Computational Photonics

Seminar 06, 14.06.2024

Homework 2:
Implementation of the Beam Propagation Method

- Implementation of the explicit-implicit Crank-Nicolson scheme
- Test by propagating a Gaussian beam through an inhomogeneous medium

Abbe School of Photonics | JENA
Friedrich-Schiller-Universität

Initial-value problem

- **Rearranging** the paraxial wave equation yields

$$\frac{\partial}{\partial z} v(x,z) = \left[\frac{i}{2\bar{k}}\frac{\partial^2}{\partial x^2} + i\,\frac{k^2(x)-\bar{k}^2}{2\bar{k}}\right] v(x,z) = \boldsymbol{L}v(x,z)$$

discretization of longitudinal coordinate (*z*)

discretization of transverse coordinate (*x*) ✓

$$\mathbf{L} = \frac{i}{2\bar{k}(\Delta x)^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix} + i\begin{pmatrix} W_1 & 0 & \cdots & & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & W_j & & \\ & & & \ddots & 0 \\ 0 & \cdots & & 0 & W_N \end{pmatrix}$$

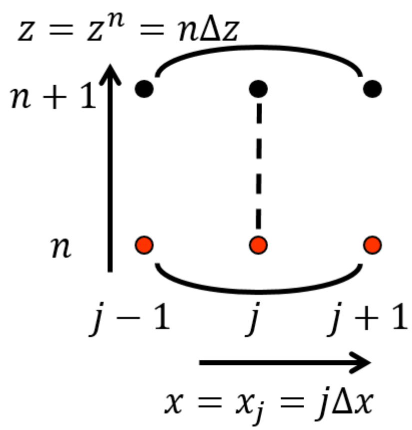
$$W(x) = \frac{k^2(x)-\bar{k}^2}{2\bar{k}}$$

Abbe School of Photonics | JENA
Friedrich-Schiller-Universität

Discretization of longitudinal coordinate

Central difference
(Explicit-Implicit scheme)

$$\frac{\partial}{\partial z} v_j(z)\bigg|_{z^{n+1/2}} \approx \frac{v_j^{n+1} - v_j^n}{\Delta z} \approx \frac{1}{2}\sum_l \left(L_{jl} v_l^n + L_{jl} v_l^{n+1}\right)$$



stable and **energy conserving**

$$\left(\mathbf{I} - \tfrac{1}{2}\Delta z \mathbf{L}\right)\mathbf{v}^{n+1} = \left(\mathbf{I} + \tfrac{1}{2}\Delta z \mathbf{L}\right)\mathbf{v}^n$$

$$\mathbf{A}\mathbf{v}^{n+1} = \mathbf{B}\mathbf{v}^n$$

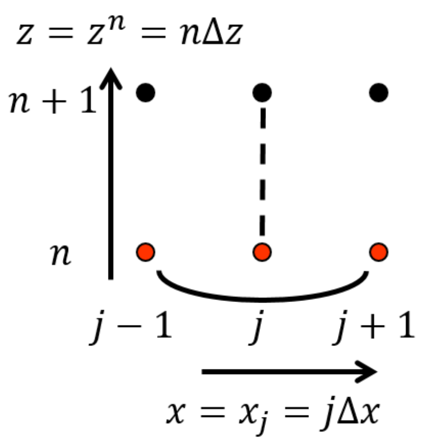
Discretization of longitudinal coordinate

Discretization of the z coordinate

$$z^n = n\Delta z$$
$$v_j^n = v_j(n\Delta z)$$

Forward difference (**Explicit** scheme)

$$\left.\frac{\partial}{\partial z} v_j(z)\right|_{z^n} \approx \frac{v_j^{n+1} - v_j^n}{\Delta z} \approx \sum_l L_{jl} v_l^n$$

