Contents

2

Beam propagation techniques in homogeneous media

This Section is devoted to the simplest but arguably the most important version of all BPM methods. Two particular classes will be discussed in detail, namely the plane-wave based method, and the BPM tailored to problems with cylindrical symmetry. Both approaches rely on the availability of accurate and efficient spectral transforms. The well-known discrete Fourier transform applies to plane-wave approach and the discrete Hankel transform is used for axially symmetric problems. Accordingly, the methods discussed here are often called spectral.

In most cases, spectral approaches require two complementary ways to store the information about the solution in the computer memory. We term them spectral-space and real-space representations.

The most important stage of the solution is accomplished in the spectral representation, with a beam-like solution given by a superposition of "elementary waves." These are the plane- or conical-waves for Fourier or Hankel-transform based methods, respectively. The elementary waves are sets of modes that constitute a basis in their respective spaces. As such they can represent arbitrary solutions, and it is the set of amplitude coefficients that the numerical algorithm must determine.

In the real-space representation we work with the numerical values of the solution. These [numerical values] are located in the nodes of a grid which samples the physical (real) space in which the beam propagates. Having an efficient algorithm that can transform a solution from the spectral to the real-space representation is crucial. This is because practical implementations almost always require that quantities of interest are calculated in the real space. Consequently, the transition between the two representations may occur frequently in a single beam propagation simulation.

Needles to say the numerical techniques discussed in this section are of a much broader applicability than in free-space beam propagation. In fact, they represent the most basic techniques that every computer-simulation practitioner must know inside-out.

2.1 Expansion into Plane Waves

As before, assume a homogeneous medium and a single-frequency ω electromagnetic field in the form of a continuous laser beam. Since the Maxwell equations are linear, one can represent quite general solutions as superpositions of plane waves. In doing so we are tacitly making use of the fact that plane waves constitute a complete system in the space of radiative (i.e. propagating) field "configurations." In that spirit, any beam-like solution can be written as a superposition of plane waves in the following form:

$$\mathbf{E}(x,y,z,t) = \int dk_x dk_y \boldsymbol{A}(k_x,k_y) \exp\left[-i\omega t + ik_x x + ik_y y + iK_z(\omega,k_x,k_y)z\right] \tag{2.1}$$

where

$$K_z(\omega,k_x,k_y) = \sqrt{\frac{\omega^2\epsilon(\omega)}{c^2} - k_x^2 - k_y^2} \tag{2.2}$$

stands for a frequency and transverse wavenumber dependent propagation "constant." For our purposes the mathematical details such as what exactly is the class of solutions that admit this kind of an expansion is not very important. The function $\boldsymbol{A}(k_x,k_y)$ can be called the spectral amplitude or the spatial spectrum. This is what really defines the beam. In the above expression, it is an arbitrary function so we need more information to specify it. This naturally comes from the initial beam condition. It is normally given as a specification of the electric field for a given value of the propagation direction coordinate z, usually $z = 0$:

$$\mathbf{E}(x,y,z=0,t=0) = \int dk_x dk_y \boldsymbol{A}(k_x,k_y) \exp\left[ik_x x + ik_y y\right] \tag{2.3}$$

What this says is that the quantity on the left is known. It represents the laser beam at the input plane separating the "outside world" from our domain of interest. Within this [domain of interest] we seek to find the solution. This is merely a specialization of the previous equation for $z = 0$, but it serves to find $\boldsymbol{A}(k_x,k_y)$ appropriate for the given problem. It can easily be inverted as follows: First, one multiplies both sides by an arbitrary plane wave $\exp\left[-iu_x x - iu_y y\right]$, and integrates over the whole space

$$\begin{aligned} &\int dxdy \mathbf{E}(x,y,z=0,t=0) \exp\left[-iu_x x - iu_y y\right] \\ &= \int dk_x dk_y \boldsymbol{A}(k_x,k_y) \int dxdy \exp\left[i(k_x-u_x)x + i(k_y-u_y)y\right] \ . \end{aligned} \tag{2.4}$$

Next, each spatial integration in the last term results in a Dirac-delta function. Note that the normalization factor depends on whether wave-numbers or wave-vectors are used in the arguments of carrier waves. Both representations are common in the BPM literature. For the wave-vector parametrization used above, one factor of 2π accompanies each delta function, and one can write

$$\int dxdy \mathbf{E}(x,y,0,0) \exp\left[-iu_x x - iu_y y\right] = \int dk_x dk_y \boldsymbol{A}(k_x,k_y)(2\pi)^2 \delta(k_x-u_x)\delta(k_y-u_y) \ . \tag{2.5}$$

Finally, the integration over $k_{x,y}$ can be trivially performed and results in

$$(2\pi)^{-2} \int dxdy \mathbf{E}(x,y,0,0) \exp\left[-iu_x x - iu_y y\right] = \boldsymbol{A}(u_x,u_y) \tag{2.6}$$

This formula specifies the initial condition for the spectral amplitude, which is simply given by the Fourier transform of the field at $z = 0$. Of course, during the propagation in the linear regime, the spectral content of the beam does not change at all and therefore the spatial spectrum obtained above from the initial condition remains valid for all times. It is correct to say that it "solves" the linear beam propagation problem in free space exactly for all propagation distances. However, this does not mean the numerical realization of the algorithm which we are going to discuss next can execute arbitrarily long steps. The numerical representation of the spatial spectrum is necessarily discrete and the computational domain is finite — this is why the formally exact solution becomes severely restricted by the numerics. These issues will be discussed in detail later.

Note: The magnetic field was not even mentioned above. This is quite common in the BPM field, where it is tacitly assumed that the magnetic and electric fields are orchestrated such that the beam propagates forward. We will show later in this course that the assumption of forward-propagating solution means that the knowledge of the electric field alone is sufficient for reconstruction of the the magnetic field. While it may seem trivial for propagation in free space, the corresponding procedure is not so simple in complex structures with various optical materials.

Note: The initial condition as specified above contains one silent assumption. This is that the initial field at $z = 0$ is transverse and that it will result in the spatial spectrum $\boldsymbol{A}(u_x, u_y)$ that is perpendicular to the wave-vector $(u_x, u_y, K_z(\omega, u_x, u_y)$: The vector $\boldsymbol{A}(u_x, u_y)$ specifies the polarization of the corresponding plane wave, and for the latter to be transverse, the polarization direction must be perpendicular to the wave-vector.

2.1.1 Basic DFT-based method

Returning to the beam propagation problem, insert the spatial spectrum back into the original expansion:

$$\mathbf{E}(x,y,z,t) = (2\pi)^{-2} e^{-i\omega t} \int dk_x dk_y \exp\left[ik_x x + ik_y y\right] \exp\left[iK_z(\omega, k_x, k_y)z\right] \times \int dr_x dr_y \exp\left[-ik_x r_x - ik_y r_y\right] \mathbf{E}(r_x, r_y, 0, 0) \ . \tag{2.7}$$

To compress the notation, subscript $\perp$ is used to denote the vectors living in the space transverse to the direction of beam propagation

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = (2\pi)^{-2} e^{-i\omega t} \int d^2 k_\perp e^{i\boldsymbol{k}_\perp \cdot \boldsymbol{r}_\perp} \exp\left[iK_z(\omega, k_x, k_y)z\right] \int d^2 r_\perp e^{-i\boldsymbol{k}_\perp \cdot \boldsymbol{r}_\perp} \mathbf{E}(\boldsymbol{r}_\perp, 0, 0) \tag{2.8}$$

It is also often easier to work with quantities in which the complex phase changes as slowly as possible. To factor out the "trivial" phase change of the whole beam, the on-axis portion of the propagation constant can be isolated as

$$\beta(\omega) = K_z(\omega, k_x = 0, k_y = 0) = \frac{\omega n(\omega)}{c} \ , \tag{2.9}$$

and taken out of the integral

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = (2\pi)^{-2} e^{-i\omega t + i\beta(\omega)z} \int d^2 k_\perp e^{i\boldsymbol{k}_\perp \cdot \boldsymbol{r}_\perp} e^{i[K_z(\omega, k_x, k_y) - \beta(\omega)]z} \int d^2 x_\perp e^{-i\boldsymbol{k}_\perp \cdot \boldsymbol{x}_\perp} \mathbf{E}(\boldsymbol{x}_\perp, 0, 0) \tag{2.10}$$

This is an exact result that gives the profile of the electric field for the arbitrary z cross-section of the beam. It constitutes the basis for the Fourier transform based Beam Propagation Method. Indeed, the formula can be read and implemented numerically as follows:

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = \text{CarrierWave } FT^{-1}\left[\text{Propagator}(z, \omega, \boldsymbol{k}_\perp) \ FT\left[\mathbf{E}(\boldsymbol{r}_\perp, 0, 0)\right]\right] \tag{2.11}$$

The first step is the Fourier transform of the initial field distribution. It transforms an input array in which the two-dimensional index corresponds to the transverse location $\boldsymbol{r}_\perp$ into an array in which the indices represent the spatial frequencies k_x, k_y. Below is the numerical representation of the spatial spectrum of the beam. The second step is the multiplication by the linear propagator

$$\mathcal{P}(\omega, k_x, k_y, \Delta z) = e^{i[K_z(\omega,k_x,k_y)-\beta(\omega)]\Delta z} \ , \tag{2.12}$$

and this is done point-by-point in the spectral space. In other words this operation is diagonal in the indices of the spatial spectrum. The propagator itself can be pre-calculated and stored in an array. This is especially advantageous if a propagation step with the same length z is to be performed several times — keep in mind that exponentiation is an expensive operation.

The last stage consists in the inverse Fourier transform that takes the "propagated" spatial spectrum and converts it into real-space field profile. Note that the propagation in free space without losses only changes relative phases between the spatial spectral components of the beam. The angular power-spectrum remains conserved.

As a very last step the overall beam phase should be updated by addition of the carrier wave phase

$$e^{-i\omega t+i\beta(\omega)z} \ .$$

In most of the BPM applications, this phase is irrelevant and the last step is omitted. However, BPM plug-ins can be part of more complex propagation algorithms in which BPM is applied to many frequency slices independently. In such cases, the overall beam phase depends on the frequency and must be properly maintained.

An important detail is the normalization. It appears in the form of the $(2\pi)^{-2}$ pre-factor in the exact formula. Its numerical counterpart is absorbed in the discrete Fourier transform and may depend on its implementation. It is a common feature in numerical libraries that raw transforms are provided that do not normalize the resulting arrays. Then, the forth-and-back transformation results in multiplication by N, the dimension of the transformed array. To compensate for this, a corresponding normalization factor must be included somewhere. One reasonable choice is to include it into the pre-calculated propagator. A programming purist can object, and rightly so, that the transform normalization factor has no natural place in the propagator. Rather, it should be kept where it belongs and that is with the subroutine that performs the transform. So, the proper normalization can be also added into either forward or backward transform. In any case, one must make sure that repeated applications of a BPM step do not change the overall amplitude or power of the propagated beam.

2.1.2 Paraxial approximation

It is often useful to restrict the beam propagation to the paraxial regime, in which the transverse components of the wave-vector are small in comparison with the longitudinal component. The argument of the linear propagator can be then Taylor expanded up to second order in transverse wavenumbers:

$$i[K_z(\omega, k_x, k_y) - \beta(\omega)]z \approx iz\left[\sqrt{\beta^2(\omega) - k_x^2 - k_y^2} - \beta(\omega)\right] \approx \frac{-iz}{2\beta(\omega)}(k_x^2 + k_y^2) \ . \tag{2.13}$$

However, it is not sufficient to change the formula that defines the beam propagator to "switch" the paraxial approximation on. If the latter is deemed useful or necessary, the implementator must make sure that the computational grid only supports the waves with spatial wavelength for which the difference between the exact and paraxial propagation is in some sense small. Otherwise, the solution may generate waves with unphysical propagation properties.

Exercise: It is sometimes said that a "typical" angle of propagation for which one can still speak of paraxial propagation is about ten degrees. Of course, any such measure is very much arbitrary — what really matters is a comparison of the exact and the approximated quantity. Use the expressions above to plot curves for the exact (i.e. non-paraxial) propagator argument and its paraxial approximation. You may do this as a function of the transverse wavenumber, perhaps relative to $\beta(\omega)$, or as a function of the plane-wave propagation angle.

