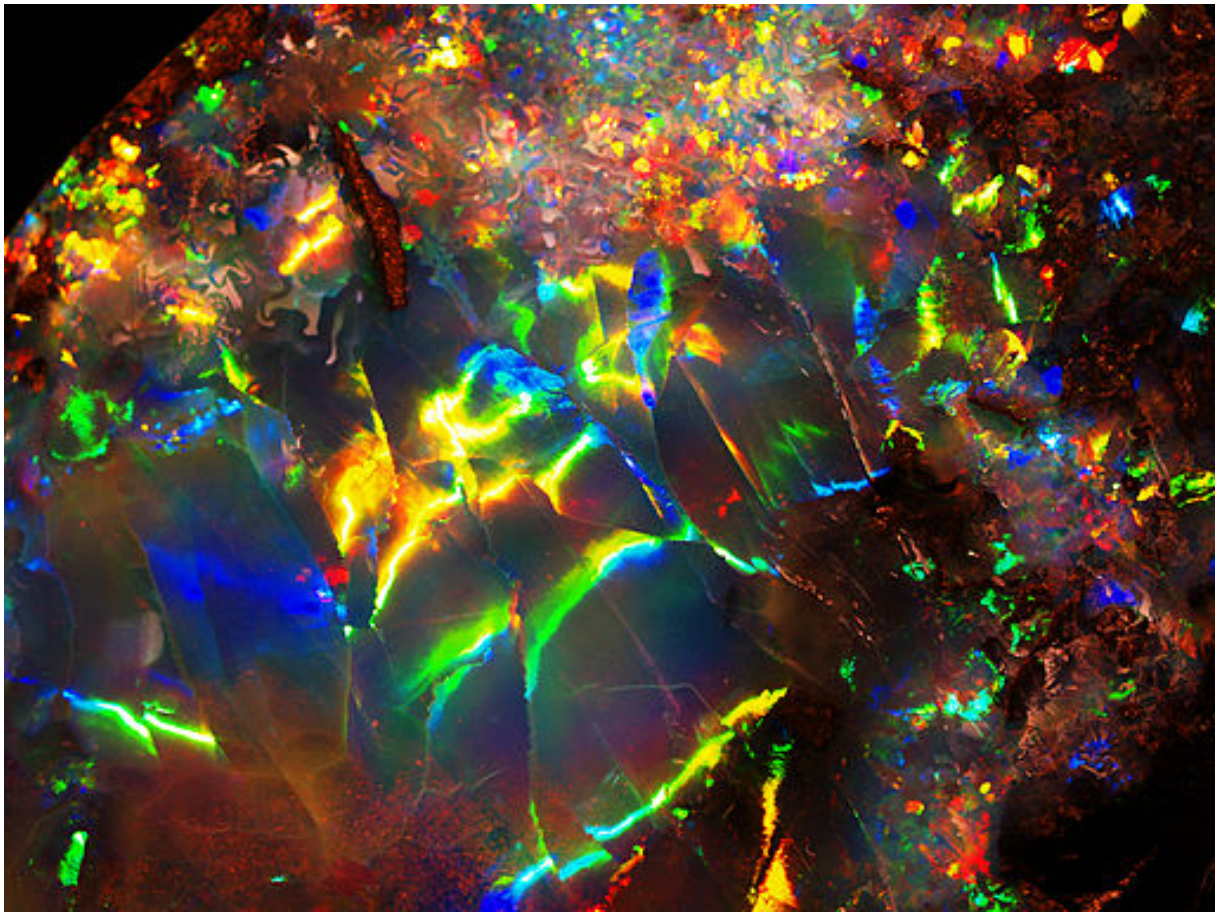
Discontinuous Galerkin Finite Element Methods for Photonic Crystals

F. Brink (s0140309)

December 5, 2013



Contents

Chapter 1

Introduction

Photonic crystals are natural or artificial periodic structures composed of two or more materials with different electromagnetic properties. If the period of the periodic structure is similar to the wavelength of an incident light beam, an interplay of refraction and interference may give rise to a photonic band gap: incident light is reflected and the amplitude of light somehow trapped inside the crystal will decay exponentially.

Applications are solar arrays, where crystals could be used to reflect incident light at the bottom of the array, fiber-optic cables, that can be 'coated' by photonic crystals to keep the light trapped inside, see for example Figure 1.1, and lasers, where a photonic crystal can be used to get a better spatial or frequential focus. If impurities are introduced similar to the impurities in semi-conductors, it is also possible to construct optical switches out of photonic crystals. An example of a photonic crystal from nature is an opal. Opals consist of densely packed spheres of silica with a diameter of a few hundred nanometers. Since this is also the frequency of visible light, a partial band gap causes different colours to reflect only at different angles of incidence. For an example, see [22] (also displayed on the front page of this report)

The size constraints on photonic crystals and their complicated structure pose some serious challenges for their synthetic production, so it is useful to be able to predict the optical properties of a photonic crystal using numerical simulations. The governing equations for the behaviour of light are the Maxwell equations. They are commonly discretised using finite differences with a leap-frog scheme for the time discretization. In the context of the Maxwell equations, this is commonly known as Yee's algorithm [32].

An alternative is to assume that the solution is harmonic in time. This removes the time derivative from the equation, replacing it with the frequency of the solution instead. The time-harmonic Maxwell equations are then solved using mode expansions, e.g. Fourier modes, which are well suited due to the periodic nature of the solution and known to be fast solution algorithms. Widely used packages like MIT Photonic Bands (MPB) [18] make use of this strategy. However, this method is also known to produce non-physical artefacts near discontinuities in the solution.

Another approach is based on Finite Element Methods [21, 28, 9]. The basic idea is to define a weak formulation of the Maxwell equations and to approximate the solution using a piecewise polynomial basis, such that each basis function is non-zero only on a finite subset of the domain. This combines high flexibility with potentially high convergence rates.

This report will be based on the research done by Sármány et al. [27] and Denissen [12]. They also use Finite Element Methods, but with the extra feature that the basis functions are allowed to be discontinuous at element boundaries. This has as a consequence that the trace of the approximate solution becomes ill-defined at element boundaries and a numerical flux has

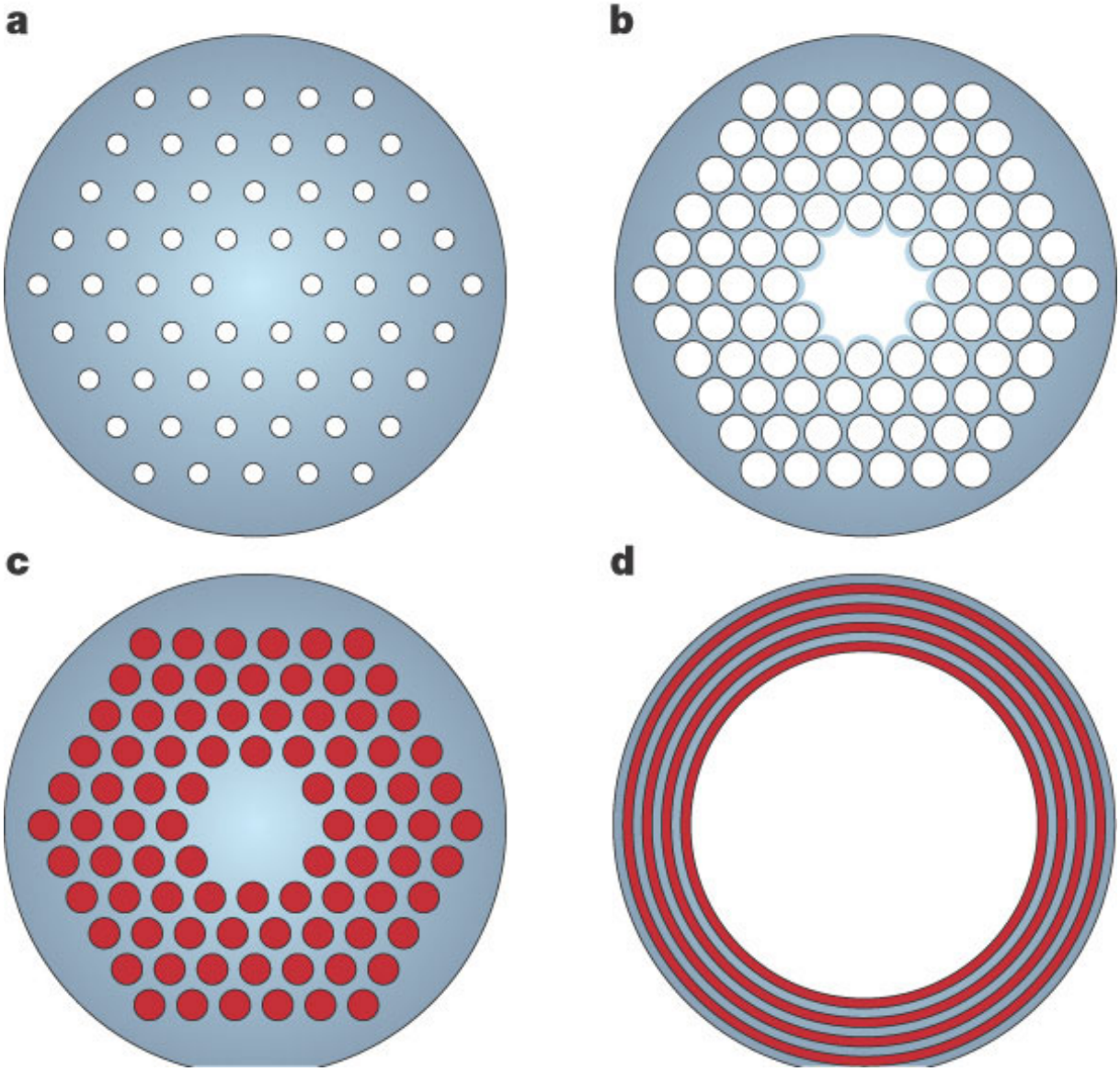


Figure 1.1: Four designs of fiber-optic cables employing photonic crystals. The white areas represent air, the blue areas represent silica and the red areas represent an unspecified high-index material. Feature sizes are typically in the nanometer ranges. Figure taken from [19].

to be introduced. The numerical flux has great impact on the stability and accuracy of the solution. For a complete discussion of numerical fluxes, see for example [3].