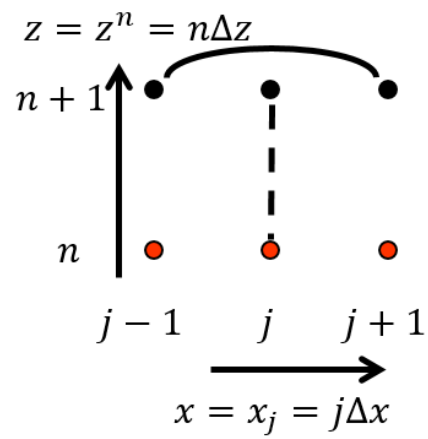


always unstable
error grows exponentially

$$\mathbf{v}^{n+1} = (\mathbf{I} + \Delta z\mathbf{L})\mathbf{v}^n$$
$$\mathbf{v}^{n+1} = \mathbf{A}\mathbf{v}^n$$

Backward difference (**Implicit** scheme)

$$\left.\frac{\partial}{\partial z} v_j(z)\right|_{z^{n+1}} \approx \frac{v_j^{n+1} - v_j^n}{\Delta z} \approx \sum_l L_{jl} v_l^{n+1}$$



stable but in general **not energy conserving**

$$\mathbf{v}^{n+1} = (\mathbf{I} - \Delta z\mathbf{L})^{-1}\mathbf{v}^n$$
$$\mathbf{v}^{n+1} = \mathbf{A}^{-1}\mathbf{v}^n$$

Tasks

1. Implement the functions `waveguide` and `gauss` to set up the problem:
   Slab waveguide with step index profile
   excited by a Gaussian initial field distribution

2. Implement the BPM using only the explicit-implicit scheme (function `beamprop_CN`)

3. Test the convergence and accuracy of obtained results vs. parameters ***dz*** and ***Nx***

4. *(Voluntary) Implement the BPM using the explicit (forward) and implicit (backward) scheme and compare the outcome to the results obtained with the Crank-Nicolson-scheme.*

Task I: Functions **waveguide**

```
def waveguide(xa, xb, Nx, n_cladding, n_core):
```

...

```
    Returns
    -------
        n : 1d-array
            Generated refractive index distribution
        x : 1d-array
            Generated coordinate vector
    '''

    x = np.linspace(-0.5*xa, 0.5*xa, Nx)
    n = np.ones_like(x)*n_cladding
    n[np.abs(x) <= 0.5*xb] = n_core
    return n, x
```

Task I: Functions **gauss**

```
def gauss(xa, Nx, w):
```

...

```
    Returns
    -------
        v : 1d-array
            Generated field distribution
        x : 1d-array
            Generated coordinate vector
    '''
    x = np.linspace(-xa/2,xa/2,Nx)
    v = np.exp(-x**2/w**2)

    return v, x
```

Task II: Functions **beamprop_CN**

```
def beamprop_CN(v_in, lam, dx, n, nd,  z_end, dz, output_step):
```

...

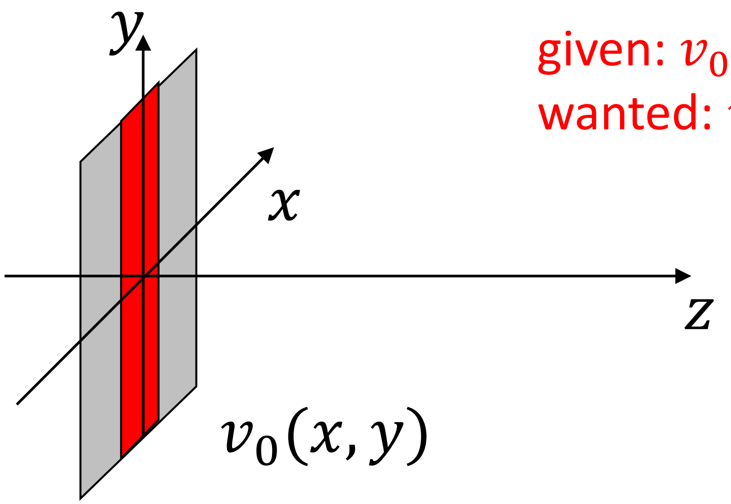
```
    Nx = len(v_in)
    Nz = round(z_end/dz)
    v = np.zeros((Nz, Nx), dtype=np.complex128)
    v[0, :] = v_in

    k0 = 2*np.pi/lam

    kbar = k0*nd
    one = np.ones((Nx,))
    l1 = sps.spdiags([one, one*-2, one],
                     [-1,       0,    1],
                     Nx, Nx)

    kj = k0*n
    W_diag = (kj**2-kbar**2)/(2*kbar)
    l2 = sps.spdiags(W_diag, 0, Nx, Nx)

    L = 1j/(2*kbar*dx**2) * l1 + 1j * l2
```

