

Parallel linear computational cost isogeometric alternating-directions (IGA-ADS) simulator of three-dimensional non-stationary problems.

Maciej Paszyński

AGH University of Science and Technology, Kraków, Poland

Collaborators:

Marcin Łoś, Jacek Leszczyński (AGH),

Luis Emilio García-Castillo (UC3, Madrid)

Julen Alvarez-Aramberri, David Pardo (BCAM, Spain)

Non-stationary Maxwell with Absorbing Boundary Conditions on a Laptop

Computational domain $[0, 600\text{km}] \times [0, 600\text{km}] \times [0, 200\text{km}]$,
Mesh $360 \times 360 \times 120$ (total of 15,552,000 elements)
IGA with quadratic B-splines, 1000 time steps, $dt = 10^{-6}$
Source $[250\text{km}, 350\text{km}] \times [250\text{km}, 350\text{km}] \times [100\text{km}, 120\text{km}]$
Ground $[0, 600\text{km}] \times [0, 600\text{km}] \times [0, 60\text{km}]$.

Non-stationary Maxwell with Absorbing Boundary Conditions on a Laptop

Background and motivation

Goal Efficient finite element simulations

- traditionally, cost of linear solver is dominant
- some problems have additional structure to exploit

Isogeometric Analysis Alternating-Directions Solver (**IGA-ADS**)

- fast $\mathcal{O}(N)$ solver for a class of time-dependent problems
- some restrictions to preserve tensor product structure
 - regular patch of elements
 - requires special time marching schemes (or explicit dynamics)
 - regular material data no longer a problem

Many applications: tumor growth simulations, advection-diffusion-reaction, Navier-Stokes

IGA-ADS can be also employed as preconditioner

Examples for today: cloud formation, pollution removal by artificially generated shock waves, and non-stationary Maxwell equations

Krzysztof Misan, Weronika Ormaniec, Adam Kania, Maciej Kozieja, Marcin Łoś, Dominik Gryboś, Jacek Leszczyński, Maciej Paszyński, [Fast Isogeometric Analysis Simulations of a Process of Air Pollution Removal by Artificially Generated Shock Waves](#), International Conference on Computational Science ICCS 2022 **Lecture Notes in Computer Science** 13352 (2022): 298-311 doi.org/10.1007/978-3-031-08757-8_26

Marcin Łoś, Maciej Woźniak, Maciej Paszyński, Andrew Lenharth, Mohammed Amber Hassaan, Keshav Pingali, [IGA-ADS: Isogeometric Analysis Finite Element Method using Alternating-Directions solver](#), **Computer Physics Communications** 217 (2017):99–116 doi.org/10.1016/j.cpc.2017.02.023

Strong form equation

We employ advection-difusion-reaction equation to model the concentration of the water vapor forming a cloud, mixed with the pollution particles.

Strong form equation

$$\frac{\partial u}{\partial t} = f + (b \cdot \nabla)u + \nabla \cdot (K \nabla u) + cu \text{ in } \Omega \times (0, T] \quad (1)$$

$$\nabla u \cdot n = 0 \text{ in } \partial\Omega \times (0, T] \quad (2)$$

$$u = u_0 \text{ in } \Omega \times 0 \quad (3)$$

where u is the concentration scalar field, b is the assumed air

velocity vector field, $K = \begin{pmatrix} K_{11} & 0 & 0 \\ 0 & K_{22} & 0 \\ 0 & 0 & K_{33} \end{pmatrix}$ is an isotropic diffusion matrix,

c is the reaction parameter, and f is the source term.

Explicit method and weak form formulations

We formulate the explicit method as:

Explicit method formulation

$$\frac{u^{t+1} - u^t}{dt} = f^t + (b \cdot \nabla)u^t + \nabla \cdot (K \nabla u^t) + cu^t \quad (4)$$

we derive the weak formulation, using test functions v , and integrating by parts the diffusion term:

Weak form formulation

$$\begin{aligned} (u^{t+1}, v) = & (u^t, v) + dt (cu^t + f^t, v) - dt (K \nabla u^t, \nabla v) \\ & + dt (b \cdot \nabla u^t, v) \quad \forall v \in V \end{aligned}$$

B-spline discretization

We discretize with B-splines over $\Omega = [0, 1]^3$:

$$u^{t+1} = \sum_{i=1, \dots, N_x; j=1, \dots, N_y; k=1, \dots, N_z} u_{ijk}^{t+1} B_i^x B_j^y B_k^z;$$
$$u^t = \sum_{i=1, \dots, N_x; j=1, \dots, N_y; k=1, \dots, N_z} u_{ijk}^t B_i^x B_j^y B_k^z$$

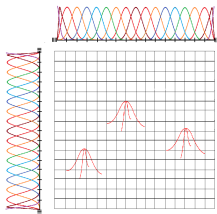


Figure: Two dimensional example of approximation with B-splines

Discrete weak formulation

B-splines for trial and test

$$\begin{aligned} & \sum_{ijk} u_{ijk}^{t+1} (B_i^x B_j^y B_k^z, B_l^x B_m^y B_n^z) = \sum_{ijk} u_{ijk}^t (B_i^x B_j^y B_k^z, B_l^x B_m^y B_n^z) - \\ & dt \left[\sum_{ijk} u_{ijk}^t \left(K \frac{\partial B_i^x}{\partial x} B_j^y B_k^z, \frac{\partial B_l^x}{\partial x} B_m^y B_n^z \right) - \sum_{ijk} u_{ijk}^t \left(K B_i^x \frac{\partial B_j^y}{\partial y} B_k^z, B_l^x \frac{\partial B_m^y}{\partial y} B_n^z \right) - \right. \\ & \left. \sum_{ijk} u_{ijk}^t \left(K B_i^x B_j^y \frac{\partial B_k^z}{\partial z}, B_l^x B_m^y \frac{\partial B_n^z}{\partial z} \right) \right] + dt \sum_{ijk} u_{ijk}^t \left(b \frac{\partial B_i^x}{\partial x} B_j^y B_k^z, B_l^x B_m^y B_n^z \right) + \\ & \sum_{ijk} u_{ijk}^t \left(b B_i^x \frac{\partial B_j^y}{\partial y} B_k^z, B_l^x B_m^y B_n^z \right) + \sum_{ijk} u_{ijk}^t \left(b B_i^x B_j^y \frac{\partial B_k^z}{\partial z}, B_l^x B_m^y B_n^z \right) \\ & \sum_{ijk} u_{ijk}^t (c B_i^x B_j^y B_k^z, B_l^x B_m^y B_n^z) + (f^t, B_l^x B_m^y B_n^z) \end{aligned}$$

$$l = 1, \dots, N_x; m = 1, \dots, N_y; n = 1, \dots, N_z$$

where $(u, v) = \int_{\Omega} u(x, y, z; t) v(x, y, z; t) dx dy dz$ for a fixed t .

Kronecker product

In general, Kronecker product matrix $\mathcal{M} = \mathcal{M}^x \otimes \mathcal{M}^y \otimes \mathcal{M}^z$ over 3D domain $\Omega = \Omega_x \times \Omega_y \times \Omega_z$ is defined as:

Kronecker product

$$\begin{aligned} \mathcal{M}_{ijklmn} &= \int B_i^x B_j^y B_k^z B_l^x B_m^y B_n^y dx dy dz = \\ &= \int B_i^x B_l^x dx \int B_j^y B_m^y dy \int B_k^z B_n^z dz = \mathcal{M}_{il}^x \mathcal{M}_{jm}^y \mathcal{M}_{kn}^z \end{aligned} \quad (5)$$

Due to the fact, that one-dimensional matrices discretized with B-spline functions are banded and they have $2p + 1$ diagonals (where p stands for the order of B-splines), since:

$$(\mathcal{M})^{-1} = (\mathcal{M}^x \otimes \mathcal{M}^y \otimes \mathcal{M}^z)^{-1} = (\mathcal{M}^x)^{-1} \otimes (\mathcal{M}^y)^{-1} \otimes (\mathcal{M}^z)^{-1}$$

we can solve our system in a linear computational cost.

Cloud formation and thermal inversion

The scalar field u represents the water vapor forming a cloud, mixed with the pollution particles.

Strong form equation

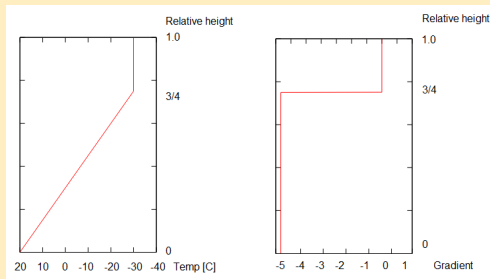
$$\begin{aligned} & \frac{\partial u(x, y, z; t)}{\partial t} + \frac{\partial T(y; t)}{\partial y} \frac{\partial u(x, y, z; t)}{\partial y} \\ & - K_x \frac{\partial^2 u(x, y, z; t)}{\partial x^2} - K_y \frac{\partial^2 u(x, y, z; t)}{\partial y^2} - K_z \frac{\partial^2 u(x, y, z; t)}{\partial z^2} = f(x, y, z; t) \\ & (x, y, z; t) \text{ in } \Omega \times (0, T] \\ & \nabla u(x, y, z; t) \cdot n(x, y, z) = 0, \quad (x, y, z; t) \text{ in } \partial\Omega \times (0, T] \\ & u(x, y, z; 0) = u_0 \text{ in } \Omega \times 0 \end{aligned}$$

where u is the concentration scalar field, where the advection is driven by the temperature gradient in the vertical direction

Temperature gradient and diffusion

Temperature gradient

$$\frac{\partial T(y; t)}{\partial y} = \begin{cases} 0 & \text{for } y > \frac{3}{4} \\ -5 & \text{for } y \leq \frac{3}{4} \end{cases} \quad (6)$$

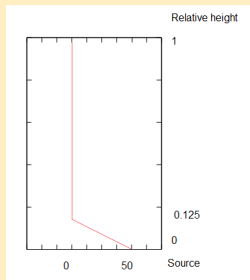


Diffusion

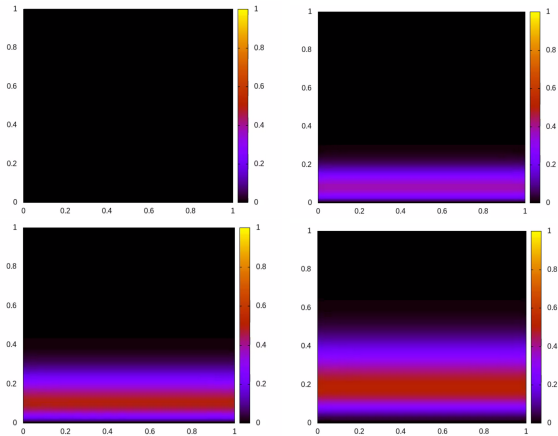
$K_x = K_y = 1.0$ the horizontal diffusion, $K_z = 0.1$ the vertical diffusion

Source term

$$f(x, y; t) = \begin{cases} 50 - 400y & \text{for } y < 0.125; \\ 0 & \text{otherwise} \end{cases} \quad (7)$$



Simulation results



Simulation results

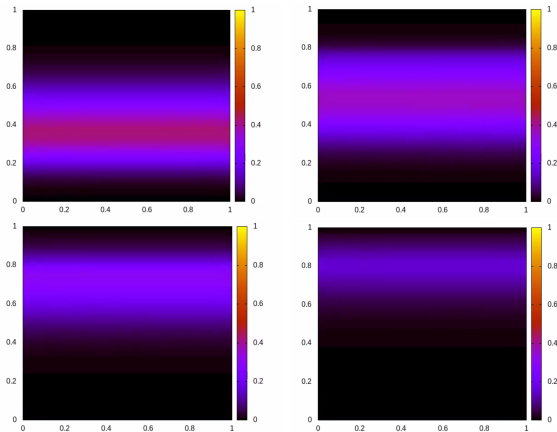


Figure: Formulation of the cloud through thermal inversion

Reducing air pollution by the use of artificial shock wave

- Due to the ground inversion pollutants can be trapped at low altitude causing damage to humans and other living organisms.
- Vertical air movement can break the inversion layer and introduce temporary upward mixing effect which in turn cause decrease in the pollution level at the lower altitudes.
- Shock wave generator [Leszczyński et.al. The method of reducing dust accumulation in the smog layer, which is the inversion layer. European Patent Office EP20217680 (2020)]
The explosions of the acetylene-air mixture reaches a pressure of about 1 MPa. One shock wave every 10 seconds.



Figure: The shock wave generator. The drop of the concentration of PM25

Shock wave generation

```
template <int t_begin, int t_end>
double clock(double t) {
if (t < t_begin || t > t_end) return 0;
t = (t - t_begin) / (t_end - t_begin);
return max(sin(2* $\pi$ *t) * cos(t), 0); }
```

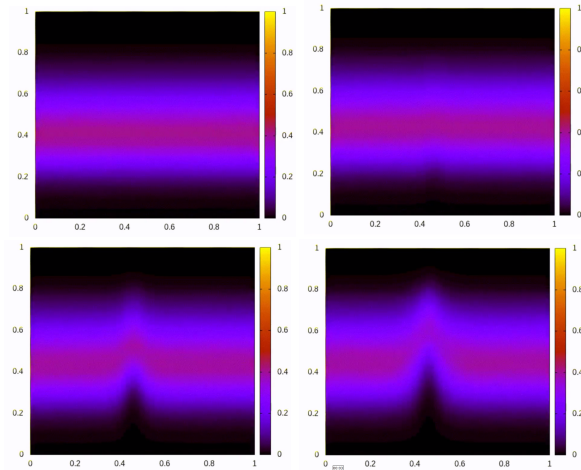
```
template <int t_begin, int t_end>
double cannon(double x, double y, double z, int
iter){
if ((x > 0.2 && x < 0.8) && (y > 0.2 && y < 0.8)) {
double t = clock<t_begin, t_end>(iter);
if (t == 0) return 0;
return 300 * (1 - z) * max(cos(2* $\pi$ *x)* cos(2* $\pi$ *y),
0)*t; } }
```

```
cannon<0, 300>(x, y, z, iter)
cannon<200, 500>(x, y, z, iter)
cannon<400, 700>(x, y, z, iter)
cannon<600, 900>(x, y, z, iter)
```

Simulation results

Two hours of parallel shared-memory solver using GALOIS.
40x40x40 elements, quadratic B-splines.
i7 8700 8th generation, 12 cores (6 hyperthreading) 16 GB RAM.
Time step $dt = 10^{-5}$, 30,000 time steps.

Simulation results



Simulation results

From the simulation results we can read that creating a shock wave resulted in a local mixing of the layers and reduction of the water vapor mixed with the polluted particles.

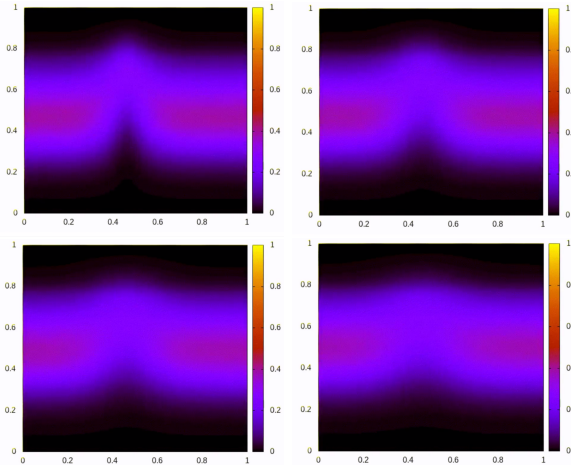


Figure: Pollution reduction by generated shock waves.

