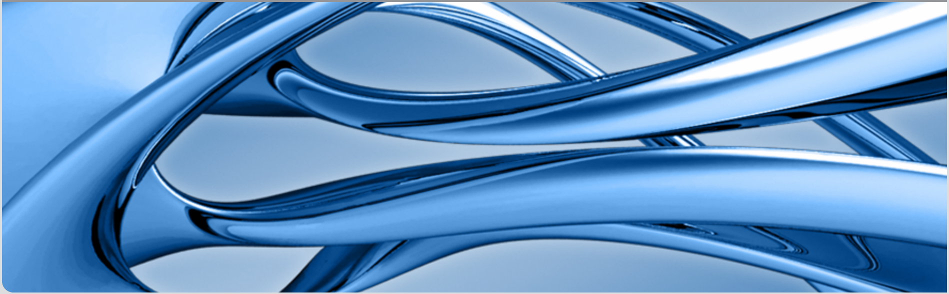
Karlsruhe Institute of Technology

First step towards
Parallel and Adaptive Computation of Maxwell's Equations

Stefan Findeisen

Supported by the DFG Research Training Group 1294 "Analysis, Simulation and Design of Nanotechnological Processes" at the KIT

Institut für Angewandte und Numerische Mathematik



The electro-magnetic Wave Problem

Karlsruhe Institute of Technology

For permeability μ and permittivity ε find an electric field $\mathbf{E}$ and a magnetic field $\mathbf{H}$ such that the linear Maxwell system

$$
\begin{aligned}
\mu\partial_t\mathbf{H} + \nabla\times\mathbf{E} &= \mathbf{0}, & \varepsilon\partial_t\mathbf{E} - \nabla\times\mathbf{H} &= \mathbf{0},\\
\nabla\cdot(\mu\mathbf{H}) &= 0, & \nabla\cdot(\varepsilon\mathbf{E}) &= 0
\end{aligned}
$$

holds for all $t \in [0, T]$. For a given initial condition $\mathbf{u}_0$ this can be written as

$$
M\partial_t\mathbf{u}(t) + A\mathbf{u}(t) = \mathbf{0} \qquad t \in [0, T]\,, \qquad \mathbf{u}(0) = \mathbf{u}_0\,,
$$

where M, A, $\mathbf{u}$ are given by

$$
M := \begin{bmatrix} \mu & 0 \\ 0 & \varepsilon \end{bmatrix}, \quad A := \begin{bmatrix} 0 & \nabla\times \\ -\nabla\times & 0 \end{bmatrix}, \quad \mathbf{u} := \begin{bmatrix} \mathbf{H} \\ \mathbf{E} \end{bmatrix}.
$$

2D reduction of Maxwell's equations

Karlsruhe Institute of Technology

Consider the 2D reduction

$$\mathbf{u} = (\mathbf{H}_1, \mathbf{H}_2, \mathbf{E}_3)\,, \quad \mathbf{H}_3 \equiv \mathbf{E}_1 \equiv \mathbf{E}_2 \equiv 0$$

for Maxwell's equations in $\Omega \subset \mathbb{R}^2$.

Decomposition for the p-refinement of the space-time domain $Q := \Omega \times (0, T)$:

- Decompose $(0, T)$ such that $[0, T] = \bigcup_{n=0}^{N-1} \overline{I_n}$, where I_n is an open interval $I_n = (t_{n-1}, t_n) \subset (0, T)$ for $n = 1, \dots, N$
- Decompose Ω such that $\overline{\Omega} = \bigcup_K \overline{K}$, where K is an open element (e.g. triangle)
- Let $h = \max \operatorname{diam}(K)$ be the size of K
- Let $\mathcal{F}_K$ be the set of faces of K

Discontinuous Galerkin Method

There exists symmetric matrices $B_1, B_2 \in \mathbb{R}^{3\times 3}$ so that the linear flux $\mathbf{F}$ is

$$A\mathbf{u} = \nabla\cdot\mathbf{F}(\mathbf{u}) = B_1\partial_{x_1}\mathbf{u} + B_2\partial_{x_2}\mathbf{u}\,.$$