Using discontinuous basis functions allows for a great flexibility in constructing the mesh, in particular for locally refined meshes that are beneficial to capture important details in the solution. The weak coupling of the elements using a numerical flux allows for a very easy treatment of discontinuities in the parameters. It also gives rise to a better sparsity structure of the discrete linear system. As an added bonus it allows for a slightly easier treatment of boundary conditions.

The reason that Discontinuous Galerkin Finite Element Methods are only a recent development is that they have relatively large memory requirements compared to classical (curl)-conforming finite element methods. Moreover, the mathematical expressions involved appear complicated at first glance. Moreover, to make good use of the sparsity structure efficient algorithms have to be developed.

The remaining challenges in finding solutions to the Maxwell equations lie in the large null-space of the curl-curl operator. Additionally, to be useful in applications a numerical solver needs to find all eigenvalues and eigenfunctions of the time-harmonic Maxwell equations without any spurious -nonphysical- modes.

As a natural result from the solution strategy the dispersion relation, linking wavenumber to frequency, is obtained.

In experimental set-ups the (local) density of states (L)DOS is usually measured. This is the number of photon states available for occupation at a specific energy level in the photonic crystal. A very large DOS will enhance spontaneous emission since there are more decay paths available than usual. An LDOS of zero means no light of this energy level can exist in the crystal, so the crystal will act as a perfect mirror for the frequencies involved.

This work considers the linear Maxwell equations for non-conductive materials. Its aim will be to provide a starting point for a mathematical model and computer code that can be used to model general photonic crystals. In particular, a discretization of the Maxwell equations will be provided and the accuracy of this discretization is verified using known convergence results from literature. Moreover, the discretized equations will be applied to simple periodic structures to demonstrate how they can be used to find the dispersion relation and (L)DOS for more general structures.

The remainder of this report can be summarised as follows. Chapter 2 will provide a short review of the Maxwell equations. In Chapter 3 a weak formulation will be set up for the Maxwell equations. This formulation will be used in Chapter 4 to provide a discretization for the time-harmonic Maxwell equations. Also in Chapter 4 the infinite periodic domain will be discussed. Using Bloch-Floquet theory a reformulation will be given based on a finite unit domain and a wave-vector k. In Chapter 5 it will be explained how these results can be used to compute the LDOS and the DOS. Chapter 6 will collect all test cases and their results to verify the accuracy of the discretization. Chapter 7 will present some additional details on the implementation of the DG algorithm. A detailled description of the code can be found in Appendix A. Finally, Chapter 8 will provide conclusions and an outlook for future work.

Chapter 2

Maxwell equations

The Maxwell equations are given as a set of four coupled differential equations. Only the case of linear isotropic media will be considered in this report. In that case some simplifications can be made in the physics and the Maxwell equations are presented as

$$\begin{cases} \nabla \cdot (\epsilon E) = \rho_f & \text{(2.1a)} \\ \nabla \times E = -\frac{1}{\mu}\frac{\partial H}{\partial t} & \text{(2.1b)} \\ \nabla \cdot \left(\frac{1}{\mu} H\right) = 0 & \text{(2.1c)} \\ \nabla \times H = J + \frac{\partial (\epsilon E)}{\partial t}, & \text{(2.1d)} \end{cases}$$

on $\Omega \subseteq \mathbb{R}^3$ where

- $\nabla = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix}$ is the gradient operator,
- ϵ is the electric permittivity,
- E is the electric field,
- ρ_f is the free charge distribution,
- μ is the magnetic permeability,
- H is the magnetic field,
- J is the free current.

Note that if (2.1a) and (2.1c) hold for the initial conditions and if $\nabla \cdot J = 0$ then (2.1b) and (2.1d) will guarantee that (2.1a) and (2.1c) hold for all time. This means they are just consistency conditions. They can be dropped from the derivations, but should be kept in mind at places where the consistency might be broken.

The equations are put in dimensionless form by introducing a reference length $\tilde{L}$ and a reference magnetic field strength $\tilde{H}_0$ and the scaling

$$x = \frac{\tilde{x}}{\tilde{L}} \qquad t = \frac{c_0 \tilde{t}}{\tilde{L}} \qquad H = \frac{\tilde{H}}{\tilde{H}_0},$$

where $c_0 = \frac{1}{\sqrt{\epsilon_0 \mu_0}}$ is the speed of light and ϵ_0 and μ_0 are the electric permittivity and magnetic permeability of vacuum. Also, scaling E and J appropriately reduces (2.1) to

$$\begin{cases} \nabla \times E = -\frac{1}{\mu_r} \frac{\partial H}{\partial t} & \text{(2.2a)} \\ \nabla \times H = J + \frac{\partial (\epsilon_r E)}{\partial t}, & \text{(2.2b)} \end{cases}$$

where ϵ_r and μ_r are the relative electric permittivity and magnetic permeability, respectively. It will be more beneficial to write the coupled set of equations as one equation. For this a choice has to be made either to use E or H as the free variable in the resulting equation. Since the eigenmodes of the E-field will be needed later on for the computation of the LDOS a choice was made to use the E-field. The H-field can always be retrieved using (2.2a) and the initial conditions if it is explicitly needed.

Take the curl of (2.2a) and the time derivative of (2.2b) and substitute $\frac{\partial}{\partial t}(\nabla \times H)$ to find

$$\nabla \times \frac{1}{\mu_r} \nabla \times E + \epsilon_r \frac{\partial^2 E}{\partial t^2} = -\frac{\partial J}{\partial t}. \tag{2.3}$$

This equation, together with the perfectly conducting boundary conditions $n \times E|_{\partial\Omega} = 0$ on a domain $\Omega \subseteq \mathbb{R}^3$ describes the behaviour of the electric field. For an infinite periodic structure, for example idealised photonic crystals, $\Omega = \mathbb{R}^3$ and it will be assumed that E, ϵ_r, μ_r and J are periodic in space and that there exists a (cubic) unit domain $\tilde{\Omega}$ that can be repeated infinitely many times in all cardinal directions. The problem then becomes to find a solution on this unit domain with the proper periodicity. See also Figure 2.1.

It will also be assumed that ϵ_r and μ_r are piecewise smooth positive functions that are constant in time.

If J is an harmonic function in time with some frequency ω it is not needed to solve the full time dependent problem. Using the Anzatz $E(t,x) = \mathrm{e}^{-\mathrm{i}\omega t}\tilde{E}(x)$ (where $J(t,x) = \mathrm{e}^{-\mathrm{i}\omega t}\tilde{J}(x)$) with $\mathrm{i} = \sqrt{-1}$ we obtain

$$\nabla \times \frac{1}{\mu_r} \nabla \times E - \omega^2 \epsilon_r E = \mathrm{i}\omega J. \tag{2.4}$$

This is a partial differential equation only in space, so in general it will be easier to solve. Note that there are no initial conditions involved in this formulation so in principle the solution might not be consistent with the free charge distribution, but this depends on the method used to solve (2.4).

If $J = 0$ (2.4) can be considered as an eigenvalue problem. The challenge here becomes to find the pairs (ω, E) such that

$$\nabla \times \frac{1}{\mu_r} \nabla \times E = \omega^2 \epsilon_r E. \tag{2.5}$$

To get more insight into the eigenvalue problem (2.5), assume that E has a Helmholz decomposition

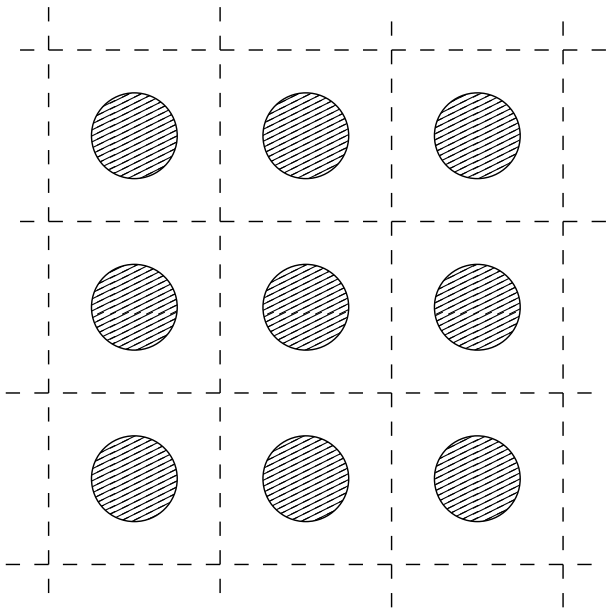


Figure 2.1: Example of a 2D photonic crystal. Displayed are 9 complete unit domains and parts of their surrounding unit domains. Domain boundaries are marked with dashed lines for emphasis.

$$E = \nabla g + \nabla \times u.$$

After applying (2.5), we obtain

$$\nabla \times \frac{1}{\mu_r} \nabla \times \nabla \times u = \omega^2 \left(\nabla g + \nabla \times u \right).$$

Taking the divergence then results in

$$0 = \omega^2 \Delta g.$$

So either $\omega = 0$ or $\Delta g = \nabla \cdot E = 0$. This means that the condition (2.1a) maps to the null-space of the curl-curl operator, so for the eigenvalue problem an additional assumption must be made that there is no free charge present. For the time harmonic formulation the free charge can also assumed to be zero. If there is a free charge present in the time harmonic problem, a correcting term can be found by solving $\Delta g = \rho_f$ and updating E accordingly.

If $\omega = 0$ and $\Delta g = 0$ then E can be linear at best. Since the eigenvalue problem is only solved on the periodic unit domain E must then be constant. One of these eigenmodes is associated with a uniform free charge density. The other two correspond to light with an infinitely long wavelength in the two polarisations it may have.

It can be concluded that the eigenvalues of the Maxwell equations without free charge in the domain are 0 (multiplicity 2) and whatever eigenvalues may be found with $\omega^2 > 0$

Chapter 3

Discontinuous Galerkin discretization

In this Chapter the weak formulation of the time-dependent Maxwell equations and the corresponding discrete linear problem will be derived.

