: An implicit method

Because the instability of the update scheme will eventually show up as an exponentially increasing "noise," it will destroy the numerical solution — an unstable method is indeed of very little use. One can suspect the the main culprit is the explicitness of the scheme, and that one could try an implicit method instead.

This can be done readily by flipping the discretization stencil of the previous section vertically, thus replacing $n \leftrightarrow n+1$ and $\Delta z \to -\Delta z$ in (3.40):

$$E_i^n = E_i^{n+1} - \frac{i\Delta z}{2k_0 \Delta x^2}(E_{i-1}^{n+1} - 2E_i^{n+1} + E_{i+1}^{n+1}) \,. \tag{3.47}$$

This scheme is called implicit, because it is not straightforward to isolate each of the updated field samples. Instead, the above represents a system of equations for E_j^{n+1} in which the number of unknowns equals the number of spatial grid points. For a practical solution, boundary conditions must be specified somehow, and the corresponding equations added to the system. It is clear that the numerical effort for one integration step $n \to n+1$ is much larger than in the case of the explicit method. On the other hand, the implicit property brings with it the desired stability.

To calculate the dispersion relation, the plane-wave based ansatz procedure can be repeated once again to obtain

$$e^{iK_z \Delta z} = \frac{1}{1 - 2i\delta(\cos k_x \Delta x - 1)} \tag{3.48}$$

This leads to the same continuum limit as the explicit method. The important difference is that the implicit scheme is stable, because it follows from the above that

$$|e^{iK_z \Delta z}| < 1 \tag{3.49}$$

for any finite discretization steps and non-zero wavenumber. Thus, the numerical noise does not grow, but it is damped instead and the method becomes stable.

However, not only numerical noise gets damped. All waves that constitute the solution will decrease in amplitude, and this means that the method introduces artificial losses into propagation. While not as destructive as instability, artificial damping is not very desirable property either, because it will, too, degrade the accuracy of the solution, albeit at a slower pace.

Put in other words, the implicit method does not conserve energy of the beam even in the loss-less free space propagation. Moreover, it is still only first-order accurate in the propagation direction, and for these reasons this update scheme is not much more useful than the explicit one. Its value for us is that is shows the way toward a stable and more accurate scheme — which will be a "mixture" of the two methods discussed so far.

3.5 Practice track: Simple finite-difference methods

3.5.1 Explicit finite-difference scheme for paraxial beam propagation

Summary:

- Numerical stability is arguably the most important property of any numerical algorithm
- The simplest scheme to apply to paraxial propagation is explicit. Unfortunately, it turns out to be unconditionally unstable, and therefore practically useless.
- The purpose of this practice package is to illustrate manifestations of numerical instability:
  - * for a while, nothing bad happens
  - * suddenly, noise develops, most often in the form "even-odd oscillations"
  - * the perturbation(s) grows extremely fast

We have shown in the theory track of this course that a simple explicit method applied to a paraxial beam propagation equation is unstable. Worse, it is *unconditionally* unstable, meaning that the numerical instability can not be avoided in a sufficiently long simulation. No matter how one sets the parameters, and in particular the length of the integration step, sooner or later noise that is necessarily present in the solution will be amplified and corrupt the solution entirely. This exercise is meant to demonstrate that the instability is in no way a negligible drawback that perhaps could be dealt with "by some other means."

Here we will look at a simple implementation of an explicit integration scheme for a paraxial beam propagation equation. Our goal is to see for ourselves the onset of numerical instability. The important take-away lesson is that while a more conservative choice of integration step can delay the point at which numerical noise overwhelms the solution, the eventual disaster can not be avoided. Thus, it is necessary to design any PDE solver such that the update scheme is at least conditionally stable.

Task 1:
Implement a program to simulate the evolution for an initial Gaussian beam propagating at a small angle with respect to the optical axis. Use the explicit, or "Euler" method as desscribed in the theory track, and implement it with periodic boundary conditions. Insert in the main integration loop a visualization step, e.g. generate a plot of the transverse beam profile, so that the evolution of the numerical solution can be observed "in real time."