The paraxial approximation propagation has an interesting property, which is that diffraction acts independently in the two transverse directions. The paraxial propagator becomes is in fact a product of two one-dimensional propagators because the exponential factorizes as such. This property becomes evident for the initial beam configurations that can be represented as a direct product of two functions, each depending only of one of the spatial variables. An important example of such a field is the Gaussian beam:

$$E(\boldsymbol{x}_\perp, 0, 0) \approx \exp[-(x^2/w_x^2 + y^2/w_y^2)] \tag{2.14}$$

for which the paraxial propagation formula

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) \approx \int d^2k_\perp e^{i\boldsymbol{k}_\perp\cdot\boldsymbol{r}_\perp} e^{-iz(k_x^2+k_y^2)/(2\beta(\omega)} \int d^2x_\perp e^{-i\boldsymbol{k}_\perp\cdot\boldsymbol{x}_\perp} \exp[-(x^2/w_x^2 + y^2/w_y^2)] \tag{2.15}$$

reduces to a direct product of factors corresponding to each transverse dimension.

Exercise: We will often utilize the well-known Gaussian beam solution as a testing tool for our beam-propagation implementations. Starting from the propagation formula above derive the expression for the propagated Gaussian beam, and implement this function for the later use in practical exercises. Your implementation should include the capability to impart a spatial phase-tilt. It will allow to model beams that propagate at an angle with respect to the axis of the computational domain.

2.1.3 Propagation step length and sampling

One can hear quite often that the spectral beam propagation method can take arbitrarily long steps because it is based on an exact formula. Most of the time this incorrect statement refers to a context in which the length of the integration step is limited by other numerical aspects. Any otherwise admissible step length turns out acceptable for the spectral BPM, so this misconception is mostly harmless. But it is still important to understand how the numerical issues seep into the exact formulation of the beam propagation problem and restrict the application of the algorithm.

Let us first see why the numerical evaluation of (2.11) becomes difficult when z is large. Consider a phase change difference between two neighboring spatial-spectrum amplitudes corresponding to the transverse wavenumber k_x, which occurs as a result of propagation over a length of z. For simplicity, paraxial approximation propagator is assumed here.

$$\Delta\phi \approx \frac{z}{2\beta(\omega)}\Delta(k_x^2) \approx \frac{z}{\beta(\omega)}k_x\Delta k_x \tag{2.16}$$

Clearly, this difference of phases must be smaller then π, otherwise there would be no way to tell how long propagation step caused it. To estimate an upper bound on the length of the step

for which the phase change difference becomes too large, let us look at the contributing factors above. The smallest difference in the wavenumber is estimated as $\Delta k_x = 2\pi/L$ where L is the length of the computational box edge. This gives

$$\Delta\phi \approx 2\pi \frac{z}{L} \frac{k_x}{\beta(\omega)} < \pi \tag{2.17}$$

where $k_x/\beta(\omega)$ is the angle of propagation and z/L represents the maximal tangent of an angle that "fits" in the propagated computational domain. This formula means that the linear propagator can only resolve a given angle-of-propagation if the corresponding ray "fits" in the box of $z \times L$, hence the limitation on how big z can be. The finer the spatial grid resolution the steeper is the maximal angle of propagation it can support. Consequently, better grid resolution will restrict the integration step more. It should be emphasized that the above is a an upper bound estimate and in practice a much more conservative value of z should be chosen.

Exercise: The above formulas are less than optimal for practical work. A much more useful way to assess if a given propagation step is acceptable under given circumstances is to have a closer look at the propagator $\mathcal{P}(\omega, k_x, k_y, \Delta z)$. It must be properly resolved for all values of transverse wavenumbers k_x, k_y. Resolution problems start to manifest for the extreme (i.e. maximal in absolute values) wave-vectors and are easy to spot when plotted: Plot both the real and imaginary parts of the propagator for a given spatial grid and several values of the integration step Δz. For what values of Δz do you start to see artifacts in your plot? Are these values in line with the argument given above?

2.1.4 Stationary phase approximation

As a preparation for the following topic, a brief review of the stationary phase method may be useful. Assume that one needs to evaluate, for a large parameter η, the following integral

$$I = \int dx f(x) \exp\left[i\eta g(x)\right] \tag{2.18}$$

where f and g are given functions. It should be intuitively clear that if η becomes large, the exponential factor will oscillate ever faster, and will effectively damp the contribution to this integral from all regions in x where $g'(x)$ is non-zero. Only regions around stationary phase points,

$$\partial_x g(x)|_{x=x_0} = 0 \tag{2.19}$$

where the exponential phase varies slowly can contribute significantly. In the vicinity of the stationary point, one can Taylor-expand the function $g(x)$ up to the second order

$$g(x) \approx g(x_0) + \frac{1}{2} g_{xx}(x_0)(x - x_0)^2 + \dots \ , \tag{2.20}$$

drop the rest, and insert the expansion in the integral to be evaluated:

$$I \approx \int dx f(x) \exp\left[i\eta(g(x_0) + \frac{1}{2} g_{xx}(x_0)(x - x_0)^2)\right] \tag{2.21}$$

Of course, the first part of the exponential argument is a constant and can be pulled out. The rest is a Gaussian-type integral, provided that the function $f(x)$ does not change too fast.

Because it was assumed that η is large the exponential averages out everywhere except $x = x_0$. Consequently, the only piece of information from $f(x)$ that remains relevant is its value at the stationary point x_0:

$$I \approx f(x_0) \exp\left[i\eta g(x_0)\right] \int dx \exp\left[i\frac{\eta}{2} g_{xx}(x_0)(x - x_0)^2\right] \tag{2.22}$$

Here one applies the well-known Gaussian integral formula to get the stationary-phase estimate of the integral:

$$I \approx f(x_0) \exp\left[i\eta g(x_0)\right] \sqrt{\frac{2\pi i}{\eta g_{xx}}} \tag{2.23}$$

A similar procedure applies to two and higher dimensions. Specifically in two dimensions the result reads

$$I \approx \frac{(2\pi i)}{\eta\sqrt{g_{xx}g_{yy} - g_{xy}g_{xy}}} e^{i\eta g(x_0,y_0)} f(x_0, y_0) \tag{2.24}$$

2.1.5 Beam profile evaluation in the far field

To further develop the sense for the role of the propagation step in the FFT based BPM, let us consider calculation of the diffraction pattern in the far-field, or Fraunhofer regime. A point was made above that despite the exact nature of the formula on which the basic FFT BPM is based, the admissible propagation step Δz must be kept sufficiently short. For very large z, depending on the initial condition the numerical formula may not work. This subsection shows what the result should be in the far-field.

Starting from (2.8)

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = e^{-i\omega t + i\beta(\omega)z} \int d^2k_\perp e^{i\boldsymbol{k}_\perp \cdot \boldsymbol{r}_\perp} e^{i[K_z(\omega,k_x,k_y) - \beta(\omega)]z} (2\pi)^{-2} \int d^2x_\perp e^{-i\boldsymbol{k}_\perp \cdot \boldsymbol{x}_\perp} \mathbf{E}(\boldsymbol{x}_\perp, 0, 0) \tag{2.25}$$

identify the Fourier transform of the initial beam profile (denoted by hat):

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = e^{-i\omega t + i\beta(\omega)z} \int d^2k_\perp e^{i\boldsymbol{k}_\perp \cdot \boldsymbol{r}_\perp} e^{i[K_z(\omega,k_x,k_y) - \beta(\omega)]z} \hat{\mathbf{E}}(\boldsymbol{k}_\perp) \ . \tag{2.26}$$

Making contact with the stationary phase approximation outlined in the previous subsection, one can identify the functions

$$f(k_x, k_y) = \hat{\mathbf{E}}(\boldsymbol{k}_\perp) \tag{2.27}$$

and

$$g(k_x, k_y) = K_z(\omega, k_x, k_y)z + k_x x + k_y y = z\sqrt{\beta(\omega) - k_x^2 - k_y^2} + k_x x + k_y y \tag{2.28}$$

to be used in the formula (2.24). The role of the large parameter is played by the propagation distance z. To find the stationary point, evaluate the partial derivatives with respect to the transverse wavenumbers

$$\frac{\partial g(k_x, k_y)}{\partial k_x} = x + z\frac{k_x}{K_z} = 0 \qquad \frac{\partial g(k_x, k_y)}{\partial k_y} = y + z\frac{k_y}{K_z} = 0 \ , \tag{2.29}$$

and find the solution for the stationary point:

$$k_x^0 = \frac{\beta x}{\sqrt{x^2 + y^2 + z^2}} \qquad k_y^0 = \frac{\beta y}{\sqrt{x^2 + y^2 + z^2}} \tag{2.30}$$

and for the value of the determinant

$$(g_{xx}g_{yy} - g_{xy})^{-1/2} = z/(x^2 + y^2 + z^2) \quad g(k_x^0, k_y^0) = \beta\sqrt{x^2 + y^2 + z^2} \tag{2.31}$$

Inserting into (2.24), one obtains the following expression for the beam amplitude in the far field

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) \approx \frac{-iz}{x^2 + y^2 + z^2}\hat{\mathbf{E}}(\boldsymbol{k}_\perp^0)e^{i\beta\sqrt{x^2+y^2+z^2}} \to \frac{-ie^{i\beta z}}{z}\hat{\mathbf{E}}\left(\frac{\beta x}{\sqrt{x^2 + y^2 + z^2}}, \frac{\beta y}{\sqrt{x^2 + y^2 + z^2}}\right) \tag{2.32}$$

This is nothing but the well-known Fraunhofer diffraction pattern; The far-field intensity pattern is governed by the Fourier transform of the input beam, but scaled in the transverse plane as indicated by the arguments of $\hat{E}$ in the last term. Clearly, as z becomes large, so must x and y for this function to show its structure. This means that at some point along z, the required size of the computational domain in x and y would be too large. Thus, in general the FFT BPM formula does not allow to reach the far-field. However, in special cases when the angular spectrum is narrow, and the computational domain is chosen large, it may be possible to calculate the far field also by FFT BPM means.

2.2 Practice track: Fourier transform technique

2.2.1 Gaussian beam solution propagating at large angles

Purpose: The purpose of this exercise is twofold. First, it illustrates how the FFT-BPM formula derived in the class can be used to obtain the well-know Gaussian beam solution. Second, and more importantly, we obtain an implementation of an exact solution which will be utilized for testing our BPM algorithms not only in this section but also in a number of other situations discussed later in the course.

Task: The importance of testing in numerical simulation can not be overstated. To test implementations of BPM methods throughout this course, we will use exactly known solutions whenever possible. Since this Section deals with the propagation in homogeneous media, exact paraxial Gaussian beam solutions are suitable for these testing purposes.

A) Starting from the general formula developed in the lecture notes, derive an analytic expression for the propagated Gaussian beam in one transverse dimension. Specify the initial beam by its waist, wavelength, and an angle that defines the direction of propagation of the beam. Since later in the course we will need beams that propagate off-axis at an angle which large, i.e.not paraxial, it will be useful to have an initial condition function that is valid for arbitrary angle between the beam axis and the BPM propagation axis.

B) Write a function to implement the beam formula derived.

C) Generalize the result to two transverse dimension.

Solution:

A) First, let us derive the solution for an initially collimated Gaussian beam characterized by its waist w. We choose the frame of reference such that the beam waist occurs at the origin, and the propagation is along the positive z-axis. Later we will transform the result into the coordinate system of the computational domain.