Multiplying $A\mathbf{u}$ with a test function $\mathbf{v}_K$ and integrating in K yields

$$\begin{aligned}(A\mathbf{u}, \mathbf{v}_K)_{0,K} &= \int_K \nabla\cdot\mathbf{F}(\mathbf{u})\cdot\mathbf{v}_K\,\mathrm{d}x \\ &= -\int_K \mathbf{F}(\mathbf{u})\cdot\nabla\mathbf{v}_K\,\mathrm{d}x + \sum_{f\in\mathcal{F}_K}\int_f \mathbf{n}_{K,f}\cdot\mathbf{F}(\mathbf{u})\cdot\mathbf{v}_K\,\mathrm{d}a\,.\end{aligned}$$

Choose p_K, set $V_{K,h} = \mathbb{P}_{p_K}(K)^3$ and define $V_h = \left\{\mathbf{v}_h \in \mathrm{L}_2(\Omega,\mathbb{R}^3)\colon \mathbf{v}_h\big|_K \in V_{K,h}\right\}$.
Depending on a numerical flux $\mathbf{n}_K\cdot\mathbf{F}^*(\mathbf{u}_h)$ on $f\in\mathcal{F}_K$ we define $A_h\mathbf{u}_h \in V_h$ by

$$(A_h\mathbf{u}_h, \mathbf{v}_K)_{0,K} = -\int_K \mathbf{F}(\mathbf{u}_h)\cdot\nabla\mathbf{v}_K\,\mathrm{d}x + \sum_{f\in\mathcal{F}_K}\int_f \mathbf{n}_{K,f}\cdot\mathbf{F}^*(\mathbf{u}_h)\cdot\mathbf{v}_K\,\mathrm{d}a$$

for all $\mathbf{u}_h \in V_h$ and $\mathbf{v}_K \in V_{K,h}$ and all K.

Discontinuous Galerkin Method

There exists symmetric matrices $B_1, B_2 \in \mathbb{R}^{3\times 3}$ so that the linear flux $\mathbf{F}$ is

$$A\mathbf{u} = \nabla\cdot\mathbf{F}(\mathbf{u}) = B_1\partial_{x_1}\mathbf{u} + B_2\partial_{x_2}\mathbf{u}\,.$$

Multiplying $A\mathbf{u}$ with a test function $\mathbf{v}_K$ and integrating in K yields

$$\begin{aligned}(A\mathbf{u}, \mathbf{v}_K)_{0,K} &= \int_K \nabla\cdot\mathbf{F}(\mathbf{u})\cdot\mathbf{v}_K\,\mathrm{d}x \\ &= -\int_K \mathbf{F}(\mathbf{u})\cdot\nabla\mathbf{v}_K\,\mathrm{d}x + \sum_{f\in\mathcal{F}_K}\int_f \mathbf{n}_{K,f}\cdot\mathbf{F}(\mathbf{u})\cdot\mathbf{v}_K\,\mathrm{d}a\,.\end{aligned}$$

Choose p_k, set $V_{K,h} = \mathbb{P}_{p_K}(K)^3$ and define $V_h = \left\{\mathbf{v}_h \in \mathrm{L}_2(\Omega,\mathbb{R}^3)\colon \mathbf{v}_h\big|_K \in V_{K,h}\right\}$.
Depending on a numerical flux $\mathbf{n}_K\cdot\mathbf{F}^*(\mathbf{u}_h)$ on $f\in\mathcal{F}_K$ we define $A_h\mathbf{u}_h \in V_h$ by

$$(A_h\mathbf{u}_h, \mathbf{v}_K)_{0,K} = -\int_K \mathbf{F}(\mathbf{u}_h)\cdot\nabla\mathbf{v}_K\,\mathrm{d}x + \sum_{f\in\mathcal{F}_K}\int_f \mathbf{n}_{K,f}\cdot\mathbf{F}^*(\mathbf{u}_h)\cdot\mathbf{v}_K\,\mathrm{d}a$$

for all $\mathbf{u}_h \in V_h$ and $\mathbf{v}_K \in V_{K,h}$ and all K.

dG Method for Electro-magnetic Waves

For $f \in \mathcal{F}_K$ let K_f be the neighboring tetrahedron with $f = \partial K \cap \partial K_f$.
The outer unit normal on f is denoted by $\mathbf{n}_{K,f}$, and we set $[\mathbf{v}]_{K,f} := \mathbf{v}_{K_f} - \mathbf{v}_K$.