3.1 Function spaces and notation

Given a tessellation $\mathcal{T}_h$ of Ω, such that ϵ_r and μ_r are both constant inside each element, define

$$H\left(\text{curl};\mathcal{T}_h\right)=\left\{u\in\left[L^2\left(\Omega\right)\right]^3:\nabla\times u|_{_K}\in\left[L^2\left(K\right)\right]^3\quad\forall K\in\mathcal{T}_h\right\}.$$

For some subset F of Ω, $(\cdot,\cdot)_F$ denotes the standard $L^2(F)$ inner product. Also define an element-wise gradient operator $\nabla_h|_{_K}=\nabla|_{_K}\,,\quad\forall K\in\mathcal{T}_h$.

At an interface of two elements define the tangential jump and the average of $u\in\mathbb{R}^3$ as:

$$\begin{aligned}\left[\!\left[u\right]\!\right]_{_T}&=n^L\times u^L+n^R\times u^R,\\ \left\{\!\left\{u\right\}\!\right\}&=\frac{u^L+u^R}{2},\end{aligned}$$

Where $\cdot^L$ and $\cdot^R$ denote the trace of the left element and the right element at their common boundary, respectively, and n is the outward pointing normal vector of the element at the interface. By their definition $n^L=-n^R$. The element that is considered left can be chosen arbitrarily. At the domain boundaries choose $\left\{\!\left\{u\right\}\!\right\}=u^L$ and $\left[\!\left[u\right]\!\right]_{_T}=n\times u^L$.

Denote by $\mathcal{F}_h^i$ the set of all interfaces of pairs of elements. The set of all element boundaries that coincide with domain boundaries is denoted by $\mathcal{F}_h^b$. The union of these sets is denoted by $\mathcal{F}_h$. Now we derive the following useful identity

$$
\begin{aligned}
&\sum_{K\in\mathcal{T}_h} (n\times u, v)_{\partial K} \\
&= \sum_{F\in\mathcal{F}_h^i} \int_F \left(n^L\times u^L\right)\cdot v^L + \left(n^R\times u^R\right)\cdot v^R \,\mathrm{d}s + \sum_{F\in\mathcal{F}_h^b}\int_F \left(n^L\times u^L\right)\cdot v^L\,\mathrm{d}s \\
&= \sum_{F\in\mathcal{F}_h^i} \frac{1}{2}\int_F \left(n^L\times u^L\right)\cdot v^L - \left(n^L\times v^L\right)\cdot u^L + \left(n^R\times u^R\right)\cdot v^R - \left(n^R\times v^R\right)\cdot u^R\,\mathrm{d}s \\
&+ \sum_{F\in\mathcal{F}_h^b}\int_F \left(n^L\times u^L\right)\cdot v^L\,\mathrm{d}s \\
&= \sum_{F\in\mathcal{F}_h^i} \frac{1}{2}\int_F \left(n^L\times u^L\right)\cdot v^L - \left(n^L\times v^L\right)\cdot u^L + \left(n^R\times u^R\right)\cdot v^R - \left(n^R\times v^R\right)\cdot u^R \\
&\qquad + \left(n^L\times u^L\right)\cdot v^R - \left(n^L\times u^L\right)\cdot v^R - \left(n^L\times v^L\right)\cdot u^R + \left(n^L\times v^L\right)\cdot u^R\,\mathrm{d}s \\
&+ \sum_{F\in\mathcal{F}_h^b}\int_F \left(n^L\times u^L\right)\cdot v^L\,\mathrm{d}s \\
&= \sum_{F\in\mathcal{F}_h^i} \int_F \left(n^L\times u^L\right)\cdot \{\!\{v\}\!\} - \left(n^L\times v^L\right)\cdot \{\!\{u\}\!\} \\
&\qquad + \left(n^R\times u^R\right)\cdot \{\!\{v\}\!\} - \left(n^R\times v^R\right)\cdot \{\!\{u\}\!\}\,\mathrm{d}s \\
&+ \sum_{F\in\mathcal{F}_h^b}\int_F \left(n^L\times u^L\right)\cdot v^L\,\mathrm{d}s \\
&= \sum_{F\in\mathcal{F}_h^i} \int_F [\![u]\!]_T \cdot \{\!\{v\}\!\} - [\![v]\!]_T \cdot \{\!\{u\}\!\}\,\mathrm{d}s + \sum_{F\in\mathcal{F}_h^b}\int_F [\![u]\!]_T\cdot\{\!\{v\}\!\}\,\mathrm{d}s. \qquad (3.1)
\end{aligned}
$$

Also define, implicitly, the following lifting operators:

$$
\begin{aligned}
\left(\mathcal{L}\left(u\right), v\right)_\Omega &= \sum_{F\in\mathcal{F}_h^i}\int_F u\cdot [\![v]\!]_T\,\mathrm{d}s, \\
\left(\mathcal{R}\left(u\right), v\right)_\Omega &= \sum_{F\in\mathcal{F}_h}\int_F u\cdot \{\!\{v\}\!\}\,\mathrm{d}s, \\
\left(\mathcal{R}_F\left(u\right), v\right)_\Omega &= \int_F u\cdot\{\!\{v\}\!\}\,\mathrm{d}s \qquad\qquad \forall F\in\mathcal{F}_h.
\end{aligned}
$$

3.2 Discretisation of the Maxwell equations

For the discretization of the Maxwell equations the same approach will be followed as the one followed by Sármány et al. [25, 26, 27] and Denissen [12]. Split Ω in a tessellation of tetrahedra $\mathcal{T}_h$, such that ϵ_r and μ_r are both constant inside each tetrahedron.

For the spatial discretization introduce an auxiliary variable q to turn (2.3) into the following coupled first order system

$$\begin{cases} \nabla \times \mu_r^{-1} q + \epsilon_r \dfrac{\partial^2 E}{\partial t^2} = -\dfrac{\partial J}{\partial t} & \text{(3.2a)} \\ q - \nabla \times E = 0, & \text{(3.2b)} \end{cases}$$

where $E, q \in H(\mathrm{curl}; \mathcal{T}_h)$. Now multiply (3.2) with arbitrary test functions $\phi, \pi \in H(\mathrm{curl}; \mathcal{T}_h)$ and integrate over Ω

$$\begin{cases} \left(-\dfrac{\partial J}{\partial t}, \phi\right)_\Omega = \displaystyle\sum_{K \in \mathcal{T}_h} \int_K \left(\nabla \times \mu_r^{-1} q(t,x)\right) \cdot \phi(x) + \epsilon_r \frac{\partial^2 E(t,x)}{\partial t^2} \cdot \phi(x) \, \mathrm{d}x \\ (q, \pi)_\Omega = \displaystyle\sum_{K \in \mathcal{T}_h} \int_K (\nabla \times E(t,x)) \cdot \pi(x) \, \mathrm{d}x, \end{cases}$$

After integration by parts we obtain

$$\begin{cases} \left(-\dfrac{\partial J}{\partial t}, \phi\right)_\Omega = \displaystyle\sum_{K \in \mathcal{T}_h} \int_K \mu_r^{-1} q(t,x) \cdot (\nabla \times \phi(x)) + \epsilon_r \frac{\partial^2 E(t,x)}{\partial t^2} \cdot \phi(x) \, \mathrm{d}x \\ \qquad + \displaystyle\int_{\partial K} \left(n \times \mu_r^{-1} q(t,x)\right) \cdot \phi(x) \, \mathrm{d}s & \text{(3.4a)} \\ (q, \pi)_\Omega = \displaystyle\sum_{K \in \mathcal{T}_h} \int_K E(t,x) \cdot (\nabla \times \pi(x)) \, \mathrm{d}x + \int_{\partial K} (n \times E(t,x)) \cdot \pi(x) \, \mathrm{d}s. & \text{(3.4b)} \end{cases}$$

Since $\mu_r^{-1} q$ and E have multiple valued traces in the boundary integrals we replace them with their numerical fluxes $\mu_r^{-1} q^*$ and E^* and integrate (3.4b) by parts again, resulting in

$$\begin{cases} \left(-\dfrac{\partial J}{\partial t}, \phi\right)_\Omega = \displaystyle\sum_{K \in \mathcal{T}_h} \left(\mu_r^{-1} q, \nabla \times \phi\right)_K + \left(\epsilon_r \frac{\partial^2 E}{\partial t^2}, \phi\right)_K + \left(n \times \mu_r^{-1} q^*, \phi\right)_{\partial K} \\ (q, \pi)_\Omega = \displaystyle\sum_{K \in \mathcal{T}_h} \int_K (\nabla \times E(t,x)) \cdot \pi(x) \, \mathrm{d}x \\ \qquad + \displaystyle\int_{\partial K} (n \times (E^*(t,x) - E(t,x))) \cdot \pi(x) \, \mathrm{d}s, \end{cases}$$

There is no need to replace π by π^* because, given a test function, its trace is known. Now rewrite the boundary integrals using (3.1) as

$$
\begin{cases}
\left(-\dfrac{\partial J}{\partial t},\phi\right)_\Omega = \displaystyle\sum_{K\in\mathcal{T}_h}\left(\mu_r^{-1}q,\nabla\times\phi\right)_K + \left(\epsilon_r\dfrac{\partial^2 E}{\partial t^2},\phi\right)_K + \sum_{F\in\mathcal{F}_h^b}\left(\left[\!\left[\mu_r^{-1}q^*\right]\!\right]_T,\left\{\!\left\{\phi\right\}\!\right\}\right)_F \\
\qquad\qquad + \displaystyle\sum_{F\in\mathcal{F}_h^i}\int_F \left[\!\left[\mu_r^{-1}q^*(t,x)\right]\!\right]_T\cdot\left\{\!\left\{\phi(x)\right\}\!\right\} - \left[\!\left[\phi(x)\right]\!\right]_T\cdot\left\{\!\left\{\mu_r^{-1}q^*(t,x)\right\}\!\right\}\mathrm{d}s \qquad (3.6a)\\
(q,\pi)_\Omega = \displaystyle\sum_{K\in\mathcal{T}_h}(\nabla\times E,\pi)_K + \sum_{F\in\mathcal{F}_h^b}\left(\left[\!\left[E^*-E\right]\!\right]_T,\left\{\!\left\{\pi\right\}\!\right\}\right)_F \\
\qquad\qquad + \displaystyle\sum_{F\in\mathcal{F}_h^i}\int_F \left[\!\left[E^*(t,x)-E(t,x)\right]\!\right]_T\cdot\left\{\!\left\{\pi(x)\right\}\!\right\} \\
\qquad\qquad\qquad - \left[\!\left[\pi(x)\right]\!\right]_T\cdot\left\{\!\left\{E^*(t,x)-E(t,x)\right\}\!\right\}\mathrm{d}s. \qquad (3.6b)
\end{cases}
$$