Task II: Functions **beamprop_CN**

```
def beamprop_CN(v_in, lam, dx, n, nd,  z_end, dz, output_step):
```

...

```
    Nx = len(v_in)
    Nz = round(z_end/dz)
    v = np.zeros((Nz, Nx), dtype=np.complex128)
    v[0, :] = v_in
```



given: $v_0\,(x) = v(x, z = 0)$
wanted: $v(x, z)$

Task II: Functions **beamprop_CN**

```
def beamprop_CN(v_in, lam, dx, n, nd,  z_end, dz, output_step):
```

...

```
    Nx = len(v_in)
    Nz = round(z_end/dz)
    v = np.zeros((Nz, Nx), dtype=np.complex128)
    v[0, :] = v_in

    k0 = 2*np.pi/lam

    kbar = k0*nd
    one = np.ones((Nx,))
    l1 = sps.spdiags([one, one*-2, one],
                     [-1,      0,   1],
                     Nx, Nx)

    kj = k0*n
    W_diag = (kj**2-kbar**2)/(2*kbar)
    l2 = sps.spdiags(W_diag, 0, Nx, Nx)

    L = 1j/(2*kbar*dx**2) * l1 + 1j * l2
```

$$\mathbf{L} = \mathbf{l}_1 + \mathbf{l}_2$$

$$\boldsymbol{l}_1 = \frac{i}{2\bar{k}(\Delta x)^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix}$$

$$\boldsymbol{l}_2 = i\begin{pmatrix} W_1 & 0 & \cdots & & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & W_j & & \\ & & & \ddots & 0 \\ 0 & \cdots & & 0 & W_N \end{pmatrix}$$

Task II: Functions **beamprop_CN**

```
def beamprop_CN(v_in, lam, dx, n, nd,  z_end, dz, output_step):
```

...

```
    A = sps.eye(Nx) - 0.5*dz*L
    B = sps.eye(Nx) + 0.5*dz*L

    for m in range(Nz-1):
        v[m+1, :] = sps.linalg.spsolve(A, B.dot(v[m, :]))

    v_out = v[::output_step]
    z = np.arange(0, z_end, dz*output_step)

    return v_out, z
```

$$\left(\mathbf{I}-\tfrac{1}{2}\Delta z\mathbf{L}\right)\mathbf{v}^{n+1}=\left(\mathbf{I}+\tfrac{1}{2}\Delta z\mathbf{L}\right)\mathbf{v}^{n}$$
$$\mathbf{A}\mathbf{v}^{n+1}=\mathbf{B}\mathbf{v}^{n}$$

Task II: Functions **beamprop_CN vs. ex-/implicit**

$$\left(\mathbf{I} - \tfrac{1}{2}\Delta z\mathbf{L}\right)\mathbf{v}^{n+1} = \left(\mathbf{I} + \tfrac{1}{2}\Delta z\mathbf{L}\right)\mathbf{v}^{n}$$
$$\mathbf{A}\mathbf{v}^{n+1} = \mathbf{B}\mathbf{v}^{n}$$

Crank-Nicolson:

```
for m in range(Nz-1):
    v[m+1, :] = sps.linalg.spsolve(A, B.dot(v[m, :]))
```

$$\mathbf{v}^{n+1} = (\mathbf{I} + \Delta z\mathbf{L})\mathbf{v}^{n}$$
$$\mathbf{v}^{n+1} = \mathbf{A}\mathbf{v}^{n}$$