Future work: Physics Informed Neural Networks

$$PINN(x, y, z; t) = u(x, y, z; t)$$

$$\frac{\partial PINN(x, y, z; t)}{\partial t} = \frac{\partial u(x, y, z; t)}{\partial t}$$

$$\frac{\partial PINN(x, y, z; t)}{\partial y} = \frac{\partial u(x, y, z; t)}{\partial y}$$

$$\frac{\partial^2 PINN(x, y, z; t)}{\partial w^2} = \frac{\partial^2 u(x, y, z; t)}{\partial w^2}, w \in \{x, y, z\}$$

$$LOSS_{PDE}(x, y, z; t) = \frac{\partial PINN(x, y, z; t)}{\partial t} +$$

$$\frac{\partial T(y; t)}{\partial t} \frac{\partial PINN(x, y, z; t)}{\partial y} - K_x \frac{\partial^2 PINN(x, y, z; t)}{\partial x^2}$$

$$- K_y \frac{\partial^2 PINN(x, y, z; t)}{\partial y^2} - K_z \frac{\partial^2 PINN(x, y, z; t)}{\partial z^2}$$

$$- f(x, y, z; t) \quad \text{for } (x, y, z) \in \Omega, t \in [0, T]$$

Future work: Physics Informed Neural Networks

$$PINN(x, y, z; t) = u(x, y, z; t)$$

$$\frac{\partial PINN(x, y, z; t)}{\partial t} = \frac{\partial u(x, y, z; t)}{\partial t}$$

$$\frac{\partial PINN(x, y, z; t)}{\partial w} = \frac{\partial u(x, y, z; t)}{\partial w} \quad w \in \{x, y, z\}$$

$$LOSS_{Init}(x, y, z; t) = PINN(x, y, z; 0) - u_0(x, y, z; 0)$$

for $(x, y, z) \in \Omega$,

$$LOSS_{BC}(x, y, z; t) = \nabla PINN(x, y, z; t) \cdot n(x, y, z) \text{ for } (x, y, z) \in \Omega,$$

here $n \in (+/- 1, 0, 0), (0, +/- 1, 0), (0, 0, +/- 1)$ so

$$\nabla PINN(x, y, z; t) \cdot n(x, y, z) = \frac{+/- \partial PINN(x, y, z; t)}{\partial w}, \text{ where } w \in \{x, y, z\}.$$

Future work: Physics Informed Neural Networks

$$I^{(0)} = x, I^{(1)} = \sigma^{(1)} \left(W^{(1)} I^{(0)} + b^{(1)} \right), \dots$$
$$\dots, I^{(n)} = \sigma^{(n)} \left(W^{(n)} I^{(n-1)} + b^{(n)} \right), I^{(out)} = W^{(out)} I^{(n)} + b^{(out)}.$$

Select $x \in \Omega \cup \partial\Omega$, select $t \in (0, T]$, compute $\frac{\partial \text{LOSS}_{PDE}(x;t)}{\partial w_{ij}^{(k)}}$, $\frac{\partial \text{LOSS}_{PDE}(x;t)}{\partial b_i^{(k)}}$

$$w_{ij}^{(k)} = w_{ij}^{(k)} - \eta * \frac{\partial \text{LOSS}_{PDE}(x;t)}{\partial w_{ij}^{(k)}} \quad b_i^{(k)} = b_i^{(k)} - \eta * \frac{\partial \text{LOSS}_{PDE}(x;t)}{\partial b_i^{(k)}}$$

Select $x \in \partial\Omega$, compute $\frac{\partial \text{LOSS}_{BC}(x)}{\partial w_{ij}^{(k)}}$, $\frac{\partial \text{LOSS}_{BC}(x)}{\partial b_i^{(k)}}$

$$w_{ij}^{(k)} = w_{ij}^{(k)} - \eta * \frac{\partial \text{LOSS}_{BC}(x;t)}{\partial w_{ij}^{(k)}} \quad b_i^{(k)} = b_i^{(k)} - \eta * \frac{\partial \text{LOSS}_{BC}(x;t)}{\partial b_i^{(k)}}$$

Select $x \in \Omega$, $t = 0$, compute $\frac{\partial \text{LOSS}_{Init}(x;0)}{\partial w_{ij}^{(k)}}$, $\frac{\partial \text{LOSS}_{Init}(x;0)}{\partial b_i^{(k)}}$

$$w_{ij}^{(k)} = w_{ij}^{(k)} - \eta * \frac{\partial \text{LOSS}_{Init}(x;0)}{\partial w_{ij}^{(k)}} \quad b_i^{(k)} = b_i^{(k)} - \eta * \frac{\partial \text{LOSS}_{Init}(x;0)}{\partial b_i^{(k)}}$$

6 (9/1)

Variational Physics Informed Neural Networks

$$PINN(x, y, z; t) = u(x, y, z; t)$$

$$\frac{\partial PINN(x, y, z; t)}{\partial t} = \frac{\partial u(x, y, z; t)}{\partial t}$$

$$\frac{\partial PINN(x, y, z; t)}{\partial w} = \frac{\partial u(x, y, z; t)}{\partial w} \quad w \in \{x, y, z\}$$

$$\begin{aligned} LOSS_{WEAK} = & \left(\frac{\partial PINN(x, y, z; t)}{\partial t}, v(x, y, z) \right) + \\ & \left(K_x \frac{\partial PINN(x, y, z; t)}{\partial x}, \nabla v(x, y, z) \right) \\ & - \left(K_y \frac{\partial PINN(x, y, z; t)}{\partial y}, \nabla v(x, y, z) \right) \\ & - \left(K_z \frac{\partial PINN(x, y, z; t)}{\partial z}, \nabla v(x, y, z) \right) \\ & - \left(\frac{\partial T(y; t)}{\partial t} \frac{\partial PINN(x, y, z; t)}{\partial y} - f(x, y, z; t), v(x, y, z) \right) \end{aligned}$$

Variational Physics Informed Neural Networks

$$I^{(0)} = x, I^{(1)} = \sigma^{(1)} \left(W^{(1)} I^{(0)} + b^{(1)} \right), \dots$$
$$\dots, I^{(n)} = \sigma^{(n)} \left(W^{(n)} I^{(n-1)} + b^{(n)} \right), I^{(out)} = W^{(out)} I^{(n)} + b^{(out)}.$$

Select v (test function), $t \in [0, T]$, compute $\frac{\partial \text{LOSS}_{WEAK}(v; t)}{\partial w_{ij}^{(k)}}$, $\frac{\partial \text{LOSS}_{WEAK}(v; t)}{\partial b_i^{(k)}}$

$$w_{ij}^{(k)} = w_{ij}^{(k)} - \eta * \frac{\partial \text{LOSS}_{WEAK}(v; t)}{\partial w_{ij}^{(k)}} \quad b_i^{(k)} = b_i^{(k)} - \eta * \frac{\partial \text{LOSS}_{WEAK}(v; t)}{\partial b_i^{(k)}}$$

Select $x \in \Omega$, $t = 0$, compute $\frac{\partial \text{LOSS}_{Init}(x; 0)}{\partial w_{ij}^{(k)}}$, $\frac{\partial \text{LOSS}_{Init}(x; 0)}{\partial b_i^{(k)}}$

$$w_{ij}^{(k)} = w_{ij}^{(k)} - \eta * \frac{\partial \text{LOSS}_{Init}(x; 0)}{\partial w_{ij}^{(k)}} \quad b_i^{(k)} = b_i^{(k)} - \eta * \frac{\partial \text{LOSS}_{Init}(x; 0)}{\partial b_i^{(k)}}$$

for $\eta \in (0, 1)$.

Maxwell equations

Maxwell equations in vacuum:

$$\partial_t \mathbf{E} = \frac{1}{\epsilon_0} \nabla \times \mathbf{H}$$

$$\partial_t \mathbf{H} = -\frac{1}{\mu_0} \nabla \times \mathbf{E}$$

$$\nabla \cdot \mathbf{E} = 0$$

$$\nabla \cdot \mathbf{H} = 0$$

\mathbf{E} – electric field, \mathbf{H} – magnetic field

ϵ_0 – permittivity, μ_0 – permeability

Splitting of rotation

$$\nabla \times = \begin{bmatrix} 0 & -\partial_3 & \partial_2 \\ \partial_3 & 0 & -\partial_1 \\ -\partial_2 & \partial_1 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & \partial_2 \\ \partial_3 & 0 & 0 \\ 0 & \partial_1 & 0 \end{bmatrix}}_{C_1} - \underbrace{\begin{bmatrix} 0 & \partial_3 & 0 \\ 0 & 0 & \partial_1 \\ \partial_2 & 0 & 0 \end{bmatrix}}_{C_2}$$

Important properties

$$C_1 \circ C_2 = \begin{bmatrix} \partial_2^2 & 0 & 0 \\ 0 & \partial_3^2 & 0 \\ 0 & 0 & \partial_1^2 \end{bmatrix} \quad C_2 \circ C_1 = \begin{bmatrix} \partial_3^2 & 0 & 0 \\ 0 & \partial_1^2 & 0 \\ 0 & 0 & \partial_2^2 \end{bmatrix}$$

$$C_1 \circ C_1 = \begin{bmatrix} 0 & \partial_2 \partial_1 & 0 \\ 0 & 0 & \partial_3 \partial_2 \\ \partial_1 \partial_3 & 0 & 0 \end{bmatrix} \quad C_2 \circ C_2 = \begin{bmatrix} 0 & 0 & \partial_3 \partial_1 \\ \partial_1 \partial_2 & 0 & 0 \\ 0 & \partial_2 \partial_3 & 0 \end{bmatrix}$$

Time discretization

$$\partial_t \mathbf{E} = \frac{\mathbf{E}^{n+\frac{1}{2}} - \mathbf{E}^n}{\tau/2} = \frac{1}{\varepsilon} \nabla \times \mathbf{H}; \quad \partial_t \mathbf{H} = \frac{\mathbf{H}^{n+\frac{1}{2}} - \mathbf{H}^n}{\tau/2} = -\frac{1}{\mu} \nabla \times \mathbf{E}$$

Substep 1

$$\mathbf{E}^{n+\frac{1}{2}} = \mathbf{E}^n + \frac{\tau}{2\varepsilon} \underbrace{\left(C_1 \mathbf{H}^{n+\frac{1}{2}} - C_2 \mathbf{H}^n \right)}_{\nabla \times \mathbf{H}}$$

$$\mathbf{H}^{n+\frac{1}{2}} = \mathbf{H}^n - \frac{\tau}{2\mu} \underbrace{\left(C_1 \mathbf{E}^n - C_2 \mathbf{E}^{n+\frac{1}{2}} \right)}_{\nabla \times \mathbf{E}}$$

Substep 2

$$\mathbf{E}^{n+1} = \mathbf{E}^{n+\frac{1}{2}} + \frac{\tau}{2\varepsilon} \underbrace{\left(C_1 \mathbf{H}^{n+\frac{1}{2}} - C_2 \mathbf{H}^{n+1} \right)}_{\nabla \times \mathbf{H}}$$

$$\mathbf{H}^{n+1} = \mathbf{H}^{n+\frac{1}{2}} - \frac{\tau}{2\mu} \underbrace{\left(C_1 \mathbf{E}^{n+1} - C_2 \mathbf{E}^{n+\frac{1}{2}} \right)}_{\nabla \times \mathbf{E}}$$

Time discretization – step 1

$$\mathbf{E}^{n+\frac{1}{2}} = \mathbf{E}^n + \frac{\tau}{2\varepsilon} \left(C_1 \mathbf{H}^{n+\frac{1}{2}} - C_2 \mathbf{H}^n \right)$$

$$\mathbf{H}^{n+\frac{1}{2}} = \mathbf{H}^n - \frac{\tau}{2\mu} \left(C_1 \mathbf{E}^n - C_2 \mathbf{E}^{n+\frac{1}{2}} \right)$$

$$\begin{aligned} \mathbf{E}^{n+\frac{1}{2}} &= \mathbf{E}^n + \frac{\tau}{2\varepsilon} \left(C_1 \left[\underbrace{\mathbf{H}^n - \frac{\tau}{2\mu} \left(C_1 \mathbf{E}^n - C_2 \mathbf{E}^{n+\frac{1}{2}} \right)}_{\mathbf{H}^{n+\frac{1}{2}}} \right] - C_2 \mathbf{H}^n \right) \\ &= \mathbf{E}^n + \frac{\tau}{2\varepsilon} \underbrace{(C_1 \mathbf{H}^n - C_2 \mathbf{H}^n)}_{\nabla \times \mathbf{H}^n} - \frac{\tau^2}{4\varepsilon\mu} C_1^2 \mathbf{E}^n + \frac{\tau^2}{4\varepsilon\mu} C_1 C_2 \mathbf{E}^{n+\frac{1}{2}} \end{aligned}$$

$$\left(1 - \frac{\tau^2}{4\varepsilon\mu} C_1 C_2 \right) \mathbf{E}^{n+\frac{1}{2}} = \mathbf{E}^n + \frac{\tau}{2\varepsilon} \nabla \times \mathbf{H}^n - \underbrace{\frac{\tau^2}{4\varepsilon\mu} C_1^2 \mathbf{E}^n}_{\text{inconvenient}}$$

$$\begin{aligned}\left(1 - \frac{\tau^2}{4\varepsilon\mu} C_1 C_2\right) \mathbf{E}^{n+\frac{1}{2}} &= \mathbf{E}^n + \frac{\tau}{2\varepsilon} \nabla \times \mathbf{H}^n - \frac{\tau^2}{4\varepsilon\mu} C_1^2 \mathbf{E}^n \\ \left(1 - \frac{\tau^2}{4\varepsilon\mu} C_2 C_1\right) \mathbf{E}^{n+1} &= \mathbf{E}^{n+\frac{1}{2}} + \frac{\tau}{2\varepsilon} \nabla \times \mathbf{H}^{n+\frac{1}{2}} - \frac{\tau^2}{4\varepsilon\mu} C_2^2 \mathbf{E}^{n+\frac{1}{2}}\end{aligned}$$

Fully expanded first substep with $\lambda = \frac{\tau^2}{4\varepsilon\mu}$

$$\begin{aligned}(1 - \lambda \partial_2^2) E_1^{n+\frac{1}{2}} &= E_1^n + \frac{\tau}{2\varepsilon} (\nabla \times \mathbf{H}^n)_1 - \lambda \partial_2 \partial_1 E_2^n \\ (1 - \lambda \partial_3^2) E_2^{n+\frac{1}{2}} &= E_2^n + \frac{\tau}{2\varepsilon} (\nabla \times \mathbf{H}^n)_2 - \lambda \partial_3 \partial_2 E_3^n \\ (1 - \lambda \partial_1^2) E_3^{n+\frac{1}{2}} &= E_3^n + \frac{\tau}{2\varepsilon} (\nabla \times \mathbf{H}^n)_3 - \lambda \partial_1 \partial_3 E_1^n\end{aligned}$$

Weak formulation

For example, the first equation: multiply by a test function v , integrate over Ω

$$\begin{aligned} & \left(E_1^{n+\frac{1}{2}}, v \right) - \lambda \left(\partial_2^2 E_1^{n+\frac{1}{2}}, v \right) = \\ & \left(E_1^n, v \right) + \frac{\tau}{2\varepsilon} \left((\nabla \times \mathbf{H}^n)_1, v \right) - \lambda \left(\partial_2 \partial_1 E_2^n, v \right) \end{aligned}$$

Integration by parts to get rid of second derivatives:

$$\begin{aligned} \int_{\Omega} f \frac{\partial g}{\partial x_{\alpha}} dx &= - \int_{\Omega} \frac{\partial f}{\partial x_{\alpha}} g dx + \int_{\partial\Omega} fg \hat{n}_{\alpha} d\sigma \\ (f, \partial_{\alpha} g) &= - (\partial_{\alpha} f, g) + \langle f, g \hat{n}_{\alpha} \rangle \end{aligned}$$

(\cdot, \cdot) – scalar product of $L^2(\Omega)$, $\langle \cdot, \cdot \rangle$ – scalar product of $L^2(\partial\Omega)$

After integrating LHS by parts we get

$$\begin{aligned} & \left(E_1^{n+\frac{1}{2}}, v \right) + \lambda \left(\partial_2 E_1^{n+\frac{1}{2}}, \partial_2 v \right) - \lambda \left\langle \partial_2 E_1^{n+\frac{1}{2}}, v n_2 \right\rangle \\ & \left(E_2^{n+\frac{1}{2}}, v \right) + \lambda \left(\partial_3 E_2^{n+\frac{1}{2}}, \partial_3 v \right) - \lambda \left\langle \partial_3 E_2^{n+\frac{1}{2}}, v n_3 \right\rangle \\ & \left(E_3^{n+\frac{1}{2}}, v \right) + \lambda \left(\partial_1 E_3^{n+\frac{1}{2}}, \partial_1 v \right) - \lambda \left\langle \partial_1 E_3^{n+\frac{1}{2}}, v n_1 \right\rangle \end{aligned}$$

Let us not worry about the **boundary terms** for now

Space discretization

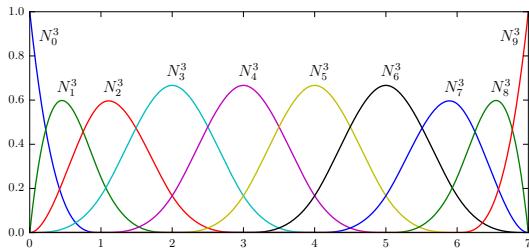
Regular domain $\Omega = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$

\mathcal{U}_h – tensor product of 1D B-spline spaces

$$\mathcal{U}_h = \text{span} \{ B_{ijk} : i = 1, \dots, N_1, j = 1, \dots, N_2, k = 1, \dots, N_3 \}$$

$$B_{ijk}(\mathbf{x}) = B_i^1(x_1) B_j^2(x_2) B_k^3(x_3)$$

where B_i^α – 1D B-spline basis functions in direction α



Space discretization (left-hand side)

$$\begin{aligned} & (B_{ijk}, B_{pqr}) + \lambda (\partial_2 B_{ijk}, \partial_2 B_{pqr}) = \\ & \int_{\Omega} B_{ijk} B_{pqr} dx + \lambda \int_{\Omega} \partial_2 B_{ijk} \partial_2 B_{pqr} dx = \\ & \int_{\Omega} B_i^1 B_j^2 B_k^3 B_p^1 B_q^2 B_r^3 dx + \lambda \int_{\Omega} \partial_2 (B_i^1 B_j^2 B_k^3) \partial_2 (B_p^1 B_q^2 B_r^3) dx = \\ & \int_{\Omega} (B_i^1 B_p^1) (B_j^2 B_q^2) (B_k^3 B_r^3) dx + \lambda \int_{\Omega} (B_i^1 B_p^1) ((B_j^2)') ((B_q^2)') (B_k^3 B_r^3) dx = \\ & \left(\int_{\Omega_1} B_i^1 B_p^1 dx_1 \right) \left(\int_{\Omega_2} B_j^2 B_q^2 dx_2 \right) \left(\int_{\Omega_3} B_k^3 B_r^3 dx_3 \right) + \\ & \lambda \left(\int_{\Omega_1} B_i^1 B_p^1 dx_1 \right) \left(\int_{\Omega_2} (B_j^2)' (B_q^2)' dx_2 \right) \left(\int_{\Omega_3} B_k^3 B_r^3 dx_3 \right) = \\ & \left(\int_{\Omega_1} B_i^1 B_p^1 dx_1 \right) \left(\int_{\Omega_2} B_j^2 B_q^2 + \lambda (B_j^2)' (B_q^2)' dx_2 \right) \left(\int_{\Omega_3} B_k^3 B_r^3 dx_3 \right) \end{aligned}$$

Assuming **the boundary terms** vanish, we are left with

$$\mathbf{L}^{(1)} = \mathbf{M}_1 \otimes (\mathbf{M}_2 + \lambda \mathbf{S}_2) \otimes \mathbf{M}_3$$

$$\mathbf{L}^{(2)} = \mathbf{M}_1 \otimes \mathbf{M}_2 \otimes (\mathbf{M}_3 + \lambda \mathbf{S}_3)$$

$$\mathbf{L}^{(3)} = (\mathbf{M}_1 + \lambda \mathbf{S}_1) \otimes \mathbf{M}_2 \otimes \mathbf{M}_3$$

Kronecker product structure \Rightarrow can be efficiently solved using ADS

$\mathbf{M}_\alpha, \mathbf{S}_\alpha$ – 1D mass and stiffness matrices

banded structure \Rightarrow linear time factorization

Boundary conditions

Do the boundary terms vanish? Depends on BC.