Continue now with the equation for q and introduce the lifting operators defined in Section 3.1

$$
(q,\pi)_\Omega = (\nabla_h\times E,\pi)_\Omega + \left(\mathcal{R}\left(\left[\!\left[E^*-E\right]\!\right]_T\right),\pi\right)_\Omega - \left(\mathcal{L}\left(\left\{\!\left\{E^*-E\right\}\!\right\}\right),\pi\right)_\Omega . \qquad (3.7)
$$

Since $\pi\in H(\mathrm{curl};\mathcal{T}_h)$ is arbitrary in (3.7) we obtain

$$
q = \nabla_h\times E + \mathcal{R}\left(\left[\!\left[E^*-E\right]\!\right]_T\right) - \mathcal{L}\left(\left\{\!\left\{E^*-E\right\}\!\right\}\right). \qquad (3.8)
$$

Use this to eliminate q from (3.6a)

$$
\begin{aligned}
\left(-\frac{\partial J}{\partial t},\phi\right)_\Omega &= \left(\mu_r^{-1}\left(\nabla_h\times E + \mathcal{R}\left(\left[\!\left[E^*-E\right]\!\right]_T\right) - \mathcal{L}\left(\left\{\!\left\{E^*-E\right\}\!\right\}\right)\right),\nabla_h\times\phi\right)_\Omega \\
&\quad + \left(\epsilon_r\frac{\partial^2 E}{\partial t^2},\phi\right)_\Omega + \sum_{F\in\mathcal{F}_h^b}\left(\left[\!\left[\mu_r^{-1}q^*\right]\!\right]_T,\left\{\!\left\{\phi\right\}\!\right\}\right)_F \\
&\quad + \sum_{F\in\mathcal{F}_h^i}\int_F \left[\!\left[\mu_r^{-1}q^*(t,x)\right]\!\right]_T\cdot\left\{\!\left\{\phi(x)\right\}\!\right\} - \left[\!\left[\phi(x)\right]\!\right]_T\cdot\left\{\!\left\{\mu_r^{-1}q^*(t,x)\right\}\!\right\}\mathrm{d}s,
\end{aligned}
$$

Using the relation for the lifting operators the weak formulation for generic numerical fluxes E^* and q^* can be transformed into

$$
\begin{aligned}
\left(-\frac{\partial J}{\partial t},\phi\right)_\Omega &= \left(\mu_r^{-1}\nabla_h\times E,\nabla_h\times\phi\right)_\Omega + \left(\epsilon_r\frac{\partial^2 E}{\partial t^2},\phi\right)_\Omega \\
&\quad + \sum_{F\in\mathcal{F}_h^b}\left(\left[\!\left[\mu_r^{-1}q^*\right]\!\right]_T,\left\{\!\left\{\phi\right\}\!\right\}\right)_F + \left(\left[\!\left[E^*-E\right]\!\right]_T,\left\{\!\left\{\mu_r^{-1}\nabla_h\times\phi\right\}\!\right\}\right)_F \\
&\quad + \sum_{F\in\mathcal{F}_h^i}\int_F \left[\!\left[\mu_r^{-1}q^*(t,x)\right]\!\right]_T\cdot\left\{\!\left\{\phi(x)\right\}\!\right\} - \left[\!\left[\phi(x)\right]\!\right]_T\cdot\left\{\!\left\{\mu_r^{-1}q^*(t,x)\right\}\!\right\} \\
&\qquad\qquad + \left[\!\left[E^*(t,x)-E(t,x)\right]\!\right]_T\cdot\left\{\!\left\{\mu_r^{-1}\nabla_h\times\phi(x)\right\}\!\right\} \\
&\qquad\qquad - \left\{\!\left\{E^*(t,x)-E(t,x)\right\}\!\right\}\cdot\left[\!\left[\mu_r^{-1}\nabla_h\times\phi(x)\right]\!\right]_T\mathrm{d}s. \qquad (3.9)
\end{aligned}
$$

3.3 Choosing a numerical flux

Before the definition of the weak formulation is completed the numerical flux still has to be chosen. This will act as the representation of E and q at element boundaries. So it is important that the flux is chosen such that, if E and $\mu_r^{-1}q$ are smooth enough, $E^* = E$ and $\mu_r^{-1}q^* = \mu_r^{-1}q$. Moreover, to prevent violating (2.1a) or (2.1c) during a time step they should be chosen such that $\nabla \cdot (\mu_r^{-1}\nabla \times E) = 0$ everywhere.

3.3.1 Interior penalty flux

With this in mind introduce the numerical fluxes that correspond to the interior penalty (IP) numerical flux

$$\begin{cases} E^* = \{\!\{E\}\!\} & \mu_r^{-1}q^* = \{\!\{\mu_r^{-1}\nabla \times E\}\!\} - a_F[\![\mu_r^{-1}E]\!]_T & \text{for internal faces,} \\ n \times E^* = 0 & \mu_r^{-1}q^* = \{\!\{\mu_r^{-1}\nabla \times E\}\!\} - a_F[\![\mu_r^{-1}E]\!]_T & \text{for boundary faces,} \end{cases}$$

where a_F can be chosen such that there are no spurious modes and the numerical discretization is stable. Insert these relations into (3.9) to obtain the space discretization according to the IP numerical flux

$$\begin{aligned} \left(-\frac{\partial J}{\partial t}, \phi\right)_\Omega \approx{}& \left(\mu_r^{-1}\nabla_h \times E, \nabla_h \times \phi\right)_\Omega + \left(\epsilon_r \frac{\partial^2 E}{\partial t^2}, \phi\right)_\Omega \\ &+ \sum_{F \in \mathcal{F}_h^b} \left([\![\{\!\{\mu_r^{-1}\nabla \times E\}\!\} - a_F[\![\mu_r^{-1}E]\!]_T]\!]_T, \{\!\{\phi\}\!\}\right)_F + \left([\![\{\!\{E\}\!\} - E]\!]_T, \{\!\{\mu_r^{-1}\nabla_h \times \phi\}\!\}\right)_F \\ &+ \sum_{F \in \mathcal{F}_h^i} \int_F [\![\{\!\{\mu_r^{-1}\nabla \times E(t,x)\}\!\} - a_F[\![\mu_r^{-1}E(t,x)]\!]_T]\!]_T \cdot \{\!\{\phi(x)\}\!\} \\ &\qquad - [\![\phi(x)]\!]_T \cdot \{\!\{\{\!\{\mu_r^{-1}\nabla_h \times E(t,x)\}\!\} - a_F[\![\mu_r^{-1}E(t,x)]\!]_T\}\!\} \\ &\qquad + [\![\{\!\{E(t,x)\}\!\} - E(t,x)]\!]_T \cdot \{\!\{\mu_r^{-1}\nabla_h \times \phi(x)\}\!\} \\ &\qquad - \{\!\{\{\!\{E(t,x)\}\!\} - E(t,x)\}\!\} \cdot [\![\mu_r^{-1}\nabla_h \times \phi(x)]\!]_T \,\mathrm{d}s. \end{aligned}$$

Eliminate the double jump and average operators to obtain the weak formulation for the IP-flux

$$\begin{aligned} \left(-\frac{\partial J}{\partial t}, \phi\right)_\Omega \approx{}& \left(\mu_r^{-1}\nabla_h \times E, \nabla_h \times \phi\right)_\Omega + \left(\epsilon_r \frac{\partial^2 E}{\partial t^2}, \phi\right)_\Omega \\ &+ \sum_{F \in \mathcal{F}_h^i} \int_F -[\![\phi(x)]\!]_T \cdot \{\!\{\mu_r^{-1}\nabla_h \times E(t,x)\}\!\} - [\![E(t,x)]\!]_T \cdot \{\!\{\mu_r^{-1}\nabla_h \times \phi(x)\}\!\} \\ &\qquad + a_F[\![\phi(x)]\!]_T \cdot [\![\mu_r^{-1}E(t,x)]\!]_T \,\mathrm{d}s - \sum_{F \in \mathcal{F}_h^b} \left([\![E]\!]_T, \{\!\{\mu_r^{-1}\nabla_h \times \phi\}\!\}\right)_F. \end{aligned} \tag{3.10}$$

3.3.2 Numerical flux in Brezzi formulation

As an alternative to the IP-flux it is also possible to define the numerical flux as

$$\begin{cases} E^* = \{\!\{E\}\!\} & \mu_r^{-1}q^* = \{\!\{\mu_r^{-1}q\}\!\} - \eta_F\{\!\{\mu_r^{-1}\mathcal{R}_F([\![E]\!]_T)\}\!\} & \text{for internal faces,} \\ n \times E^* = 0 & \mu_r^{-1}q^* = \{\!\{\mu_r^{-1}q\}\!\} - \eta_F\{\!\{\mu_r^{-1}\mathcal{R}_F([\![E]\!]_T)\}\!\} & \text{for boundary faces,} \end{cases} \tag{3.11}$$

where η_F can be related to the number of faces, which ensures that the numerical discretization is stable. This has the advantage that a stable choice for η_f is easier to choose than a stable choice for a_F, especially for higher order elements. Insert (3.11) into (3.9) to obtain the spatial discretization according to the Brezzi-flux (remember the definition of q from (3.8)).