By using the upwind flux, we get the operator

$$\begin{aligned}\big(A_h(\mathbf{H}_h, \mathbf{E}_h), (\phi_{K,h}, \psi_{K,h})\big)_{0,K} &= (\nabla \times \mathbf{E}_{K,h}, \phi_{K,h})_{0,K} - \big(\nabla \times \mathbf{H}_{K,h}, \psi_{K,h}\big)_{0,K} \\ &+ \sum_{f \in \mathcal{F}_K} \Big(\frac{c_{K_f}\varepsilon_{K_f}}{c_K \varepsilon_K + c_{K_f}\varepsilon_{K_f}} \big(\mathbf{n}_{K,f} \times [\mathbf{E}_h]_{K,f}, \phi_{K,h}\big)_{0,f} \\ &\quad - \frac{c_{K_f}\mu_{K_f}}{c_K \mu_K + c_{K_f}\mu_{K_f}} \big(\mathbf{n}_{K,f} \times [\mathbf{H}_h]_{K,f}, \psi_{K,h}\big)_{0,f} \\ &\quad + \frac{1}{c_K \mu_K + c_{K_f}\mu_{K_f}} \big(\mathbf{n}_{K,f} \times (\mathbf{n}_{K,f} \times [\mathbf{E}_h]_{K,f}), \psi_{K,h}\big)_{0,f} \\ &\quad + \frac{1}{c_K \varepsilon_K + c_{K_f}\varepsilon_{K_f}} \big(\mathbf{n}_{K,f} \times (\mathbf{n}_{K,f} \times [\mathbf{H}_h]_{K,f}), \phi_{K,h}\big)_{0,f} \Big)\end{aligned}$$

for $(\mathbf{E}_h, \mathbf{H}_h) \in V_h$ and $(\psi_{K,h}, \phi_{K,h}) \in V_{K,h}$.

(for details see Hesthaven and Warburton 2002)

Time Integration

Consider the semi-discrete problem

$$M_h \partial_t \mathbf{u}_h(t) + A_h \mathbf{u}_h(t) = \mathbf{0} \quad \text{for } t \in [0, T] \quad \text{subject to } \mathbf{u}_h(0) = \mathbf{u}_{h,0}.$$

To solve this problem, we use the implicit midpoint rule

$$\mathbf{u}_h^n = \mathbf{u}_h^{n-1} + (t_n - t_{n-1}) \left(M_h + \frac{t_n - t_{n-1}}{2} A_h \right)^{-1} A_h \mathbf{u}_h^{n-1} \qquad \text{for } n = 1, \dots, N$$

with step size $t_n - t_{n-1}$ and initial condition $\mathbf{u}_h^0 = \mathbf{u}_{h,0}$.

Properties of the implicit midpoint rule:

- No CFL (Courant-Friedrichs-Lewy) condition $\Rightarrow$ allows larger time steps $t_n - t_{n-1}$
- Convergence order 2 in time
- The upwind flux guarantees that $(M_h + \frac{t_n - t_{n-1}}{2} A_h)$ is positive definite
- Costs: solve $(M_h + \frac{t_n - t_{n-1}}{2} A_h)\widetilde{\mathbf{u}}_h^n = A_h \mathbf{u}_h^{n-1}$ in each step

Hence the solution $\mathbf{u}_h$ is computed sequentially on the slices $S_n := \Omega \times I_n \subset Q$.

