Report

Beam Propagation Method

Anne Weber, Felix Wechsler, Mingxuan Zhang

Group 4

June 10, 2020

Abbe School of Photonics

Friedrich-Schiller-Universität Jena

Contents

1 Introduction

In this report we present details of how to implement a beam propagation method. We are using the slowly varying envelope approximation (SVEA) and implement it with a 2nd order Crank-Nicolson method. We furthermore demonstrate this algorithm on a 1D guided mode and point out details how to optimize the numerical performance.

2 Physical Background

For simplicity we assume that we are in the paraxial regime of light propagation. Starting with the scalar Helmholtz equation for the wave's complex amplitude $U(\mathbf{r}, \omega)$ given by

$$\triangle U(\mathbf{r}, \omega) + k^2 U(\mathbf{r}) = 0 \tag{1}$$

we introduce an ansatz function which separates the fast oscillation in z direction from the slowly varying envelope:

$$U(\mathbf{r}) = A(\mathbf{r}) \exp(i\bar{k}z) \ . \tag{2}$$

Herein k is the wave number and thus $\bar{k}$ represents an average value, which must be chosen suited to the problem. $A(\mathbf{r})$ is the wave's envelope. However, important to note is, that this ansatz Equation 2 does not introduce any simplifications so far, it is just an alternative representation.
To derive the paraxial Helmholtz equation we now assume that $A(\mathbf{r})$ is slowly varying over the direction of propagation z. This allows us to neglect some second order derivatives in Equation 1. The final result is the paraxial Helmholtz equation:

$$\frac{\partial A(\mathbf{r})}{\partial z} = \frac{i}{2\bar{k}} \left(\frac{\partial^2 A(\mathbf{r})}{\partial x^2} + \frac{\partial^2 A(\mathbf{r})}{\partial y^2} \right) + i \frac{k^2 - \bar{k}^2}{2\bar{k}} A(\mathbf{r}) \tag{3}$$

In the following sections we explain how to numerically solve this equation.

3 Numerical Implementation

In this part we show the numerical implementation. We solve the problem in two dimensions. The coordinates are x and z, where z is the propagation direction. So the envelope will be characterized by x.

3.1 Background

In Equation 3 we can discretize the second order derivative with:

$$\frac{\partial^2 A(x,z)}{\partial x^2} \approx \frac{A_{j-1}^n - 2A_j^n + A_{j+1}^n}{(\Delta x)^2} \tag{4}$$

where A_j^n is the amplitude at position j at iteration step n (equivalent to z position at $n \cdot \Delta z$). x as position can be expressed as $j \cdot \Delta x$. The first order derivative at the left-hand side of Equation 3 can be expressed with a 2nd order central difference scheme:

$$\frac{\partial A(x,z)}{\partial z} \approx \frac{A_j^{n+1} - A_j^{n-1}}{2\Delta z} \tag{5}$$

We can define now an operator matrix for the differentiation with respect to x:

$$\tilde{\mathcal{L}} = \begin{bmatrix} a_1 & b_1 & 0 & \ddots \\ b_1 & a_2 & b_2 & \ddots \\ 0 & b_2 & a_3 & \ddots \\ \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{6}$$

where $a_j = \frac{-i}{k(\Delta x)^2} + i\frac{k_0^2 n_j^2 - \bar{k}^2}{2k}$ and $b_j = \frac{i}{2k(\Delta x)^2}$. n_j is the refractive index at position $n(j \cdot \Delta x)$. Finally, combining Equation 3, Equation 4 and Equation 6 yields the Crank-Nicolson Method:

$$\left(\mathbb{1} - \frac{\Delta z}{2}\tilde{\mathcal{L}}\right)\mathbf{A}^{n+1} = \left(\mathbb{1} + \frac{\Delta z}{2}\tilde{\mathcal{L}}\right)\mathbf{A}^n \tag{7}$$

where $\mathbf{A}^n = [A_1^n, A_2^n, .., A_N^n]^T$ is a vector of the field at different x positions and $\mathbb{1}$ the identity matrix. N is the total amount of sampled x positions.

3.2 Implementation

Equation 7 is the equation we want to solve. A naive approach would be to explicitly invert the matrix on the left-hand side. However, this matrix (let us call it `Ml`) is sparse and would possibly result in a non-sparse inverted matrix. This can be computationally expensive. A better approach is to factorize the matrix using the function `factorized(Ml)` from the package `scipy.sparse.linalg` [Vir+20]. This function performs a LU decomposition and returns a function which can be used to solve Equation 7 efficiently. This linear system of equation can be solved for every

step in z direction and the results are stored. Via this procedure one obtains the field evolution over z.