Explicit:

```
for m in range(Nz-1):
    v[m+1, :] = A.dot(v[m, :])
```

$$\mathbf{v}^{n+1} = (\mathbf{I} - \Delta z\mathbf{L})^{-1}\mathbf{v}^{n}$$
$$\mathbf{v}^{n+1} = \mathbf{A}^{-1}\mathbf{v}^{n}$$

Implicit:

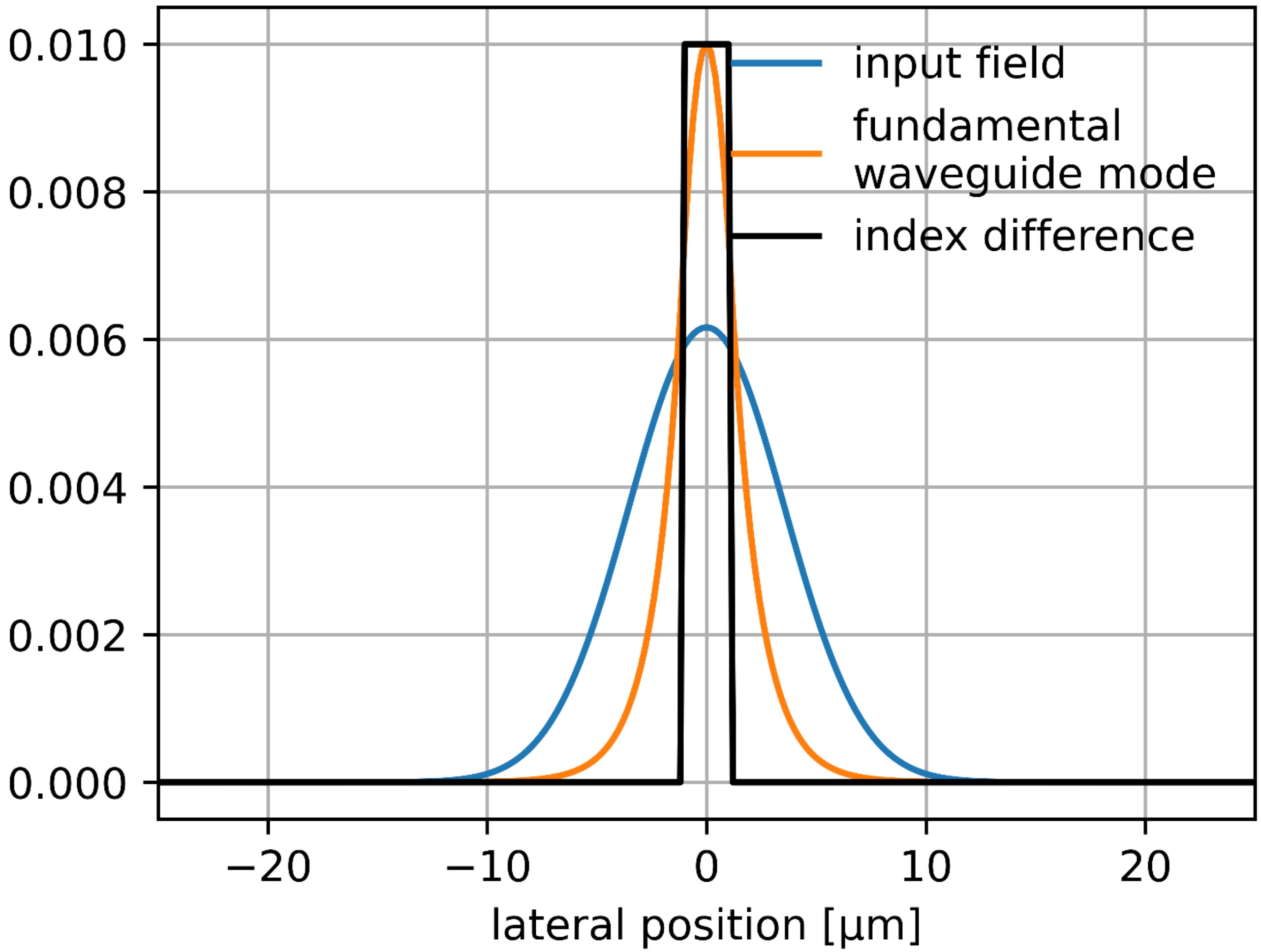
```
for m in range(Nz-1):
    v[m+1, :] = sps.linalg.spsolve(A, v[m, :])
```

