Computational Photonics
Tutorial 5:
Finite element method (for the scalar wave equation)

5 July 2023

Dr. Markus Nyman, markus.nyman@kit.edu
M.Sc. Nigar Asadova, nigar.asadova@kit.edu

Exercises for everyone:

- Task 1: 1D FEM for the scalar wave equation

Extended exercises:

- Task 2*: 2D FEM for the scalar wave equation

Reminder: sparse matrices

For FEM it is convenient to construct the system matrices incrementally:

- create a dictionary-of-keys matrix with `dok_matrix`
- fill it
- transform to compressed sparse column format with `.tocsc()`
- now linear algebra will be faster (for example `eigs`)

You can use `.todense()` to make a dense matrix for debugging purposes.

1D equation to solve

$$\frac{\partial^2 u(x)}{\partial x^2} + k_0^2 \epsilon u(x) = \beta^2 u(x)$$

$$(\mathbf{S} + \mathbf{W})\mathbf{u} = \beta^2 \mathbf{M}\mathbf{u}$$

where

$$S_{m,n} = -\int \frac{\partial \phi_m(x)}{\partial x} \frac{\partial \phi_n(x)}{\partial x} dx$$
$$W_{m,n} = k_0^2 \int \phi_m(x) \epsilon(x) \phi_n(x) dx$$
$$M_{m,n} = \int \phi_m(x) \phi_n(x) dx$$

Generalized eigenvalue problem with system matrix $\mathbf{S} + \mathbf{W}$ and mass matrix $\mathbf{M}$, solvable with `scipy.sparse.linalg.eigs`.

Piecewise linear basis functions

Prototype element $x \in [0,1]$, only two prototype basis functions on it:

$$
\begin{aligned}
\phi_0(x) &= 1 - x \\
\phi_1(x) &= x
\end{aligned}
$$

Integrals needed for later

$$
\begin{aligned}
\int_0^1 \phi_0(x)\phi_0(x)dx &= \frac{1}{3} & \qquad \int_0^1 \frac{\partial \phi_0(x)}{\partial x}\frac{\partial \phi_0(x)}{\partial x}dx &= 1 \\
\int_0^1 \phi_0(x)\phi_1(x)dx &= \frac{1}{6} & \qquad \int_0^1 \frac{\partial \phi_0(x)}{\partial x}\frac{\partial \phi_1(x)}{\partial x}dx &= -1 \\
\int_0^1 \phi_1(x)\phi_1(x)dx &= \frac{1}{3} & \qquad \int_0^1 \frac{\partial \phi_1(x)}{\partial x}\frac{\partial \phi_1(x)}{\partial x}dx &= 1
\end{aligned}
$$

Integrals on an arbitrary element

Transformation from arbitrary element to prototype element (notation: element's untransformed basis functions ϕ_E, prototype basis functions ϕ)

$$\int_{x_a}^{x_b} \phi_{E,m}(x)\phi_{E,n}(x)dx = (x_b - x_a)\int_0^1 \phi_m(x')\phi_n(x')dx'$$

$$\int_{x_a}^{x_b} \frac{\partial \phi_{E,m}(x)}{\partial x}\frac{\partial \phi_{E,n}(x)}{\partial x}dx = \frac{x_b - x_a}{(x_b - x_a)^2}\int_0^1 \frac{\partial \phi_m(x')}{\partial x'}\frac{\partial \phi_n(x')}{\partial x'}dx'$$

On the RHS, the integrals are exactly the same as the ones on the previous slide, and the prefactors reflect the fact that different elements (intervals) can have different sizes.

Practical implementation of 1D FEM

Defining the mesh and distribution of ϵ:

- Define nodes $[x_0, x_1, x_2, \ldots]$.
- Element n is an interval between nodes x_n and x_{n+1}.
- Let ϵ be constant on each element: $[\epsilon_0, \epsilon_1, \ldots]$