4 Results

In this section we present the findings ouf our simulations.
As initial field we choose a Gaussian beam:

$$A^0 = \exp\left(-\frac{x^2}{w^2}\right) \tag{8}$$

The refractive index distribution is a simple step-index profile with rectangular step width x_b and the two refractive indices n_{cladding} and n_{core}. The simulation result can be seen in Figure 1.

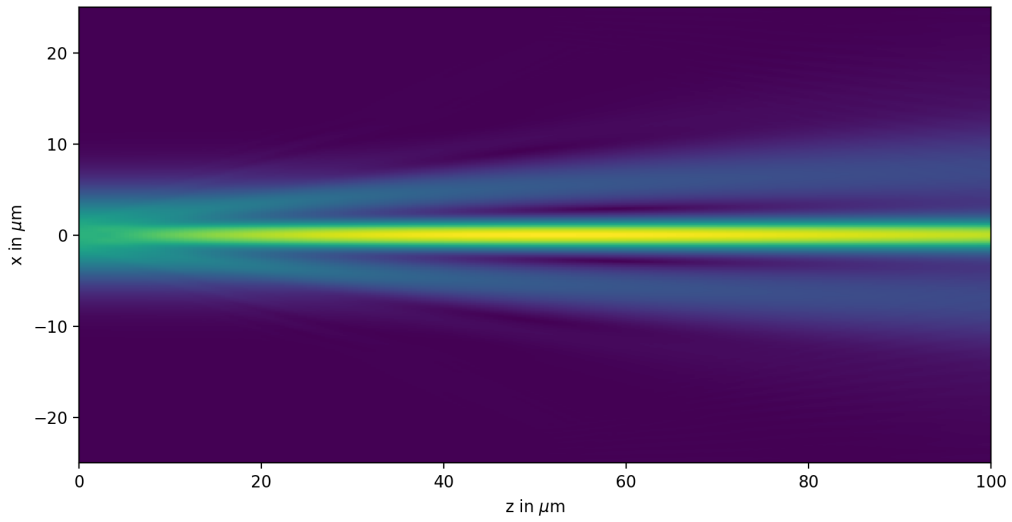

Figure 1: Propagation of a Gaussian beam with $w = 5\,\mu\text{m}$, $\Delta z = 0.5\,\mu\text{m}$, $\lambda = 1\,\mu\text{m}$, $\Delta x = 0.5\,\mu\text{m}$, $x_b = 2\,\mu\text{m}$, $n_{\text{cladding}} = 1.46$, $n_{\text{core}} = 1.45$, $\bar{k} = k_0 \cdot 1.455$.

We observe that only the central part of the Gaussian beam is guided since the waveguide's width is smaller than the beam's. Hence, the non-central part spreads out.

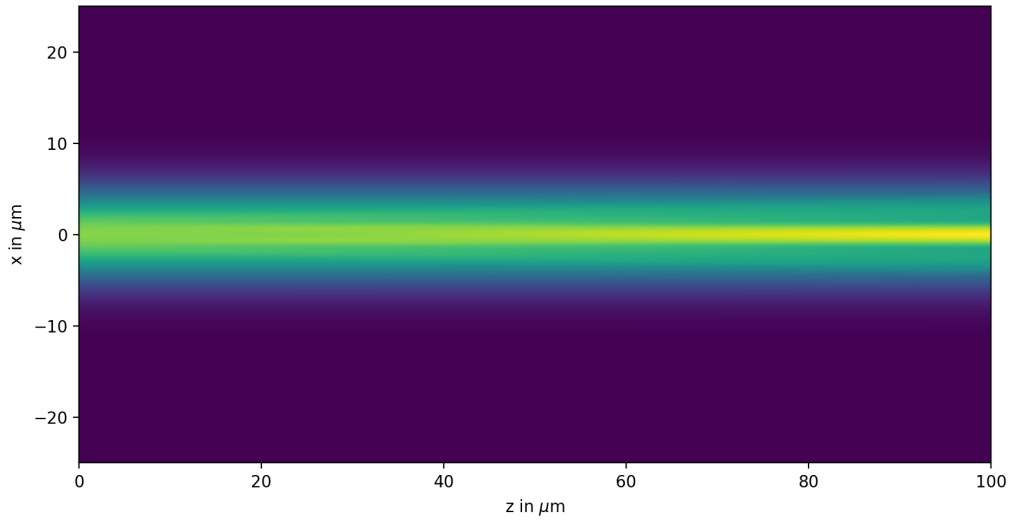

Figure 2: Propagation of a Gaussian beam with the same parameters as in Figure 1, but $\bar{k} = k_0 \cdot 3$.