One possible formulation:

$$\mathbf{E} \times \hat{\mathbf{n}} = 0 \quad \mathbf{H} \cdot \hat{\mathbf{n}} = 0$$

Boundary of the cube $\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$, $\hat{\mathbf{n}} = \pm \hat{\mathbf{e}}_k$ on Γ_k

- $E_2 = E_3 = 0$ on Γ_1
- $E_1 = E_3 = 0$ on Γ_2
- $E_1 = E_2 = 0$ on Γ_3

\Rightarrow boundary terms vanish

Problem These BC reflect waves and are not suitable for simulating an antenna

Absorbing Boundary Conditions (ABC)

What we need is

$$-\hat{\mathbf{n}} \times \mathbf{H} + \frac{1}{\eta} \hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \mathbf{E}) = \mathbf{U}$$

where $\eta = \sqrt{\mu/\epsilon}$, \mathbf{U} – incident field

Issues

- introduces coupling between E and H
- interferes with the splitting

Solution Alternative formulation $\partial_t(BC)$

$$-\hat{\mathbf{n}} \times \underbrace{\partial_t \mathbf{H}}_{-\frac{1}{\mu} \nabla \times \mathbf{E}} + \frac{1}{\eta} \hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \partial_t \mathbf{U}$$

$$\hat{\mathbf{n}} \times \frac{1}{\mu} \nabla \times \mathbf{E} + \frac{1}{\eta} \hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \partial_t \mathbf{U}$$

Absorbing boundary conditions

Since $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c}) \mathbf{b} - (\mathbf{a} \cdot \mathbf{b}) \mathbf{c}$ we have

$$\begin{aligned} \hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) &= \nabla(\hat{\mathbf{n}} \cdot \mathbf{E}) - (\hat{\mathbf{n}} \cdot \nabla) \mathbf{E} \\ -\hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) &= \partial_t \mathbf{E} - \underbrace{(\hat{\mathbf{n}} \cdot \partial_t \mathbf{E}) \hat{\mathbf{n}}}_{\substack{\text{perpendicular component} \\ \text{of } \partial_t \mathbf{E}}} \\ &\quad \underbrace{\hspace{10em}}_{\substack{\text{tangential component} \\ \text{of } \partial_t \mathbf{E}}} \end{aligned}$$

For $\hat{\mathbf{n}} = \pm \hat{\mathbf{e}}_k \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$ over the cube

$$\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) = \pm (\nabla E_k - \partial_k \mathbf{E}) = \hat{\mathbf{n}}_k \begin{bmatrix} \partial_1 E_k - \partial_k E_1 \\ \partial_2 E_k - \partial_k E_2 \\ \partial_3 E_k - \partial_k E_3 \end{bmatrix}$$

$k=1,2,3$, $\partial_k \in \{\partial_1, \partial_2, \partial_3\} = (\text{like}) = \{\partial_x, \partial_y, \partial_z\}$, $\hat{\mathbf{n}}_k \in \{-1, 1\}$

Absorbing boundary conditions

Back to b.c. $\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) + \frac{\mu}{\eta} \hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \mu \partial_t \mathbf{U}$
using (for a given k we have two non-zero components on a cube)

$$\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) = \pm (\nabla E_k - \partial_k \mathbf{E}) = \hat{\mathbf{n}}_k \begin{bmatrix} \partial_1 E_k - \partial_k E_1 \\ \partial_2 E_k - \partial_k E_2 \\ \partial_3 E_k - \partial_k E_3 \end{bmatrix}$$

and

$$-\hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \partial_t \mathbf{E} - \underbrace{(\hat{\mathbf{n}} \cdot \partial_t \mathbf{E}) \hat{\mathbf{n}}}_{\substack{\text{perpendicular component} \\ \text{of } \partial_t \mathbf{E} = 0 \text{ here}}}$$

tangential component
of $\partial_t \mathbf{E}$

On the cube this amounts to

- on Γ_1 , $\hat{\mathbf{n}} = \pm \hat{\mathbf{e}}_1$:

$$\hat{\mathbf{n}}_1 (\nabla \times \mathbf{E})_3 = -\mu U_2 - \frac{\mu}{\eta} \partial_t E_2$$
$$\hat{\mathbf{n}}_1 (\nabla \times \mathbf{E})_2 = \mu U_3 + \frac{\mu}{\eta} \partial_t E_3$$

Absorbing boundary conditions

Back to b.c. $\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) + \frac{\mu}{\eta} \hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \mu \partial_t \mathbf{U}$
using (for a given k we have two non-zero components on a cube)

$$\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) = \pm (\nabla E_k - \partial_k \mathbf{E}) = \hat{\mathbf{n}}_k \begin{bmatrix} \partial_1 E_k - \partial_k E_1 \\ \partial_2 E_k - \partial_k E_2 \\ \partial_3 E_k - \partial_k E_3 \end{bmatrix}$$

and

$$-\hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \partial_t \mathbf{E} - \underbrace{(\hat{\mathbf{n}} \cdot \partial_t \mathbf{E}) \hat{\mathbf{n}}}_{\substack{\text{perpendicular component} \\ \text{of } \partial_t \mathbf{E} = 0 \text{ here}}}$$

tangential component
of $\partial_t \mathbf{E}$

On the cube this amounts to

- on Γ_2 , $\hat{\mathbf{n}} = \pm \hat{\mathbf{e}}_2$:

$$\hat{\mathbf{n}}_2 (\nabla \times \mathbf{E})_3 = -\mu U_1 - \frac{\mu}{\eta} \partial_t E_1$$
$$\hat{\mathbf{n}}_2 (\nabla \times \mathbf{E})_1 = \mu U_3 + \frac{\mu}{\eta} \partial_t E_3$$

Absorbing boundary conditions

Back to b.c. $\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) + \frac{\mu}{\eta} \hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \mu \partial_t \mathbf{U}$
using (for a given k we have two non-zero components on a cube)

$$\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}) = \pm (\nabla E_k - \partial_k \mathbf{E}) = \hat{\mathbf{n}}_k \begin{bmatrix} \partial_1 E_k - \partial_k E_1 \\ \partial_2 E_k - \partial_k E_2 \\ \partial_3 E_k - \partial_k E_3 \end{bmatrix}$$

and

$$-\hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \partial_t \mathbf{E}) = \partial_t \mathbf{E} - \underbrace{(\hat{\mathbf{n}} \cdot \partial_t \mathbf{E}) \hat{\mathbf{n}}}_{\substack{\text{perpendicular component} \\ \text{of } \partial_t \mathbf{E} = 0 \text{ here}}}$$

tangential component
of $\partial_t \mathbf{E}$

On the cube this amounts to

- on Γ_3 , $\hat{\mathbf{n}} = \pm \hat{\mathbf{e}}_3$:

$$\begin{aligned} \hat{\mathbf{n}}_3 (\nabla \times \mathbf{E})_2 &= -\mu U_1 - \frac{\mu}{\eta} \partial_t E_1 \\ \hat{\mathbf{n}}_3 (\nabla \times \mathbf{E})_1 &= \mu U_2 + \frac{\mu}{\eta} \partial_t E_2 \end{aligned}$$

Absorbing boundary conditions

How to use it? (integration by parts in the weak form results in the following boundary terms) [we skip all other parts here]

- LHS:

$$-\lambda \partial_2^2 E_1^{n+\frac{1}{2}} \rightarrow -\lambda \left\langle \partial_2 E_1^{n+\frac{1}{2}}, v \hat{n}_2 \right\rangle$$

$$-\lambda \partial_3^2 E_2^{n+\frac{1}{2}} \rightarrow -\lambda \left\langle \partial_3 E_2^{n+\frac{1}{2}}, v \hat{n}_3 \right\rangle$$

$$-\lambda \partial_1^2 E_3^{n+\frac{1}{2}} \rightarrow -\lambda \left\langle \partial_1 E_3^{n+\frac{1}{2}}, v \hat{n}_1 \right\rangle$$

- RHS:

$$-\lambda \partial_2 \partial_1 E_2^n \rightarrow -\lambda \left\langle \partial_1 E_2^n, v \hat{n}_2 \right\rangle$$

$$-\lambda \partial_3 \partial_2 E_3^n \rightarrow -\lambda \left\langle \partial_2 E_3^n, v \hat{n}_3 \right\rangle$$

$$-\lambda \partial_1 \partial_3 E_1^n \rightarrow -\lambda \left\langle \partial_3 E_1^n, v \hat{n}_1 \right\rangle$$

Idea Change $\mathbf{E}^{n+\frac{1}{2}}$ to \mathbf{E}^n and put it on the RHS

Absorbing boundary conditions

$$\lambda \langle \partial_2 E_1^n, \nu n_2 \rangle - \lambda \langle \partial_1 E_2^n, \nu n_2 \rangle = -\lambda \left\langle \underbrace{-\partial_2 E_1^n + \partial_1 E_2^n}_{(\nabla \times \mathbf{E}^n)_3}, \nu n_2 \right\rangle$$

$$\lambda \langle \partial_3 E_2^n, \nu n_3 \rangle - \lambda \langle \partial_2 E_3^n, \nu n_3 \rangle = -\lambda \left\langle \underbrace{-\partial_3 E_2^n + \partial_2 E_3^n}_{(\nabla \times \mathbf{E}^n)_1}, \nu n_3 \right\rangle$$

$$\lambda \langle \partial_1 E_3^n, \nu n_1 \rangle - \lambda \langle \partial_3 E_1^n, \nu n_1 \rangle = -\lambda \left\langle \underbrace{-\partial_1 E_3^n + \partial_3 E_1^n}_{(\nabla \times \mathbf{E}^n)_2}, \nu n_1 \right\rangle$$

Each boundary term is non-zero on exactly one of $\Gamma_1, \Gamma_2, \Gamma_3$

Absorbing boundary conditions

Using boundary conditions we can rewrite components of $\nabla \times \mathbf{E}$ as

$$\left\langle \underbrace{-\partial_2 E_1^n + \partial_1 E_2^n}_{(\nabla \times \mathbf{E}^n)_3}, \nu n_2 \right\rangle = \left\langle \mu U_1 + \frac{\mu}{\eta} \partial_t E_1, \nu \right\rangle_{\Gamma_2}$$

$$\left\langle \underbrace{-\partial_3 E_2^n + \partial_2 E_3^n}_{(\nabla \times \mathbf{E}^n)_1}, \nu n_3 \right\rangle = \left\langle \mu U_2 + \frac{\mu}{\eta} \partial_t E_2, \nu \right\rangle_{\Gamma_3}$$

$$\left\langle \underbrace{-\partial_1 E_3^n + \partial_3 E_1^n}_{(\nabla \times \mathbf{E}^n)_2}, \nu n_1 \right\rangle = \left\langle \mu U_3 + \frac{\mu}{\eta} \partial_t E_3, \nu \right\rangle_{\Gamma_1}$$

and approximate the time derivative as

$$\partial_t \mathbf{E}^n \approx \frac{\mathbf{E}^n - \mathbf{E}^{n-1}}{\tau} \quad \partial_t \mathbf{E}^{n+\frac{1}{2}} \approx \frac{\mathbf{E}^{n+\frac{1}{2}} - \mathbf{E}^n}{\tau/2}$$

Full formulation

Let $*$ = $n + \frac{1}{2}$ and

$$b_{\alpha\beta}(u, v) = \lambda(\partial_\alpha u, \partial_\beta v) \quad a_\alpha(u, v) = (u, v) + b_{\alpha\alpha}(u, v)$$

$$c_\alpha(\mathbf{F}, v) = \frac{\tau}{2\varepsilon} ((\nabla \times \mathbf{F})_\alpha, v) \quad \gamma_\alpha(u, v) = \lambda \mu \langle u, v \rangle_{\Gamma_\alpha}$$

$$a_2(\mathbf{E}_1^*, v) = (\mathbf{E}_1^n, v) + b_{12}(\mathbf{E}_2^n, v) + c_1(\mathbf{H}^n, v) - \gamma_2 \left(U_1 + \frac{1}{\eta} \partial_t \mathbf{E}_1^n, v \right)$$

$$a_3(\mathbf{E}_2^*, v) = (\mathbf{E}_2^n, v) + b_{23}(\mathbf{E}_3^n, v) + c_2(\mathbf{H}^n, v) - \gamma_3 \left(U_2 + \frac{1}{\eta} \partial_t \mathbf{E}_2^n, v \right)$$

$$a_1(\mathbf{E}_3^*, v) = (\mathbf{E}_3^n, v) + b_{31}(\mathbf{E}_1^n, v) + c_3(\mathbf{H}^n, v) - \gamma_1 \left(U_3 + \frac{1}{\eta} \partial_t \mathbf{E}_3^n, v \right)$$

$$a_3(\mathbf{E}_1^{n+1}, v) = (\mathbf{E}_1^*, v) + b_{13}(\mathbf{E}_3^*, v) + c_1(\mathbf{H}^*, v) - \gamma_3 \left(U_1 + \frac{1}{\eta} \partial_t \mathbf{E}_1^*, v \right)$$

$$a_1(\mathbf{E}_2^{n+1}, v) = (\mathbf{E}_2^*, v) + b_{21}(\mathbf{E}_1^*, v) + c_2(\mathbf{H}^*, v) - \gamma_1 \left(U_2 + \frac{1}{\eta} \partial_t \mathbf{E}_2^*, v \right)$$

$$a_2(\mathbf{E}_3^{n+1}, v) = (\mathbf{E}_3^*, v) + b_{32}(\mathbf{E}_2^*, v) + c_3(\mathbf{H}^*, v) - \gamma_2 \left(U_3 + \frac{1}{\eta} \partial_t \mathbf{E}_3^*, v \right)$$

Full formulation $* = n + \frac{1}{2}$

$$a_2(E_1^*, v) = (E_1^n, v) + b_{12}(E_2^n, v) + c_1(\mathbf{H}^n, v) - \gamma_2 \left(U_1 + \frac{1}{\eta} \partial_t E_1^n, v \right)$$

$$a_3(E_1^{n+1}, v) = (E_1^*, v) + b_{13}(E_3^*, v) + c_1(\mathbf{H}^*, v) - \gamma_3 \left(U_1 + \frac{1}{\eta} \partial_t E_1^*, v \right)$$

$$a_3(E_2^*, v) = (E_2^n, v) + b_{23}(E_3^n, v) + c_2(\mathbf{H}^n, v) - \gamma_3 \left(U_2 + \frac{1}{\eta} \partial_t E_2^n, v \right)$$

$$a_1(E_2^{n+1}, v) = (E_2^*, v) + b_{21}(E_1^*, v) + c_2(\mathbf{H}^*, v) - \gamma_1 \left(U_2 + \frac{1}{\eta} \partial_t E_2^*, v \right)$$

$$a_1(E_3^*, v) = (E_3^n, v) + b_{31}(E_1^n, v) + c_3(\mathbf{H}^n, v) - \gamma_1 \left(U_3 + \frac{1}{\eta} \partial_t E_3^n, v \right)$$

$$a_2(E_3^{n+1}, v) = (E_3^*, v) + b_{32}(E_2^*, v) + c_3(\mathbf{H}^*, v) - \gamma_2 \left(U_3 + \frac{1}{\eta} \partial_t E_3^*, v \right)$$

Numerical example – scattering problem

Manufactured solution problem on $\Omega = [0, 1] \times [0, 1] \times [0, 20]$

$$\mathbf{E}(\mathbf{x}, t) = \cos\left(\omega_0 \left(t - \frac{x_3}{c_0}\right)\right) g\left(t - \frac{x_3}{c_0}\right) \hat{\mathbf{e}}_1$$
$$\mathbf{H}(\mathbf{x}, t) = \frac{1}{\eta_0} \cos\left(\omega_0 \left(t - \frac{x_3}{c_0}\right)\right) g\left(t - \frac{x_3}{c_0}\right) \hat{\mathbf{e}}_2$$

where

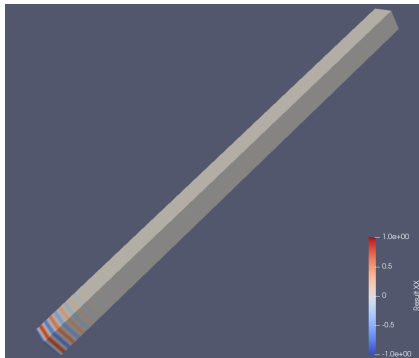
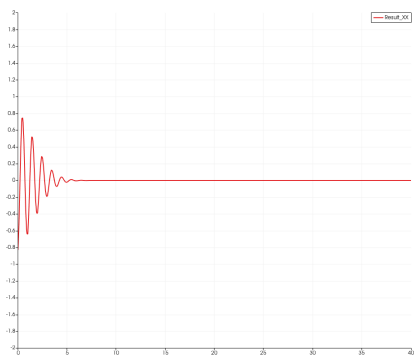
$$g(s) = \begin{cases} \exp\left(-\frac{1}{2} \left(\frac{s-t_0}{\sigma}\right)^2\right) & s \geq 0 \\ 0 & s < 0 \end{cases}$$

c_0 – speed of light, $\omega_0 = 2\pi f_0$, $f_0 = 2c_0$, $\sigma = 4/f_0$

Discretization

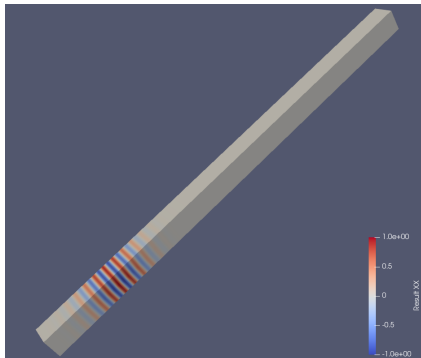
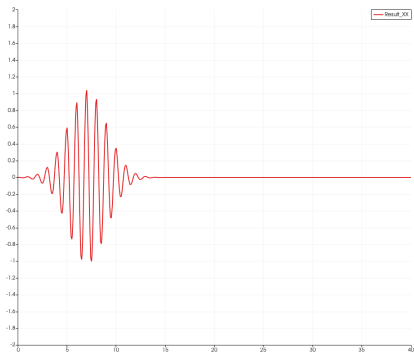
- mesh size $4 \times 4 \times 100$
- time step $\tau = 2.5 \times 10^{-11}$, 4000 steps (total time 100 ns)