Constructing the system

- Create empty sparse matrices $\mathbf{S}$, $\mathbf{W}$ and $\mathbf{M}$.
- For each element n with length $L_n = x_{n+1} - x_n$, and corresponding field unknowns u_n and u_{n+1}:
  - Add to second derivative operator:
    - $S_{n,n} += -1/L_n$, this is $-1/L_n \int \partial_x \phi_0(x') \partial_x \phi_0(x') dx'$
    - $S_{n+1,n+1} += -1/L_n$, this is $-1/L_n \int \partial_x \phi_1(x') \partial_x \phi_1(x') dx'$
    - $S_{n,n+1} += 1/L_n$, this is $-1/L_n \int \partial_x \phi_0(x') \partial_x \phi_1(x') dx'$
    - $S_{n+1,n} += 1/L_n$, this is $-1/L_n \int \partial_x \phi_1(x') \partial_x \phi_0(x') dx'$
  - Add to wave number operator:
    - $W_{n,n} += k_0^2 \epsilon_n L_n \frac{1}{3}$, this is $k_0^2 \epsilon_n L_n \int \phi_0(x') \phi_0(x') dx'$
    - $W_{n+1,n+1} += k_0^2 \epsilon_n L_n \frac{1}{3}$
    - $W_{n,n+1} += k_0^2 \epsilon_n L_n \frac{1}{6}$
    - $W_{n+1,n} += k_0^2 \epsilon_n L_n \frac{1}{6}$
  - Add to mass matrix: same as wave number operator but without $k_0^2 \epsilon_n$.

Task 1: 1D FEM

Implement the FEM with linear basis functions for 1D wave equation as outlined above.
Two test cases:

- Sanity check for matrix construction: $\lambda_0 = 1$, nodes $x = [0, 1, 2, 3]$, $\epsilon = 1$ everywhere. The resulting matrices are given on the next slide.
- 1D silicon waveguide:
  - $\lambda_0 = 1\,\mu\text{m}$
  - Domain width $4\,\mu\text{m}$, core width $0.8\,\mu\text{m}$
  - $\epsilon = 2.25$ in cladding, $\epsilon = 12$ in core
  - Mesh: use fine discretization ($0.01\,\mu\text{m}$ step) in the core and in its immediate surroundings ($x \in [-0.6, 0.6]$ um), coarse discretization elsewhere ($0.05\,\mu\text{m}$ step)
  - "Natural" boundary conditions (no need to separately implement): $du/dx = 0$
  - Plot the first six eigenmodes (five guided, one unguided) and their β

Task 1 sanity check

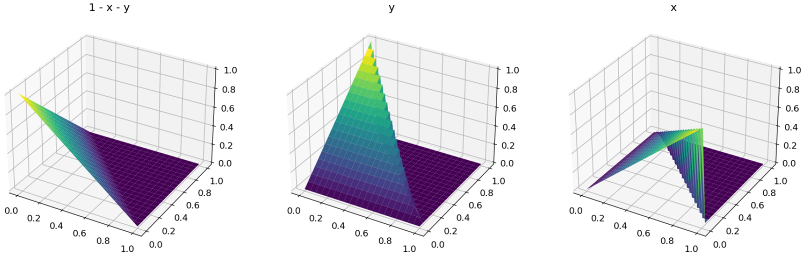
```
In [9]: diff2x.todense()
Out[9]:
matrix([[-1.,  1.,  0.,  0.],
        [ 1., -2.,  1.,  0.],
        [ 0.,  1., -2.,  1.],
        [ 0.,  0.,  1., -1.]])

In [10]: wavenumop.todense()
Out[10]:
matrix([[13.15947253,  6.57973627,  0.        ,  0.        ],
        [ 6.57973627, 26.31894507,  6.57973627,  0.        ],
        [ 0.        ,  6.57973627, 26.31894507,  6.57973627],
        [ 0.        ,  0.        ,  6.57973627, 13.15947253]])

In [11]: mass.todense()
Out[11]:
matrix([[0.33333333, 0.16666667, 0.        , 0.        ],
        [0.16666667, 0.66666667, 0.16666667, 0.        ],
        [0.        , 0.16666667, 0.66666667, 0.16666667],
        [0.        , 0.        , 0.16666667, 0.33333333]])
```

2D basis functions

Example: triangle elements, linear basis functions
Prototype element: a triangle with the vertices (0,0), (1,0), (0,1).



$$\int_0^1 \int_0^{1-x} (1-x-y)(1-x-y)dydx = \frac{1}{12}$$

$$\int_0^1 \int_0^{1-x} (1-x-y)xdydx = \frac{1}{24}$$

$\ldots$

All required 2D integrals

Points of the triangle in `scipy.spatial.Delaunay`:
point 0: (1,0), point 1: (0,1), point 2: (0,0).