The beam profile at $z = 0$ is a simple Gaussian function characterized by w:

$$\exp[-x^2/w^2] \ .$$

The first step is to obtain its Fourier transform:

$$\int_{-\infty}^{+\infty} dx \exp[-ikx] \exp[-x^2/w^2] \ . \tag{2.33}$$

This can be evaluated using the Gaussian-integral formula

$$\int_{-\infty}^{+\infty} dx \exp[-ax^2 + bx] = \sqrt{\frac{\pi}{a}} e^{\frac{b^2}{4a}} \ . \tag{2.34}$$

In this case we have $a = 1/w^2$ and $b = -ik$, and obtain

$$\int_{-\infty}^{+\infty} dx \exp[-ikx] \exp[-x^2/w^2] = \sqrt{\pi w} \exp[-k^2 w^2/4] \tag{2.35}$$

The next FFT-BPMstep is the application of the *paraxial* propagator,

$$\sqrt{\pi w} \exp[-k^2 w^2/4] \exp[-izk^2/(2\beta)] \ . \tag{2.36}$$

The second Fourier transform takes the solution back to real space:

$$(2\pi)^{-1} \int_{-\infty}^{+\infty} dk \exp[+ikx] \sqrt{\pi w} \exp[-k^2 w^2/4] \exp[-izk^2/(2\beta)] \ . \tag{2.37}$$

Here one collects like terms in the argument of the exponential in order to identify parameters a, b for yet another application of the above Gaussian-integral formula, and the result becomes:

$$\frac{1}{\sqrt{1 + \frac{2iz}{\beta w^2}}} \exp\left[-\frac{x^2}{w^2(1 + \frac{2iz}{\beta w^2})} \right] \tag{2.38}$$

This is a paraxial solution with its waist at $z = 0$. However, this in general will not be suitable for initial conditions in our BPM simulations. Should we need an initial condition such that it would focus at a distance d from where we initiate the BPM integration, we simply replace $z \to z - d$.

The solution we have is written in terms of the complex beam parameter with the minor difference (from the standard formulas) that we have imposed the normalization to unity at $z = 0$ and $x = 0$.

As usual, the dependence on the propagation coordinate is only through its ratio to the characteristic length-scale of the beam which is the Rayleigh range

$$\frac{2z}{\beta w^2} = \frac{z}{z_R} \qquad z_R = \frac{\pi w^2}{\lambda} = \frac{\beta w^2}{2} \ .$$

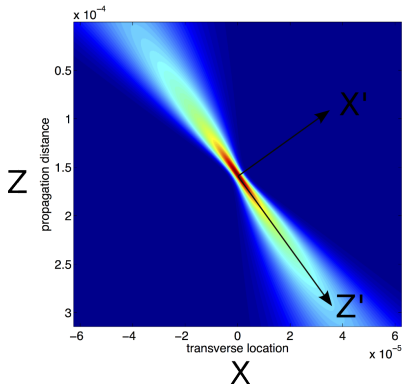
A Gaussian beam propagating at an angle, "placed" in the computational domain can be obtained by transforming the above result into a rotated coordinate system:

$$z' = z \cos\alpha + x \sin\alpha \quad , \quad x' = -z \sin\alpha + x \cos\alpha \ . \tag{2.39}$$

However, one must remember that the whole solution also contains the carrier wave, so the transformed (rotated) solution is

$$\frac{1}{\sqrt{1 + \frac{2iz'}{\beta w^2}}} \exp\left[-\frac{x'^2}{w^2(1 + \frac{2iz'}{\beta w^2})} \right] e^{+i\beta z'} \tag{2.40}$$

B) This function is implemented in the instructor's solution *GaussianBeam1DRotated.m*. The coordinate system used in this functions are illustrated in the following figure.



Gaussian beam coordinate systems utilized in the derivation of the solution. The intensity of a beam propagating at an angle is shown in the coordinate system (x,z) of the computational domain. The primed coordinates (x' and z') span the frame of reference which originates in the waist point of the beam and is oriented along its propagation direction.

A note is in order concerning the way beams propagating at an angle are realized in some beam propagation simulations. If the angle of propagation is not too large, the primed variables can be approximated by the original ones, except the z' that appears in the carrier-wave exponential (why?). The first-order approximation results in a phase-factor

$$e^{+i\beta z'} \approx \exp[i\beta z]\exp[i\beta\alpha x] \approx \exp[i\beta z]\exp[ik_\perp x] \ . \tag{2.41}$$

Thus, adding a phase-front tilt to the normal Gaussian will make it propagate at an angle. It is important to remember that this approximation is only good for small angles.

C) The generalization for two transverse dimensions is based on the fact that for any initial condition in the form a direct product

$$F(x)G(y) \ ,$$

the *paraxial* beam propagation formula factorizes into two one-dimensional contributions from each x and y. This is rather straightforward, and left to the Reader to show that if the beam waist is the same in both transverse direction, the solution becomes

$$\frac{1}{1+\frac{2iz'}{\beta w^2}}\exp\left[-\frac{x'^2+y'^2}{w^2(1+\frac{2iz'}{\beta w^2})}\right]e^{+i\beta z'} \ . \tag{2.42}$$

2.2.2 Implementation of FFT-BPM, paraxial and non-paraxial regimes

Task 1: Implement a one-dimensional beam propagation method based on expansion into plane waves, using FFT in the paraxial approximation. Set up the simulation such that it starts from the initial condition given by the function in the previous exercise. Let the algorithm propagate the solution through multiple propagation steps, and compare the resulting numerical solution with the exact one, given again by the same formula as the initial condition, this time evaluated for the appropriate propagation distance. Near-perfect agreement should be obtained for propagation along axis (zero beam angle).

Solution:

The implementation of the Fourier transform based beam propagation method is rather straightforward. There are three central portions of the program:

- Coordinate system. It is practical to precalculate coordinate arrays for both real-space and spectral-space representation. The later means to calculate the grid of spatial wavenumbers, for example as in the following listing:

Listing 2.1. 1D-Maxwell integration step implementation

```
% LX stands for the computational domain dimension
LX = ...

% dk represents the ''wave-number quantum''
dk = 2*pi/LX

% NX is the number of grid points in the spatial domain
NX = ...

% define transverse wavenumbers = spectral coordinates
kx = zeros(1,NX);
for k=0:NX/2
        kx(1+k) = dk*k;
end
for k=NX/2+1:NX-1
        kx(1+k) = dk*(k - NX);
end
```

This method honors the layout of frequencies normally used in implementations of FFTs. The first half of the arrays contains positive spatial frequencies, while the second half carries their negative counterparts.

- Spectral propagator. Having calculated the array of spatial wavenumbers, the propagator can be precalculated for the given integration step size dz:

Listing 2.2. 1D-Maxwell integration step implementation

```
% k0 stands for the reference wavenumber
k0 = 2.0*pi/lambda

% two versions of the propagator array below:

% propagator: paraxial version
px = exp(-1i*(kx.*kx)/(2*k0)*dz +1i*k0*dz);

% nonparaxial propagator
px = exp(+1i*dz*( (sqrt(k0*k0 - kx.*kx) ) ) );
```

Note that this implementation preserves the absolute phase of the solution as it includes also the phase change of the carrier wave. This is not strictly necessary, but needed if we want to compare not only the intensity but also the real part of the complex amplitude (which represents the electric field) between the exact and numerical solutions.

- Propagation step. Here we show the example in Matlab, where we can define a "one-liner" function to perform the whole update which returns the array holding the new solution amplitude:

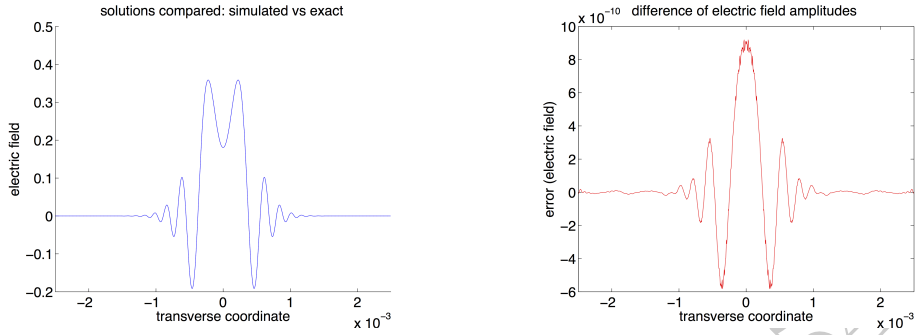
Listing 2.3. 1D-Maxwell integration step implementation

```
% define one linear step
```

```
2 LinearStep = @(amplitudein,propagator) ifft(propagator.*fft(amplitudein));
```

Note that in this case the forward and inverse transforms are properly normalized. With some other FFT implementation, the user must remember to ensure that the norm of the solutions remains preserved.

The rest of the simulation code is not specific to the method discussed, and implements the test based on comparison with the exactly known solution. To make this test non-trivial, the spatial profile of the beam must undergo substantial changes. One possibility is to start with a focused beam and let it evolve through its focal point. This is illustrated in the instructor's solution stored in the file named *pParaxialTest.m*.



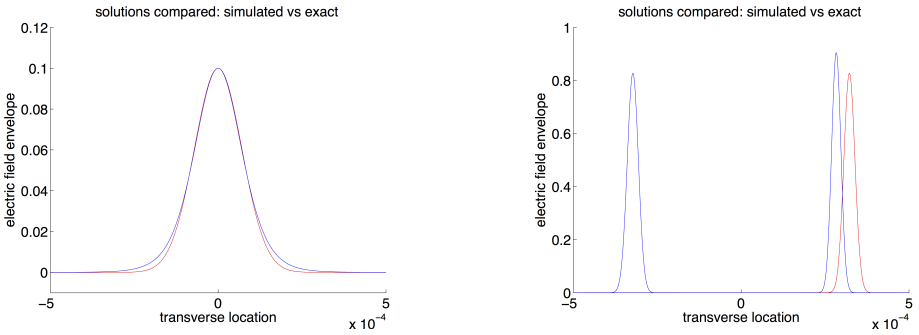
Test of 1D FFT-BPM implementation: Propagation of a Gaussian beam ($\lambda =$ 800 nm) beyond its focus in which it attains the waist of 100 micron. The total propagation length was ten Rayleigh ranges. The whole simulation was executed in fifty steps. The left panel shows comparison of the numerical and analytic solution, with two curves that are practically indistinguishable. The panel on the right shows the deviation between the simulated and analytic electric field amplitudes. This error increases with the number of integration steps taken, as the error accumulate. However, as expected the gap between the analytic and simulated results is very small.

Task 2: Implement in your script also the possibility to use the exact, non-paraxial propagator. Execute the simulation, and observe the deviations between the exact paraxial and numerical solution. Try to find a regime in which these deviations manifest as clearly as possible.

Solution:

a) One obvious possibility to induce non-paraxial effects is a tightly focused beam. Instructor's solution *pNonaparaxialWaist.m* illustrates such a regime. As the name suggests, the divergence of the beam is controlled by choosing its waist to be of the order of the wavelength. The reader should experiment with the simulation parameters in order to verify that unless the beam approaches the wavelength scale, the difference between paraxial and exact solution is insignificant.

b) A more stringent test of the FFT-BPM program in the non-paraxial regime is made possible simulating a beam propagating at a steep angle with respect to the axis of the computational domain. This arrangement is also more relevant to situations one often encounters in practical beam-propagation simulations. Instructor's solution *pNonparaxialAngle.m* contains an example.



Left: Non-paraxial effects in a tightly focused ($w = 1$ micron) Gaussian beam ($\lambda = 800$ nm). The non-paraxial solution (blue) exhibits slightly wider non-Gaussian beam pedestal.
Right: Non-paraxial effects in the Gaussian beam propagating at an angle of 0.35 radians (≈ 20 degrees) over a distance of 40 Rayleigh ranges. Focus with $w = 15$ micron attained roughly in the middle of the simulated propagation distance. The initial beams is the peak in the left-hand portion of the figure, the propagated beam profiles are on the right. The paraxial solution, shown in blue, propagates at a shallower angle. Obviously, propagation at an angle is more sensitive to non-paraxiality.

Puzzler: Script *pPuzzler.m* is intended to simulate at-an-angle propagation. It uses the same method as the one tested previously, yet a simulation run reveals a clearly incorrect behavior. The beam seems to propagate in a wrong direction, which may be a bit unexpected for what is an "exact" algorithm that passed a test. It is the reader's task to identify a remedy to this problem, and explain its origin.