Solution:

Listing 3.1. Simple implementation of an explicit BPM scheme

```
% 1-D Finite-Difference beam propagator, explicit method
% illustrates instability

% parameters
lambda = 800.0e-09;    % wavelength
wx     = 40.0e-06;     % beam waist
LX     = 1.0e-03;      % domain size
NX     = 512;          % grid points
dz     = 25.0e-06;     % integration step
LZ     = 0.0014;       % propagation distance
steps  = LZ/dz;        % controls the main loop

```

```
13 % derived parameters
14 k0 = 2*pi/lambda;
15 dx = LX/NX;
16
17 % array of coordinates
18 cx = dx*(linspace(0,NX-1,NX)-NX/2);
19
20 % amplitude holder, define an initial condition
21 % beam propagating at angle
22 am0 = GaussianBeam1D(cx,0.0,wx,k0,0,k0/50.0);
23
24 % cefficient for the FD scheme
25 pcoeff = -1i*dz/(2.0*k0*dx*dx);
26
27 % define function to execute linear step
28 LinearStep = @(A,C) A + C*(circshift(A,[2,1]) -2.0*A + circshift(A,[2,-1]));
29
30 % initailize the working array for the evolving amplitude
31 am1 = am0;
32
33 % main integration loop
34 for s=1:steps
35   am1 = LinearStep(am1,pcoeff);
36   plot(cx,real(am1));
37   % tune this to adjust visualization speed
38   pause(0.1);
39 end
```

The above script is self-explanatory, but a comment on how the integration step is realized might be in order. Line 28 defines a function that embodies the finite difference integration scheme. The last term represents the second-order derivative approximation

$$\approx A(i-1) - 2A(i) + A(i+1) \ ,$$

which is realized as a weighted sum of suitably shifted vector arrays. Here we use the circular shift function which does two things: it will align the vector components that are to be combined, and it will automatically take care of the periodic boundary conditions.

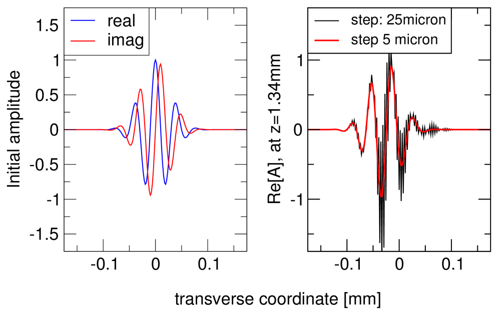
Task 2:

Use your program to explore several parameter choices to visualize typical instability properties. Set up the script to propagate a beam at an angle — this will result in "nontrivial" change of the numerical solution with the propagation distance. A "useful" simulated propagation distance is such that the waveform visibly shifts to one side in the computational domain.

With this set up, perform simulations that illustrate the following:

- No matter what the input parameters, in particular the propagation step, simulation never reaches a "useful distance". For example, with the initial beam waist of 40 micron, the instability onset is at mere 2 mm propagation distance for a 25 micron propagation step. Decreasing the latter to 5 micron only shifts the instability to about 3 mm.
- Instability grows from noise even for a very smooth initial condition. Beam profiles with steep gradients of phase are even more susceptible.

- If the initial condition contains a cusp (e.g. at the domain edge with periodic boundary conditions), or some other sharp feature, it can seed the unstable growth very quickly. In this exercise, use a wider beam to induce such a situation. When the beam waist is increased from 40 to 140 micron, its tail reaches the domain edge and creates a cusp with its opposite end because of the periodic boundary conditions. As a consequence, the first spots at which numerical noise starts to grow rapidly, are at the domain edges.



Numerical instability onsent in the beam propagation simulation utilizing an explict finite-difference integration scheme. The initial condition (left) is a Gaussian beam propagating at an angle. The panel on the right shows the real part of the propagated amplitude after 1.34 mm. The black curve is the result of interation with the step of 25 micron, and shows the very beginning of the instability growth. While a shorter integration step (red curve) can remove the numerical noise at this propagation distance, the instability onset is merely delayed.

Summary:

- Implementation of an implicit finite-difference update scheme is a bit more involved than the simple explicit method studied previously, but the task is considerably simplified by the higher-level language programming (Matlab in this case) and/or utilization of linear algebra libraries.
- Validation and accuracy testing is easiest if an exact solution is available. In this case, the paraxial Gaussian beam is used.
- Visualization of the dispersion relation is convenient with the method introduced in the study of Maxwell equations solver. Implicit character of the method introduces numerical diffusion which in turn makes propagation constants complex and thus "blurs" the dispersion relation of numerical waves.
- Convergence study is an important element of any numerical experiment. The mutual "interaction" of transverse and longitudinal grid spacings is illustrated in the measured convergence curves.