$$\begin{aligned}\left(-\frac{\partial J}{\partial t},\phi\right)_\Omega \approx{}& \left(\mu_r^{-1}\nabla_h\times E,\nabla_h\times\phi\right)_\Omega+\left(\epsilon_r\frac{\partial^2 E}{\partial t^2},\phi\right)_\Omega\\ &+\sum_{F\in\mathcal{F}_h^b}\left(\left[\!\left\{\mu_r^{-1}q\right\}\!\right]-\eta_F\left[\!\left\{\mu_r^{-1}\mathcal{R}_F\left(\left[E\right]_T\right)\right\}\!\right]_T,\left\{\!\left\{\phi\right\}\!\right\}\right)_F+\left(\left[\!\left\{E\right\}\!\right]-E\right]_T,\left\{\!\left\{\mu_r^{-1}\nabla_h\times\phi\right\}\!\right\}\right)_F\\ &+\sum_{F\in\mathcal{F}_h^i}\int_F\left[\!\left\{\mu_r^{-1}\nabla_h\times E\left(t,x\right)+\mu_r^{-1}\mathcal{R}\left(\left[\!\left\{E\left(t,x\right)\right\}\!\right]-E\left(t,x\right)\right]_T\right)\right.\\ &\qquad-\mu_r^{-1}\mathcal{L}\left(\left\{\!\left\{E\left(t,x\right)\right\}\!\right\}-E\left(t,x\right)\right\}\right)\!\right\}-\eta_F\left[\!\left\{\mu_r^{-1}\mathcal{R}_F\left(\left[E\left(t,x\right)\right]_T\right)\right\}\!\right]_T\cdot\left\{\!\left\{\phi\left(x\right)\right\}\!\right\}\\ &\qquad+\left[\phi\left(x\right)\right]_T\cdot\left[\!\left\{\eta_F\left[\!\left\{\mu_r^{-1}\mathcal{R}_F\left(\left[E\left(t,x\right)\right]_T\right)\right\}\!\right]-\left\{\!\left\{\mu_r^{-1}\nabla_h\times E\left(t,x\right)\right\}\!\right\}\right.\right.\\ &\qquad+\mu_r^{-1}\mathcal{R}\left(\left[\!\left\{E\left(t,x\right)\right\}\!\right]-E\left(t,x\right)\right]_T\right)-\mu_r^{-1}\mathcal{L}\left(\left\{\!\left\{E\left(t,x\right)\right\}\!\right\}-E\left(t,x\right)\right\}\right)\!\right\}\!\right\}\\ &\qquad+\left[\!\left\{E\left(t,x\right)\right\}\!\right]-E\left(t,x\right)\right]_T\cdot\left\{\!\left\{\mu_r^{-1}\nabla_h\times\phi\left(x\right)\right\}\!\right\}\\ &\qquad\left.-\left\{\!\left\{\left\{\!\left\{E\left(t,x\right)\right\}\!\right\}-E\left(t,x\right)\right\}\cdot\left[\mu_r^{-1}\nabla_h\times\phi\left(x\right)\right]_T\,\mathrm{d}s,\right.\end{aligned}$$

Again eliminate the double jump and average operators and reorder the terms

$$\begin{aligned}\left(-\frac{\partial J}{\partial t},\phi\right)_\Omega \approx{}& \left(\mu_r^{-1}\nabla_h\times E,\nabla_h\times\phi\right)_\Omega+\left(\epsilon_r\frac{\partial^2 E}{\partial t^2},\phi\right)_\Omega\\ &+\sum_{F\in\mathcal{F}_h^i}\int_F-\left[E\left(t,x\right)\right]_T\cdot\left\{\!\left\{\mu_r^{-1}\nabla_h\times\phi\left(x\right)\right\}\!\right\}-\left[\phi\left(x\right)\right]_T\cdot\left\{\!\left\{\mu_r^{-1}\nabla_h\times E\left(t,x\right)\right\}\!\right\}\\ &\qquad+\left(n_F+\eta_F\right)\left(\mu_r^{-1}\mathcal{R}_F\left(\left[\phi\left(x\right)\right]_T\right)\right)\cdot\mathcal{R}_F\left(\left[E\left(t,x\right)\right]_T\right)\,\mathrm{d}s\\ &-\sum_{F\in\mathcal{F}_h^b}\left(\left[E\right]_T,\left\{\!\left\{\mu_r^{-1}\nabla_h\times\phi\right\}\!\right\}\right)_F,\end{aligned}\tag{3.12}$$

where n_F is the number of faces per tetrahedron.

3.4 Finite elements

We now introduce the finite element $(\mathcal{K},\mathcal{P},\mathcal{N})$. Let $\mathcal{K}$ be an element in a tessellation of tetrahedra $\mathcal{T}_h$. For the rest of this section, a tetrahedron $\mathcal{K}\in\mathcal{T}_h$ has vertices $\{v_1,v_2,v_3,v_4\}$, edges $\{e_{12},e_{13},e_{14},e_{23},e_{24},e_{34}\}$, such that e_{ij} has adjacent vertices v_i and v_j. The element $\mathcal{K}$ has faces $\{F_1,F_2,F_3,F_4\}$. Number the faces such that F_i is opposite to v_i, e_{ij} has tangent vector τ_{ij} pointing from i to j and F_i has an outward pointing normal n_i. For a graphical overview see Figure 3.1. Also define the barycentric coordinates as

$$\lambda_i=\begin{cases}1, & x=v_i\\ 0, & x\in F_i,\end{cases},$$

such that λ_i is linear. For an easy construction of a hierarchical basis we use Legendre polynomials. For $x\in[-1,1]$ they are recursively defined as follows

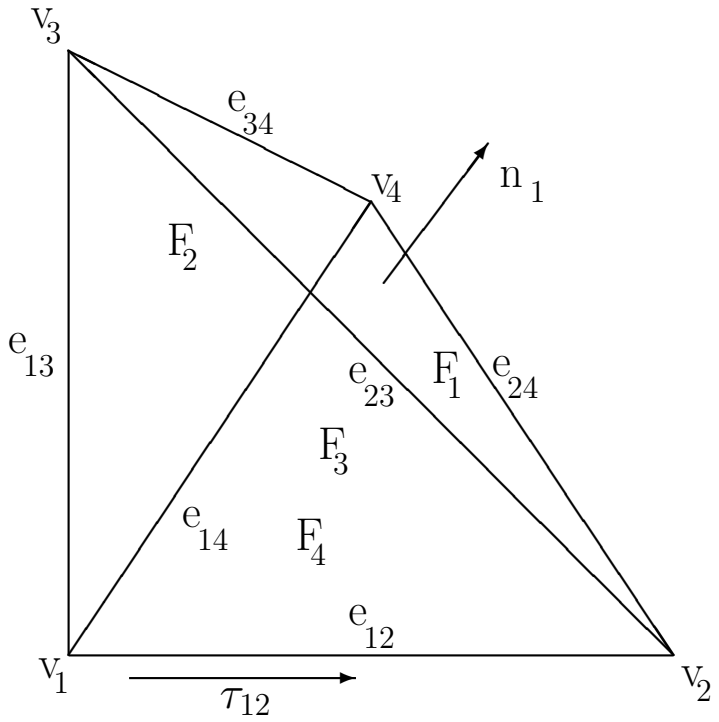


Figure 3.1: Finite element with all vertices, edges and faces marked. The vectors τ_{12} and n_1 are also drawn.

$$\begin{cases} L_0(x) = 1 \\ L_1(x) = x \\ L_p(x) = \dfrac{2p-1}{p} x L_{p-1}(x) - \dfrac{p-1}{p} L_{p-2}(x) \quad p \geq 2. \end{cases}$$

Let $\mathcal{P}_p \subseteq H(\text{curl}; \mathcal{T}_h)$ be the space of polynomials of order p in $\mathbb{R}^3$ and let $\mathcal{N}$ be its dual space.

Introduce the following set of hierarchic basis functions of H (curl) conforming finite elements proposed by Ainsworth and Coyle [2]. We consider the following cases:

edges

- $\phi_{0,AB}(x) = \lambda_A(x) \nabla\lambda_B(x) - \lambda_B(x) \nabla\lambda_A(x)$
- $\phi_{1,AB}(x) = -\lambda_A(x) \nabla\lambda_B(x) - \lambda_B(x) \nabla\lambda_A(x)$
- $\phi_{i,AB}(x) = \frac{2i-1}{i} L_{i-1}(\lambda_B(x) - \lambda_A(x)) \phi_{1,A,B}(x) - \frac{i-1}{i} L_{i-2}(\lambda_B(x) - \lambda_A(x)) \phi_{0,A,B}(x)$ $2 \leq i \leq p$

edge based faces

- $\phi_{i,e,D,AB}(x) = \lambda_A(x) \lambda_B(x) L_i(\lambda_B(x) - \lambda_A(x)) \nabla\lambda_C(x) \quad 0 \leq i \leq p-2$

faces

- $\phi_{0,D,l,m}(x) = \lambda_A(x) \lambda_B(x) \lambda_C(x) L_l(\lambda_B(x) - \lambda_A(x)) L_m(\lambda_C(x) - \lambda_A(x)) (v_B - v_A)$ $0 \leq l \leq l+m \leq p-3$

- $\phi_{1,D,l,m}(x) = \lambda_A(x)\lambda_B(x)\lambda_C(x) L_l(\lambda_B(x) - \lambda_A(x)) L_m(\lambda_C(x) - \lambda_A(x))(v_C - v_A)$
  $0 \le l \le l+m \le p-3$

face based interior

- $\phi_{F,D,l,m}(x) = \lambda_A(x)\lambda_B(x)\lambda_C(x) L_l(\lambda_B(x) - \lambda_A(x)) L_m(\lambda_C(x) - \lambda_A(x)) \nabla\lambda_D(x)$
  $0 \le l \le l+m \le p-3$

interior

- $\phi_{K,k,l,m,n}(x) =$
  $\lambda_0(x)\lambda_1(x)\lambda_2(x)\lambda_3(x) L_l(\lambda_1(x) - \lambda_0(x)) L_m(\lambda_2(x) - \lambda_0(x)) L_n(\lambda_3(x) - \lambda_0(x)) v_k$
  $0 \le l \le l+m \le l+m+n \le p-4$

Next, we show that the hierarchic basis based on the $H(\mathrm{curl})$ conforming finite elements of Ainsworth and Coyle is a basis for $\mathcal{P}_p$. For all x we have

$$\begin{aligned}
&\sum_{i,A,B} w_{i,A,B}\phi_{i,A,B}(x) \\
&+ \sum_{i,D,AB} w_{i,e,D,AB}\phi_{i,e,D,AB}(x) \\
&+ \sum_{D,l,m} \left(w_{0,D,l,m}\phi_{0,D,l,m}(x) + w_{1,D,l,m}\phi_{1,D,l,m}(x)\right) \\
&+ \sum_{D,l,m} w_{F,D,l,m}\phi_{F,D,l,m}(x) \\
&+ \sum_{k,l,m,n} w_{K,k,l,m,n}\phi_{K,k,l,m,n}(x) = 0. \qquad (3.13)
\end{aligned}$$

Consider the edge AB with $A < B$. Note that $\nabla\lambda_i \cdot \tau_{AB}$ is 1 if $i = A$ or $i = B$ and 0 otherwise. Also λ_i is 0 on edge AB unless $i = A$ or $i = B$. So by restricting (3.13) to the edge AB and multiplying it with τ_{AB} (3.13) turns into

$$\sum_i w_{i,A,B}\phi_{i,A,B}(x)\Big|_{AB} \cdot \tau_{AB} = 0.$$

The edge functions are linearly independent hence it follows that $w_{i,A,B} = 0$. Now consider the face F_j. We have the relations $\nabla\lambda_j \times n_j = 0$ and $\lambda_j = 0$ on F_j, so restrict (3.13) to F_j and take the cross product with n_j to obtain

$$\sum_{i,AB} w_{i,D,AB}\phi_{i,D,AB}(x)\Big|_{F_j} \times n_j + \sum_{l,m} \left(w_{0,D,l,m}\phi_{0,D,l,m}(x)\,|_{F_j} + w_{1,D,l,m}\phi_{1,D,l,m}(x)\,|_{F_j}\right) \times n_j = 0.$$

The edge based face functions are linearly independent so restrict again to the edge AB to show that $w_{i,D,AB} = 0$. It is now easy to see that also $w_{0,D,l,m} = 0$ and $w_{1,D,l,m} = 0$.