Results – scattering problem



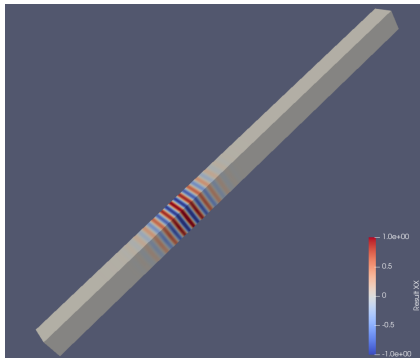
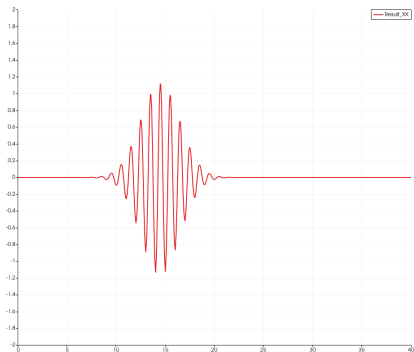
Step 500 ($t = 12.5$ ns)

Results – scattering problem



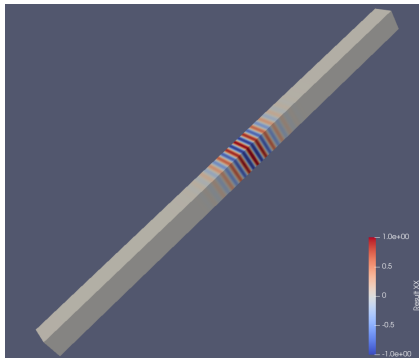
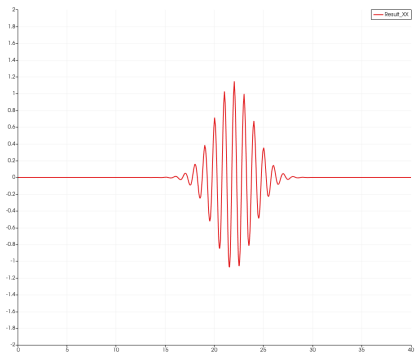
Step 1000 ($t = 25$ ns)

Results – scattering problem



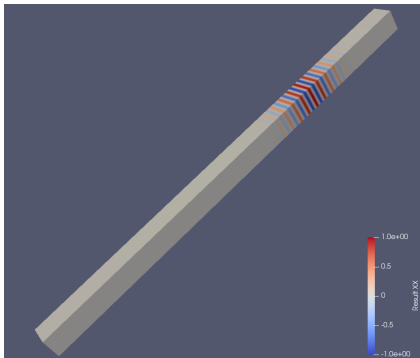
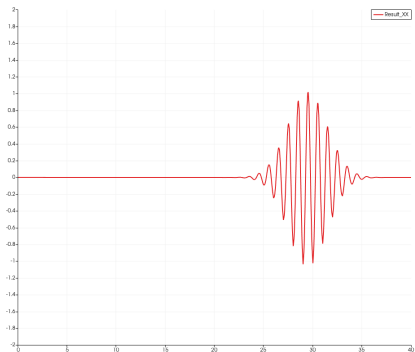
Step 1500 ($t = 37.5$ ns)

Results – scattering problem



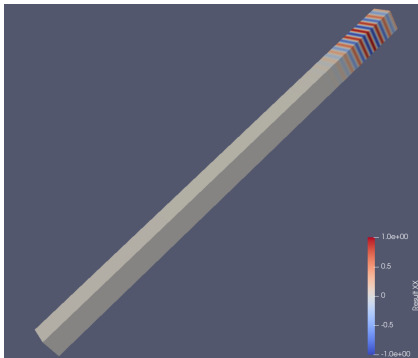
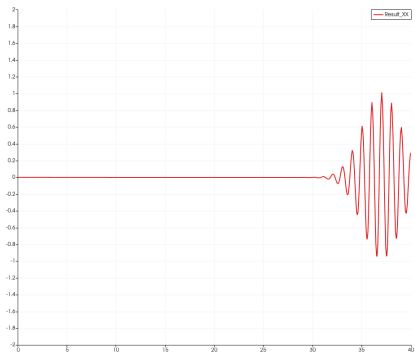
Step 2000 ($t = 50$ ns)

Results – scattering problem



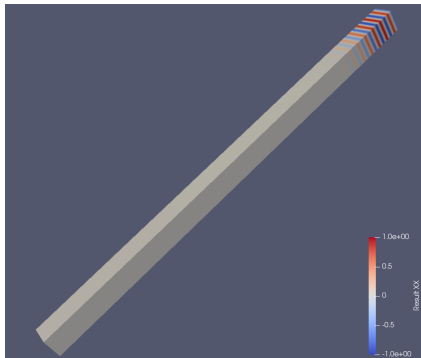
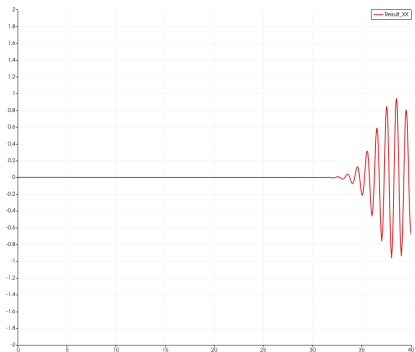
Step 2500 ($t = 62.5$ ns)

Results – scattering problem



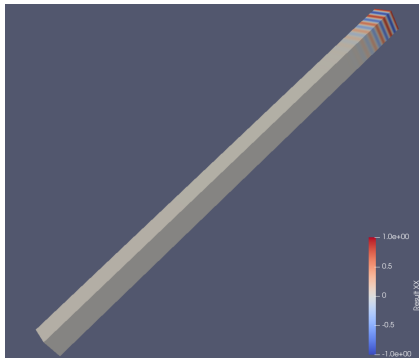
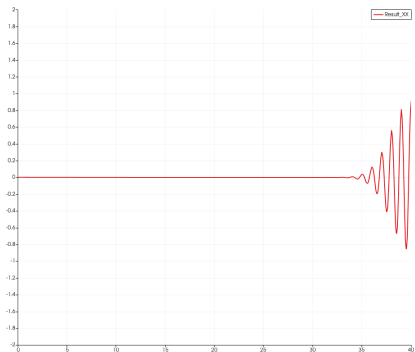
Step 3000 ($t = 75$ ns)

Results – scattering problem



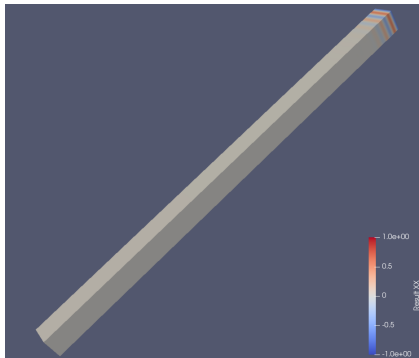
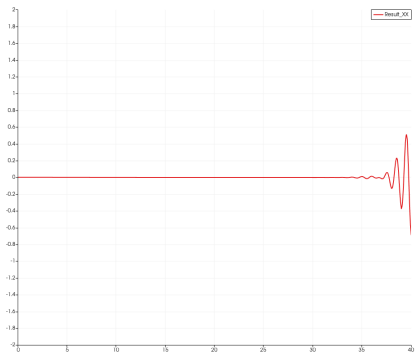
Step 3100 ($t = 77.5$ ns)

Results – scattering problem



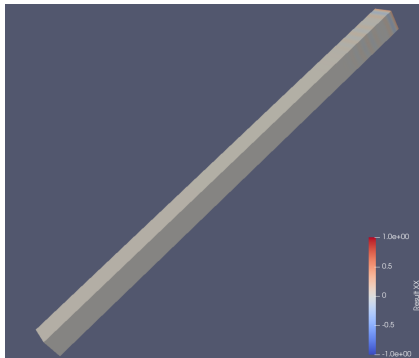
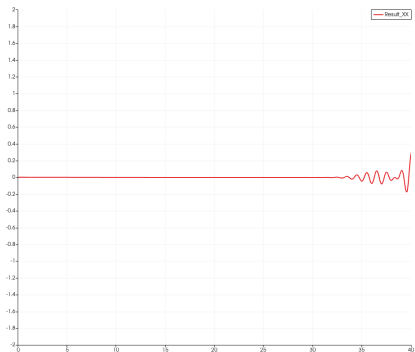
Step 3200 ($t = 80$ ns)

Results – scattering problem



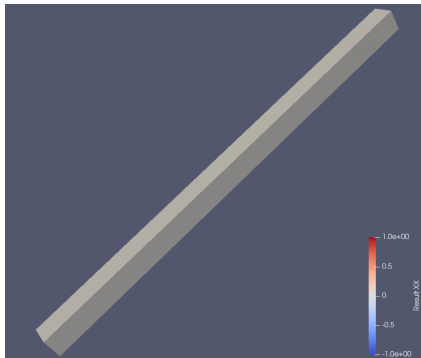
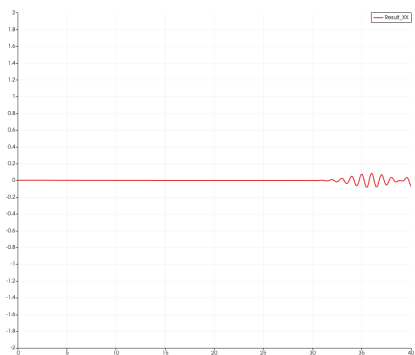
Step 3300 ($t = 82.5$ ns)

Results – scattering problem



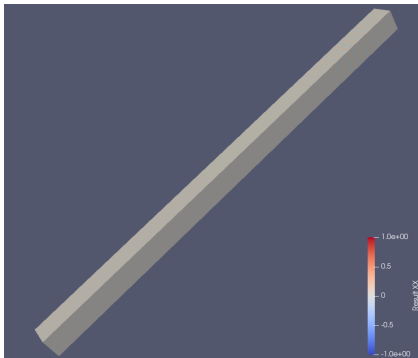
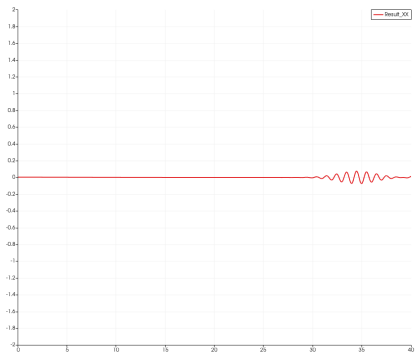
Step 3400 ($t = 85$ ns)

Results – scattering problem



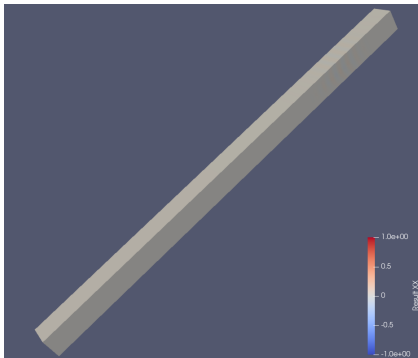
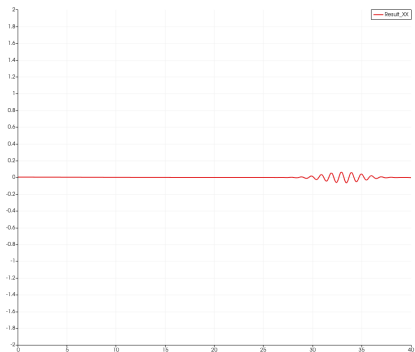
Step 3500 ($t = 87.5$ ns)

Results – scattering problem



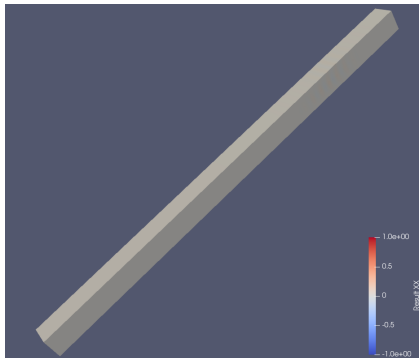
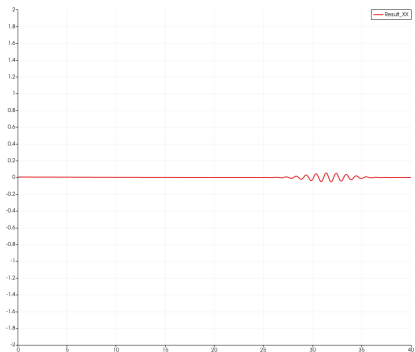
Step 3600 ($t = 90$ ns)

Results – scattering problem



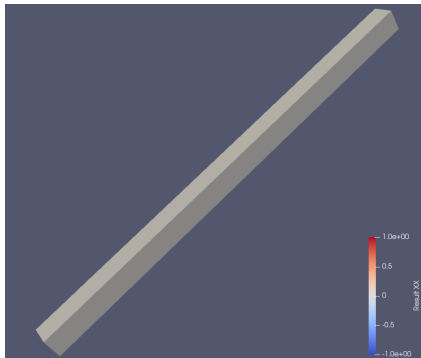
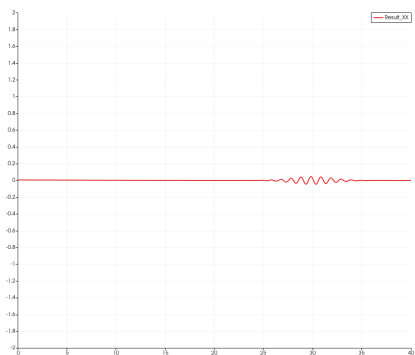
Step 3700 ($t = 92.5$ ns)

Results – scattering problem



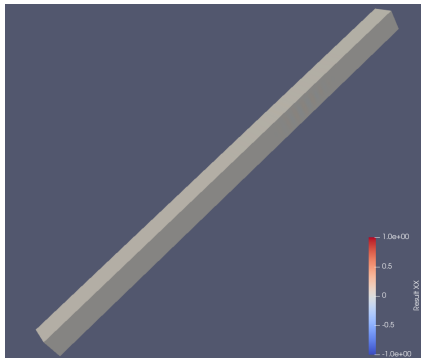
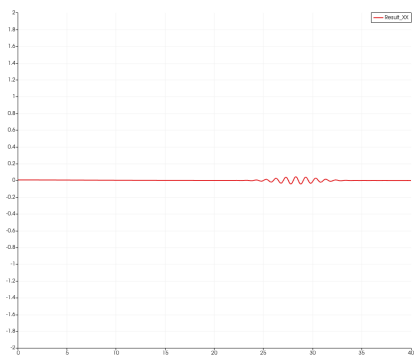
Step 3800 ($t = 95$ ns)

Results – scattering problem



Step 3900 ($t = 97.5$ ns)

Results – scattering problem



Step 4000 ($t = 100$ ns)

Code verification with manufactured solution

For $\Omega = [0, 1]^3$, for $\epsilon = 1$ and $\mu = 1$ we define

$$u_{\kappa, \lambda}^1(x, t) = \begin{bmatrix} \sin(\kappa\pi x_2) \sin(\lambda\pi x_3) \cos(\sqrt{\kappa^2 + \lambda^2}\pi t) \\ 0 \\ 0 \\ 0 \\ -\frac{\lambda}{\sqrt{\kappa^2 + \lambda^2}} \sin(\kappa\pi x_2) \cos(\lambda\pi x_3) \sin(\sqrt{\kappa^2 + \lambda^2}\pi t) \\ \frac{\kappa}{\sqrt{\kappa^2 + \lambda^2}} \cos(\kappa\pi x_2) \sin(\lambda\pi x_3) \sin(\sqrt{\kappa^2 + \lambda^2}\pi t) \end{bmatrix}$$

for $\kappa, \lambda \in \mathcal{N}$, $\kappa, \lambda \neq 0$.

Code verification with manufactured solution

For $\Omega = [0, 1]^3$, for $\epsilon = 1$ and $\mu = 1$ we define

$$u_{\kappa, \lambda}^2(x, t) = \begin{bmatrix} 0 \\ \sin(\kappa\pi x_1) \sin(\lambda\pi x_3) \cos(\sqrt{\kappa^2 + \lambda^2}\pi t) \\ 0 \\ -\frac{\lambda}{\sqrt{\kappa^2 + \lambda^2}} \sin(\kappa\pi x_1) \cos(\lambda\pi x_3) \sin(\sqrt{\kappa^2 + \lambda^2}\pi t) \\ 0 \\ \frac{\kappa}{\sqrt{\kappa^2 + \lambda^2}} \cos(\kappa\pi x_1) \sin(\lambda\pi x_3) \sin(\sqrt{\kappa^2 + \lambda^2}\pi t) \end{bmatrix}$$

for $\kappa, \lambda \in \mathcal{N}$, $\kappa, \lambda \neq 0$.