3.5.2 Implementation of the implicit BPM scheme

It is often convenient for both, the derivation and the practical implementation in a computer program, to represent the core of the numerical algorithm in the form that uses matrices, or operators. Because such a notation is practically a rule in the articles on beam propagation methods, here we take the opportunity to practice this approach in the most simple case. The method will also be utilized in the subsequent sections, and in particular in different variations on the Crank-Nicolson theme...

Write the implicit BPM-update scheme in the "operator" form

$$L^{(-)}E_{new} = E_{old} \ ,$$

where the matrix $L^{(-)}$ acts on the to be computed vector (array) storing the one-dimensional complex amplitude E_{new} representing the profile of the beam. Explicitly, the elements of the resulting vector are

$$\left(L^{(-)}E\right)_i = \sum_k L^{(-)}_{ik} E_k = \{E_i - \delta(E_{i-1} - 2E_i + E_{i+1})\} \ ,$$

with δ standing for the usual factor

$$\delta = \frac{i\Delta z}{2k_0 \Delta x^2}$$

formed from the “nonphysical” parameters characterizing the discrete form of our BPM equation. It will be illuminating to think of δ as the parameter that controls the departure of operator $L^{(-)}$ from the identity matrix. Similar matrices, and in particular $L^{(+)}$, which will differ only in the sign in front of δ, will soon appear in our derivations.

In this practice package, we implement the BPM simulator in Matlab, and in doing so we take advantage of its matrix-handling functionality. We simply define the above matrix equation, and ask Matlab to apply whatever solver it deems appropriate to solve it. The core of the main program will contain lines similar to (see the instructor‘s solution in *Main.m*):

Listing 3.2. Main loop of an implicit BPM scheme

```
%% PROPAGATE
Eold = E0;                % starting from the initial condition E0

   for k=1:nsteps
      z    = k*dz;        % keep track of the actual propagtion distance
      Enew = LM\Eold;     % finds solution to LM Enew = Eold
      Eold = Enew;        % current beam profile always resides in Eold...

      Observer_Report     % output/visualization after each step
   end
%% END PROPAGATE
```

The matrix or operator defining the implicit update scheme happens to be tri-diagonal. This is an important property that in actuality affects and guides the design of many core BPM building blocks. However, at this point we leave it to the linear system solver to recognize the specific type of the matrix (and hope that the solver will use it to its advantage). We will look at how to solve this kind of the system efficiently later in this course.

The most error-prone step in setting up a BPM simulation is often the stage when the linear system matrix is defined. In this case, $L^{(-)}$ consists of the diagonal and two adjacent upper- and lower-diagonals. The following lines (from *Method.m*) tell Matlab to insert the corresponding matrix elements into a sparse matrix LM holding operator $L^{(-)}$:

Listing 3.3. Left-hand-side operator as a sparse matrix

```

  delta      = dz/(2*k0*dx^2);
  idelta     = 1i*delta;

  LM_diag = zeros(NX,1);             % NX is the grid dimension
  LM_diag(1:NX) = 1 + 2*idelta;      % the diagonal
  offdiag = ones(NX-1,1)*idelta;     % off-diagonals are shorter
```

```

% the following places diagonals into a sparse matrix:
LM  =  sparse(1:NX-1,2:NX,   -offdiag,  NX,NX)+...
       sparse(1:NX,  1:NX,    LM_diag,  NX,NX)+...
       sparse(2:NX,  1:NX-1,-offdiag,  NX,NX);
```

The reason that we utilize a *sparse* matrix holder here is that most the matrix elements in $L^{(-)}$ are zeros. This fact not only reduces the necessary memory storage, but also makes calculation with such a matrix (e.g. matrix-vector multiplication) much more efficient.