Now still restrict (3.13) to F_j, but without taking the cross product with n_j, to see that

$$\sum_{l,m} w_{F,D,l,m}\phi_{F,D,l,m}(x)\,|_{F_j} = 0,$$

and $w_{F,D,l,m} = 0$. This leaves

$$\sum_{k,l,m,n} w_{K,k,l,m,n}\phi_{K,k,l,m,n}(x) = 0.$$

The interior bubble functions are also independent, so $w_{K,k,l,m,n} = 0$. It can be known that $[\mathcal{P}_p(K)]^3$ has dimension $\frac{(p+1)(p+2)(p+3)}{2}$. The linearly independent set proposed by Ainsworth and Coyle is a basis of $[\mathcal{P}_p(K)]^3$ with dimension

$$\begin{aligned}&6(p+1) + 4*3(p-1) + 4(p-1)(p-2) + 2(p-1)(p-2) + \frac{(p-1)(p-2)(p-3)}{2}\\ =&\frac{(p+1)(p+2)(p+3)}{2},\end{aligned}$$

so indeed the proposed hierarchic basis is a basis for $[\mathcal{P}_p(K)]^3$.

3.5 Discontinuous Galerkin discretization of the Maxwell equations

Now express, for every element in $\mathcal{T}_h$, the unknown field $E(t,x)$ as an expansion into the basis functions

$$E(t,x) \approx \sum_{i=1}^{N_p} E_i(t)\,\phi_i(x) \quad \forall x \in \mathcal{K} \quad \forall \mathcal{K} \in \mathcal{T}_h. \tag{3.14}$$

First consider a discretization using the Brezzi-flux. Insert the basis function expansion (3.14) into (3.12) to get

$$\begin{aligned}\left(-\frac{\partial J}{\partial t}, \phi_j\right)_\Omega = &\sum_{i=1}^{N_p} \frac{\partial^2 E_i}{\partial t^2} \sum_{K\in\mathcal{T}_h} (\epsilon_r\phi_i, \phi_j)_K + \sum_{i=1}^{N_p} E_i \Bigg(\sum_{K\in\mathcal{T}_h} \left(\mu_r^{-1}\nabla\times\phi_i, \nabla\times\phi_j\right)_K\\ &+ \sum_{F\in\mathcal{F}_h^i} \int_F -[\![\phi_i(x)]\!]_T \cdot \{\!\{\nabla\times\phi_j(x)\}\!\} - \{\!\{\mu_r^{-1}\nabla\times\phi_i(x)\}\!\}\cdot[\![\phi_j(x)]\!]_T\\ &\qquad + (n_F+\eta_F)\left(\mu_r^{-1}\mathcal{R}_F([\![\phi_i(x)]\!]_T)\right)\cdot\mathcal{R}_F([\![\phi_j(x)]\!]_T)\,\mathrm{d}s\\ &- \sum_{F\in\mathcal{F}_h^b} \left([\![\phi_i]\!]_T, \{\!\{\mu_r^{-1}\nabla\times\phi_j\}\!\}\right)\Bigg) \quad j = 1,\ldots,N_p.\end{aligned}$$

Similarly, for a discretization using the IP-flux, insert the basis function expansion (3.14) into (3.10) to get

$$\left(-\frac{\partial J}{\partial t},\phi_j\right)_\Omega = \sum_{i=1}^{N_p}\frac{\partial^2 E_i}{\partial t^2}\sum_{K\in\mathcal{T}_h}(\epsilon_r\phi_i,\phi_j)_K + \sum_{i=1}^{N_p}E_i\Bigg(\sum_{K\in\mathcal{T}_h}\left(\mu_r^{-1}\nabla\times\phi_i,\nabla\times\phi_j\right)_K$$
$$+\sum_{F\in\mathcal{F}_h}\int_F -[\![\phi_i(x)]\!]_T\cdot\{\!\{\nabla\times\phi_j(x)\}\!\} - \{\!\{\mu_r^{-1}\nabla\times\phi_i(x)\}\!\}\cdot[\![\phi_j(x)]\!]_T$$
$$+a_F[\![\mu_r^{-1}\phi_i(x)]\!]_T\cdot[\![\phi_j(x)]\!]_T\,\mathrm{d}s$$
$$-\sum_{F\in\mathcal{F}_h^b}\left([\![\phi_i]\!]_T,\{\!\{\mu_r^{-1}\nabla\times\phi_j\}\!\}\right)\Bigg)\quad j=1,\ldots,N_p.$$

In both cases these relations constitute a system of linear ODEs

$$\mathcal{M}\frac{\partial^2\vec{E}}{\partial t^2}+\mathcal{S}\vec{E}=\vec{J}, \tag{3.15}$$

for suitable choices of $\mathcal{M}$, $\mathcal{S}$ and $\vec{J}$. Equation (3.15) can be solved using any stable and consistent time-stepping method. Actually doing this is left to the reader as the time-dependent formulation is not the main concern of this report. (But see the notes on implementation.)

3.6 Periodic boundary conditions

Up to this point everything was derived for homogeneous boundary conditions. However, an important class of applications, viz. photonic crystals, has an infinite, but periodic, domain. A periodic domain, in this context, is taken to mean a domain where all given functions are periodic in every spatial direction. In this case E is also assumed to be periodic and the computational domain Ω is chosen such that the periods of the parameters and E fit nicely into Ω. On the top boundary of Ω cell faces are linked to cell faces of the bottom boundary to mimic the periodic behaviour. The same is done in the other spatial directions.

There are no explicit dependencies on x in the face contributions to $\mathcal{S}$ so it is safe to consider these linked boundary faces as if they are internal faces in the discretization.

Chapter 4

Time-harmonic formulation

For the time-harmonic Maxwell equations the assumption is made that the solution to (2.3) and the source term are of the form

$$E(t,x) = \mathrm{e}^{-\mathrm{i}\omega t}\bar{E}(x),$$
$$J(t,x) = \mathrm{e}^{-\mathrm{i}\omega t}\bar{J}(x).$$

This reduces (2.3) to

$$\nabla \times \mu_r^{-1} \nabla \times \bar{E} - \omega^2 \epsilon_r \bar{E} = \mathrm{i}\omega \bar{J}. \tag{4.1}$$

Notice that the derivation in the previous chapter is independent of the temporal contribution of E so we can immediately write down the discrete system of equations

$$\left(\mathcal{S} - \omega^2 \mathcal{M}\right) \vec{\bar{E}} = \vec{\bar{J}}, \tag{4.2}$$

Where $\vec{\bar{J}} = \left\{\left(\mathrm{i}\omega\bar{J}, \phi_i\right)_\Omega\right\}$ and $\mathcal{M}$ and $\mathcal{S}$ remain unchanged.

4.1 Solving linear systems of equations

For solving linear systems of equations several methods are available. In this section we consider the general linear problem $Ax = b$ instead of the specific problem $\left(\mathcal{S} - \omega^2\mathcal{M}\right)\vec{\bar{E}} = \vec{\bar{J}}$. For sparse systems of equations we use a Krylov subspace method.

In the Krylov subspace method we construct the matrix $V_k = \left(v_1,\ v_2,\ v_3,\ \ldots\ , v_k\right)$, where v_k forms a basis of the Krylov subspace. The goal is then to find stationary values of $(AV_k y - b)^T B (AV_k y - b)$ for some matrix B. The choice of B depends on the specific Krylov method and B will rarely be needed explicitly. This means that a solution y_k of

$$V_k^T A^T B A V_k y_k = V_k^T A^T B b \tag{4.3}$$

has to be found. This is viable because the dimension of this problem is much smaller than the dimension of the original problem. The dimension k of the Krylov subspace is increased iteratively until either $Ax - b$ is small enough or some other stopping criterion is reached.

This means a Krylov subspace method can be completely described by a strategy to construct v_{k+1} and a choice of B. For this work the minres algorithm was used. In the remainder of this

section a description of minres will be provided. First, a choice for B will be made and the strategy to construct v_{k+1} will be given. Then, it is shown that in the case of minres the vectors v_i are orthonormal. Finally, a construction strategy of x_{k+1} will be provided that makes use of the iterative nature and saves a lot of double work. The derivation presented here will be based on the derivation presented in [24].