2.2.3 Simulation of beam propagation in the Fraunhofer regime

Purpose: This exercise illustrates numerical difficulties in FFT-BPM propagation in far field. Concretely, we look at:

a) how cyclic (periodic) boundary conditions imposed by the Fourier transform prevent a truly long-distance simulated beam propagation,
b) simple numerical technique of a boundary guard, which is a kind of "poor-man's" method to mimic transparent boundary conditions at the computational box edge,
a) and effects of grid-resolution in the spectral domain, that show up especially in case of an initial condition with sharp edge(s).

The most important lesson of this exercise is that while such "poor man's" transparent boundary conditions can be fully sufficient in some cases, one must be always rather careful and look out for numerical artifacts introduced due to the modified boundary.

Task: Calculate the Fraunhofer diffraction pattern of a simple slit through the FFT-based beam propagation. The aim of this exercise is to explore a situation which "amplifies" certain numerical issues connected with FFT-BPM, and beam propagation in general.

Solution:

There are three Matlab scripts, *pFarField_1.m*, *pFarField_2.m*, and *pFarField_3.m* in this exercise package. They are essentially identical, provided for easy comparison of different simulation regimes. The situation simulated represents the Fraunhofer diffraction on a one-dimensional slit. The diffracted field is propagated sufficiently far, so that an approximation of the far-field Fraunhofer pattern is obtained.

Script *pFarField_1.m* utilizes a sufficiently large computational domain, and the boundary-guard is not used. This example shows that a reasonable approximation of the far-field pattern can be obtained if the domain is sufficiently large, and its spatial resolution is fine enough. Of course, it makes the computation relatively expensive. Here, in this one-dimensional case it may not be an issue, but it can in general (especially in two dimensions and in time-dependent problems) make the simulation prohibitively expensive.

This motivates script *pFarField_2.m* which is an attempt to achieve far field simulation with a smaller domain. Inspection of the boundary region in the simulated beam profile at the final distance reveals artifacts that are due to wings of the pattern coupling back because of the cyclic boundary conditions imposed by the FFT method.

Script *pFarField_3.m* switches on the boundary guard. It is designed to eliminate the problem with the periodic domain boundaries. Readers should inspect the script to see what implementation of the guard was chosen, and experiment with its settings.

In general, one has to choose the thickness of the boundary-guard layer. This is a part of the domain that is essentially "sacrificed," and not useful for obtaining any information form the simulated solution. Another parameter of the guard is the steepness with which its absorption increases toward the edge of the domain. If it is too steep, it can itself induce strong reflection, and thus make the domain effectively smaller.

By playing with these characteristics, it should be easy to realize that this way of handling the unwanted "reflections" from the boundaries is a rather subtle business. The take-away lesson is that with these simple boundary conditions, one has to pay keen attention to possible numerical artifacts.

One could say that the remedies we have used to improve the fidelity of our simulation on a smaller lattice were designed to control waves with extreme transverse wavenumbers. These are the components of the solution that propagate at steepest angles, and therefore experience first the finite size of the computational domain. In line with the observations made in one-dimensional Maxwell solver, it is the waves which belong to the edges of the resolved numerical bandwidth that tend to cause problems.

Note: An alternative way to deal with the extreme wavenumbers in the beam propagation is to design a propagator that will damp all waves with wavenumbers higher than a chosen threshold. This can be realized as a boundary guard in the spectral domain. Similar to the situation in the real space, the transition between the unperturbed and damped regions of transverse wavenumbers must be sufficiently gradual. If this transition is sharp, wave-forms arise in the real space representation that exhibit long non-decaying "tails," with characteristic wavenumbers, namely those separating the damped and un-damped wave-vectors.

2.2.4 Strongly non-paraxial regime: Poisson‘s bright spot

Summary:

- This exercise shows how a 1-D implementation of the FFT-BPM can be easily modified for two transverse dimensions...
- ... and illustrates the FFT-BPM strength in an application that bridges different proagation regimes, from strongly non-paraxial to paraxial.
- In this context, the computational complexity becomes an important issue and we touch upon question of performance and parallelization.

The physical context chosen for this work-package is that of diffraction of a collimated beam on a circular, completely opaque screen. This gives rise to the famous effect called Poisson‘s bright spot, or spot of Arago. A nice paper by R. Lucke (Eur. J. Phys. 27 (2006) 193) and related comment by G.S. Smith (Eur. J. Phys. 27 (2006) L21L23) are convenient sources to read on the mathematical background and in particular about two versions of diffraction formulas, Rayleigh-Sommerfeld and Fresnel-Kirchhoff integrals.

Form the numerical point of view, the calculation of the diffracted field in the space beyond the circular obstacle, and especially in the close vicinity of the optical axis and close to the screen, presents a good opportunity to realize how important are the non-physical parameters underlying simulations, such as the grid resolution. The reader will quickly appreciate that a calculation of the very same quantity may require very different numerical efforts depending on the choice of the simulated problem geometry.

Task 1: Simulation implementation
Possibly starting from the source developed for the previous exercise, write a program to calculate the diffracted field behind the circular obstacle when illuminated by a finite-diameter collimated beam. This will require an gextension of the FFT-BPM to two transverse dimensions.

Solution

The extension of the code from a previous exercise is not difficult. Let us comment on couple of points where mistakes occur often. As in the case with a single trnasverse dimension, one needs to prepare the linear propagator which in turn requires to calculate first the “allowable” spatial wavenumbers. These are of course obtained the same way as in one dimension. However, the realization of the propagator must be specifically two-dimensional because of the *non-paraxial* difraction regime that needs to be properly captured (recall that while paraxial regime produces propagating beam which is a direct product of two one-dimensional beams diffracting in their respective dimensions, such a reduction does not occur in general). For simplicity, it is assumed that the size of the computational box is the same along both transverse axes which we take as x and y, both sampled with NX grid points. Then in C language, the core part of the propagator implementation could be written as in this example:

Listing 2.4. Non-paraxial BPM propagator implementation in C

```
1
2   /* calculate transverse wavenumbers */
3   double kx[NX];
4   for(x=0;x<NX/2;x++) kx[x] = dk*x;
5   for(    ;x<NX;  x++) kx[x] = dk*(x-NX);
6
7   /* 2D propagator holder, use FFTW memory allocation */
```

```
8    fftw_complex* pxy;
9    pxy = fftw_malloc(NX*NX*sizeof(fftw_complex));
10
11   for(x=0;x<NX;x++) {
12     for(y=0;y<NX;y++) {
13       // paraxial: comment out for this exercise!
14       pxy[x+NX*y] = cexp(-I*(kx[x]*kx[x] + kx[y]*kx[y])/(2*k0)*dz )/(NX*NX);
15
16       // non-paraxial: use this version here!
17       pxy[x+NX*y]  = cexp(+I*(csqrt(k0*k0 - kx[x]*kx[x] - kx[y]*kx[y])));
18       pxy[x+NX*y] *= cexp(-I*k0*dz )/(NX*NX);
19       }
20   }
```

In the last expression, the reference carrier-wave phase is taken out. This is mostly inconsequential in the BPM context, since one is rarely interested in the absolute beam phase. Also note that when one uses a fast Fourier transform library, as FFTW in this example, it is better for performance to use the corresponding routines to allocate the propagator memory.

The physical nature of the simulated problem requires that the spectral beam propagation method is applied as a series of shorter integration steps, and at least a poor man's absorbing boundary guard is applied periodically to the solution. The reason for this is related to the occurrence of the complete spectrum of transverse wavenumbers, from very small ones, through waves that propagate almost perpendicularly to the axis, to even evanescent waves. One roughly say that each "'bundle" of plane waves that constitute a cone gives rise to the amplitude profile at certain distance from the screen. Aiming at reconstructing the latter from zero to "infinity," one must include waves with steep propagation angles, and the wavepacket that thse constitute reach the computational boundary very quickly. This is where they must be continually (or sufficiently often) destroyed by the boundary guard.

As a side note, the above also works in the opposite direction. If one only needs to claculate the beam profile at a specific large distance from the screen, a paraxial propagator will suffice as long as it carries the waves with proagation angles subtended by the axis and a line that connects the observation point on axis with the edge of the screen. In fact, this is exactly what was done in the other practical exercise with the Poisson's bright spot.

Going back to the boundary guard implementation, keep in mind that there is no universal recipe to set this up. The following code snippet is just such an example:

Listing 2.5. Simple boundary guard

```
1  /* 2D boundary guard holder, allocated with FFTW routine */
2  double* bxy;
3  bxy = fftw_malloc(NX*NX*sizeof(double));
4  for(x=0;x<NX;x++) {
5    for(y=0;y<NX;y++) {
6      // super-gaussian shape example: experiment with parameters!
7      bxy[x+NX*y] = exp(-pow((cx[x]*cx[x] + cx[y]*cx[y])/(LX*LX/5.0),8.0) );
8      }
9  }
```

It is to be emphasized that suitable parameters that control the shape of the guard should be obtained with experimentation and careful testing. The profile of the guard must not be too steep to minimize wave reflection from it. Moreover, a too frequent application of even a gradual absorption profile eventually creates an effectively smaller computational domain with a reflective

boundary, thus defeating it own purpose. This problem is a good place to experiment with these simple absorbing boundary conditions. The reader will surely find parameter setting that will work well and many more that will not work at all. The take-home lesson should be that of extreme caution in working with boundary guards of this simplest kind.

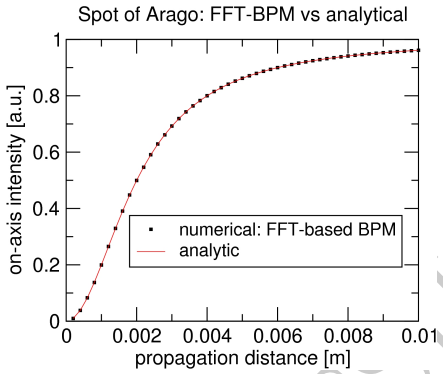
Finally, the main loop of the code is pretty much the same as in one dimension, alternating Fourier transform (in 2D), multiplication by the above propagator, following by the inverse Fourier transform. Periodically, this step should be followed by annihilating the outgoing waves that near the computational domain boundary.

The reader can find an instructors solution example in the corresponding practical exercise folder. Both single-thread and parallelized version in C and Matlab are provided as examples.

Task 2: On-axis intensity of the Poisson's bright spot
Reproduce in simulation the analytical expression for the on-axis intensity of the Poisson's bright spot. With a standing for the radius of a circular obstacle perpendicular to the beam propagation axis, the amplitude at the propagation distance z is

$$U(z) = \frac{z}{\sqrt{a^2 + z^2}} \ . \tag{2.43}$$

The simulation should also capture the immediate vicinity of the opaque screen, which will need the application of the non-paraxial propagator. It is left up to the reader to choose concrete parameters. The main goal is to obtain data which will show an agreement with the analytic target at least as good as that shown in the figure:



Analytic and simulated profile of the on-axis intensity vs propagation distance. Beam with a supper-Gaussian cross-section intensity profile diffracts around a circular opaque obstacle and gives rise to the famous Poisson's bright spot (the picture shows its intensity). The agreement between analytic and simulated solutions is essentially perfect, but this is only achievable with a reasonable numerical effort when the parameters of the problem are chosen appropriately (see text).

Solution:
At this point, I strongly encourage the reader to stop reading, and attempt the solution. One should quickly realize that it is not an easy simulation. Insuficient grid resolution may be suspect, but it can happen that even grids with several thousand points across will not produce a satisfactory agreement with the analytic formula... The deviations will mainly occur in the vicinity of the screen, while a better agrement may appear further down the axis.

Clearly, the non-paraxial nature of the optical field just beyond the screen is difficult to capture numerically, and is limitted by the maximal wavenumbers that are resolved by the numerical propagator. Points on the axis that are close to the screen are iluinated by waves that propagate essentialy perpendicularly to the beam propagation direction as a whole. Consequently, the transverse resolution of the numerical grid must be better than the wavelength of the beam.

However, because the problem states that the parameters of the simulated geometry can be freely chosen, we can cheat and by-pass the above described difficulties. Note that the wavelength

does not explicity appear in the formula (2.43)! If for a given radius a the wavelength is so large that a/λ is a modest number, the simulation becomes very easy. This is indeed how the results shown in the figure were obtained. With $a = 2$ mm, and $\lambda = 100\mu$m, a grid of 4096×4096 point turns out to be sufficient. The solution was sampled in 50 steps with a boundary guard applied after each FFT-BPM step.