Note that in any other than Matlab programming environment, one would store the elements of the matrix in auxiliary arrays. For now we trust Matlab to execute these operations in an efficient way. The drawbacks of such an approach is that the user does not enforce the most efficient method explicitly. The obvious advantage of it is that it only takes a few lines of the code to realize what is actually a non-trivial program, and this is just what we want here. The above lines of code implement the most important part of the implicit method studied in this exercise. The rest is "support" code that realizes the initial conditions, simulation parameters, and visualization and result outputs. Program *Main.m* calls the corresponding routines and executes the simulation that demonstrates that the implicit BPM method is indeed stable. The run starts from an initially focused Gaussian beam and propagates the solution beyond its focus region. Variation of the "nonphysical" parameters such as Δx, Δz, L_x, ... shows that the instability that degraded the explicit method has been successfully eliminated...

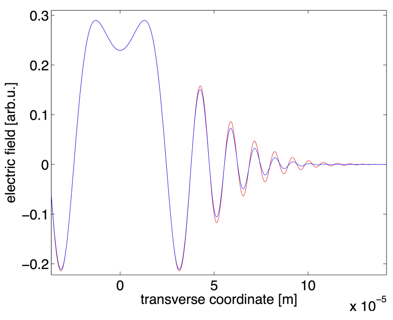
Validation and accuracy testing

Every program, no matter how trivial, must be tested. Gaussian beam solution (for single transverse dimension) is utilized here to test the BPM implementation. The initial condition is generated by *GaussianBeam1D.m*, and the BPM solver propagates this over certain propagation distance where it is compared to the target solution also generated by *GaussianBeam1D.m*.

Program in *Test.m* executes this numerical versus analytic comparison. Readers are invited to explore various parameter settings and get a sense of:

- if the numerical solution agrees with the analytic-formula target
- where (in the real space) the deviations between the two show up
- what parameters control these deviations
- what properties of the numerical method underline this behavior

The following figure illustrates such a comparison run.



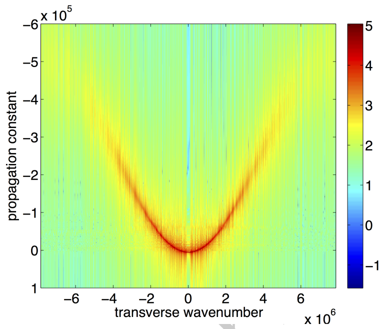
Testing the implicit BPM implementation versus the analytic Gaussian beam solution. Initially collimated Gaussian beam is propagated over several Rayleigh ranges, and the numerical (blue) and analytic (red) solutions for the electric field are compared here. The agreement of the two in the on-axis region around $x \approx 0$ indicates that the implementation of the method probably works as it should. The gap between the two solutions is clearly visible further from the axis, but it can be controlled (decreased) by a more conservative choice of simulation parameters.

The very first question that this comparative simulation invites is of course if the deviation seen between the analytic and numerical results is due to the limitation of the method itself, or it is caused by how the method was implemented. Since the analytic solution is generated by the formula from the *paraxial* Gaussian beam, one should expect that if the method is properly implemented, one must be able to choose simulation parameters to control (minimize) the error. In other words, one expects this method to converge to the continuum limit of the paraxial beam propagation equation, to which the Gaussian beam is an exact solution. It is left to the reader to verify that the numerical solution can be made closer to its analytic counterpart by choosing a sufficiently fine grid resolution Δx, and/or the integration step Δz.

It is important to give a proper interpretation to the error observed in the numerical solution. In the above figure it becomes obvious that the error appears largest away from axis. This is because it is the wave components that propagate at an angle w.r.t. the axis that contribute most to this portion of the solution. At the same time, we know from our experience with the numerical Maxwell solver that these waves must suffer from the effects of *numerical* dispersion. In other words, they do not propagate the same way their continuum-limit counterparts do. This is the reason why the error is most visible off-axis. To verify that the reason just put forward indeed applies, the reader could execute a simulation with a single plane wave propagating at an angle, and study the error as a function of its propagation angle. The manifestation of the numerical dispersion, and more specifically of its deviation from the continuum limit, is that the numerical waves propagate at angles different (smaller) from those that should correspond to their transverse wavenumbers.