Minres is based on the choice $B = I$, $v_1 = \frac{b}{\|b\|}$ and

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1},$$

where $\alpha_j = v_j^T A v_j$, $\beta_1 = \|b\|$, $v_0 = 0$ and $\beta_{j+1} \geq 0$ is chosen such that $\|v_{j+1}\| = 1$. This construction guarantees that the v_j are normal. Now show the v_j are orthogonal with the help of the inproducts

$$\begin{aligned}
\beta_{j+1} v_{j+1} \cdot v_j =& v_j^T A v_j - \alpha_j ||v_j|| - \beta_j v_j \cdot v_{j-1} = -\beta_j v_j \cdot v_{j-1}, \\
\beta_{j+1} v_{j+1} \cdot v_{j-1} =& v_j^T A v_{j-1} - \alpha_j v_j \cdot v_{j-1} - \beta_j ||v_{j-1}|| \\
=& v_j^T \left(\beta_j v_j + \alpha_{j-1} v_{j-1} + \beta_{j-1} v_{j-2}\right) - \alpha_j v_j \cdot v_{j-1} - \beta_j ||v_{j-1}|| \\
=& \beta_{j-1} v_j \cdot v_{j-2} + \left(\alpha_{j-1} - \alpha_j\right) v_j \cdot v_{j-1}, \\
\beta_{j+1} v_{j+1} \cdot v_{j-k} =& v_j^T \left(\beta_{j-k+1} v_{j-k+1} + \alpha_{j-k} v_{j-k} + \beta_{j-k} v_{j-k-1}\right) \\
& - \alpha_j v_j \cdot v_{j-k} - \beta_j v_{j-1} \cdot v_{j-k}, \quad k > 1.
\end{aligned}$$

So by induction v_{j+1} is orthogonal to each of $v_1, ..., v_j$. Their construction guarantees that the v_j have unit length. This means they are also orthonormal.

The construction procedure for v_{j+1} can be rewritten as

$$A v_j = \beta_j v_{j-1} + \alpha_j v_j + \beta_{j+1} v_{j+1}, \quad \forall 0 \leq j \leq k.$$

Equivalently, using matrices

$$A V_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T,$$

where

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ & \ddots & \ddots & \ddots \\ & & \beta_k & \alpha_k \end{pmatrix}.$$

By definition $\beta_1 v_1 = b$ and thanks to the orthonormality $V_k^T V_k = I_k$, the k by k identity matrix. With this information the system (4.3) can be rewritten into

$$\begin{cases} T_k^T T_k y_k + \beta_{k+1}^2 e_k e_k^T y_k = \beta_1 T_k e_1, \\ x_k = V_k y_k. \end{cases}$$

The matrix $T_k^T T_k y_k + \beta_{k+1}^2 e_k e_k^T$ is penta-diagonal and symmetric, so solving this linear system is easy. Moreover, every iteration only introduces five new entries in the lower right part of this matrix while leaving the rest of the matrix unchanged. The dense linear system solver is programmed to make use of this advantage. Knowing x_k the residual $A x_k - b$ can be computed to use as a stopping criterion.

An additional advantage of this strategy is that it is known from the time dependent problem that the operator $S-\omega^2 M$ will not introduce a non-zero divergence into the numerical solution. The algorithm presented here is based on repeated application of $S-\omega^2 M$ to the right hand side J, so if J is divergence free the obtained solution will also be divergence free.

4.2 Eigenvalue problem and the k-shifted formulation

If $\bar{J}=0$ in (4.1) then $\mathcal{S}\bar{E}=\omega^2\mathcal{M}\bar{E}$. This is a generalized eigenproblem in ω^2. Knowing the eigenvalues can be advantageous in finding time dependent solutions. The eigenvalues can also be used to find the band structure of a photonic crystal.

There is, however, one problem with finding this band structure. These computations require a solution to the periodic formulation. So, necessarily any eigenmode found will fit in the computational domain chosen to model the real domain. To circumvent this problem drop the assumption that the periodicity of E fits in the computational domain and instead assume

$$\bar{E}\left(x\right)=\mathrm{e}^{\mathrm{i}k\cdot x}\tilde{E}\left(x\right), \tag{4.5}$$

for some vector k, where $\tilde{E}$ is periodic such that the period fits in the domain. For a 1D example where the k-shifted formulation is necessary, see Figure 4.1.

Bloch-Floquet theory is used to analyse equation (4.5). By construction $\tilde{\Omega}$ is the fundamental domain of a Bravais lattice. More specifically there are three linearly independent vectors r_1, r_2 and r_3 that generate the lattice

$$\Lambda=\left\{\sum_{j=1}^{3}l_j r_j | l_1,l_2,l_3\in\mathbb{Z}\right\},$$

such that $\mathbb{R}^3=\bigcup\limits_{a\in\Lambda}\tilde{\Omega}+a$, where the shifted domains overlap only at the boundaries. Construct reciprocal vectors $\hat{r}$ such that $r_i\cdot\hat{r_j}=2\pi\delta_{ij}$ and the reciprocal lattice

$$\hat{\Lambda}=\left\{\sum_{j=1}^{3}l_j \hat{r_j} | l_1,l_2,l_3\in\mathbb{Z}\right\}.$$

If $\hat{a}\in\hat{\Lambda}$ and $a\in\Lambda$, then $\hat{a}\cdot a=2n\pi$, $n\in\mathbb{Z}$ so

$$\mathrm{e}^{\mathrm{i}(k+\hat{a})\cdot x}\tilde{E}\left(x\right)=\mathrm{e}^{\mathrm{i}(k\cdot x+\hat{a}\cdot x+\hat{a}\cdot a)}\tilde{E}\left(x\right)=\mathrm{e}^{\mathrm{i}k\cdot x}\mathrm{e}^{\mathrm{i}\hat{a}\cdot(x+a)}\tilde{E}\left(x+a\right).$$

So k is equivalent to $k+\hat{a}$ and (4.5) needs only be solved for the Brillouin zone, the set of all k closer to the origin than to any other $\hat{a}\in\Lambda$.

Before reformulating most of the equations discussed in Chapter 3, define some new operators

$$\begin{aligned}\nabla_k\times g&=(\mathrm{i}k+\nabla)\times g=\mathrm{i}k\times g+\nabla\times g,\\ \nabla_k\cdot g&=(\mathrm{i}k+\nabla)\cdot g=\mathrm{i}k\cdot g+\nabla\cdot g,\\ \nabla_k f&=(\mathrm{i}k+\nabla)\,f=\mathrm{i}kf+\nabla f,\end{aligned}$$

for all scalar functions f and all vector functions g. Also, derive a version of Stokes' theorem

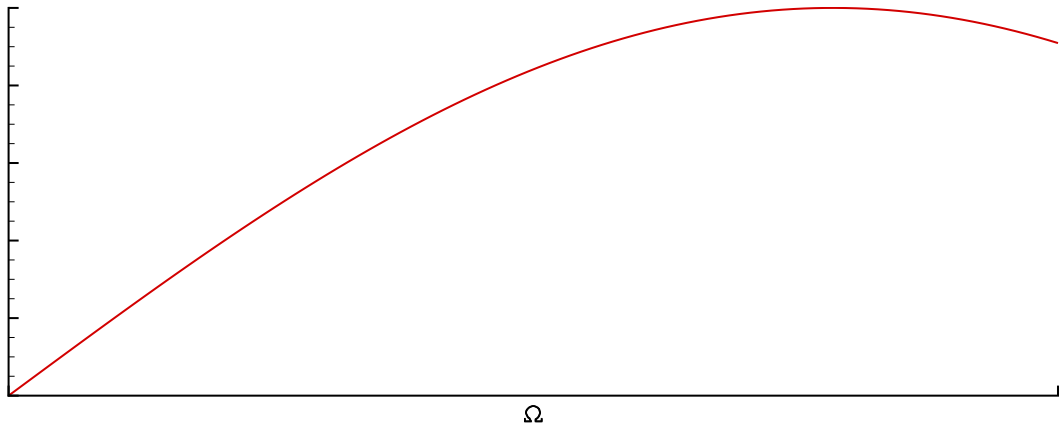


Figure 4.1: Periodic function with period π, while the domain width is only 1. To accommodate this function a frequency shift is required.

$$
\begin{aligned}
(\nabla_k \times g, \phi)_\Omega &= \int_\Omega (\nabla_k \times g(x)) \cdot \overline{\phi(x)}\, \mathrm{d}x \\
&= \int_\Omega (\mathrm{i}k \times g(x)) \cdot \overline{\phi(x)} + (\nabla \times g(x)) \cdot \overline{\phi(x)}\, \mathrm{d}x \\
&= \int_\Omega g(x) \cdot \overline{\mathrm{i}k \times \phi(x)} + g(x) \cdot \overline{\nabla \times \phi(x)}\, \mathrm{d}x + \int_{\partial\Omega} (n \times g(x)) \cdot \overline{\phi(x)}\, \mathrm{d}s \\
&= (g, \nabla_k \times \phi)_\Omega + (n \times g, \phi)_{\partial\Omega}\,,
\end{aligned}
$$

where the overbar denotes complex conjugation. With these tools in hand we transform the time-harmonic Maxwell equation into.

$$
\begin{aligned}
0 &= \nabla \times \mu_r^{-1} \nabla \times \mathrm{e}^{\mathrm{i}k\cdot x} \tilde{E}(x) - \omega^2 \epsilon_r \mathrm{e}^{\mathrm{i}k\cdot x} \tilde{E}(x) \\
&= \nabla \times \mu_r^{-1} \mathrm{e}^{\mathrm{i}k\cdot x} \nabla_k \times \tilde{E}(x) - \omega^2 \epsilon_r \mathrm{e}^{\mathrm{i}k\cdot x} \tilde{E}(x) \\
&= \nabla_k \times \mu_r^{-1} \nabla_k \times \tilde{E}(x) - \omega^2 \epsilon_r \tilde{E}(x)\,. \qquad (4.6)
\end{aligned}
$$

Next, we introduce the auxiliary variable q

$$
\begin{cases}
\left(\nabla_k \times \mu_r^{-1} q, \phi\right)_\Omega = \omega^2 \left(\epsilon_r \tilde{E}, \phi\right)_\Omega \\
\left(\nabla_k \times \tilde{E}, \pi\right)_\Omega = (q, \pi)_\Omega\,.
\end{cases}
$$

integrate by parts

$$\begin{cases} \sum_{K\in\mathcal{T}_h} \left(\mu_r^{-1}q, \nabla_k\times\phi\right)_K + \left(n\times\mu_r^{-1}q,\phi\right)_{\partial K} = \omega^2\left(\epsilon_r\tilde{E},\phi\right)_\Omega \\ \sum_{K\in\mathcal{T}_h} \left(\tilde{E}, \nabla_k\times\pi\right)_K + \left(n\times\tilde{E},\pi\right)_{\partial K} = (q,\pi)_\Omega\,, \end{cases}$$