Code verification with manufactured solution

For $\Omega = [0, 1]^3$, for $\epsilon = 1$ and $\mu = 1$ we define

$$u_{\kappa, \lambda}^3(x, t) = \begin{bmatrix} 0 \\ 0 \\ \frac{\sin(\kappa\pi x_1) \sin(\lambda\pi x_2) \cos(\sqrt{\kappa^2 + \lambda^2}\pi t)}{\sqrt{\kappa^2 + \lambda^2}} \\ -\frac{\lambda}{\sqrt{\kappa^2 + \lambda^2}} \sin(\kappa\pi x_1) \cos(\lambda\pi x_2) \sin(\sqrt{\kappa^2 + \lambda^2}\pi t) \\ \frac{\kappa}{\sqrt{\kappa^2 + \lambda^2}} \cos(\kappa\pi x_1) \sin(\lambda\pi x_2) \sin(\sqrt{\kappa^2 + \lambda^2}\pi t) \\ 0 \end{bmatrix}$$

for $\kappa, \lambda \in \mathcal{N}$, $\kappa, \lambda \neq 0$.

Code verification with manufactured solution

There first manufactured solution function is

$$\mathbf{u}_A(x, t) = \gamma u_{1,1}^1(x, t) + 2\gamma u_{1,1}^2(x, t) + 3\gamma u_{1,1}^3(x, t) \quad (8)$$

Notice that \mathbf{u}_A has six components, where the first three components denote \mathbf{E} and the last three components denote \mathbf{H} .

The parameter γ is selected in such a way that $\|\mathbf{u}_A(x, 0)\|_{L^2(\Omega)} = 1$.

We define

$$\underbrace{\begin{bmatrix} \sin(\pi x_2) \sin(\pi x_3) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{u_{1,1}^1(x,0)}, \quad \underbrace{\begin{bmatrix} 0 \\ \sin(\pi x_1) \sin(\pi x_3) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{u_{1,1}^2(x,0)}, \quad \underbrace{\begin{bmatrix} 0 \\ 0 \\ \sin(\pi x_1) \sin(\pi x_2) \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{u_{1,1}^3(x,0)}.$$

$$\begin{aligned}
1 &= \|\mathbf{u}_A(x, 0)\|_{L^2(\Omega)}^2 = \\
&\int_{[0,1]^3} \|\gamma u_{1,1}^1(x, 0) + 2\gamma u_{1,1}^2(x, 0) + 3\gamma u_{1,1}^3(x, 0)\|^2 dx_1 dx_2 dx_3 = \\
&\int_{[0,1]^3} \left\| \begin{bmatrix} \gamma \sin(\pi x_2) \sin(\pi x_3) \\ 2\gamma \sin(\pi x_1) \sin(\pi x_3) \\ 3\gamma \sin(\pi x_1) \sin(\pi x_2) \\ 0 \\ 0 \\ 0 \end{bmatrix} \right\|^2 dx_1 dx_2 dx_3 = \\
&\int_{[0,1]^3} (\gamma^2 \sin^2(\pi x_2) \sin^2(\pi x_3) + 4\gamma^2 \sin^2(\pi x_1) \sin^2(\pi x_3)) \\
&\quad + \int_{[0,1]^3} (9\gamma^2 \sin^2(\pi x_1) \sin^2(\pi x_2)) dx_1 dx_2 dx_3 = \\
&(\gamma^2 \frac{1}{2} \frac{1}{2} + 4\gamma^2 \frac{1}{2} \frac{1}{2} + 9\gamma^2 \frac{1}{2} \frac{1}{2}) = (\frac{1}{4}\gamma^2 + \gamma^2 + \frac{9}{4}\gamma^2) = \frac{14}{4}\gamma^2 \quad (9)
\end{aligned}$$

Code verification with manufactured solution

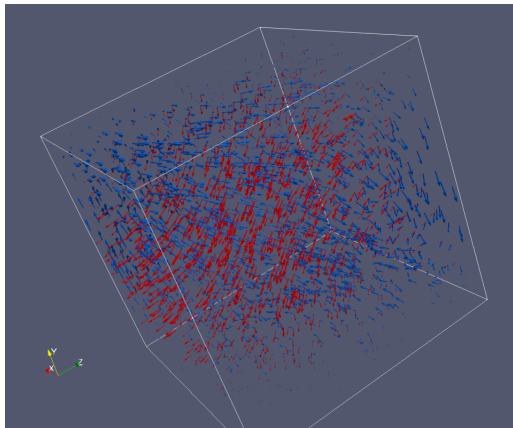


Figure: Electric (red) and magnetic (blue) vector fields, resulting from the problem with manufactured solution.

Code verification with manufactured solution

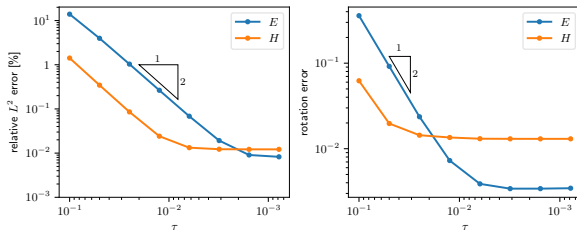


Figure: Order of the time integration scheme as measured using $\int E^2(\mathbf{x})d\mathbf{x}$ norm (left) and $\int(\nabla \times E(\mathbf{x}))^2d\mathbf{x}$ norm (right) for electric (blue) vector field and using $\int H^2(\mathbf{x})d\mathbf{x}$ norm (left) and $\int(\nabla \times H(\mathbf{x}))^2d\mathbf{x}$ norm (right) for magnetic (orange) vector fields. The problem with manufactured solution over the computational mesh with $16 \times 16 \times 16$ elements.

Code verification with manufactured solution

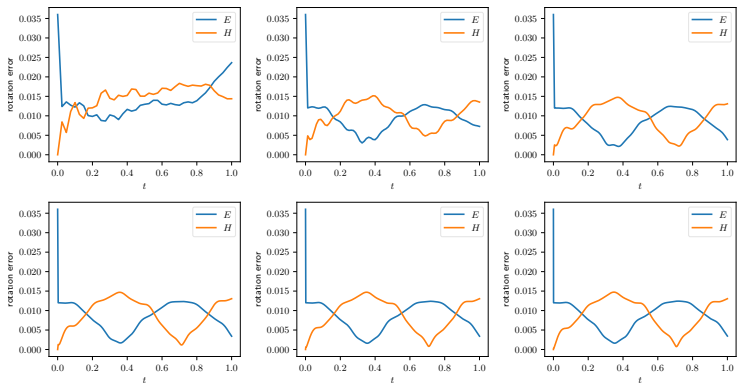


Figure: H-curl error of electric (blue) and magnetic (orange) vector fields. The problem with manufactured solution over the computational mesh with $16 \times 16 \times 16$ elements, for the time interval $[0,1]$, with $\#$ time step 40,80,160 (first), 320, 640, 1280 (second row).

Antenna problem

For the antenna problem, $\mathbf{U} = 0$ and there is an additional term on the RHS representing the antenna (electric dipole):

$$\begin{aligned}\partial_t \mathbf{E} &= \frac{1}{\varepsilon_0} \nabla \times \mathbf{H} - \mathbf{J}_{\text{imp}} \\ \partial_t \mathbf{H} &= -\frac{1}{\mu_0} \nabla \times \mathbf{E}\end{aligned}$$

where \mathbf{J}_{imp} is non-zero on a very thin, short part of Ω

Remark In the weak formulation it comes up as a line integral

$$\dots - \int_{\Gamma} \mathbf{J}_{\text{imp}} \cdot \mathbf{v} \, d\sigma$$

as a limit of $(\mathbf{J}_{\text{imp}}, \mathbf{v})$ as the width $\rightarrow 0$

Numerical example – antenna in vacuum

Domain $\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$

$$\mathbf{J}_{\text{imp}}(\mathbf{x}, t) = \begin{cases} J_0(x_3, t) \hat{\mathbf{e}}_3 & x_1 = x_2 = 0, x_3 \in [-l/2, l/2] \\ 0 & \text{elsewhere} \end{cases}$$

where $J_0(x, t) = g(t) \sin \omega_0 t$ and

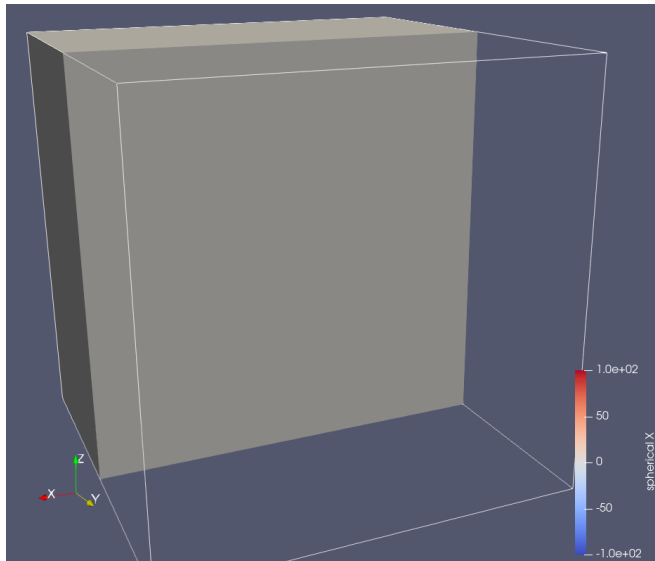
$$g(t) = 1 - \exp(-t/\sigma)$$

with $l = 1/50$, $\omega_0 = 2\pi f_0$, $f_0 = 2c_0$, $\sigma = 2/f_0$

Discretization

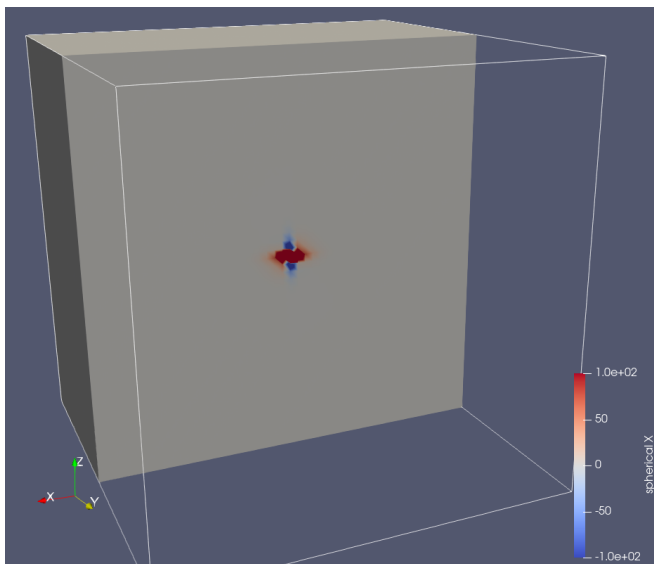
- mesh size $100 \times 100 \times 100$
- time step $\tau = 2.5 \times 10^{-11}$, 200 steps (total time 5 ns)

Results – antenna in vacuum



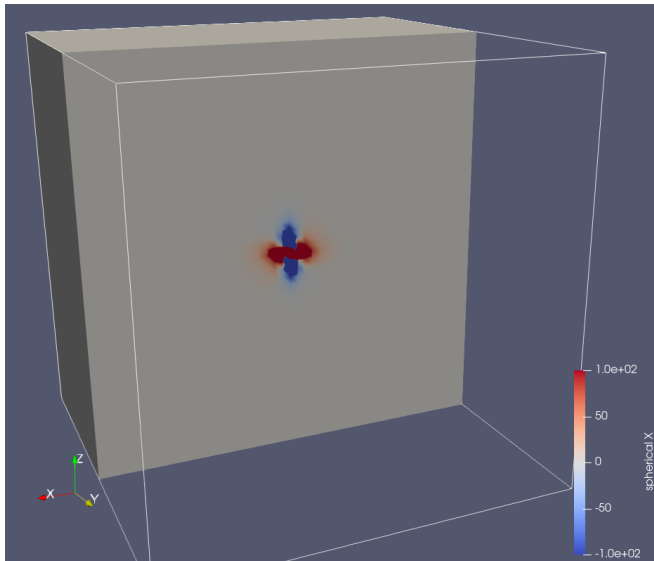
Step 0 ($t = 0$ ns)

Results – antenna in vacuum



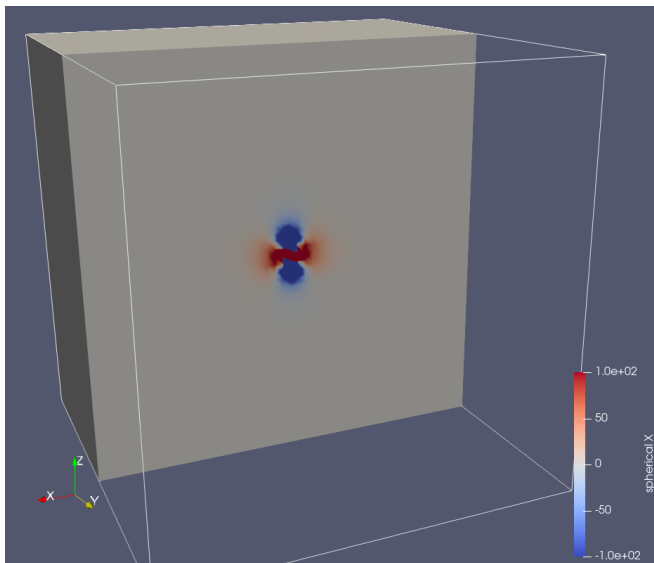
Step 10 ($t = 0.25$ ns)

Results – antenna in vacuum



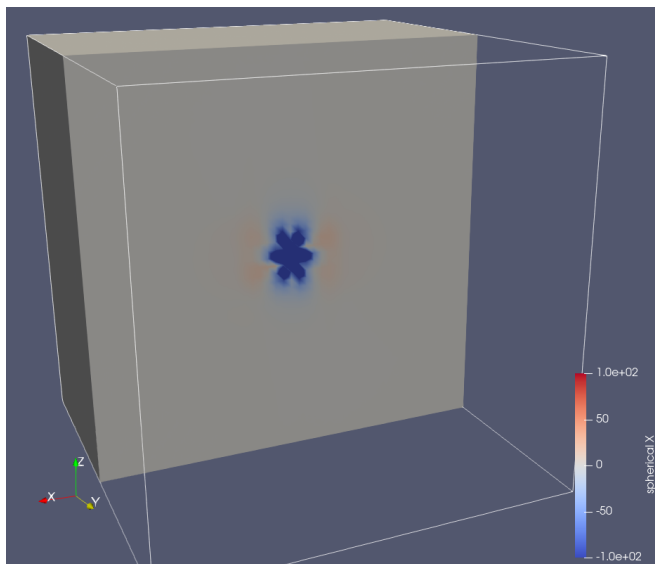
Step 20 ($t = 0.5 \text{ ns}$)

Results – antenna in vacuum



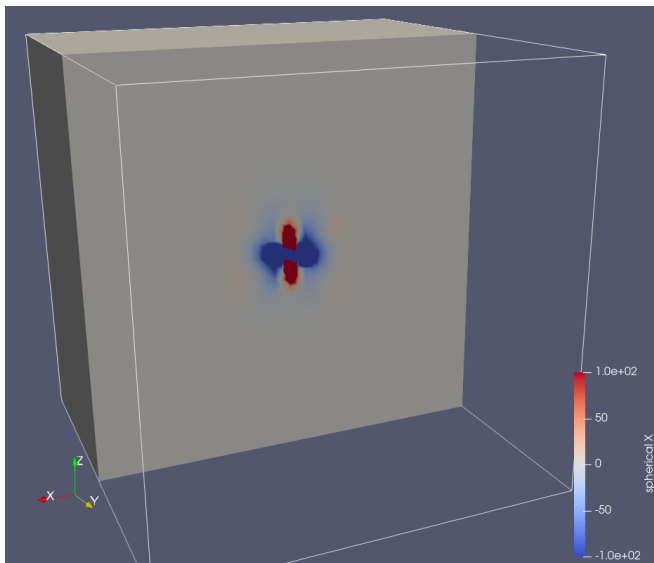
Step 30 ($t = 0.75$ ns)

Results – antenna in vacuum



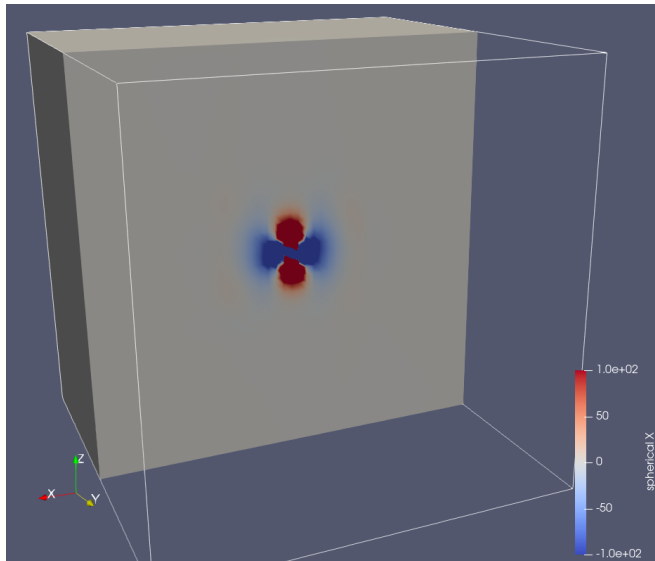
Step 40 ($t = 1 \text{ ns}$)