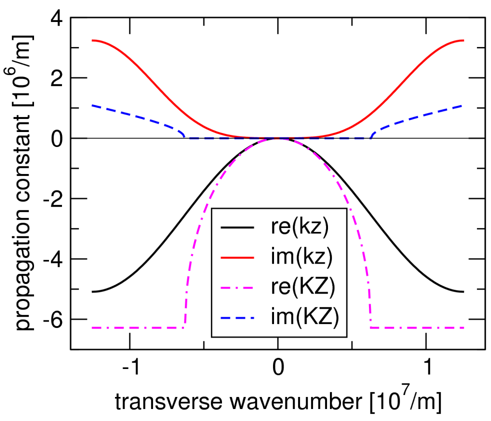
3.5.3 Mapping the numerical dispersion relation

It is instructive to investigate the numerical dispersion with the mapping method previously applied to the one-dimensional Maxwell solver. The initial condition in this method is white noise for both the real and imaginary parts of the solution. This is numerically evolved over some propagation distance, while snapshots of the beam profiles are stored after every integration step. The resulting transverse×longitudinal spatial profile of the complex envelope is Fourier transformed to obtain a spatial spectrum. This spectrum reveals energy accumulation in the vicinity of the locus that corresponds to the numerical wave dispersion curve.



Spatial spectrum of the BPM-propagated (initial) white noise reveals the dispersion relation for all possible waves supported by the numerical algorithm. The parabolic structure in the center corresponds to the continuum limit dispersion relation of the paraxial beam propagation equation. Waves with higher transverse wavenumbers are strongly damped — this is the consequence of the implicit nature of the integration scheme. It manifests in the blurring of the curve at larger transverse wavenumbers.

This figure shows sharply defined longitudinal propagation constants for waves with small transverse wavenumbers. At higher transverse wavenumbers, the imaginary part of the numerical propagation constant causes damping of such waves. This has two consequences, both visible in the figure. First, there is less "energy" in these waves because it was partially dissipated during the propagation. Second, their energy is spread, or blurred over a region of propagation constants, this blurring corresponding to the finite lifetime of the wave.



Comparison of the numerical dispersion curves for the implicit finite-difference BPM (full lines), and the exact non-paraxial beam propagation (dashed lines). Curves shown were generated for $\lambda = 1\mu$m, and grid spacing of $\Delta x = 0.25\lambda$. The propagation step is 10^{-8}m. Imaginary parts of the numerical and non-paraxial curves were scaled by a factor of fifty and one tenth, respectively.

That the measured dispersion relation behaves the way it is supposed to can be checked by evaluating the theoretical dispersion formula derived in the theory track. The figure above shows the real and imaginary parts of the numerical propagation constant versus transverse wavenumber. For comparison, the exact non-paraxial curves are also shown. They indicate the width of the light cone in this figure, and thus show which waves are physical, and which should not be present in the BPM simulation.

The fact that all numerical waves are damped is most important, because dissipation affects not only the noise but also the physical solution we seek to find. While in the context of the beam propagation one does not really need to capture waves with transverse wavenumbers that represent either steep propagation angles or are even beyond the light cone for the given wavelength, even paraxial rays are damped and this is clearly not desirable. The same issue becomes even more serious in quantum mechanics, where the preservation of energy in the beam corresponds to the unitary nature of quantum wavefunction evolution. We will deal with this difficulty shortly...

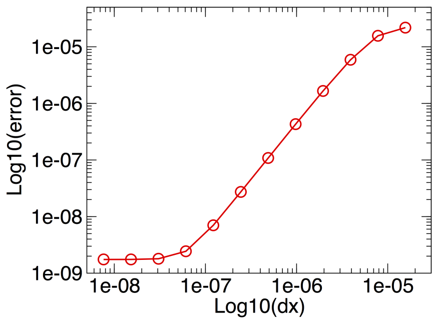
3.5.4 Convergence study

In this practice package, we look at the basics of performing a convergence check of a BPM-method implementation. We have implemented an implicit finite-difference method to solve the basic paraxial beam-propagation equation. The discretization scheme uses a three-point estimated of the second spatial derivative, and as such it should result in a quadratic converge of the numerical solution to the exact one. This rate of convergence will be tested here.

For this simple case, one can take advantage of the fact that an exact solution to the continuum equation exist, namely in the form of a (paraxial) Gaussian beam. This is used to generate both the initial conditions for the numerical simulation, and the target solution that this simulation should approach.