Instructor's examples provided in the practice folder in files *cmp.c* and *instructors.c* for serial and parallel implmentation in C, and in *p1.m* for a Matlab version.

This is the first practical problem in this course in which the compute time becomes an issue, and wishes that the simulation would run just a bit faster. So it is a good opportunity to experiment with parallelization, too. While Matlab can parallelize for free, and possible even without users knowledge, certain amount of work investment is required in a compiled language. The example given in this practice package is for the OpenMP, pragma-based loop parallelization.

Task 3:
How sharp is the pattern in this diffraction scenario is, depends on how sharp and smooth the boundary of the obstacle screen is. Design a modification of the initial condition that mimics a fuzzy obstacle edge, and show that when the fuzziness affects one Fresnel zone, the central part of the pattern starts to degrade.

2.2.5 Calculation of the longitudinal vector component

Summary:

- Vectorial nature of light can be easily implemented into spectral beam propagation methods.
- The longitudinal component of the electric field amplitude need not be simulated along the whole propagation distance. Rather, it can be calculated from the divergence condition only when needed.

Background:
We pointed out in the introductory section that the beam propagation method usually works only with the transverse field components, whether electric or magnetic. But sometimes, one needs the longitudinal vector component of the field, for example for calculation of a nonlinear response.

It is possible to write the evolution equations for the longitudinal field component, and these can be solved along with the equations for the dominant polarization(s). This becomes especially simple in the case of a spectral method applied to a homogeneous medium, since the algorithm is exactly the same as for the transverse field components. Different polarization components are completely de-coupled, and each satisfies the same wave equation. The identical FFT-BPM algorithm therefore operates on each polarization component of the electric field. Naturally, one must make sure that the initial condition exhibits zero divergence, otherwise the solution could not represent a solution to Maxwell equations. However, such an approach would be unnecessary...

In the case of spectral FFT-based method, the longitudinal component can be obtained easily without actually "propagating" it. This is possible because the simulated medium is homogeneous, and the divergence constraint can yield the z-component exactly. The electric Gauss law in the real-space representation,

$$\nabla.\boldsymbol{E}(x, y, z) = 0$$

translates into the spectral-space representation

$$A_x(k_x, k_y, z)k_x + A_y(k_x, k_y, z)k_y + k_z A_z(k_x, k_y, z) = 0$$

where the z component of the wavevector can be fixed by requiring that the dispersion relation is satisfied, namely

$$k_x^2 + k_y^2 + k_z^2 = \frac{\omega^2 n(\omega)^2}{c^2} \ .$$

Thus, having calculated $A_{x,y}(k_x, k_y, z)$ for the given z where E_z is required, one transforms the $E_{x,y}(x, y, z)$ into spectral representation, and a calculates

$$A_z(k_x, k_y, z) = -\frac{A_x k_x + A_y k_y}{\sqrt{k_0^2 - k_x^2 - k_y^2}} \ .$$

This formula must of course be restricted to those wave-vectors that result in a real-valued k_z. Testing for this is usually necessary because the longitudinal components only become significant in non-paraxial regimes. In such situations, BPM usually utilizes spectral space grid (k_x, k_y) that encompasses regions corresponding to evanescent waves. They would normally carry negligible amplitudes.

Task 1:

- Starting from the implementation of the 2D FFT-BPM from the previous practice package, add a function which will calculate the E_z component of the electric field. You may assume that $E_y = 0$, or that the dominant field components is oriented along axis x.
- Set up a simulation to illustrate that a significant longitudinal field evolves in the focal point of a beam. Note the ratio between the field strength E_X and E_z, and how it depends on the focal length.

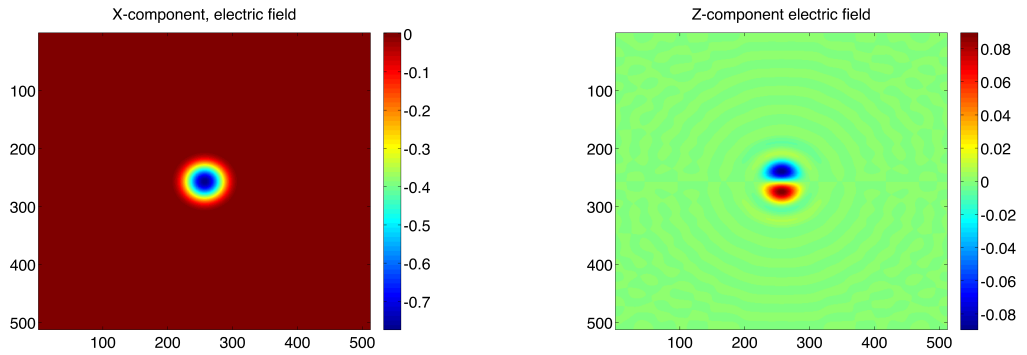
Solution:
Instructor's solution example is stored in *pEzInFocus.m*, for a simulation showing a tightly focused beam. It executes the "scalar" (single polarization component) simulation exactly as before. When the final propagation distance is reached, the E_z field is calculated, using the formula above, as follows:

Listing 2.6. Calculation of the longitudinal field

```
% calculate the spatial spectrum of Ex:
spatialspectrum = fft2(amplitudeX);

% evaluate the wave-number dependent factor Ez/Ex:
operator = zeros(NX,NX);
for x=1:NX
    for y=1:NX
        if( k0^2 > kx(x)^2 + kx(y)^2 )
          operator(x,y) = -kx(x)/sqrt(k0^2 - kx(x)^2 - kx(y)^2);
        end
    end
end

% apply to spatial spectrum and transform back to real space
amplitudeZ = ifft2(spatialspectrum.*operator);
```

To emphasize the strength of the longitudinal vector component, one needs a tight focus. One convenient was to set up a simulation is to use the analytic Gaussian beam formula evaluated at a given distance before the beam focus. In the example that follows the beam waist was chosen equal to the wavelength $\lambda = 633$nm. Note that the fact that this analytic formula does not describe accurately tightly focused beam solutions is irrelevant: It does represent *some* initial condition, and the longitudinal part is not initially specified. The beam is propagated into focus, and E_Z is calculated there. The result is illustrated in the figure that follows.

Comparison of the two panels shows that the intensity ratio E_z/E_x is about one tenth. Thus, even in the most tightly focused beam, the longitudinal component is relatively weak. This is one of the rationales for neglecting E_z in many beam propagation approaches.



Dominant E_x (left) and longitudinal E_z (right) fields in the focal point. Propagation of E_x was simulated over four Rayleigh range length, and then E_z was evaluated form the $\nabla\dot{\mathbf{E}} = 0$ condition. Note the ratio between the field strengths in the two panels. Horizontal and vertical axes correspond to array indices of the computational grid.

Task 2:
Set up a similar simulation illustrating the longitudinal field component in a beam that propagates at a steep angle w.r.t. the axis. The high-angle Gaussian beam solution developed previously can be utilized to create an initial condition.

The instructor‘s example can be inspected in *pEzAtSteepAngle.m*.

2.3 Expansion into Bessel Beams

In this section we discuss the Discrete Hankel Transform based Beam Propagation Method (DHT BPM). It is correct to say that it is just a special case of the previously described FFT BPM and namely one that deals with axially symmetric field distributions. This is of course a very important case which deserves attention. There are numerous applications in optics and the corresponding numerical tools have use in many different areas.

The formulation derived next is fully analogous to the plane-wave expansion case. First we concentrate on the traits that are common to both methods and then we continue with the discussion of differences later. Once again, we assume a homogeneous medium, and in it a beam-like propagating solution of Maxwell equations characterized by a single angular frequency ω.

To keep things simple, we work in the scalar approximation and thus neglect the fact that the electromagnetic waves are vectorial and transverse. Vectorial Bessel beams will be discussed in the subsequent subsection.

One could approach the derivation the same way we adopted for plane waves. That is, using the fact that Bessel functions constitute a complete system one can construct an appropriate basis to expand a general beam solution. However, Bessel beams are somewhat less intuitive to work with - at least in comparison with the well-known plane waves. In particular, one needs two kinds of functions to create a complete orthogonal basis set for a general (i.e. non-symmetric) field configuration. The expansion in the polar angle direction would be the same as the discrete Fourier, while the radial expansion would use Bessel function of various orders. To avoid this complexity we choose a different approach here. This lead us directly to the special case of cylindrically symmetric solutions and expansion in terms of zero-order Bessel functions.

2.3.1 Basic DHT-based method

Recall the general plane wave expansion utilized in the previous subsection:

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = (2\pi)^{-2} e^{-i\omega t + i\beta(\omega) z} \int d^2 k_\perp e^{i\boldsymbol{k}_\perp \cdot \boldsymbol{r}_\perp} e^{i[K_z(\omega, k_x, k_y) - \beta(\omega)]z} \int d^2 x_\perp e^{-i\boldsymbol{k}_\perp \cdot \boldsymbol{x}_\perp} \mathbf{E}(\boldsymbol{x}_\perp, 0, 0) \ , \tag{2.44}$$

and specialize it for the case of axially symmetric initial field distribution. In other words:

$$\mathbf{E}(\boldsymbol{x}_\perp, z = 0, t = 0) = \mathbf{E}(\rho = \sqrt{x^2 + y^2}) \ .$$

Note that this symmetry requirement is less trivial than it may seem and that it already hides the scalar approximation mentioned earlier. This equation says that the polarization vector of the beam is the same along every coordinate circle — this is not the case for truly general Maxwell solutions. For this initial condition, the last integral above can be rewritten such that it will lead to the integral representation of a Bessel function. In polar coordinates, it reads

$$\int d^2 x_\perp e^{-i\boldsymbol{k}_\perp \cdot \boldsymbol{x}_\perp} \mathbf{E}(\boldsymbol{x}_\perp, 0, 0) = \int_0^{2\pi} d\phi \int_0^\infty \rho d\rho e^{-ik_\perp \rho \cos(\theta - \phi)} \mathbf{E}(\rho, 0, 0) \tag{2.45}$$

where θ and ϕ are the polar angles of the wave-vector $\boldsymbol{k}_\perp$ and of the vector $\boldsymbol{x}_\perp$, respectively. The integration over ϕ gives the integral representation of the Bessel function,

$$J_0(k_\perp \rho) = \frac{1}{2\pi} \int_0^{2\pi} d\phi e^{-ik_\perp \rho \cos(\theta - \phi)} \tag{2.46}$$

and one gets

$$\int d^2 x_\perp e^{-i\boldsymbol{k}_\perp \cdot \boldsymbol{x}_\perp} \mathbf{E}(\boldsymbol{x}_\perp, 0, 0) = 2\pi \int_0^\infty \rho d\rho J_0(k_\perp \rho) \mathbf{E}(\rho, 0, 0) \ , \tag{2.47}$$

independently of the value of θ. This is expected because the Fourier transform of the radially symmetric function must also posses the same symmetry — it only depends on the modulus of the wave-vector $\boldsymbol{k}_\perp$ but not of its direction. Note that the linear propagator is also symmetric, i.e.

$$\mathcal{P}(\omega, k_x, k_y, z) = e^{i[K_z(\omega, k_x, k_y) - \beta(\omega)]z} = e^{i[\sqrt{\beta(\omega)^2 - (k_x^2 + k_y^2)} - \beta(\omega)]\Delta z} \equiv e^{i[K_z(\omega, k_\perp) - \beta(\omega)]z} \ , \tag{2.48}$$

$$K_z(\omega, k) = \sqrt{\frac{\omega^2 \epsilon(\omega)}{c^2} - k^2} \equiv \sqrt{\beta(\omega)^2 - k^2} \ , \tag{2.49}$$

and it means that the integration over the transverse wave-number plane in (2.45) can be performed in terms of Bessel functions the same way as just done for the integration in real-space.

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = e^{-i\omega t + i\beta(\omega)z} \int_0^\infty k_\perp dk_\perp J_0(k_\perp r) e^{i[K_z(\omega, k_\perp) - \beta(\omega)]z} \int_0^\infty \rho d\rho J_0(k_\perp \rho) \mathbf{E}(\rho, 0, 0) \ , \quad (2.50)$$

This is the sought expansion of a radially symmetric Maxwell solution into Bessel beams of zero order. It is also the basis for the BPM based on the discrete Hankel transform.