In Figure 2 we can observe the effect when $\bar{k}$ is chosen too large. The approximation of a slowly varying amplitude is violated and the two figures are not identical. The figure indicates that the beam would be better guided. One effect due to the perfectly conducting boundaries can be seen in Figure 3.

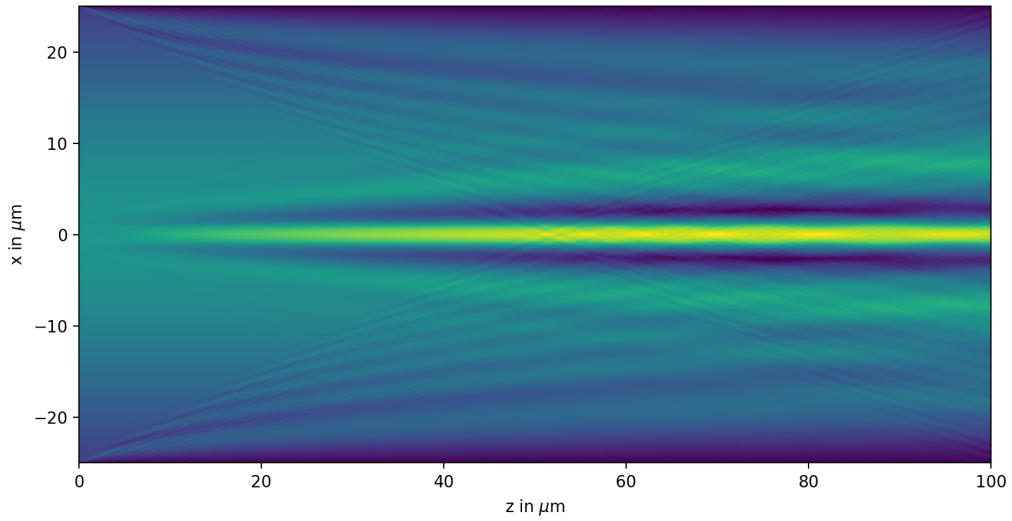

Figure 3: Propagation of a Gaussian beam with the same parameters as in Figure 1 but $w = 25\,\mu\text{m}$.

Some waves are reflected at the top and the bottom. Therefore one needs to choose a larger simulation grid or more suitable boundary conditions.

4.1 Computational Performance

Furthermore we provide some data for the computational performance. Our setup for all measurements was Python 3.8.2, SciPy 1.3.2, NumPy 1.17.3 and a Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz. The datatype for the performance critical part was single float. For the parameters of Figure 1 the elapsed time of the LU method was (0.012 ± 0.001) s. Instead of doing the factorization, we could also solve the system of equations in every step via `spsolve`. For this the elapsed time was (0.044 ± 0.005) s. So doing the factorization already improves the performance by a factor of 4. The slowest approach is the explicit inversion (once before the **for**-loop) of `Ml`. This approach takes (0.81 ± 0.01) s, being a factor of roughly 70 slower than the factorization.

4.2 Convergence Rate

We can vary the step sizes both Δz and Δx to check the convergence rate. By theory it is quadratically convergent for both lateral and traversal step size. First we can calculate a reference field with high resolution in x. Then we simulate another field with a lower x resolution. To compare the high resolution field with the low resolution field, we need to downscale the high resolution field to be the same size as the smaller one. After the downscaling we can calculate the mean error of the fields via:

$$\text{error}_x = \frac{1}{N_x} \sum_i \left| |A_i^n| - |A_i^{\text{ref}}| \right| \tag{9}$$

where A_i^{ref} is the high resolution reference field. Because the field values in general are complex, we use the absolute values. In Figure 4 we see the error plotted over the different step sizes Δx. The orange curve indicates a fitted function $a \cdot (\Delta x)^b$ with $a = (3.36 \pm 0.02) \cdot 10^{-5}$ and $b = 2.09 \pm 0.01$. b is the rate of convergence. Since b is larger than 2 we observe a convergence which is slightly higher than expected.