Indexing of basis functions that corresponds to this indexing:

$$\phi_0(x,y) = x, \;\; \phi_1(x,y) = y, \;\; \phi_2(x,y) = 1 - x - y$$

Integrals over the prototype element

$$\iint \phi_0\phi_0 dydx = \iint \phi_1\phi_1 dydx = \iint \phi_2\phi_2 dydx = \frac{1}{12}$$
$$\iint \phi_0\phi_1 dydx = \iint \phi_0\phi_2 dydx = \iint \phi_1\phi_2 dydx = \frac{1}{24}$$
$$\iint dydx = \frac{1}{2}$$

Transformation to/from prototype element in 2D

Mapping the triangle element E defined by points $(x, y) = (x_0, y_0)$, (x_1, y_1), (x_2, y_2) to $(x', y') = (0, 0)$, $(1, 0)$, $(0, 1)$ can be done with an affine transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{A}(\begin{bmatrix} x \\ y \end{bmatrix} - \mathbf{b})$$

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \mathbf{b} = \begin{bmatrix} x_0 & y_0 \end{bmatrix}, \mathbf{A}^{-1} = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}$$

Basis function on the arbitrary element defined from the prototype basis function

$$\phi_E(x, y) = \phi(ax + by, cx + dy) = \phi(x', y')$$

Gradients of the basis functions of an arbitrary element

$$\nabla \phi_E(x, y) = \mathbf{A}^T \nabla' \phi(x', y')$$

where $\nabla = (\partial/\partial x, \partial/\partial y)$, $\nabla' = (\partial/\partial x', \partial/\partial y')$.

Transformation of integrals in 2D

All integrals pick up a prefactor from the change of *dxdy*:

$$\iint_E \phi_{E,m}(x,y)\phi_{E,n}(x,y)dxdy = (\det \mathbf{A}^{-1}) \iint_{\text{proto}} \phi_m(x',y')\phi_n(x',y')dx'dy'$$

With linear elements ($\nabla\phi_E(x,y)$ constant) the gradient integrals are simple:

$$\begin{aligned}\iint_E \nabla\phi_{E,m}(x,y)\cdot\nabla\phi_{E,n}(x,y)dxdy &= \nabla\phi_{E,m}\cdot\nabla\phi_{E,n}(\det \mathbf{A}^{-1}) \iint_{\text{proto}} dx'dy' \\ &= (\mathbf{A}^T\nabla\phi_m)\cdot(\mathbf{A}^T\nabla\phi_n)(\det \mathbf{A}^{-1}) \iint_{\text{proto}} dx'dy'\end{aligned}$$

Practical implementation of 2D FEM

N points, N_e elements
Mesh is defined by:

- `points` array (Nx2): each row has the x and y coordinates of a point
- simplices array (N_ex3): each row has 3 indices to the `points` array, defining the 3 point of the triangle element

The transformations **A** can be easily calculated from these.

For this exercise, code is provided for meshing and visualization, based on `scipy.spatial.Delaunay` and `matplotlib.tri.Triangulation`; the transformations **A** are automatically determined.

Practical implementation of 2D FEM

Build the system and mass matrices element by element:

- Get this element's indices of the three points (i_1, i_2 and i_3) and the transform $\mathbf{A}$.
- Calculate gradients of the element's basis functions: $\nabla\phi_E(x, y) = \mathbf{A}^T\nabla\phi(x, y)$
- For the pair of points (i_1, i_2) in the triangle (and associated basis functions):
  - Add to Laplace operator $L_{i_1,i_2} -= 1/2(\det \mathbf{A}^{-1})(\nabla\phi_{E,1} \cdot \nabla\phi_{E,2})$
  - Add to wave number operator $W_{i_1,i_2} += k_0^2\epsilon_E(\det \mathbf{A}^{-1})(\int dx'dy'\phi_1\phi_2)$
  - Add to mass matrix $M_{i_1,i_2} += (\det \mathbf{A}^{-1})(\int dx'dy'\phi_1\phi_2)$
- Do the same for every other pair of basis functions (9 pairs in total).

Task 2*: 2D FEM

Implement FEM with linear basis functions for 2D scalar wave equation. Test case: silicon nanowire in air (just for testing; the scalar approximation is actually pretty bad here)

- $\lambda_0 = 1\,\mu\text{m}$, core radius $0.3\,\mu\text{m}$
- $\epsilon = 12$ in the nanowire, $\epsilon = 1$ elsewhere
- Circular domain and mesh (you can use attached code for meshing and plotting)
- "Natural boundary condition" $\partial u/\partial \mathbf{n} = 0$ (no special implementation necessary)
- Plot the first five modes.