Exercise: Derive the above result directly from the expansion into Bessel beams. Write the electric field as a superposition

$$\mathbf{E}(r, z, t) = \int k dk \boldsymbol{A}(k) \exp\left[-i\omega t + iK_z(\omega, k)z\right] J_0(kr) \ ,$$

convince yourself that this expression does obey the wave equation. Determine the spectral amplitude $A(k)$ from the initial condition with the help of the following orthogonality relation for Bessel functions

$$\int_0^\infty r dr J_0(kr) J_0(ur) = \frac{1}{k} \delta(k - u)$$

Hint: To show that the wave equation is indeed satisfied, you will need the Bessel differential equation and its relation to the axially symmetric Laplacian differential operator.

The expression we have just derived has the same structure as the corresponding expansion into plane waves, with the difference that two-dimensional Fourier transforms are replaced by the radial Hankel transforms:

$$\text{Hankel}[f(r)] \equiv \int_0^\infty r dr J_0(kr) f(r) \ . \quad (2.51)$$

Note that the Hankel transform is also its own inverse (you can show this using the above mentioned orthogonality relation for Bessel functions). Thus, in complete analogy with the FFT BPM, we may write down the basis for the radially symmetric BPM:

$$\mathbf{E}(\boldsymbol{r}_\perp, z, t) = \text{CarrierWave} \ \times \ \text{Hankel}\left[\text{Propagator}(z, \omega, \boldsymbol{k}_\perp) \ \text{Hankel}\left[\mathbf{E}(\boldsymbol{r}_\perp, 0, 0)\right]\right] \quad (2.52)$$

2.3.2 Connection to Wave and Helmholtz Equations

It is instructive to approach the decomposition into Bessel beams also from the direction of the wave equation. For a fixed angular frequency (i.e. monochromatic light) the latter goes over into the Helmholtz equation. The Helmholtz equation in cylindrical coordinates reads as

$$A_{zz} + A_{rr} + \frac{1}{r} A_r + \frac{1}{r^2} A_{\phi\phi} + \frac{\omega^2 n^2(\omega)}{c^2} A = 0 \ . \quad (2.53)$$

One can seek the solution with the ansatz

$$A \to \mathcal{A}(r) e^{im\phi} e^{iK_z z} \ . \quad (2.54)$$

For the Helmholtz to be satisfied, the following must hold for the ansatz amplitude

$$+r^2 \mathcal{A}_{rr} + r\mathcal{A}_r + \left[r^2 \left(\frac{\omega^2 n^2(\omega)}{c^2} - K_z^2 \right) - m^2 \right] \mathcal{A} = 0 \ . \quad (2.55)$$

This equation should be compared with the Bessel equation of order m:

$$x^2 J_m^{''}(x) + x J_m^{'}(x) + (x^2 - m^2) J_m(x) = 0 \tag{2.56}$$

to obtain

$$\mathcal{A} = J_m(k_\perp r) \qquad k_\perp = \sqrt{\frac{\omega^2 n^2(\omega)}{c^2} - K_z^2}\ . \tag{2.57}$$

These solutions are the "eigenmodes" of the free-space wave equation:

$$\text{Bessel beam} = J_m(k_\perp r) e^{im\phi} e^{iK_z z - i\omega t}\ . \tag{2.58}$$

Different values of the "magnetic" quantum number m represent different angular momenta. The cylindrically symmetric solution corresponds to $m = 0$, i.e. the zero-order Bessel beam. The transverse wavenumber $k_\perp$ and the angular frequency determine the angle between the axis and the propagation directions in the bundle of plane waves that make up the Bessel beam. The plane wave propagation vectors "populate" a cone with a corresponding angle — this is why these solutions are often referred to as conical waves.

Exercise: General solutions of the wave equation can be equivalently represented as superpositions of plane waves or Bessel beams. This means that all elements of one basis must be superpositions of the "vectors" form the other basis. Research how a plane wave can be written in terms of Bessel functions.

2.3.3 Paraxial and non-paraxial solutions

The above derivations were done for a general, non-paraxial propagation. However, it will be shown later in connection with the paraxial beam propagation equation that Bessel solutions can represent both paraxial and non-paraxial situations. The difference between them lies only in their respective propagation constants (independently of the order m):

$$K_z^{exact}(\omega, k_\perp) = \sqrt{\frac{\omega^2 n^2(\omega)}{c^2} - k_\perp^2} \qquad K_z^{parax}(\omega, k_\perp) = \frac{\omega n(\omega)}{c} - \frac{c}{2\omega n(\omega)} k_\perp^2 \tag{2.59}$$

2.3.4 Discrete Hankel Transform

Our next task is to translate (2.52) into a practical numerical algorithm. What we need is the Hankel counterpart of the discrete Fourier transform. It is no accident that such a transform exists and that it has many properties that are similar to those of Fourier transforms. Unlike the latter, Hankel transforms come in different orders. These orders correspond to different polar-angle dependence of the function transformed (as m in the Bessel beam expression 2.58). We will initially restrict ourselves to the zero order as is appropriate for axially symmetric functions.

The discrete Hankel transform is an approximation of its continuum version

$$F(k) = \int_0^\infty r dr J_0(kr) f(r) \quad , \quad f(r) = \int_0^\infty k dk J_0(kr) F(k)\ , \tag{2.60}$$

which connects the "original" $f(r)$ to its "transform" $F(r)$. However, the above formulas are identical for the forward and backward transformation and imply that the Hankel transform is also its own inverse. This is a property which we will want to preserve in the discrete version.

To verify that the above integrals are indeed compatible one must use the orthogonality relation valid for the Bessel functions:

$$\int_0^\infty r dr J_0(kr) J_0(ur) = \frac{1}{k}\delta(k-u) \ . \tag{2.61}$$

Thanks to this the following string of equalities

$$\begin{aligned} F(k) &= \int_0^\infty r dr J_0(kr) f(r) = \int_0^\infty r dr J_0(kr) \int_0^\infty u du J_0(ur) F(u) \\ &= \int_0^\infty u du F(u) \int_0^\infty r dr J_0(kr) J_0(ur) = \int_0^\infty u du F(u) \frac{\delta(k-u)}{k} = F(k) \end{aligned} \tag{2.62}$$

confirms that the forward and backward transforms are the same.

DHT acts on a vector of values located at points given by the zeros of a Bessel function. This compares to the case of the discrete Fourier transform where samples are located at points given by zeros of sine and/or cosine functions.

Besides the assumption that the function we aim to transform has a radial symmetry we also need to require that its support is compact. In other words, the function must be identically equal to zero for r larger than the radius of our computational domain R_{max}. Moreover, this function must be so-called band-width limited. This means that it does not contain arbitrarily high spatial frequencies. This in turn means that only a finite number of spectral amplitudes is needed to represent the function.

The following are the properties of computational grids that support numerical Hankel transforms:

Computational grid in the real space

- Spatial-grid point locations scale with R_{max}:

$$r_k = R_{max}\frac{\alpha_k}{\alpha_M} \quad , \quad k = 1, 2, \ldots, M-1 \tag{2.63}$$

  where α_k is the k-th zero of Bessel function J_0,
- M determines the grid resolution and therefore the maximal supported spatial frequency
- The very first grid point lies at non-zero distance from the axis $r = 0$.
- The last point r_{M-1} is also away from the boundary R_{max}
- It is tacitly assumed that the function vanishes at the boundary $f(R_{max}) = 0$ and beyond.
- Spatial resolution is roughly R_{max}/M
- Grid points are not spaced at equal distances! They are a bit more apart closer to the center.

Computational grid in the spectral space

- Grid points in the spectral space are also given by Bessel zeros:

$$k_n = \frac{\alpha_n}{R_{max}} \quad , \quad k = 1, 2, \ldots, M-1 \ . \tag{2.64}$$

- The minimal possible spatial frequency is α_1/R_{max} ($\alpha_1 \approx 2.4048$), a counterpart of $2\pi/L$ for discrete Fourier transform.
- There is no grid point at zero spatial frequency, which means that one can't represent a truly constant function.
- This is related to the requirement that both the function $f(r)$ and its spectrum $F(k)$ must vanish at large values of their respective arguments.

- Spectral-space and real-space grids are essentially identical. One could (but should not!) choose a system of units in which the grids would coincide.

So the "original" and "transformed" functions are represented at these points as arrays stored in the computer memory,

$$F_n \equiv F(k_n) \qquad f_k \equiv f(r_k) \ , \tag{2.65}$$

and the Discrete Hankel Transform (DHT) is nothing but a matrix multiplication by the same matrix H, in both directions:

$$F_n = \sum_k H_{nk} f_k \quad , \quad f_k = \sum_n H_{kn} F_n \ . \tag{2.66}$$

Johnson ([1] H. Fisk Johnson, *Computing discrete Hankel transform*, Computer Physics Communications 43(1987)181.) gives the explicit form for the matrix elements of H as

$$H_{mn} = \frac{2}{\alpha_M} \frac{J_0(\alpha_m \alpha_n / \alpha_M)}{J_1^2(\alpha_n)} \ . \tag{2.67}$$

To calculate these we need high-accuracy implementations of the Bessel function themselves, as well as of auxiliary functions that can locate up to a few thousand of Bessel roots. The corresponding evaluation is usually done in the initialization stage and the matrix H_{mn} is stored together with the coordinate arrays in real (r_k) and spectral (k_n) spaces. Because the transformation is an inverse of itself we only need to store a single matrix.

However, it also means that we better have the following property satisfied:

$$\sum_k H_{nk} H_{km} = \delta_{nm} \tag{2.68}$$

Reference [1] also provides a proof that (2.68) holds in the limit of large M. It is shown that in this limit the repeated application of H yields

$$\sum_k H_{nk} H_{km} = \frac{4}{\alpha_M^2} \sum_{k=1}^{M-1} \frac{J_0(\alpha_n \alpha_k / \alpha_M)}{J_1^2(\alpha_k)} \frac{J_0(\alpha_k \alpha_m / \alpha_M)}{J_1^2(\alpha_m)} = \delta_{nm} \tag{2.69}$$

which is orthogonality relation (11) in Ref. [1]. It is to be noted that for finite M, the above is not strictly true. Nevertheless, even for smallest Ms the accuracy is high. For M about few hundred, which is the typical range in practice, non-diagonal values of $\sum_k H_{nk} H_{km}$ are of the order of 1×10^{-25}.

Before going into applications of DHT let us make a few notes to compare DHT to the more common FFT. Unlike the latter, DHT is represented by a dense, in fact a completely full matrix. As such this numerical transformation is "slow." The complexity of a single evaluation scales the same way as any matrix-vector multiplication, i.e. the computation time grows as M^2. This has a very pronounced effect on the performance, of course.

Typically, one uses a few hundred points to represent a radially symmetric function. Up to a few thousand grid points may be practical, depending on the available computer speed. It is sometimes possible to take advantage of the fact that the Hankel transform is real and the matrix-vector multiplication can be coded as such.

The availability of libraries for DHT is much less that that for DHT. Fortunately, Bessel function and Bessel function zero implementations are quite common. With those in hand it is easy to program DHT by straightforward coding of the above formulas.

2.3.5 Application of DHT to free-space propagation

The discrete Hankel transform numerical formula is completely analogous to that for the Fourier-transform based beam propagation method. For the sake of completeness, here is the expression corresponding to (2.52) — it shows explicitly that this method is restricted to axially symmetric situations:

$$E(r,z,t) = \text{CarrierWave} \;\times\; \text{Hankel}\left[\text{Propagator}(z,\omega,k_\perp)\;\text{Hankel}\left[E(\rho,0,0)\right]\right] \tag{2.70}$$

It is also to be understood that it is a scalar formula. Therefore it can only represent a vector field in an approximation in which the beam is more or less linearly polarized. The vector case is briefly discussed later in this section.

In the above expression, it was assumed that the normalization of the numerical implementation of the Hankel transform is chosen such that the inverse is equal to the transform itself. Of course, the same holds for the "distribution" of the normalization factors as for the Fourier-based case. So there is a degree of freedom here — this is something to pay attention to when utilizing a third-party implementation of DHT.