Time Integration

Consider the semi-discrete problem

$$M_h \partial_t \mathbf{u}_h(t) + A_h \mathbf{u}_h(t) = \mathbf{0} \quad \text{for } t \in [0, T] \quad \text{subject to } \mathbf{u}_h(0) = \mathbf{u}_{h,0}.$$

To solve this problem, we use the implicit midpoint rule

$$\mathbf{u}_h^n = \mathbf{u}_h^{n-1} + (t_n - t_{n-1}) \left(M_h + \frac{t_n - t_{n-1}}{2} A_h \right)^{-1} A_h \mathbf{u}_h^{n-1} \qquad \text{for } n = 1, \dots, N$$

with step size $t_n - t_{n-1}$ and initial condition $\mathbf{u}_h^0 = \mathbf{u}_{h,0}$.

Properties of the implicit midpoint rule:

- No CFL (Courant-Friedrichs-Lewy) condition $\Rightarrow$ allows larger time steps $t_n - t_{n-1}$
- Convergence order 2 in time
- The upwind flux guarantees that $(M_h + \frac{t_n - t_{n-1}}{2} A_h)$ is positive definite
- Costs: solve $(M_h + \frac{t_n - t_{n-1}}{2} A_h)\widetilde{\mathbf{u}}_h^n = A_h \mathbf{u}_h^{n-1}$ in each step

Hence the solution $\mathbf{u}_h$ is computed sequentially on the slices $S_n := \Omega \times I_n \subset Q$.

Space-Time Cells

Define space-time cells

$$\tau = K_\tau \times I_\tau$$

which consist of a spatial element K_τ and a local time interval $I_\tau = (t_\tau^{\text{min}}, t_\tau^{\text{max}})$ and decompose Q into a finite number of open space-time elements $\tau \subset Q$ such that

$$\overline{Q} = \bigcup_{\tau \in \mathcal{T}} \overline{\tau}.$$

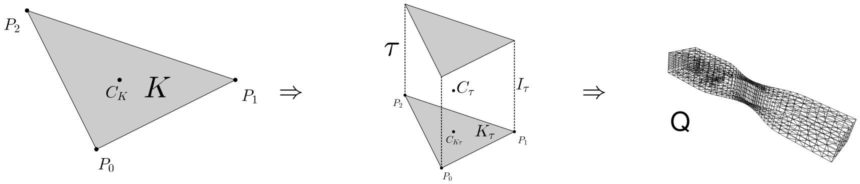


Figure: spatial element, space-time element, space-time mesh

Space-time dG approximation

For every τ choose polynomial degrees p_τ for the ansatz in space, and define the local test spaces $H_{\tau,h} = \mathbb{P}_{p_\tau}(K_\tau)^3$ and the test space

$$H_h = \left\{ \mathbf{v}_h \in \mathrm{L}_2(Q)^3 \colon \mathbf{v}_h|_\tau \in H_{\tau,h} \right\}.$$

For the ansatz space, we define the affine space depending on the initial condition

$$U_h = \Big\{ \mathbf{u}_h \in \mathrm{H}^1(0,T;\mathrm{L}_2(\Omega)^3) \colon \mathbf{u}_h(0) = \mathbf{u}_0 \text{ and for all } \tau \in \mathcal{T} \text{ and } (\mathbf{x},t) \in \tau$$

$$\mathbf{u}_h(\mathbf{x},t) = \frac{t_\tau^{\max} - t}{t_\tau^{\max} - t_\tau^{\min}} \mathbf{w}_{\tau,h}(\mathbf{x}, t_\tau^{\min}) + \frac{t - t_\tau^{\min}}{t_\tau^{\max} - t_\tau^{\min}} \mathbf{v}_{\tau,h}(\mathbf{x}),$$

$$\text{where } \mathbf{w}_{\tau,h} \in U_h|_{[0,t_\tau^{\min}]} \text{ and } \mathbf{v}_{\tau,h} \in H_{\tau,h} \Big\}$$

Let A_h be the discontinuous Galerkin operator with upwind flux approximating A.