Results – antenna in vacuum



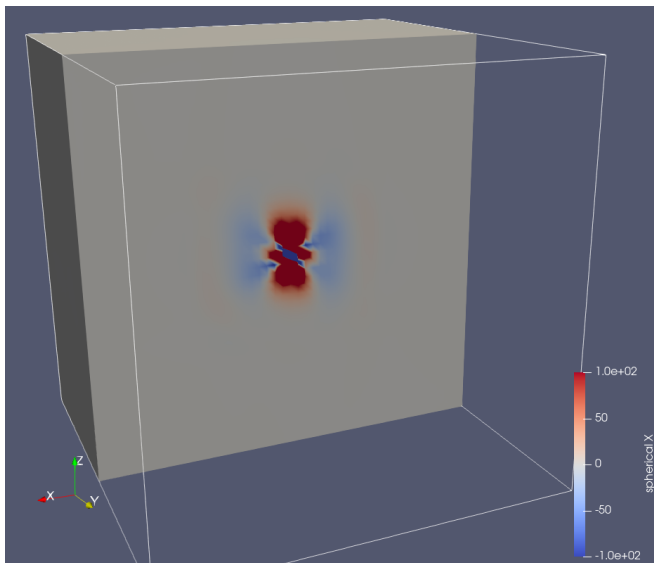
Step 50 ($t = 1.25 \text{ ns}$)

Results – antenna in vacuum



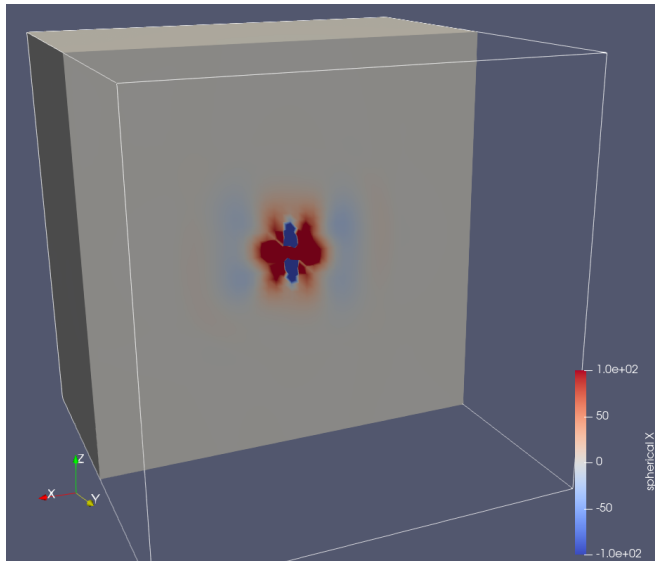
Step 60 ($t = 1.5 \text{ ns}$)

Results – antenna in vaccuum



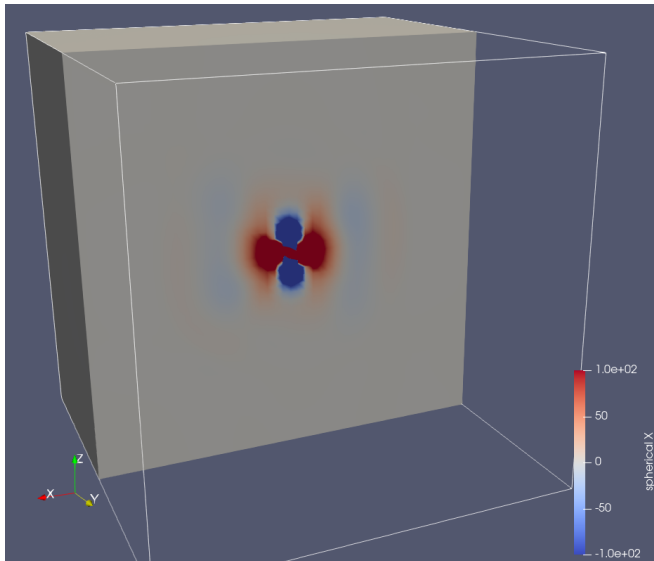
Step 70 ($t = 1.75 \text{ ns}$)

Results – antenna in vaccuum



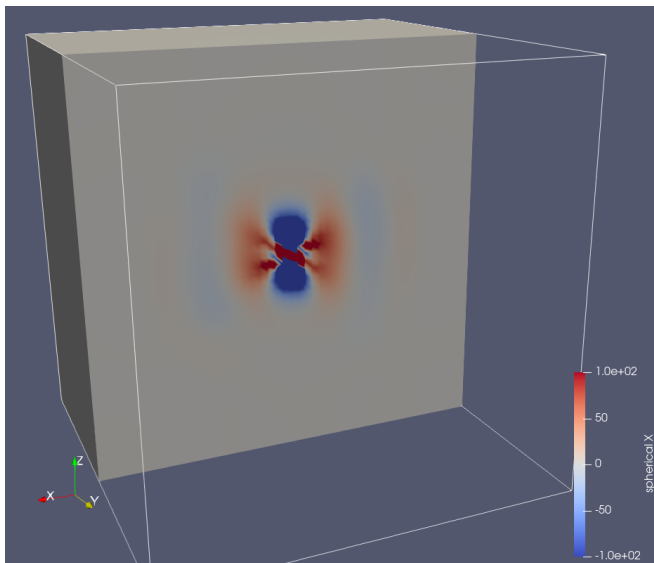
Step 80 ($t = 2 \text{ ns}$)

Results – antenna in vaccuum



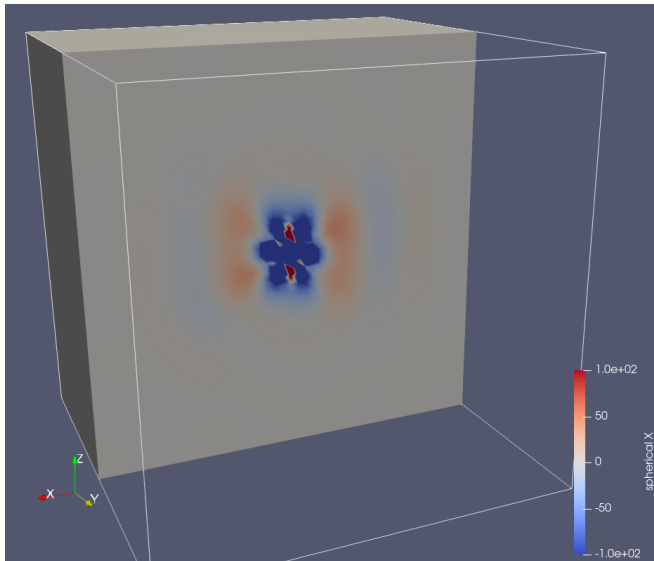
Step 90 ($t = 2.25$ ns)

Results – antenna in vaccuum



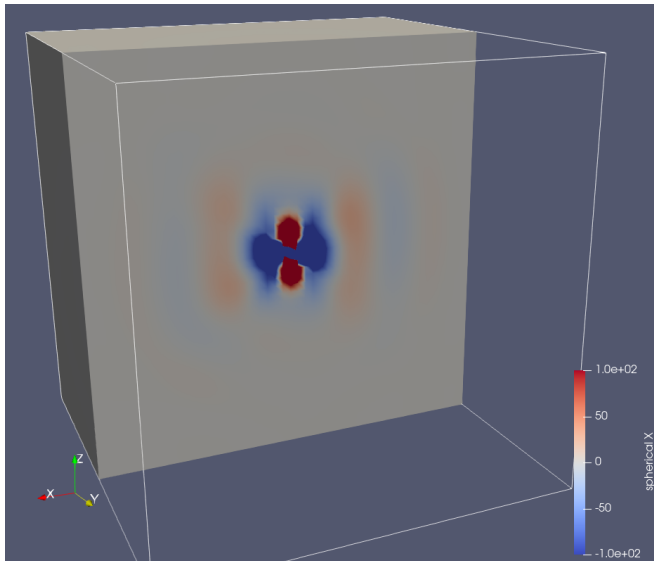
Step 100 ($t = 2.5 \text{ ns}$)

Results – antenna in vacuum



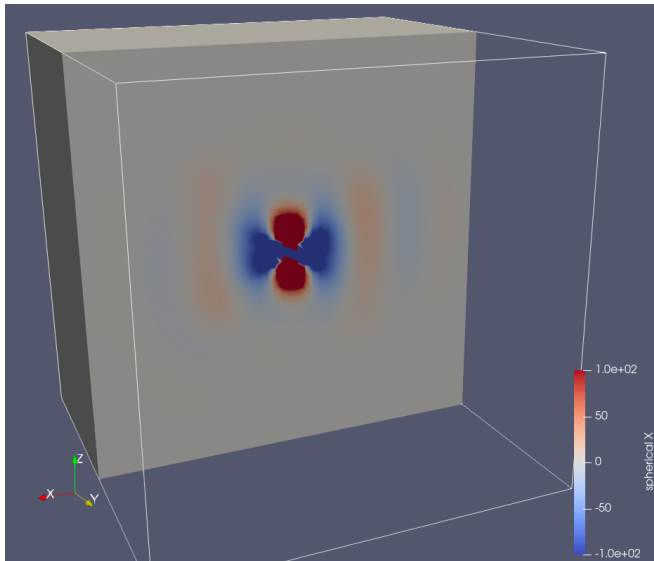
Step 110 ($t = 2.75$ ns)

Results – antenna in vaccuum



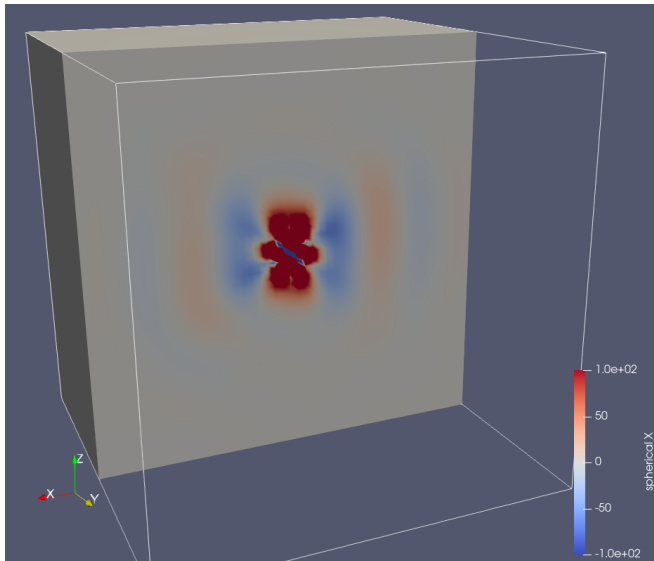
Step 120 ($t = 3 \text{ ns}$)

Results – antenna in vacuum



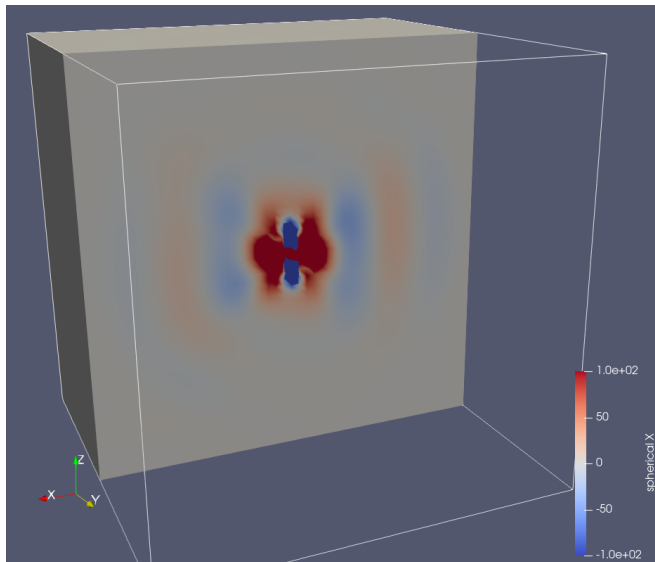
Step 130 ($t = 3.25$ ns)

Results – antenna in vacuum



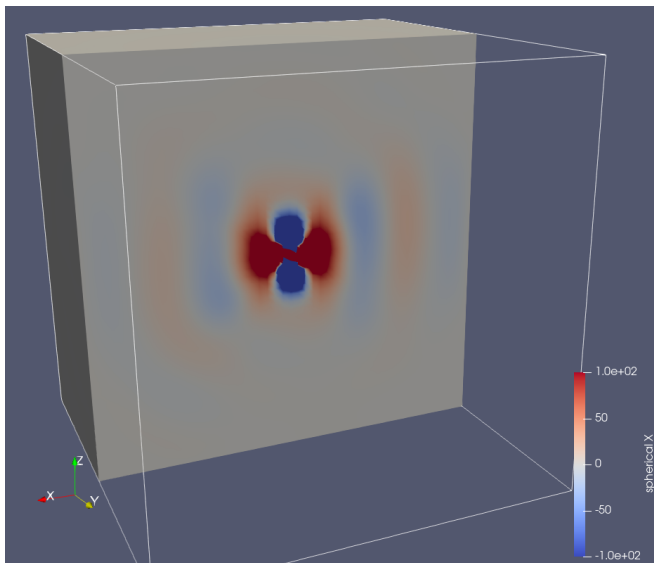
Step 140 ($t = 3.5$ ns)

Results – antenna in vaccuum



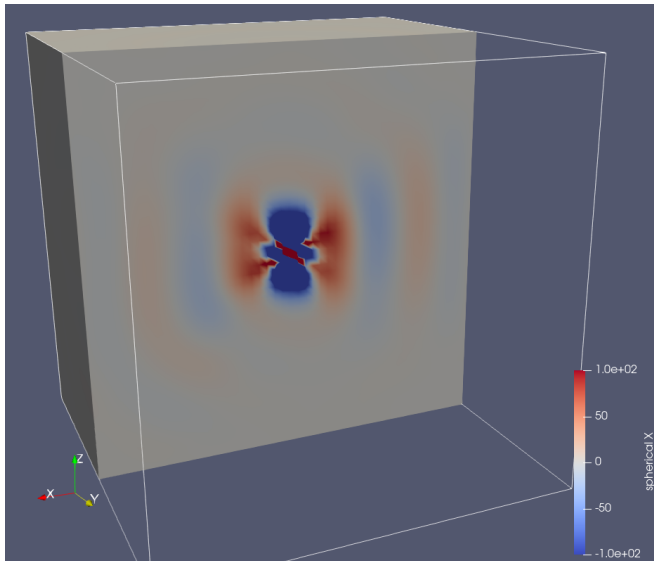
Step 150 ($t = 3.75$ ns)

Results – antenna in vaccuum



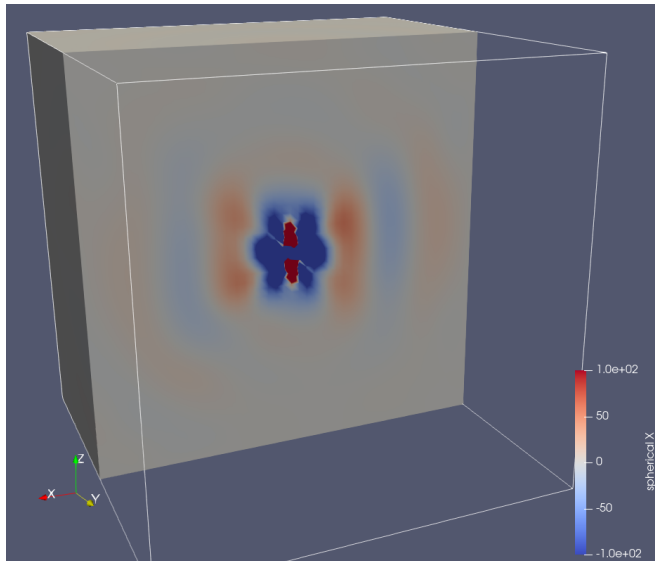
Step 160 ($t = 4 \text{ ns}$)

Results – antenna in vaccuum



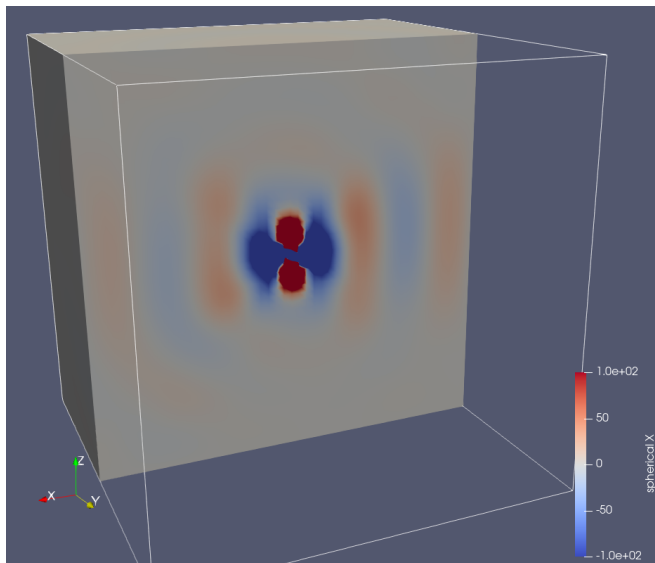
Step 170 ($t = 4.25$ ns)

Results – antenna in vaccuum



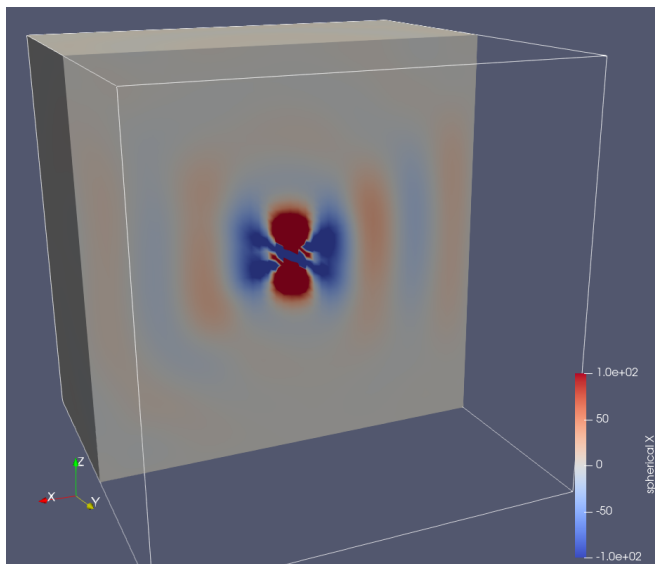
Step 180 ($t = 4.5 \text{ ns}$)