Task II: Test parameters

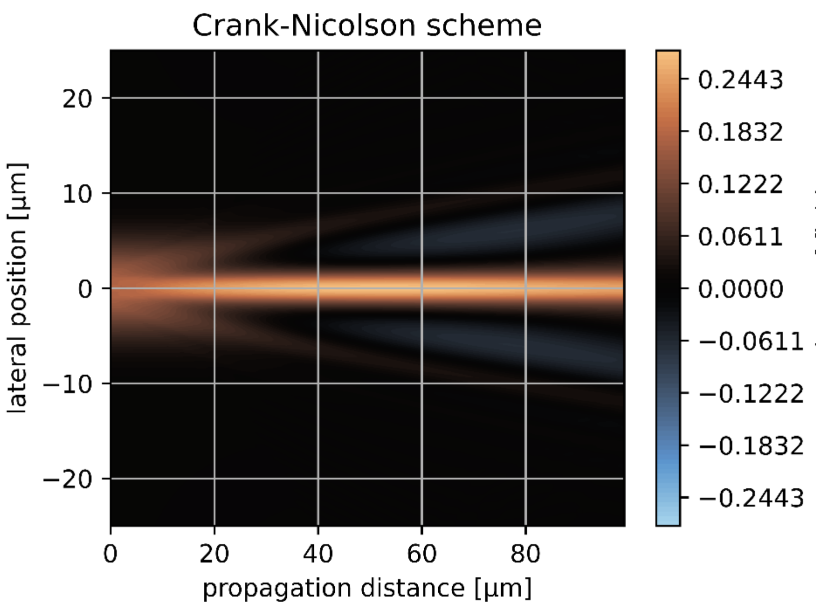
- Test your implementation with the following parameters:
  - Waveguide: `xa = 50µm`, `xb = 2µm`, `Nx = 251`, `n_cladding = 1.45`, `n_core = 1.46`
  - Initial field: `w = 5.0µm`
  - Solver: `z_end = 100µm`, `dz = 0.5µm`, `nd = 1.455`, `lambda = 1µm`

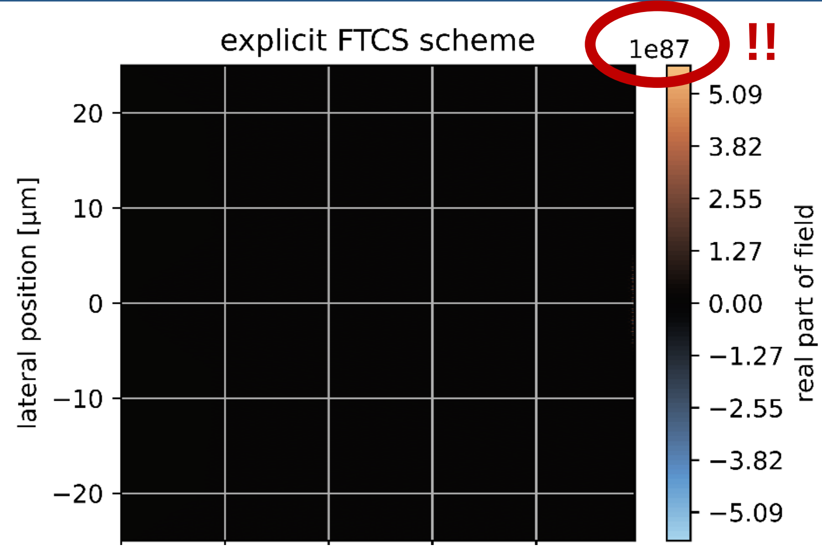
- The solution will vary slowly along z, you can choose `output_step` larger than 1 (e.g. `output_step*dz = 1.0µm`)