Lemma

Let $L_h = M_h \partial_t + A_h$ and $\mathbf{f} \in \mathrm{L}_2(Q)^3$. The discrete solution $\mathbf{u}_h \in U_h$ of the implicit midpoint rule is characterized by the variational equation

$$(L_h \mathbf{u}_h, \mathbf{v}_h)_Q = (\mathbf{f}, \mathbf{v}_h)_Q, \qquad \mathbf{v}_h \in H_h.$$

Space-time dG approximation

For every τ choose polynomial degrees p_τ for the ansatz in space, and define the local test spaces $H_{\tau,h} = \mathbb{P}_{p_\tau}(K_\tau)^3$ and the test space

$$H_h = \left\{ \mathbf{v}_h \in \mathrm{L}_2(Q)^3 \colon \mathbf{v}_h|_\tau \in H_{\tau,h} \right\}.$$

For the ansatz space, we define the affine space depending on the initial condition

$$U_h = \Big\{ \mathbf{u}_h \in \mathrm{H}^1(0,T;\mathrm{L}_2(\Omega)^3) \colon \mathbf{u}_h(0) = \mathbf{u}_0 \text{ and for all } \tau \in \mathcal{T} \text{ and } (\mathbf{x},t) \in \tau$$

$$\mathbf{u}_h(\mathbf{x},t) = \frac{t_\tau^{\max} - t}{t_\tau^{\max} - t_\tau^{\min}} \mathbf{w}_{\tau,h}(\mathbf{x}, t_\tau^{\min}) + \frac{t - t_\tau^{\min}}{t_\tau^{\max} - t_\tau^{\min}} \mathbf{v}_{\tau,h}(\mathbf{x}),$$

$$\text{where } \mathbf{w}_{\tau,h} \in U_h|_{[0,t_\tau^{\min}]} \text{ and } \mathbf{v}_{\tau,h} \in H_{\tau,h} \Big\}$$

Let A_h be the discontinuous Galerkin operator with upwind flux approximating A.

Lemma

Let $L_h = M_h \partial_t + A_h$ and $\mathbf{f} \in \mathrm{L}_2(Q)^3$. The discrete solution $\mathbf{u}_h \in U_h$ of the implicit midpoint rule is characterized by the variational equation

$$(L_h \mathbf{u}_h, \mathbf{v}_h)_Q = (\mathbf{f}, \mathbf{v}_h)_Q, \qquad \mathbf{v}_h \in H_h.$$

Data Structure

Our implementation is done in `C++` and the data structure is organized as follows:

- (Time-)cells, faces, edges are identified by their geometric midpoints
- Hash maps containers are used for the data
- Geometric midpoints are used as hash keys

```
class Cell : public vector<Point> {...};
class Cells : public hash_map<Point,Cell*,Hash> {...};
```

```
class Interval {
 const double t_min;
 const double t_max;
  ...};
```

```
class TCell {
 const Cell* C;
 const Interval* I;
  ...};
```

```
class TCells : public hash_map<Point,TCell*,Hash> {
 tcell tcells () const { return tcell(begin());}
 tcell tcells_end () const { return tcell(end()); }
 tcell find_cell (const Point& z) const { return tcell(find(z)); }
  ...};
```

The use of hash maps allows us to distribute cells among the different processes and solve the problem in parallel. Every cell is stored on only one master process and communicates its data to other processes, if needed.

Data Structure

Our implementation is done in `C++` and the data structure is organized as follows:

- (Time-)cells, faces, edges are identified by their geometric midpoints
- Hash maps containers are used for the data
- Geometric midpoints are used as hash keys

```
class Cell : public vector<Point> {...};
class Cells : public hash_map<Point,Cell*,Hash> {...};
```

```
class Interval {
 const double t_min;
 const double t_max;
  ...};
```

```
class TCell {
 const Cell* C;
 const Interval* I;
  ...};
```

```
class TCells : public hash_map<Point,TCell*,Hash> {
 tcell tcells () const { return tcell(begin());}
 tcell tcells_end () const { return tcell(end()); }
 tcell find_cell (const Point& z) const { return tcell(find(z)); }
  ...};
```

The use of hash maps allows us to distribute cells among the different processes and solve the problem in parallel. Every cell is stored on only one master process and communicates its data to other processes, if needed.