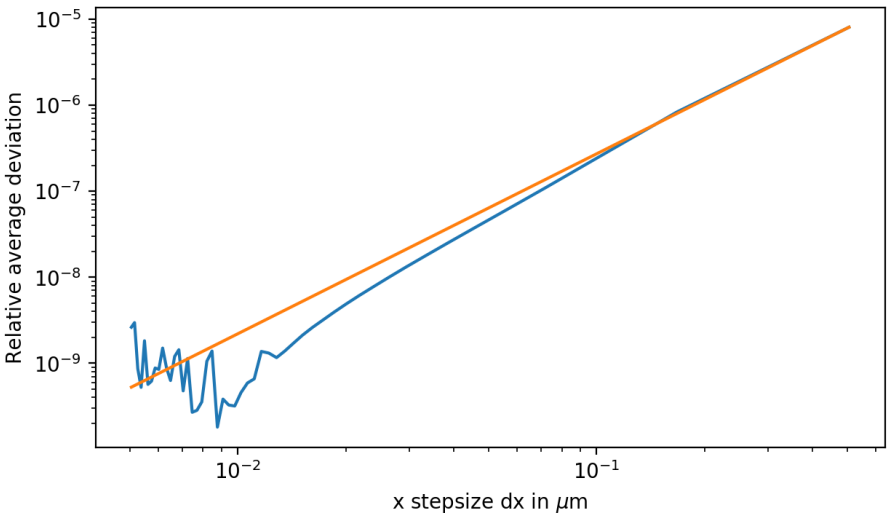

Figure 4: Convergence behaviour with different x sampling. Same set of parameters as in Figure 1.

For the error with different Δz we do not need to scale the reference field because we can only evaluate the field at the final z position. Figure 5 shows the convergence behaviour for different z sampling. The fitted parameters of the orange curve are $a = (5.08 \pm 0.06) \cdot 10^{-5}$ and $b = 0.96 \pm 0.02$. For z the convergence is therefore one order lower than expected. For the difference we do not have a reasonable explanation.

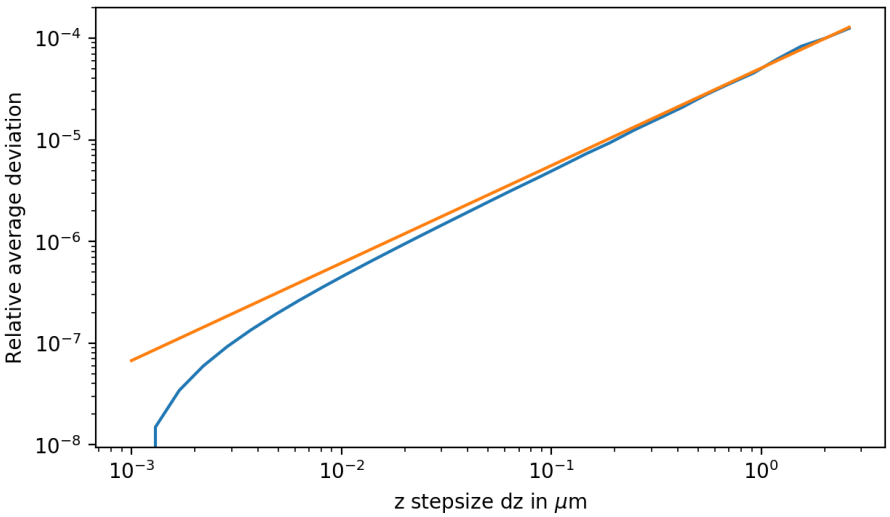

Figure 5: Convergence behaviour with different z sampling. Same set of parameters as in Figure 1.

5 Conclusion

In this report we presented the physical background of the beam propagation. Following we showed details of the numerical implementation and the discretization in propagation direction z and traversal coordinate x. Additionally we included considerations on computation efficiency, regarding the solving of the principal equation. At the end we provided some insights into the convergence behaviour of the Crank-Nicolson scheme. Despite the method being a second order method we did not achieve the same convergence rate for Δz. For Δx we were able to confirm the convergence order.

References

[PRK20] T. Pertsch, C. Rockstuhl, and T. Kaiser. *Beam Propagation Method - Slowly Varying Envelope Approximation*. Lecture Notes on Computational Photonics. 2020.

[Vir+20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. 1. 0. Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.