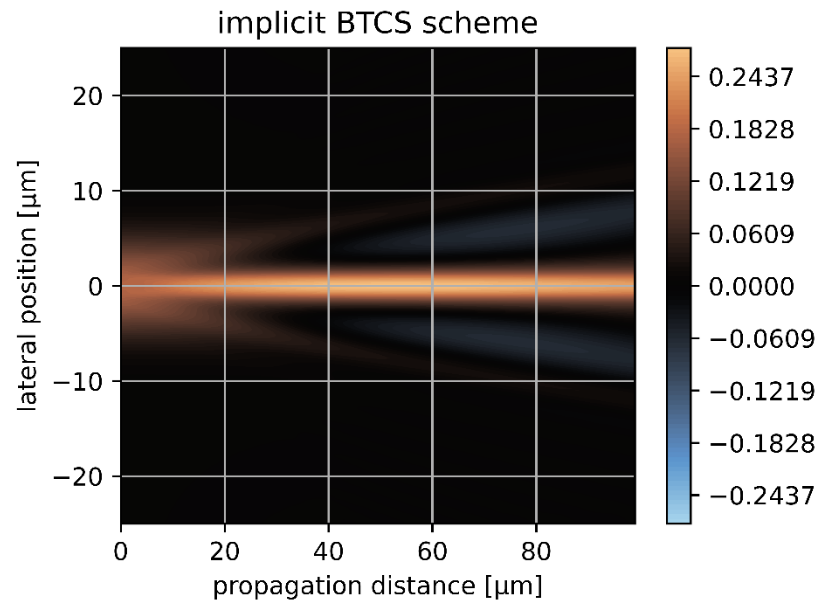
Task II: Test parameters



Task II: Test parameters

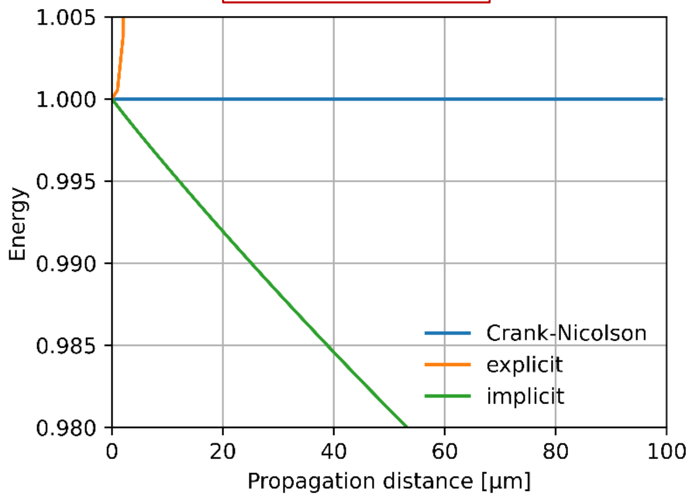




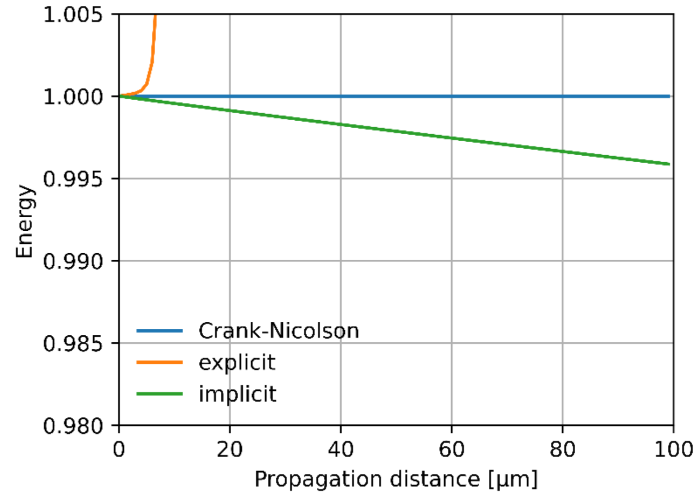


Task III: Testing the effect of dz for different schemes

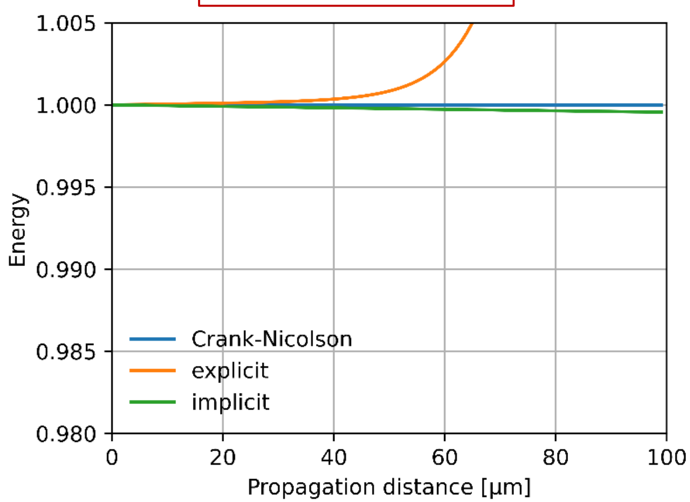
dz = 0.5 µm



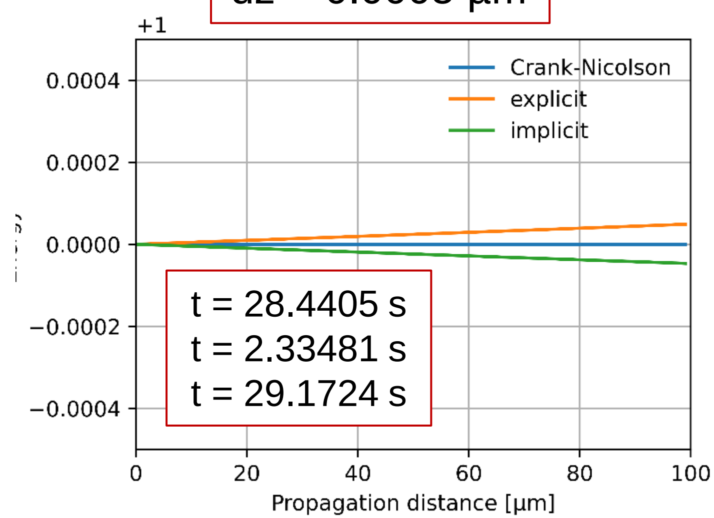
dz = 0.05 µm



dz = 0.005 µm

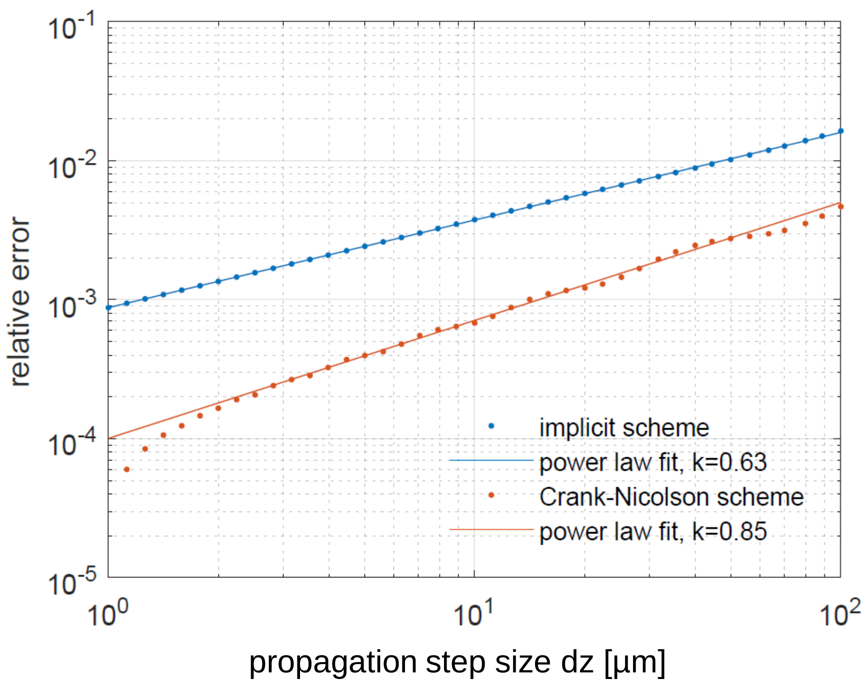


dz = 0.0005 µm