Data Structure

Our implementation is done in `C++` and the data structure is organized as follows:

- (Time-)cells, faces, edges are identified by their geometric midpoints
- Hash maps containers are used for the data
- Geometric midpoints are used as hash keys

```
class Cell : public vector<Point> {...};
class Cells : public hash_map<Point,Cell*,Hash> {...};

class Interval {
 const double t_min;
 const double t_max;
  ...};

class TCell {
 const Cell* C;
 const Interval* I;
  ...};

class TCells : public hash_map<Point,TCell*,Hash> {
 tcell tcells () const { return tcell(begin());}
 tcell tcells_end () const { return tcell(end()); }
 tcell find_cell (const Point& z) const { return tcell(find(z)); }
  ...};
```

The use of hash maps allows us to distribute cells among the different processes and solve the problem in parallel. Every cell is stored on only one master process and communicates its data to other processes, if needed.

Data Structure

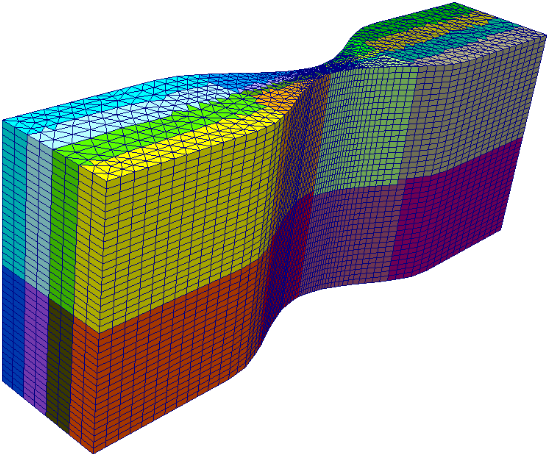


Figure: Distribution of the space-time cells on 32 processes

Heuristic Indicator and p-Refinement

p-adaptive refinement in space:

- Compute $\mathbf{u}_h$ with lowest polynomial degree $p = 0$ on all slices S_n
- Refinement indicator η_τ of a cell τ:

$$\eta_\tau^2 := \sum_{f \in \mathcal{F}_\tau} \eta_f^2 \quad \text{where } \eta_f^2 := h_f \left\| (\mathbf{F}^*(\mathbf{u}_h) - \mathbf{F}(\mathbf{u}_h)) \cdot \mathbf{n}_f \right\|_{\mathrm{L}_2(f)}^2 ,$$

  with flux and numerical flux $\mathbf{F}$ and $\mathbf{F}^*$, and area h_f, and the outer normal vector $\mathbf{n}_f$ of the face f.
- Increase the polynomial degree p on the space-time cell $\tau \subset S_n$, if

$$\eta_\tau > (1 - \theta) \max_{\tau \in S_n} \eta_\tau$$

  holds for η_τ and a given parameter $\theta \in [0, 1]$, e.g. $\theta = 0.99$.

Heuristic Indicator and space-time Refinement

Karlsruhe Institute of Technology

Space-time refinement:

- Same heuristic as for *p*-refinement
- If $\tau = K_\tau \times I_\tau$ should be refined in time, then
  1. Split $I_\tau = (t_\tau^{\min}, t_\tau^{\max})$ into I_τ^1 and I_τ^2, s.t. $\overline{I_\tau} = \overline{I_\tau^1} \cup \overline{I_\tau^2}$, where $I_\tau^1 := (t_\tau^{\min}, t_\tau^{1/2})$, $I_\tau^2 := (t_\tau^{1/2}, t_\tau^{\max})$ and $t_\tau^{1/2} := 0.5(t_\tau^{\min} + t_\tau^{\max})$
  2. Replace τ by two new space-time cells $\tau^1 = K_\tau \times I_\tau^1$ and $\tau^2 = K_\tau \times I_\tau^2$
- Perform the implicit midpoint rule twice on time refined space-time cells

Performance of the heuristic approach:

- **High** polynomial degrees and **small** time steps are used in areas where a single wavefront is located
- **Lowest** polynomial degrees and **larger** time steps are used in areas with absence of a wave

Heuristic Indicator and space-time Refinement

Space-time refinement:

- Same heuristic as for p-refinement
- If $\tau = K_\tau \times I_\tau$ should be refined in time, then
  1. Split $I_\tau = (t_\tau^{\min}, t_\tau^{\max})$ into I_τ^1 and I_τ^2, s.t. $\overline{I_\tau} = \overline{I_\tau^1} \cup \overline{I_\tau^2}$,
     where $I_\tau^1 := (t_\tau^{\min}, t_\tau^{1/2})$, $I_\tau^2 := (t_\tau^{1/2}, t_\tau^{\max})$ and $t_\tau^{1/2} := 0.5(t_\tau^{\min} + t_\tau^{\max})$
  2. Replace τ by two new space-time cells $\tau^1 = K_\tau \times I_\tau^1$ and $\tau^2 = K_\tau \times I_\tau^2$
- Perform the implicit midpoint rule twice on time refined space-time cells

Performance of the heuristic approach:

- **High** polynomial degrees and **small** time steps are used in areas where a single wavefront is located
- **Lowest** polynomial degrees and **larger** time steps are used in areas with absence of a wave

Numerical Experiment

We consider:

- Unstructured triangular mesh in a locally tapered domain $\Omega \subset (0, 10) \times (-1, 1)$ with reflecting boundary conditions

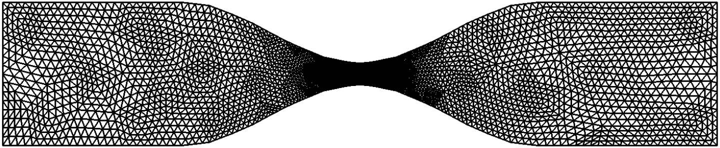


Figure: Unstructured triangular mesh in tapered domain Ω

- Constant parameters $\mu = \varepsilon = 1$
- Initial condition $\mathbf{u}_0(x) = \begin{cases} (0, 0, \cos(4\pi x_1 - 3\pi) + 1)^T, & \text{for } 1 \leq x_1 \leq 1.5, \\ \mathbf{0}, & \text{else.} \end{cases}$
- Final time $T = 8$ with constant initial step size $t_n - t_{n-1} = 0.1$

Numerical Experiment

$t = 0:$

$t = 2:$

$t = 4:$

$t = 6:$

$t = 8:$

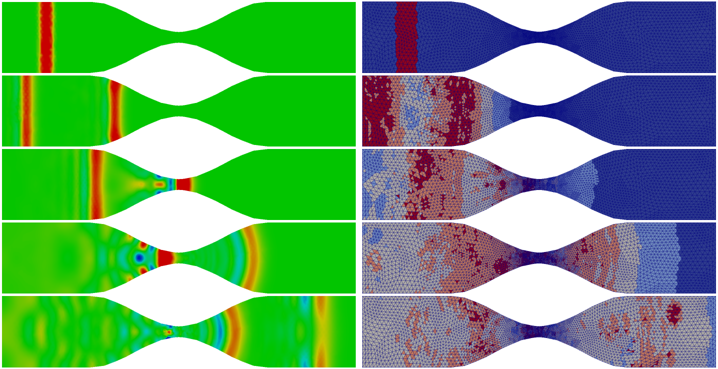


Figure: Initial distribution of $\mathbf{E}_3$ and polynomial degrees and time evolution