Results – antenna in vaccuum



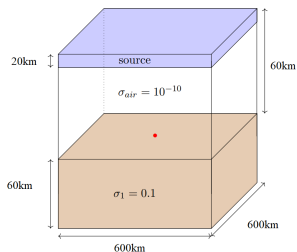
Step 190 ($t = 4.75 \text{ ns}$)

Results – antenna in vaccuum



Step 200 ($t = 5 \text{ ns}$)

Numerical example – magnetotelluric problem



$$\partial_t \mathbf{E} = \frac{1}{\epsilon_0} (\nabla \times \mathbf{H} - \sigma \mathbf{E} - \mathbf{J}_{\text{imp}})$$

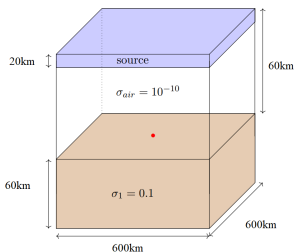
$$\partial_t \mathbf{H} = -\frac{1}{\mu_0} (\nabla \times \mathbf{E} - \mathbf{M}_{\text{imp}})$$

$$\mathbf{J}_{\text{imp}} = (1, 1, 0)\delta(t) \text{ for } t = 0$$

$$\mathbf{M}_{\text{imp}} = 0$$

$\epsilon = 0.1$ for the ground, $\epsilon = 10^{-10}$ for the air
Absorbing Boundary Condition ABC

Numerical example – magnetotelluric problem



Computational domain $[0, 600\text{km}] \times [0, 600\text{km}] \times [0, 200\text{km}]$,
Computational mesh $360 \times 360 \times 120$ results in 15,552,000 elements

Source

$[250\text{km}, 350\text{km}] \times [250\text{km}, 350\text{km}] \times [100\text{km}, 120\text{km}]$

Ground level at 60km

1000 time steps, $dt = 10^{-6}$

Numerical example – magnetotelluric problem

Numerical example – magnetotelluric problem

Numerical example – magnetotelluric problem

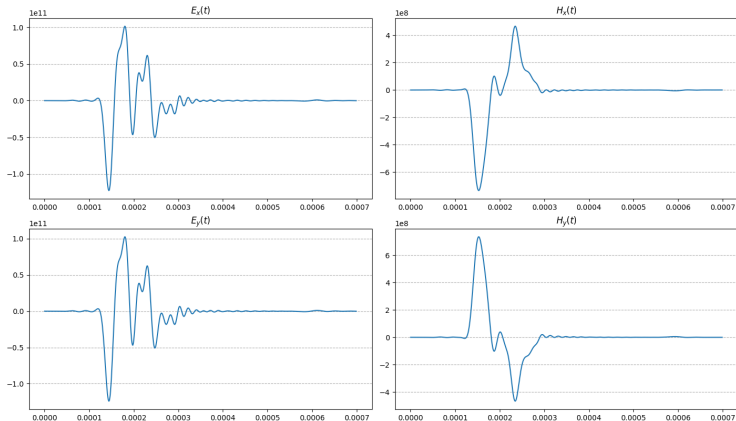


Figure: Components of electric and magnetic field as recorded at the receiver

Numerical example – magnetotelluric problem

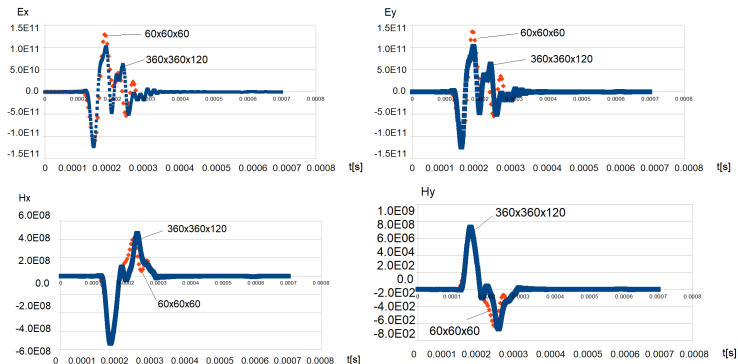


Figure: Convergence on meshes $100 \times 100 \times 60$ versus $360 \times 360 \times 120$

Integration loop – parallel version

```
s for each element  $E = [\xi_{l_x}, \xi_{l_x+1}] \times [\xi_{l_y}, \xi_{l_y+1}] \times [\xi_{l_z}, \xi_{l_z+1}]$  in parallel do
   $U^{loc} \leftarrow 0$ ;
  for each quadrature point  $\xi = (X_{k_x}, X_{k_y}, X_{k_z})$  do
     $\mathbf{x} \leftarrow \Psi_E(\xi)$ ;
     $W \leftarrow w_{k_x} w_{k_y} w_{k_z}$ ;
     $u, Du \leftarrow 0$ ;
    for  $l \in \mathcal{I}(E)$  do
       $u \leftarrow u + U_l^{(t)} \mathcal{B}_l(\xi)$ ;
       $Du \leftarrow Du + U_l^{(t)} \nabla \mathcal{B}_l(\xi)$ ;
    for  $l \in \mathcal{I}(E)$  do
       $v \leftarrow \mathcal{B}_l(\xi)$ ;
       $Dv \leftarrow \nabla \mathcal{B}_l(\xi)$ ;
       $U_l^{loc} \leftarrow U_l^{loc} + W |E| b(u, v + \Delta t F(u, Du, v, Dv))$ ;
  synchronized
    for  $l \in \mathcal{I}(E)$  do
       $U_l^{(t+1)} \leftarrow U_l^{(t+1)} + U_l^{loc}$ 
```

Implementation: `Galois::for_each`, `Galois::Runtime::LL::SimpleLock`

Execution times (1/6)

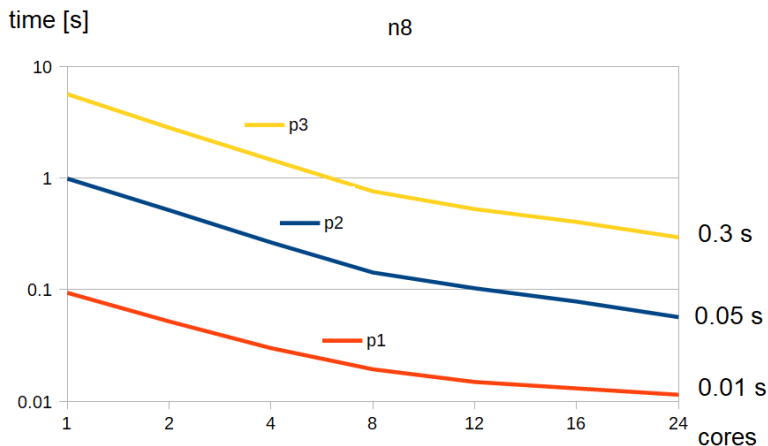


Figure: Execution time over the computational mesh of size $8 \times 8 \times 8$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Execution times (2/6)

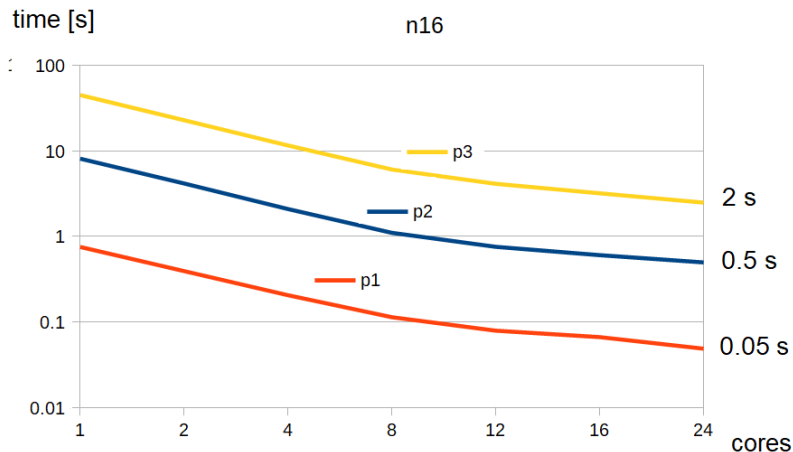


Figure: Execution time over the computational mesh of size $16 \times 16 \times 16$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Execution times (3/6)

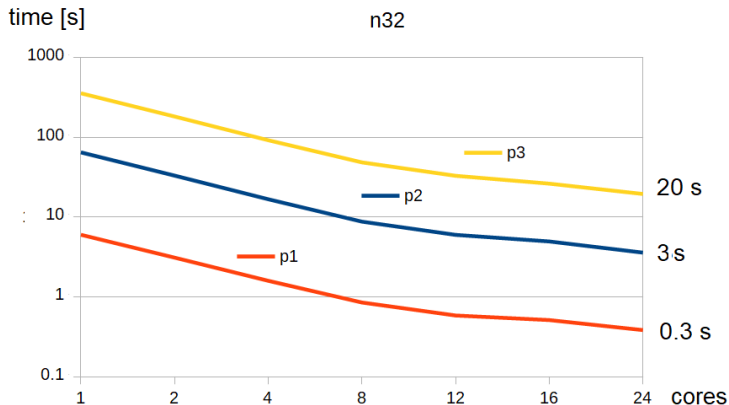


Figure: Execution time over the computational mesh of size $32 \times 32 \times 32$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Execution times (4/6)

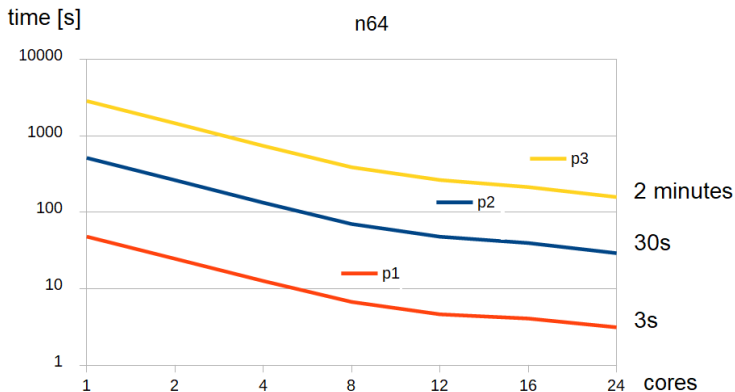


Figure: Execution time over the computational mesh of size $64 \times 64 \times 64$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Execution times (5/6)

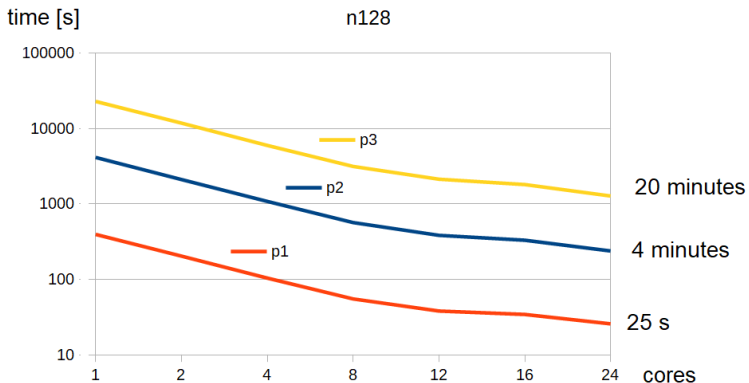


Figure: Execution time over the computational mesh of size $128 \times 128 \times 128$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Execution times (6/6)

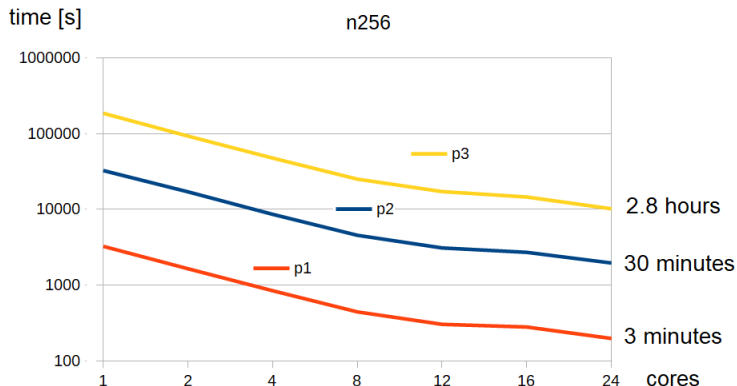


Figure: Execution time over the computational mesh of size $256 \times 256 \times 256$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Speedup (1/6)

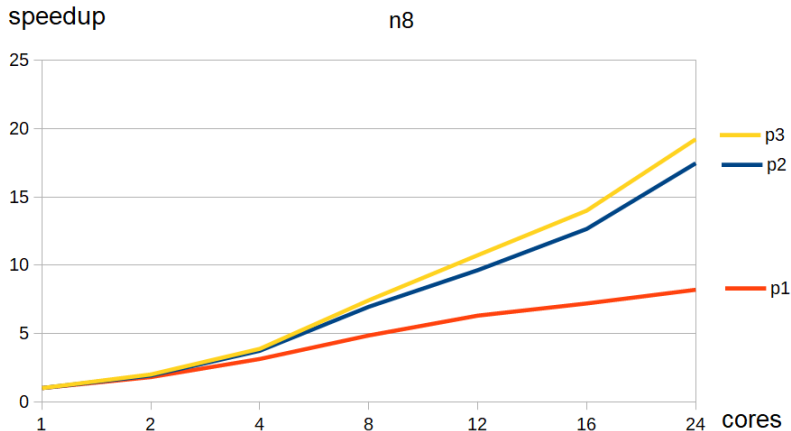


Figure: Speedup over the computational mesh of size $8 \times 8 \times 8$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Speedup (2/6)

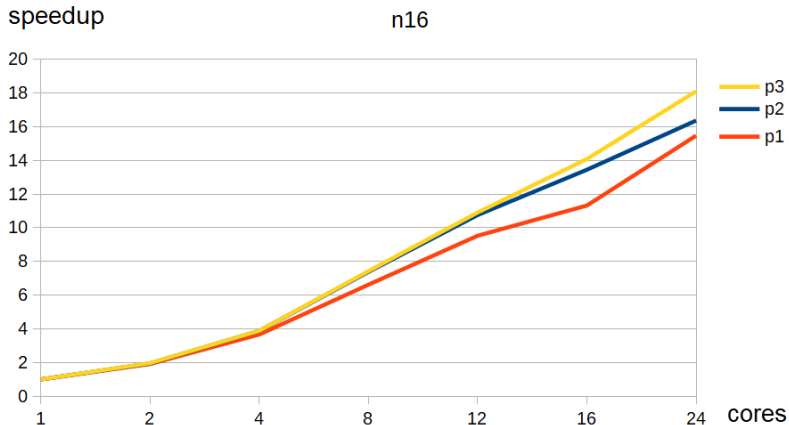


Figure: Speedup over the computational mesh of size $16 \times 16 \times 16$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Speedup (3/6)

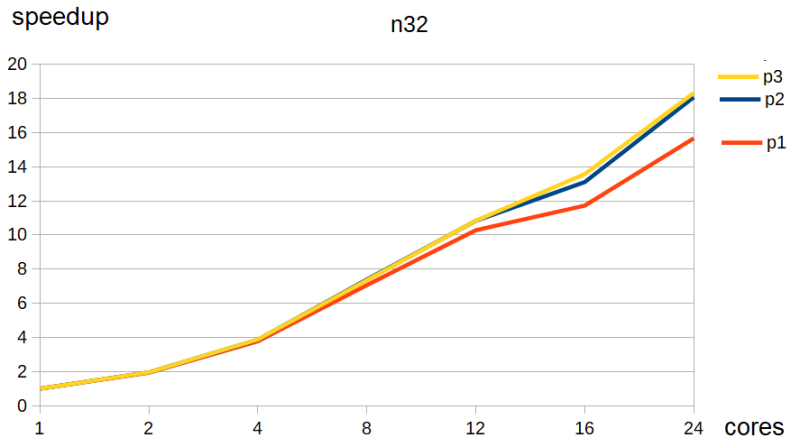


Figure: Speedup over the computational mesh of size $32 \times 32 \times 32$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Speedup (4/6)

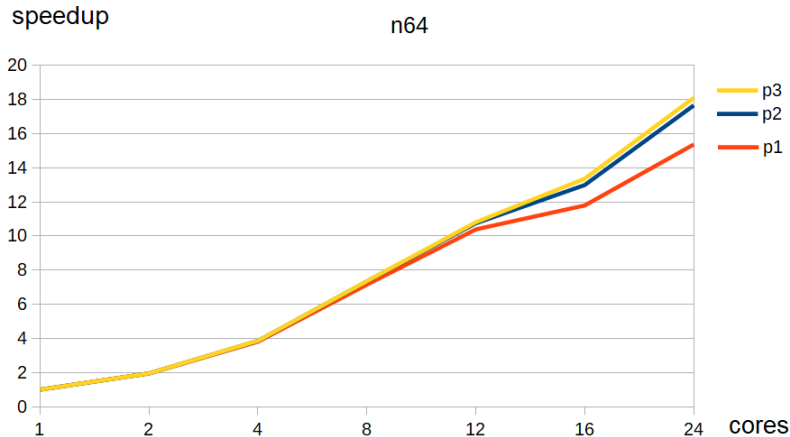


Figure: Speedup over the computational mesh of size $64 \times 64 \times 64$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Speedup (5/6)

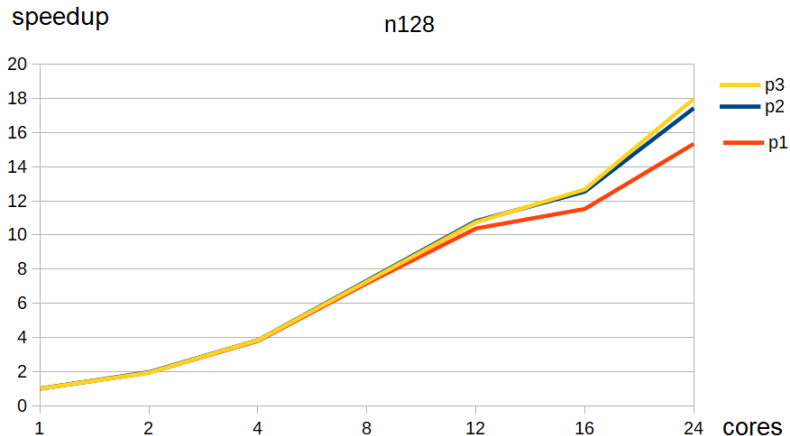


Figure: Speedup over the computational mesh of size $128 \times 128 \times 128$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Speedup (6/6)

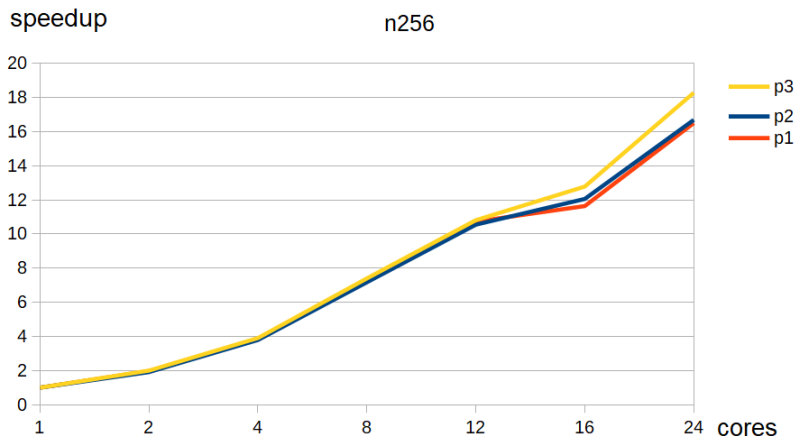


Figure: Speedup over the computational mesh of size $256 \times 256 \times 256$ elements, for a number of cores=1,2,4,8,16,24 for linear, quadratic and cubic B-splines.