Task III: Convergence scaling

```
Propagation length = 20 µm
Gaussian width = 2.0 µm
dx = 0.005 µm
```

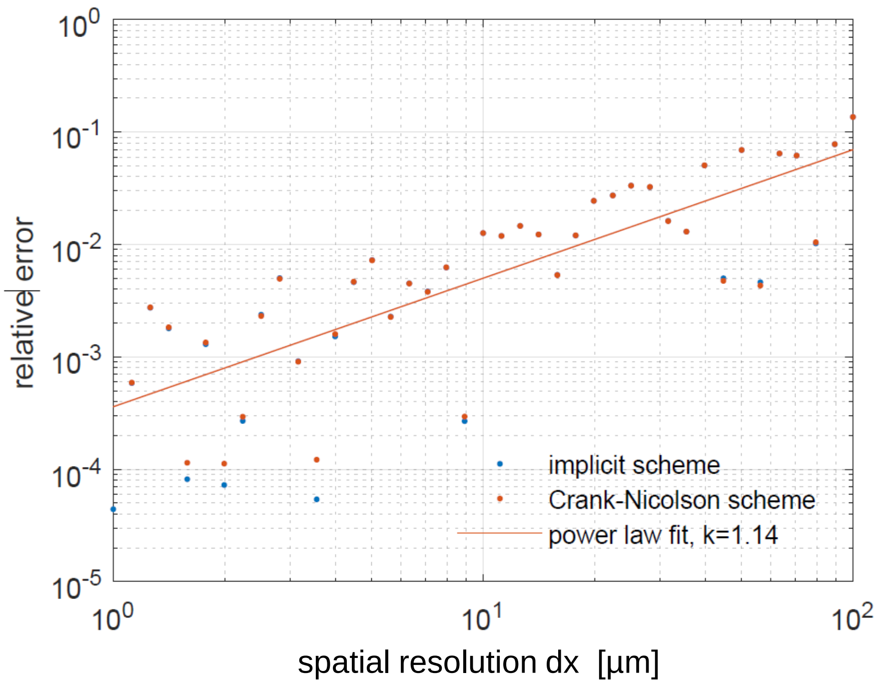


$$e = \sqrt{\sum_{i=1}^{N_x} \left| v_i(z_{\text{end}}) - v_i^*(z_{\text{end}}) \right|^2}.$$

solution for ***given*** *dz* — *solution for* ***smallest*** *dz*

Task III: Convergence scaling

```
Propagation length = 20 µm
Gaussian width = 2.0 µm
dz = 0.005 µm
```



Solution in the center of the waveguide for ***given*** *dx*

Solution in the center of the waveguide for ***smallest*** *dx*

$$e = \frac{\left| v_{(N_x-1)/2+1}(z_{\text{end}}) - v^*_{(N_x-1)/2+1}(z_{\text{end}}) \right|}{\left| v^*_{(N_x-1)/2+1}(z_{\text{end}}) \right|}$$

- Erratic change of relative error with spatial resolution due to refractive index step
- Take home message: convergence may be dependent on geometry / meshing !!