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
```

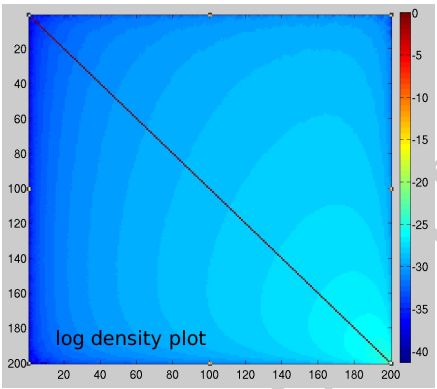
```
14  lastzero  = allzeros(Nr+1);
15
16  % create symmetric auxiliary
17  auxmatrix = besselj(0, 1/lastzero*subzeros*(subzeros'));
18
19  % auxiliary
20  auxvector = (besselj(1,subzeros)).^(-2);
21
22  % T will hold the transformation matrix
23  DHT.T     = 2/lastzero*auxmatrix*diag(auxvector);
24
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:

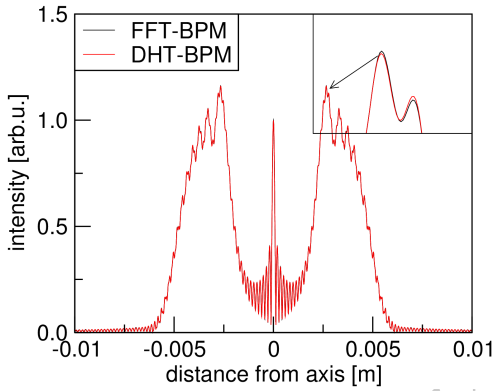
Listing 2.8. DHT implementation in Matlab

```
% 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
```