Weak scalability(1/6)

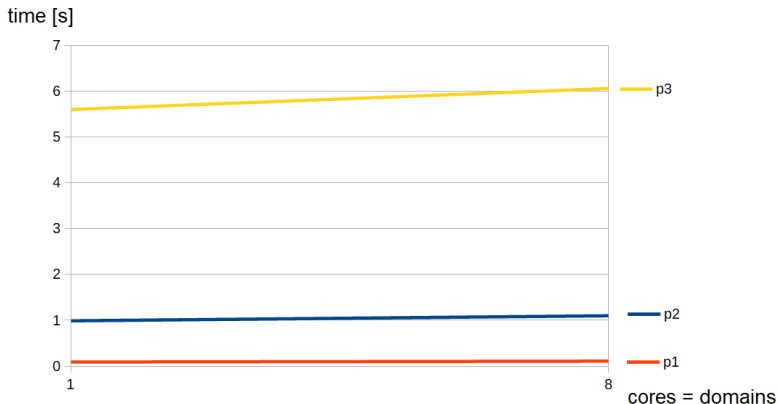


Figure: Weak scalability measured on 1 domain $8 \times 8 \times 8$ elements per 1 core versus 8 subdomains, a total of $2 \times 8 \times 2 \times 8 \times 2 \times 8$ elements per 8 cores. Measurements for linear, quadratic and cubic B-splines.

Weak scalability(2/6)

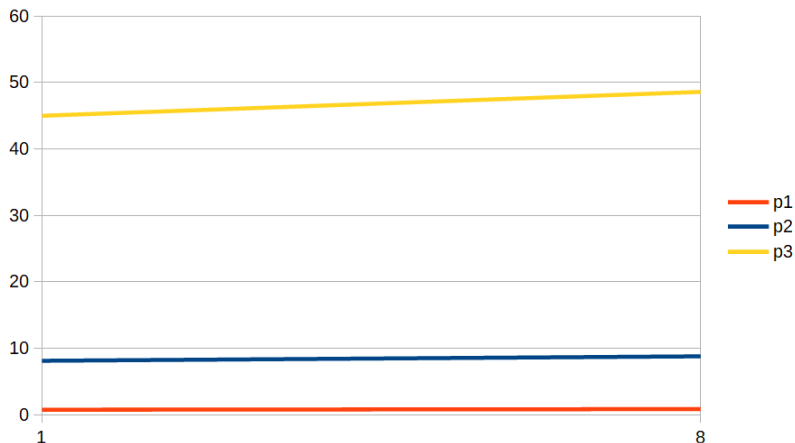


Figure: Weak scalability measured on 1 domain $16 \times 16 \times 16$ elements per 1 core versus 8 subdomains, a total of $2 \times 16 \times 2 \times 16 \times 2 \times 16$ elements per 8 cores. Measurements for linear, quadratic and cubic B-splines.

Weak scalability(3/6)

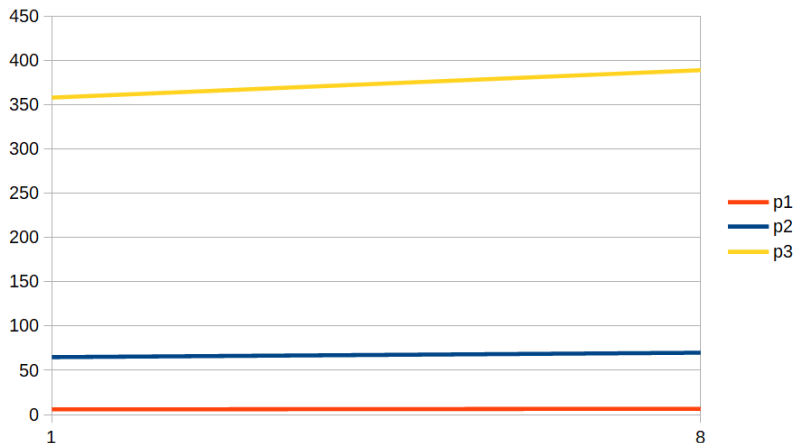


Figure: Weak scalability measured on 1 domain $32 \times 32 \times 32$ elements per 1 core versus 8 subdomains, a total of $2 \times 32 \times 2 \times 32 \times 2 \times 32$ elements per 8 cores. Measurements for linear, quadratic and cubic B-splines.

Weak scalability(4/6)

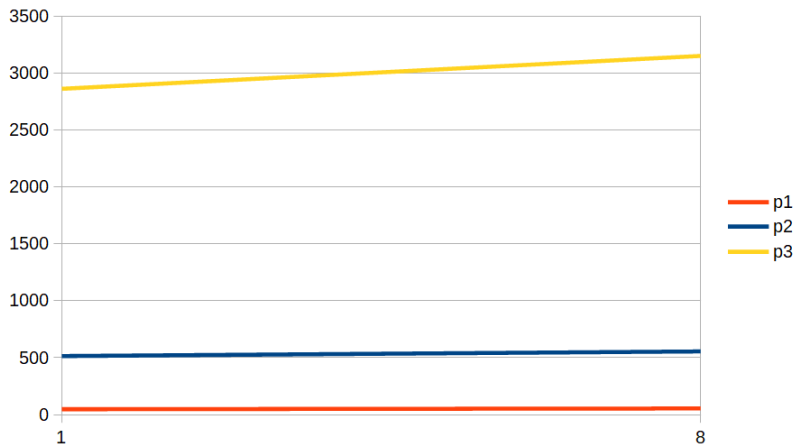


Figure: Weak scalability measured on 1 domain $64 \times 64 \times 64$ elements per 1 core versus 8 subdomains, a total of $2 \times 64 \times 2 \times 64 \times 2 \times 64$ elements per 8 cores. Measurements for linear, quadratic and cubic B-splines.

Weak scalability(5/6)

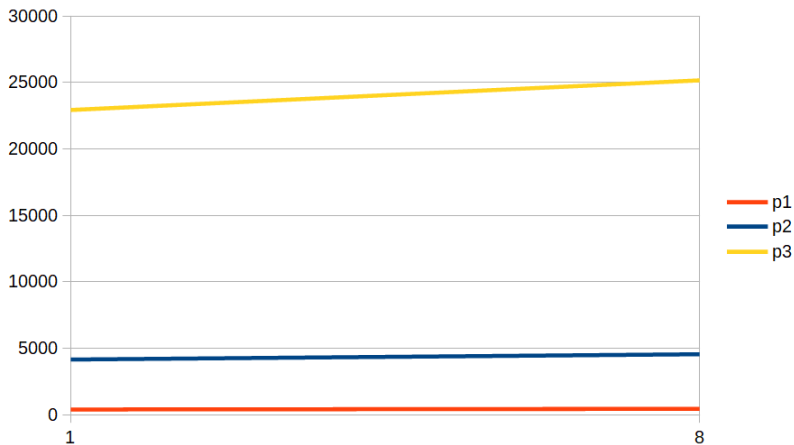


Figure: Weak scalability measured on 1 domain $128 \times 128 \times 128$ elements per 1 core versus 8 subdomains, a total of $2 \times 128 \times 2 \times 128 \times 2 \times 128$ elements per 8 cores. Measurements for linear, quadratic and cubic B-splines.

Alternating Direction Solver

Idea exploit Kronecker product structure of the matrix

Generally, consider

$$\mathbf{Lx} = \mathbf{b}$$

with $\mathbf{L} = \mathbf{A} \otimes \mathbf{B}$, where \mathbf{A} is $n \times n$, \mathbf{B} is $m \times m$

Definition of Kronecker (tensor) product:

$$\mathbf{L} = \begin{bmatrix} \mathbf{A} B_{11} & \mathbf{A} B_{12} & \cdots & \mathbf{A} B_{1m} \\ \mathbf{A} B_{21} & \mathbf{A} B_{22} & \cdots & \mathbf{A} B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A} B_{m1} & \mathbf{A} B_{m2} & \cdots & \mathbf{A} B_{mm} \end{bmatrix}$$

Alternating Direction Solver – 2D

Let

$$\mathbf{x}_i = (x_{i1}, \dots, x_{in})^T$$

$$\mathbf{b}_i = (b_{i1}, \dots, b_{in})^T$$

We can rewrite the system as a block matrix equation:

$$\begin{cases} \mathbf{A}B_{11}\mathbf{x}_1 + \mathbf{A}B_{12}\mathbf{x}_2 + \cdots + \mathbf{A}B_{1m}\mathbf{x}_m = \mathbf{b}_1 \\ \mathbf{A}B_{21}\mathbf{x}_1 + \mathbf{A}B_{22}\mathbf{x}_2 + \cdots + \mathbf{A}B_{2m}\mathbf{x}_m = \mathbf{b}_2 \\ \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ \mathbf{A}B_{m1}\mathbf{x}_1 + \mathbf{A}B_{m2}\mathbf{x}_2 + \cdots + \mathbf{A}B_{mm}\mathbf{x}_m = \mathbf{b}_m \end{cases}$$

Consider each component of \mathbf{x}_i and $\mathbf{y}_i \Rightarrow$ family of linear systems

$$\begin{cases} B_{11}x_{1i} + B_{12}x_{2i} + \cdots + B_{1m}x_{mi} = y_{1i} \\ B_{21}x_{1i} + B_{22}x_{2i} + \cdots + B_{2m}x_{mi} = y_{2i} \\ \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ B_{m1}x_{1i} + B_{m2}x_{2i} + \cdots + B_{mm}x_{mi} = y_{mi} \end{cases}$$

for each $i = 1, \dots, n$

\Rightarrow linear systems with matrix \mathbf{B}

Alternating Direction Solver – 2D

Two steps – solving systems with **A** and **B** in different *directions*

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} y_{11} & y_{21} & \cdots & y_{m1} \\ y_{12} & y_{22} & \cdots & y_{m1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & y_{mn} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{21} & \cdots & b_{m1} \\ b_{12} & b_{22} & \cdots & b_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1n} & b_{2n} & \cdots & b_{mn} \end{bmatrix}$$

$$\begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mm} \end{bmatrix} \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ x_{21} & \cdots & x_{2n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix}$$

Non-constant material coefficients

We cannot factor out μ, ε immediately:

$$(1 - \lambda \partial_1^2) E_3 \rightarrow \left(1 - \frac{\tau^2}{4} \varepsilon^{-1} \partial_1 \mu^{-1} \partial_1\right) E_3$$

Weak formulation:

$$(E_3, v) + \frac{\tau^2}{4} (\mu^{-1} \partial_1 E_3, \partial_1 (\varepsilon^{-1} v))$$

Matrix associated with the above:

$$A_{ijk,pqr} = \int_{\Omega} \left\{ B_{ijk} B_{pqr} + \frac{\tau^2}{4} \mu^{-1} \partial_1 B_{ijk} \partial_1 (\varepsilon^{-1} B_{pqr}) \right\} dx$$

Non-constant material coefficients

Idea for each test function, approximate ε , μ by a constant

$$\varepsilon \approx \varepsilon_{ijk}, \quad \mu \approx \mu_{ijk} \quad \lambda_{ijk} = \frac{\tau^2}{4 \mu_{ijk} \varepsilon_{ijk}}$$

$$\begin{aligned} \tilde{A}_{ijk,pqr} &= \int_{\Omega} \{ B_{ijk} B_{pqr} + \lambda_{ijk} \partial_1 B_{ijk} \partial_1 B_{pqr} \} dx \\ &= \underbrace{\left[(B_i^1, B_p^1) + \lambda_{ijk} ((B_i^1)', (B_p^1)') \right]}_{Q_{ip}^{jk}} \underbrace{(B_j^2, B_q^2)}_{M_{jq}} \underbrace{(B_k^3, B_r^3)}_{M_{kr}} \end{aligned}$$

Almost a Kronecker product structure:

$$\tilde{A}_{ijk,pqr} = Q_{ip}^{jk} M_{jq} M_{kr}$$

but the first matrix varies depending on the full test function index

Alternating Direction Solver – extension

Generalization of the Kronecker product structure:

$$\mathbf{L} = \begin{bmatrix} \mathbf{A} B_{11} & \mathbf{A} B_{12} & \cdots & \mathbf{A} B_{1m} \\ \mathbf{A} B_{21} & \mathbf{A} B_{22} & \cdots & \mathbf{A} B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A} B_{m1} & \mathbf{A} B_{m2} & \cdots & \mathbf{A} B_{mm} \end{bmatrix}$$

⇓

$$\mathbf{L} = \begin{bmatrix} \mathbf{A}_1 B_{11} & \mathbf{A}_1 B_{12} & \cdots & \mathbf{A}_1 B_{1m} \\ \mathbf{A}_2 B_{21} & \mathbf{A}_2 B_{22} & \cdots & \mathbf{A}_2 B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_m B_{m1} & \mathbf{A}_m B_{m2} & \cdots & \mathbf{A}_m B_{mm} \end{bmatrix}$$

Alternating Direction Solver – generalization

$$\begin{cases} B_{11}\mathbf{x}_1 + B_{12}\mathbf{x}_2 + \cdots + B_{1m}\mathbf{x}_m = \mathbf{y}_1 = \mathbf{A}_1^{-1}\mathbf{b}_1 \\ B_{21}\mathbf{x}_1 + B_{22}\mathbf{x}_2 + \cdots + B_{2m}\mathbf{x}_m = \mathbf{y}_2 = \mathbf{A}_2^{-1}\mathbf{b}_2 \\ \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ B_{m1}\mathbf{x}_1 + B_{m2}\mathbf{x}_2 + \cdots + B_{mm}\mathbf{x}_m = \mathbf{y}_m = \mathbf{A}_m^{-1}\mathbf{b}_m \end{cases}$$

$$\begin{cases} B_{11}x_{1i} + B_{12}x_{2i} + \cdots + B_{1m}x_{mi} = y_{1i} \\ B_{21}x_{1i} + B_{22}x_{2i} + \cdots + B_{2m}x_{mi} = y_{2i} \\ \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ B_{m1}x_{1i} + B_{m2}x_{2i} + \cdots + B_{mm}x_{mi} = y_{mi} \end{cases}$$

for each $i = 1, \dots, n$

Conclusions

- Efficiency begins at the level of problem formulation
- Efficient solvers may require molding the problem to obtain the right structure
- For non-stationary problems, a lot of that molding can be done on the level of designing the time-stepping scheme

- Some restrictions imposed by ADS can be (partially) lifted
- Nevertheless, some special structure is still necessary
- Applications to advection-diffusion, Maxwell equations, Navier-Stokes

Marcin Łoś, Ignacio Muga, Judit Muñoz-Matute, Maciej Paszyński, [Isogeometric Residual Minimization Method \(iGRM\) with direction splitting for non-stationary advection–diffusion problems](#)

Computers & Mathematics with Applications, 79(2) (2020)
213-229

doi.org/10.1016/j.camwa.2019.06.023

Marcin Łoś, Ignacio Muga, Judit Muñoz-Matute, Maciej Paszyński, [Isogeometric residual minimization \(iGRM\) for non-stationary Stokes and Navier–Stokes problems](#),

Computers & Mathematics with Applications, 95(1) (2021)
200-214.

doi.org/10.1016/j.camwa.2020.11.013