Unlike the FFT BPM, this method works with dense transformation matrices. While in the FFT BPM the transformation matrix is never stored or even evaluated explicitly, here one has to pre-calculate the transformation matrix and keep it stored in the memory. To express the method a bit more explicitly, we may write it in the matrix notation (leaving out the time dependence inherited from the carrier wave). Given the radial size of the computational domain R_{max}, the corresponding spectrum of transverse wavenumbers k_n is pre-calculated first, together with the transformation matrix. Then the propagation formula is simply the following matrix product:

$$E(r_k,z) \equiv E_k(z) = \sum_{n,m,l} H_{kl} P_{lm}(z,\omega) H_{mn} E_n(0) \tag{2.71}$$

where H_{kl} stands for the Hankel transform matrix, and the diagonal propagator matrix is calculated for the given step length z as

$$P_{lm}(z,\omega) = \delta_{lm} e^{izK_z(\omega,k_m)} \quad , \quad K_z(\omega,k) = \sqrt{\frac{\omega^2 n^2(\omega)}{c^2} - k^2}\;. \tag{2.72}$$

K_z can be replaced by its paraxial approximation if needed. In the non-paraxial case, when the grid resolution is so fine that the argument of the square root may become negative for high transverse wavenumbers, the imaginary part has to be chosen such that the wave is damped on propagation.

One difference from the FFT-based case is worthwhile to note. If the propagation step z was not to change, and many propagation steps were to be executed, the whole action of the propagation here expressed in three matrix multiplications could be stored in a single "full propagator" matrix. This would effectively eliminate one matrix-vector multiplication from the propagation scheme and make it almost twice as fast (note that the propagator application is diagonal and therefore relatively inexpensive). However, one has to keep in mind that evaluation of the full propagator matrix requires matrix-matrix multiplication (plus storage of an additional dense matrix) and that itself may be more expensive than the "unaccelerated" application of the method. This is why most of the time in practice it is better to apply (2.71) as is.

To conclude this subsection let us note that it is sometimes said that radially symmetric field propagation is better, or at least equally well treated by two-dimensional FFT transform. This argument is based on the fact that the DHT is a "slow" algorithm and its complexity scales as the square of the number of points sampling the domain. This is roughly the same complexity

as that for the two-dimensional FFT, so why not to use two dimensional representation with a radially symmetric field? This reasoning is in principle correct, however, in practice the algorithm using DHT will greatly outperform the 2D-FFT method. Yet another reason to apply a dedicated algorithm to radially symmetric problems is that their sampling on a square-lattice grid is rather unnatural and introduces anisotropy in the numerical solution. These artifacts can be reduced by improving resolution, but can not be completely eliminated.

Exercise:
a) Implement the discrete Hankel transform. It is practical to design it as an object that holds all related information, including the discrete values of radial coordinates, transverse wavenumbers, and the transformation matrix itself.
b) Implement DHT-based BPM algorithm. Demonstrate correctness through a comparison with the analytic solution of the Gaussian beam propagation. Hint: this will require the possibility to switch between paraxial and non-paraxial modes in the DHT BPM.

2.3.6 Application of DHT to hollow waveguides

In all numerical simulations, the computational domain is necessarily finite, and the issue of appropriate boundary conditions comes up. In the case of spectral methods, though, this aspect is relatively less visible. This is because of the nature of the transform, be it Fourier or Hankel, the boundary conditions are "given" and not subject to user's choice.

For the DHT method, the boundary condition says that the field vanishes at the edge of the computational domain (note that the point at the very edge is not among the discrete field samples, though). One physical interpretation of such conditions is that they represent propagation in a "tube" waveguide with a perfectly conducting shell. This forces the electric field to vanish. Apart from the discretization, which can in principle be refined until artifacts decay below an acceptable level, the method solves this physical situation exactly.

However, perfectly conducting cylindrical waveguide is an idealization. Fortunately there is a relatively simple way to modify the DHT-based BPM method to cylindrical waveguides with lossy walls. An important example is hollow waveguides or glass capillaries often used in nonlinear optics of gases. In what follows the modified method is described briefly. For the mathematical background, the Reader is referred to the paper by Marcatili and Schmeitzer, entitled *Hollow metallic and dielectric waveguides for long distance optical transmission and lasers* published in The Bell System Technical Journal, p. 1784, in July 1964. This is a classical reference that serves as a basis to justify the modified DHT method. Note that a completely analogous approach applies to waveguides with a simple slab geometry.

Assume that the waveguide under consideration is a glass capillary, characterized by its inner diameter of $2a$, and the "cladding" refractive index $n_c(\omega)$.

Exact modal solutions exist for these situations and they show that there exists a set of leaky modes. These leaky modes are localized in the hollow of the waveguide, and slowly radiate into cladding (which is assumed to be infinitely extended). Thus, the modes are lossy, and are in fact superpositions of infinitely many continuum-spectrum modes that together represent resonance solutions very much resembling metastable states of quantum mechanics.

It turns out that the loss decreases with the increasing refractive index contrast between the cladding and the material, e.g. gas, in the waveguide bore. Moreover, the spatial distribution of the exact modes is very similar to those describing the idealized loss-less waveguide (which in turn are exactly our Bessel beams used so far). The difference is that the exact modes have:

- small but non-zero value at the cladding boundary;
- small imaginary part that increases close to the material interface; and
- leaky modes are defined on an infinite domain $r \to \infty$ cladding interface.

All of these aspects are neglected in the modified DHT BPM for cylindrical waveguides. In other words, the spatial shape of the modal fields remains the same (in particular their domain is restricted to the inside of the waveguide), but their propagation constants are modified as to approximate the propagation constants of the exact modes.

For example for the vacuum in the hollow waveguide, the real parts of propagation constants remain unchanged from their values as defined within the DHT BPM:

$$\beta_n(\lambda) = \frac{2\pi}{\lambda}\sqrt{1 - \left(\frac{j_{0n}\lambda}{2\pi a}\right)^2}\,, \tag{2.73}$$

and the imaginary part is then added to β_n in the form

$$\alpha_n = \left(\frac{j_{0n}}{2\pi}\right)^2 \frac{\lambda^2}{a^3}\frac{1}{\sqrt{n_{cl}^2 - 1}} \tag{2.74}$$

This causes exponential decay on propagation. With the loss sharply increasing with the decreasing radius a of the waveguide. The above is valid in the paraxial approximation, but this hardly presents any restriction for applications. This is because the whole method modification is only good for weak losses. The important property of this method is that the propagation reflects the fact that the higher-order modes (i.e. those with more radial zeros) suffer higher propagation losses. As a consequence, the propagation acts as a spatial filter, gradually eliminating sharp spatial features in the beam. Asymptotically, the beam shape tends to that of the fundamental Bessel mode.

2.3.7 Vectorial Bessel Beams

For the FFT BPM, the vectorial nature of the electromagnetic wave poses no serious problem. The only issue it brings is that the polarization vector of each spectral amplitude must be always chosen orthogonal to the wave-vector that spectral amplitude belongs to. The situation is a bit more complicated with the radial DHT-based method. This has to do with the different radial grid sampling in different-order transforms.

The full general expansion of vector electromagnetic wave into vector Bessel beams is quite involved. Here we restrict our attention to the derivation of a correction to the effectively scalar approach discussed up to this point.

Let us therefore look at the radially symmetric analogues of plane-wave Maxwell solutions. As plane waves are a complete system any solution can be written in this form

$$\boldsymbol{\psi} = \int \boldsymbol{A}(\boldsymbol{k}) \exp\left[i\boldsymbol{k}\cdot\boldsymbol{r} - i\omega t\right] \tag{2.75}$$

Here the wavelength is fixed to ω, so we are effectively solving the Helmholtz equation. Because of the dispersion relation, it must hold that $k^2 = \omega^2 n(\omega)^2/c^2$. Due to the transverse nature of the wave, we also require that $\nabla\cdot\boldsymbol{\psi}$ must vanish, which in turn implies $\boldsymbol{k}\cdot\boldsymbol{A}(\boldsymbol{k}) = 0$. The above solution ansatz is z-invariant solution such that upon propagation it only acquires a phase change $\psi(z + \Delta z) = e^{i\beta\Delta z}\psi(z)$ where we must take $\beta = k_z = \sqrt{\omega^2 n(\omega)^2/c^2 - k_\perp^2}$. With this notation, and in polar coordinates, the thought after solution is

$$\boldsymbol{\psi}(r,\Phi) = \int_0^{2\pi} d\phi \int_0^{\infty} k_\perp dk_\perp \boldsymbol{A}(k_\perp,\phi) \exp\left[ik_\perp r(\cos\phi\cos\Phi + \sin\phi\sin\Phi) + i\beta z - i\omega t\right] \quad (2.76)$$

This is quite a general wave, which we want to restrict to a fixed value of the transverse wavenumber $k_\perp$. This is where the wave becomes conical. Since ω has already been fixed, fixing the amplitude of $k_\perp$ select the angle θ the angle between $\boldsymbol{k}$ and the optical axis $\boldsymbol{z}$.

Next we choose the spectral amplitude vector. The ansatz is such that it ensures that the wave or Helmholtz equation is satisfied. We only need to pay attention to the transversality or divergence constraint. Since the goal is to obtain a vector-corrected version of the Bessel beam we have been working with so far, let us choose:

$$\begin{aligned} A_x(\phi) &= \frac{1}{2\pi} A_0 \\ A_y(\phi) &= 0 \\ A_z(\phi) &= \frac{-1}{2\pi} \frac{k_\perp}{k_z} \cos\phi A_0 \end{aligned} \quad (2.77)$$

The x-component is supposed to be the dominant one and corresponds to the scalar approximation of before. The second condition says that the wave has no component in the y direction, and the third, z component is determined such that the divergence constraint is satisfied:

$$\boldsymbol{A}.\boldsymbol{k} = A_x k_x + A_z k_z = A_x k_\perp \cos\phi + A_z k_z = 0 \quad (2.78)$$

With this spectral amplitude parametrization, (2.76) needs to be evaluated, for example

$$\boldsymbol{\psi}_x = A_0 e^{i\beta z - i\omega t} \frac{1}{2\pi} \int_0^{2\pi} d\phi \exp\left[ik_\perp r\cos\phi\right] \quad (2.79)$$

This and the other non-zero component can be explicitly calculated using the integral representation for the Bessel functions

$$J_n(z) = \frac{i^{-n}}{2\pi} \int_0^{2\pi} e^{iz\cos\phi} \cos n\phi \ . \quad (2.80)$$

One obtains

$$\begin{aligned} \psi_x(r,\Phi) &= A_0 e^{ik_z z - i\omega t} J_0(k_\perp r) \\ \psi_y(r,\Phi) &= 0 \\ \psi_z(r,\Phi) &= A_0 \frac{k_\perp}{ik_z} e^{ik_z z - i\omega t} J_1(k_\perp r)\cos\Phi \end{aligned} \quad (2.81)$$

Here one can see that the wave is approximately linearly polarized in the x direction as long as $k_\perp$ is small in comparison to k_z, i.e. the angle of propagation θ in the conical wave is small. The correction comes in the form of the longitudinal field component which is concentrated close to the axis, but vanishes directly on-axis. Its angular dependence indicates that the longitudinal component is most intense above and below the axis.

As long as the longitudinal component can be neglected (for example in its contribution to non-linear medium response), the uncorrected Bessel expansion can play the role of plane-wave field representation. However, Bessel beams become *strictly radially symmetric* only in the paraxial approximation.

Yet another important observation to make here is that the longitudinal component of the vectorial Bessel beam is parameterized in the first-order Bessel function. For this order there exists the corresponding spectral transform as we discussed previously. However, it utilizes a different set of radial coordinates both in real and spectral space. This greatly complicates the practical usage of Bessel expansion in the truly vectorial situations because the different vector components must live on different grids.

2.4 Practice track: Discrete Hankel transform technique

Summary:

- Implementation of a discrete Hankel transform.
- Practical implementation of a DHT-based BPM.
- Test of DHT-BPM against the exact Gaussian beam solution.