Numerical Experiment

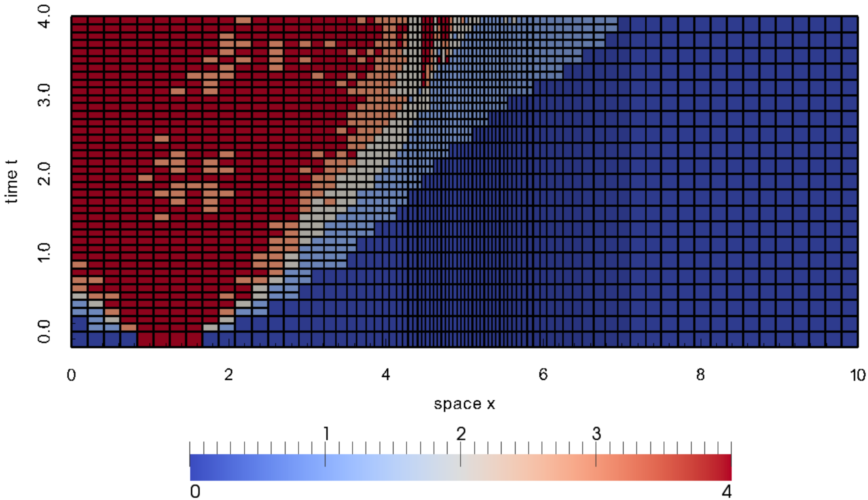


Figure: Polynomial degrees and time refinement in the time interval [0, 4]

Numerical Experiment

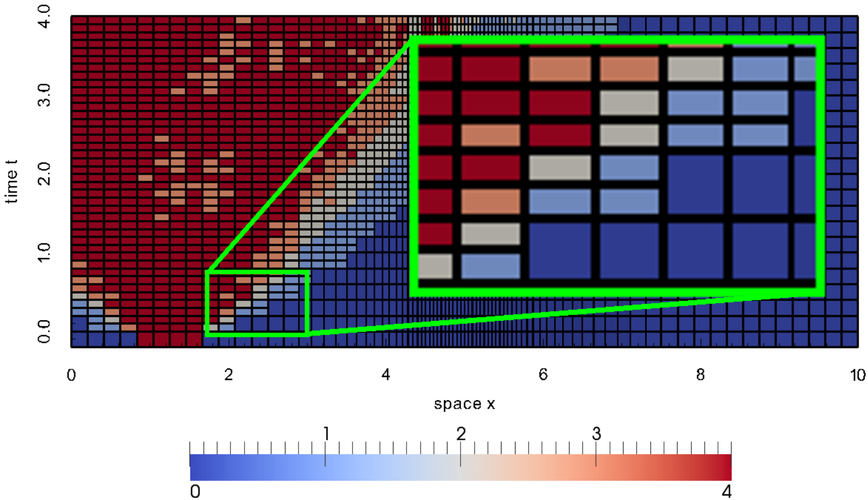


Figure: Polynomial degrees and time refinement in the time interval [0, 4]

Numerical Experiment

Adaptive polynomial degrees:

	p-adaptive		uniform	
p	degrees of freedom	$\Vert \mathbf{u}_{p+1}(T) - \mathbf{u}_p(T) \Vert_V$	degrees of freedom	$\Vert \mathbf{u}_{p+1}(T) - \mathbf{u}_p(T) \Vert_V$
0	1,469,664	0.835850	1,469,664	0.853796
1	3,303,156	0.269539	4,408,992	0.265763
2	5,424,354	0.086987	8,817,984	0.020959
3	8,100,078	0.022396	14,656,640	0.002958
4	11,494,896		22,044,960	

Table: Adaptive polynomial degrees and corresponding degrees of freedom

Numerical Experiment

Space-time adaptive (max. one refinement in time):

	p-time-adaptive		uniform	
p	degrees of freedom	$\|\mathbf{u}_{p+1}(T) - \mathbf{u}_p(T)\|_V$	degrees of freedom	$\|\mathbf{u}_{p+1}(T) - \mathbf{u}_p(T)\|_V$
0	1,469,664	0.840901	2,939,328	0.845272
1	6,051,378	0.309939	8,817,984	0.289086
2	10,040,952	0.083207	17,635,968	0.023039
3	13,678,458	0.057660	29,393,280	0.003572
4	16,445,328		44,089,920	

Table: Space-time adaptive refinement and corresponding degrees of freedom

Outlook

- Computation of the full 3D Maxwell problem
- Additional refinements in time
- *h*-adaptivity in space
- Better error indicators and estimators