To avoid accumulation of errors, and thus probe only the local error controlled by the discretization scheme, the propagation distance should not be too long. At the same time it should be long enough to create a sufficient difference between the initial and target beam profiles.

In this particular example, the goal is to verify the methods properties as the discretization of the spatial grid becomes finer and finer. To this end, a series of runs is executed, doubling the resolution (or the number of grid points) every time. The distance between the numerical and target solutions is evaluated for each resolution and the result is plotted in the log-log scale. The following figure shows an example generated by the instructor‘s solution *ConvergenceStudy.m*.

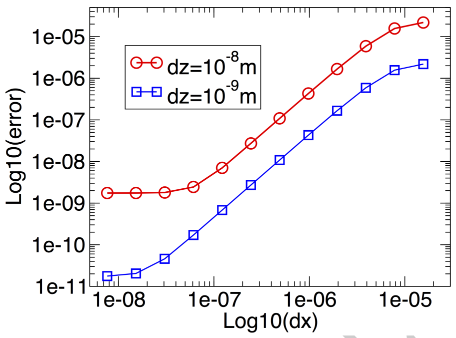


The error of the numerical solution as a function of Δx, the spatial grid‘s spacing. The slope of the linear part of the plot is close to two, corresponding to the order of accuracy of the discretized Laplacian in the paraxial beam propagation equation. The error is measured as the maximum (over the spatial extent of the grid) of $|E(x)_{num} - E(x,)_{exact}|$. Other metrics could be used, leading to a qualitatively similar result.

There are three regions in the curve shown above. The first is controlled by relatively large Δx when the error decreases with decreasing Δx but at a rate slower than one corresponding to the accuracy order one expects based on the discretization scheme. This is simply the manifestation of the fact that error$\approx \Delta x^2$ behavior is only reached in the limit of small grid spacing.

The middle of the plot is where Δx is sufficiently small to reveal quadratic decrease of the error. This is the regime we look for.

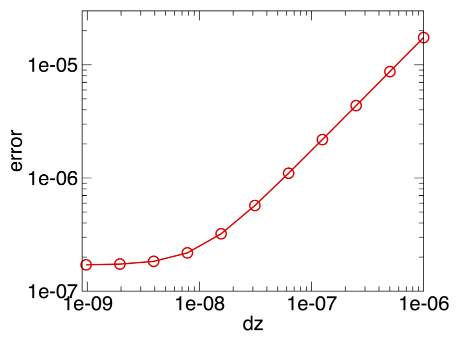
For very small Δx, the spatial grid discretization error become so small that some other error source starts to dominate. That is why decreasing Δx even further does not actually reduce the error any more. In this case, the residual error is controlled by the discreteness of the integration step Δz. This can be checked by running the same series of simulations with a smaller Δz. The reader is invited to check that it extends the region over which error decreases quadratically with Δx. Eventually, for a very small Δz, the error will behave erratically as the machine number accuracy starts to affect comparison of the exact and numerical solutions.



Convergence w.r.t. Δx for two fixed values of Δz. The minimal error achieved for small Δx obviously decreases with Δz, and this is in line with two sources or discretization errors. However, first-order accuracy in Δz suggests that the improvement should only be by about an order of magnitude, while the figure shows two orders...

Figure above compares convergence curves for two values of the integration step. As pointed out in the caption, the minimal error improvement seen at the bottom part of the curve is better than the expected one order of magnitude. This occurs because of the setup of this simulation. It is not suitable for measuring convergence w.r.t. Δz, because it does not ensure that “absolutely everything” except the controlled quantity (Δz in this case) remains unchanged during the whole convergence study.

The following figure shows a properly executed convergence check w.r.t. the integration step.



Convergence of the implicit beam propagation methods with respect to the integration step Δz. The first linear part of the plot has a slope close to unity, and this is expected since the accuracy order of the method is one for this parameter. For very small integration steps, the error levels off — this is when it becomes limited by the grid spacing Δx.

To conclude the convergence exercise, let us note that it makes it obvious that refining resolution in one dimension only does not necessarily result in an improved solution. The interplay between the non-physical numerical parameters becomes important, and very often decreasing one requires a corresponding adjustment in the other. Obviously this has an unpleasant consequence that the numerical work required increases even faster.