2.4.1 Implementing and testing DHT

Here we implement a discrete Hankel transform (DHT) function that will serve as a building block for the beam propagation method based on expansion into Bessel waves. While this task is relatively straightforward, the program must be thoroughly tested. It will be used heavily in what follows in this course. Should one encounter problems in applications that utilize DHT, there must be absolutely no doubt that the transform works as it should.

Also from the standpoint of its future use, it is useful to realize the transform as an object that carries not only the transformation matrix itself, but also the corresponding coordinates in both the real space, and in the spectral space of transverse wavenumbers. Such an approach has no additional cost, yet makes it easier and, importantly, more robust to implement various functions related to the simulation grid and the corresponding spectral beam propagator.

It should be mentioned that DHT is not yet a standard numerical tool, and we need to implement our own. Since FFT, as a counterpart of DHT, is widely available in a number of high-performance libraries, it make little sense to try to code up one's own FFT function (most likely, such an attempt would result in a function that seriously lack in performance!). The current situation with DHT is quite different. Moreover, as this exercise surely demonstrates, its implementation is not difficult at all...

Task 1: DHT implementation
To keep this exposition as simple as possible, it will be assumed that the Bessel function zeros have been pre-calculated, and are stored in a text file *J0zeros.dat* which contains the first three thousand zeros of J_0. This number is sufficient for most practical applications.

Following the formulas given in the main text, the DHT function can be realized e.g. as follows:

Listing 2.7. DHT implementation in Matlab

```
function DHT = myDHT(rmax, Nr);

% set the domain size and number of grid points
DHT.rmax  = rmax;
DHT.Nr    = Nr;

% read BesselJ0 zeros from the provided file
allzeros  = textread('J0zeros.dat');

% select subset of zeros to use
subzeros  = allzeros(1:Nr);

% this represents the outer boundary
```

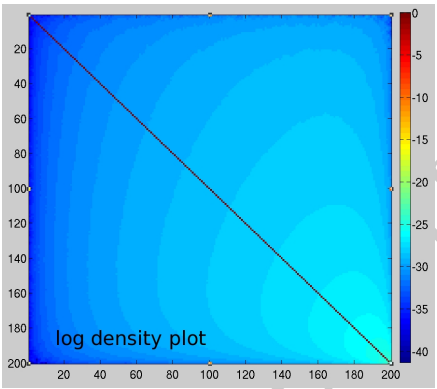
```
14  lastzero  = allzeros(Nr+1);
15
16  % create symmetric auxiliary
17  auxmatrix = besselj(0, 1/lastzero*subzeros*(subzeros'));
18
19  % auxiliary
20  auxvector = (besselj(1,subzeros)).^(-2);
21
22  % T will hold the transformation matrix
23  DHT.T     = 2/lastzero*auxmatrix*diag(auxvector);
24
25  % coordinates (radial) in real space
26  DHT.cr    = rmax*subzeros/lastzero;
27
28  % transverse wavenumbers
29  DHT.kt    = subzeros/rmax;
```

Task 2: Testing DHT matrix

The DHT has the property that it is its own inverse,

$$H.H = 1$$

which holds in the limit of the large matrix size. This relation offers a way to test our program, as illustrated in this figure:



Logarithmic density plot of the square of a Hankel-transform matrix $|H^2|$ of size $N = 200$. In the limit $N \to \infty$, it converges to a unity matrix. Here one can see that the deviations are extremely small already for what is a relatively small transformation matrix. This means that for all practical purposes, the discrete Hankel transform is its own inverse.

While the above is an important test to pass, it is of course not sufficient. It only shows that the DHT matrix represents *some* unitary transform. One must also check that it is the *correct* transform, i.e. that it transforms Bessel $J_0(k_n r)$ with k_n being the n-th transverse wavenumber, into a discrete delta function δ_{in} on the spectral grid. The same matrix must accomplish this mapping also in the reverse direction. It is left for the reader to demonstrate this.

2.4.2 Implementation of DHT-based beam propagation method

Having coded the discrete Hankel transform object, the implementation of the spectral beam propagation method for axially symmetric problems becomes quite simple.

First, one invokes the initialization function for DHT, passing arguments that give the radius of the computational domain and the number of grid points for discretization. At this stage, the DHT function will determine the allowed transverse wavenumbers, together with the location of grid point in the real space. Recall that neither of these arrays is equidistant, and that there exists no grid point on axis at zero radius. Since this step defines the computational domain for the simulation, it has to be executed immediately after setting simulation parameters, and in particular before calculation of the initial condition.

In the second step, one utilizes the vector of transverse wavenumbers calculated by the DHT function to prepare the linear propagator. Other parameters that are needed here are the propagation step Δz, and the wavenumber corresponding to the given wavelength, $k_0 = 2\pi/\lambda$. As for the propagator itself, one has two options, because the DHT-based BPM admits both paraxial and exact, non-paraxial propagation:

$$P(k_\perp) = \exp[-ik_\perp^2 \Delta z/(2k_0)]$$

gives the paraxial propagator for the transverse wavenumber k, and

$$P(k_\perp) = \exp[+i\Delta z(\sqrt{k_0^2 - k_\perp^2} - k_0)]$$

is the corresponding expression for the exact linear propagator, in which we have subtracted the carrier wave phase $e^{-i\Delta z k_0}$ so that the two versions coincide for small $k_\perp$.

Third step is to define a function that will execute one propagation step of length Δz encoded previously in the propagator. This consists in applying the Hankel transform onto a vector representing the beam amplitude at the given propagation distance, followed by point-by-point multiplication by the linear propagator, and finally going back to real space through another application of the Hankel transform.

The following listing shows Matlab realization of the above described three preparation steps for DHT-based BPM:

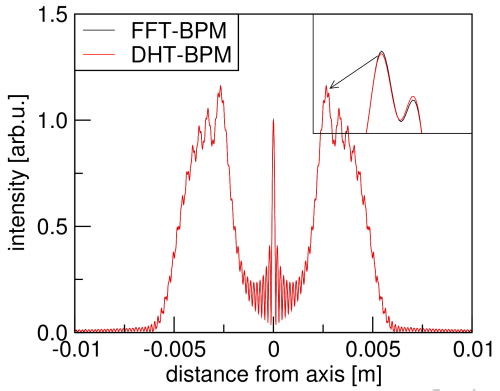
Listing 2.8. DHT implementation in Matlab

```
% prepare Hankel transform object
HT = myDHT(LR,NR);

% pre-calculate the linear propagator
pr = zeros(1,NR);
for x=1:NR

% in paraxial approximation:
    pr(x) = exp(-1i*(HT.kt(x)^2)/(2*k0)*dz );

% or non-paraxial:
    pr(x) = exp(1i*dz*(sqrt(k0*k0 - HT.kt(x)^2) - k0) );

end
pr = pr';

% define function that executes one linear step
LinearStep = @(amplitudein,propagator) HT.T*( propagator.*(HT.T*amplitudein) );
```

2.4.3 Testing DHT-BPM on the Spot of Arago problem

A particularly effective way to test simulation codes consists in double-coding, or solving the same problem with two different programs, preferably implementing two different algorithms. In this case we will compare solutions for the Poisson bright spot obtained by the FFT-BPM in two transverse dimensions and the DHT-BPM solution obtained on a one-dimensional radial grid. The two approaches share the spirit of the method, but their implementations have essentially nothing in common. Thus, passing this test will give a very strong indication that both implementations work correctly. This comparison will give the reader also the opportunity to compare the computational efficiency of the two methods. The short scripts required to execute both simulations are appended to this section for side-by-side comparison. The reader should appreciate the simplicity of the code and note the common structure of the two algorithms.

The simulations model difraction of a fourt-order super-Gaussian beam with radius of 5 mm, collimated onto a circular opaque obstacle with radius of 2 mm. The wavelength is chosen to be 633 nm. The final propagation distance is 1 meter, and this is simulated in twenty integration steps. In case of FFT-BPM, the computational domain is 3×3 cm large, with 4096×4096 grid points. For the DHT-BPM, the radius of the domain is 1.5 cm, and it is sampled with 2048 grid points. The following figure hows a comparison of the intensity profiles in the radial direction:



Comparison of FFT-based and DHT-based beam propagation methods. The solutions show the intensity of the diffraction pastern behind a circular obstacle illuminated by a collimated super-Gaussian beam. The insert shows that the two algorithms produce solutions that are barely distinguishable on the scale of this figure.

To conclude, note that the FFT-based simulation requires both a larger computational domain, and bigger computational effort. The DHT based method runs in a fraction of time needed by the FFT approach. But what is perhaps even more important, the problem setting, being axially symmetric fits the DHT method. On the contrary, data that are and should remain perfectly radially symmetric must never be put on a square-latice grid. We have learned in the Maxwell solver context that the anisotropy of the grid will show up in the results. Thus, in the given problem, the DHT methods is clearly preferred.

Listing 2.9. FFT-BPM simulation

```
% parameters
lambda = 633.0e-09;
LX     = 3.0e-02;
NX     = 4096;
dz     = 0.05;
LZ     = 1.0;

% derived parameters
k0   = 2*pi/lambda;
stps = LZ/dz;

% coordinates and transverse wavenumbers
dx   = LX/NX;
cx   = dx*(linspace(0,NX-1,NX)-NX/2);

dk   = 2*pi/LX;
kx   = zeros(1,NX);
for k=0:NX/2
      kx(1+k) = dk*k;
end
for k=NX/2+1:NX-1
      kx(1+k) = dk*(k - NX);
end

% amplitude holder, define an initial condition
% it represents a super-Gaussian
% with a hole in the center
am0 = zeros(NX,NX);
for x=1:NX
  for y=1:NX
    am0(x,y) = IC(sqrt(cx(x)^2 + cx(y)^2));
  end
end

% propagator, paraxial
pxy = zeros(NX,NX);
for x=1:NX
  for y=1:NX
  pxy(x,y) = exp(-1i*(kx(x)^2 + kx(y)^2)/(2*k0)*dz);end
end

% poor man's absorbing boundary guard
bxy = zeros(NX,NX);
for x=1:NX
  for y=1:NX
  bxy(x,y) = exp(-((cx(x)^2 + cx(y)^2)/(LX*LX/5)).^8);end
end

% define one linear step
LinearStep = @(A,P) ifft2(P.*fft2(A));

% execute propagation steps
am1 = am0;
for s=1:stps
  am1 = LinearStep(am1,pxy);
  am1 = am1.*bxy;
end



% save result for comparison with DHT-BPM
fout = fopen('amplitude_vs_x_FFT.dat','w');
for x=1:NX
  fprintf(fout,'%g %g\n',cx(x),abs(am1(x,NX/2+1)));
end
```

Listing 2.10. DHT-BPM simulation

```
% parameters - same as in p1FFT.m
lambda = 0.633e-06;
LR     = 1.5e-02;
NR     = 2048;
dz     = 0.05;
LZ     = 1.0;

% derived parameters
k0 = 2*pi/lambda;
stps = LZ/dz;

% prepare Hankel transofrm
HT = myDHT(LR,NR);

% amplitude holder, define an initial condition
% it represents a super-Gaussian
% with a hole in the center
am = zeros(1,NR);
for x=1:NR
    am(x) = IC(sqrt(HT.cr(x)^2));
end
am = am';

% propagator, paraxial
pr = zeros(1,NR);
for x=1:NR
   pr(x) = exp(-i*(HT.kt(x)^2)/(2*k0)*dz );end
pr = pr';

% poor man's absorbing boundary guard
br = zeros(1,NR);
for x=1:NR
    br(x) = exp(-( (HT.cr(x)^2)/(LR*LR/1.25)).^8 );end
br = br';

% define one linear step
LinearStep = @(A,P) HT.T*( P.*(HT.T*A) );

% execute propagation steps
for s=1:stps
  am = LinearStep(am,pr);
  am = am.*br;
end

% symmetrize radial functions for better viewing
am2save = abs([flipud(am);am]);
rc2save = [flipud(-HT.cr);HT.cr];

% save for comparison with p1FFT.m result
fout = fopen('amplitude_vs_radius_DHT.dat','w');
for x=1:2*NR
  fprintf(fout,'%g %g\n',rc2save(x),am2save(x));
end
```