Introduce q^* and E^* and integrate by parts again

$$\begin{cases} \sum_{K\in\mathcal{T}_h} \left(\mu_r^{-1}q, \nabla_k\times\phi\right)_K + \left(n\times\mu_r^{-1}q^*,\phi\right)_{\partial K} = \omega^2\left(\epsilon_r\tilde{E},\phi\right)_\Omega \\ \sum_{K\in\mathcal{T}_h} \left(\nabla_k\times\tilde{E}, \pi\right)_K + \left(n\times\left(\tilde{E}^*-\tilde{E}\right),\pi\right)_{\partial K} = (q,\pi)_\Omega\,. \end{cases}$$

The introduction of k doesn't change the face contributions, so from here we skip ahead to the weak formulation. Using a wave vector is only relevant on a periodic domain, so the boundary faces disappear. The weak formulation then becomes

$$\begin{aligned} \omega^2\left(\epsilon_r\tilde{E},\phi\right)_\Omega = &\left(\nabla_{k,h}\times\tilde{E}, \nabla_{k,h}\times\phi\right)_\Omega \\ &+ \sum_{F\in\mathcal{F}_h}\int_F [\![\mu_r^{-1}q^*(x)]\!]_T\cdot\overline{\{\!\{\phi(x)\}\!\}} - \{\!\{\mu_r^{-1}q^*(x)\}\!\}\cdot\overline{[\![\phi(x)]\!]_T} \\ &\qquad + [\![\tilde{E}^*(x)-\tilde{E}(x)]\!]_T\cdot\overline{\{\!\{\nabla_k\times\phi(x)\}\!\}} \\ &\qquad - \{\!\{\tilde{E}^*(x)-\tilde{E}(x)\}\!\}\cdot\overline{[\![\nabla_k\times\phi(x)]\!]_T}\,\mathrm{d}s. \end{aligned}$$

For the IP-flux use as numerical flux $\mu_R^{-1}q^* = \{\!\{\mu_r^{-1}\nabla_k\times\tilde{E}\}\!\} - a_F[\![\mu_r^{-1}\tilde{E}]\!]_T$ and we obtain the weak formulation

$$\begin{aligned} \omega^2\left(\epsilon_r\tilde{E},\phi\right)_\Omega = &\left(\nabla_{k,h}\times\tilde{E}, \nabla_{k,h}\times\phi\right)_\Omega \\ &+ \sum_{F\in\mathcal{F}_h}\int_F -\{\!\{\mu_r^{-1}\nabla_k\times\tilde{E}(x)\}\!\}\cdot\overline{[\![\phi(x)]\!]_T} - [\![\tilde{E}(x)]\!]_T\cdot\overline{\{\!\{\nabla_k\times\phi(x)\}\!\}} \\ &\qquad + a_F[\![\mu_r^{-1}\tilde{E}(x)]\!]_T\cdot\overline{[\![\phi(x)]\!]_T}\,\mathrm{d}s. \end{aligned} \tag{4.10}$$

For the Brezzi-flux the numerical flux defined in Section 3.3.2 is suitable, resulting in

$$\begin{aligned} \omega^2\left(\epsilon_r\tilde{E},\phi\right)_\Omega = &\left(\nabla_{k,h}\times\tilde{E}, \nabla_{k,h}\times\phi\right)_\Omega \\ &+ \sum_{F\in\mathcal{F}_h}\int_F -\{\!\{\mu_r^{-1}\nabla_k\times\tilde{E}(x)\}\!\}\cdot\overline{[\![\phi(x)]\!]_T} - [\![\tilde{E}(x)]\!]_T\cdot\overline{\{\!\{\nabla_k\times\phi(x)\}\!\}} \\ &\qquad + (n_F+\eta_F)\left(\mu_r^{-1}\mathcal{R}_F\left([\![\tilde{E}(x)]\!]_T\right)\right)\cdot\overline{\mathcal{R}_F\left([\![\phi(x)]\!]_T\right)}\,\mathrm{d}s. \end{aligned} \tag{4.11}$$

The hierarchic basisfunctions of Ainsworth and Coyle were chosen for their nice properties in the $H(\mathrm{curl};\mathcal{T}_h)$ space. Unfortunately the ∇_k operator destroys some of these. To counteract this effect also apply a shift to the basisfunctions. $\tilde{\phi}(x) = \mathrm{e}^{-\mathrm{i}k\cdot(x-x_i)}\phi(x)$, where the x_i are still

undefined. Insert the new basisfunctions into the weak formulations (4.10) and (4.11) to obtain new discretizations with shifted basisfunctions. First for the IP-flux

$$\begin{aligned}\sum_{i=1}^{N_p} E_i\omega^2\left(\epsilon_r\tilde{\phi}_i,\tilde{\phi}_j\right)_\Omega &= \sum_{i=1}^{N_p} E_i\left(\nabla_{k,h}\times\tilde{\phi}_i,\nabla_{k,h}\times\tilde{\phi}_j\right)_\Omega \\ &+ E_i\sum_{F\in\mathcal{F}_h}\int_F -\{\!\{\mu_r^{-1}\nabla_k\times\tilde{\phi}_i(x)\}\!\}\cdot\overline{[\![\tilde{\phi}_j(x)]\!]_T} - [\![\tilde{\phi}_i(x)]\!]_T\cdot\overline{\{\!\{\nabla_k\times\tilde{\phi}_j(x)\}\!\}} \\ &+ a_F[\![\mu_r^{-1}\tilde{\phi}_i(x)]\!]_T\cdot\overline{[\![\tilde{\phi}_j(x)]\!]_T}\,\mathrm{d}s \quad j=1,\dots,N_p.\end{aligned}$$

Extracting the exponents wherever possible then results in

$$\begin{aligned}&\sum_{i=1}^{N_p} E_i\mathrm{e}^{\mathrm{i}k\cdot(x_i-x_j)}\omega^2\left(\epsilon_r\phi_i,\phi_j\right)_\Omega \\ =&\sum_{i=1}^{N_p} E_i\mathrm{e}^{\mathrm{i}k\cdot(x_i-x_j)}\left(\nabla_h\times\phi_i,\nabla_h\times\phi_j\right)_\Omega \\ +&E_i\mathrm{e}^{\mathrm{i}k\cdot(x_i-x_j)}\sum_{F\in\mathcal{F}_h}\int_F -\{\!\{\mathrm{e}^{-\mathrm{i}k\cdot x}\mu_r^{-1}\nabla\times\phi_i(x)\}\!\}\cdot\overline{[\![\mathrm{e}^{-\mathrm{i}k\cdot x}\phi_j(x)]\!]_T} \\ &- [\![\mathrm{e}^{-\mathrm{i}k\cdot x}\phi_i(x)]\!]_T\cdot\overline{\{\!\{\mathrm{e}^{-\mathrm{i}k\cdot x}\nabla\times\phi_j(x)\}\!\}} \\ &+ a_F[\![\mathrm{e}^{-\mathrm{i}k\cdot x}\mu_r^{-1}\phi_i(x)]\!]_T\cdot\overline{[\![\mathrm{e}^{-\mathrm{i}k\cdot x}\phi_j(x)]\!]_T}\,\mathrm{d}s \quad j=1,\dots,N_p. \qquad (4.12)\end{aligned}$$

For the Brezzi-flux we obtain analogously

$$\begin{aligned}&\sum_{i=1}^{N_p} E_i\omega^2\left(\epsilon_r\tilde{\phi}_i,\tilde{\phi}_j\right)_\Omega \\ =&\sum_{i=1}^{N_p} E_i\left(\nabla_{k,h}\times\tilde{\phi}_i,\nabla_{k,h}\times\tilde{\phi}_j\right)_\Omega \\ +&E_i\sum_{F\in\mathcal{F}_h}\int_F -\{\!\{\mu_r^{-1}\nabla_k\times\tilde{\phi}_i(x)\}\!\}\cdot\overline{[\![\tilde{\phi}_j(x)]\!]_T} - [\![\tilde{\phi}_i(x)]\!]_T\cdot\overline{\{\!\{\nabla_k\times\tilde{\phi}_j(x)\}\!\}} \\ &+ (n_F+\eta_F)\left(\mu_r^{-1}\mathcal{R}_F\left([\![\tilde{\phi}_i(x)]\!]_T\right)\right)\cdot\overline{\mathcal{R}_F\left([\![\tilde{\phi}_j(x)]\!]_T\right)}\,\mathrm{d}s \quad j=1,\dots,N_p.\end{aligned}$$

After extracting the exponents this transforms into

$$
\begin{aligned}
&\sum_{i=1}^{N_p} E_i \mathrm{e}^{\mathrm{i}k\cdot(x_i-x_j)} \omega^2 \left(\epsilon_r \phi_i, \phi_j\right)_\Omega \\
=&\sum_{i=1}^{N_p} E_i \mathrm{e}^{\mathrm{i}k\cdot(x_i-x_j)} \left(\nabla_h \times \phi_i, \nabla_h \times \phi_j\right)_\Omega \\
&+E_i \mathrm{e}^{\mathrm{i}k\cdot(x_i-x_j)} \sum_{F\in\mathcal{F}_h} \int_F -\left\{\!\!\left\{\mathrm{e}^{-\mathrm{i}k\cdot x} \mu_r^{-1} \nabla \times \phi_i(x)\right\}\!\!\right\} \cdot \overline{\left[\!\left[\mathrm{e}^{-\mathrm{i}k\cdot x} \phi_j(x)\right]\!\right]_T} \\
&-\left[\!\left[\mathrm{e}^{-\mathrm{i}k\cdot x} \phi_i(x)\right]\!\right]_T \cdot \overline{\left\{\!\!\left\{\mathrm{e}^{-\mathrm{i}k\cdot x} \nabla \times \phi_j(x)\right\}\!\!\right\}} \\
&+(n_F+\eta_F)\left(\mu_r^{-1}\mathcal{R}_F\left(\left[\!\left[\mathrm{e}^{-\mathrm{i}k\cdot x}\phi_i(x)\right]\!\right]_T\right)\right) \cdot \overline{\mathcal{R}_F\left(\left[\!\left[\mathrm{e}^{-\mathrm{i}k\cdot x}\phi_j(x)\right]\!\right]_T\right)}\,\mathrm{d}s \quad j=1,\ldots,N_p. \qquad (4.13)
\end{aligned}
$$

The resulting eigenvalues $\omega(k)$ are called the dispersion relation.

4.3 Solving eigenvalue problems

Equations (4.12) and (4.13) represent a generalized eigenvalue problem of the form $S_k x = \lambda M_k x$. In the case of a DGFEM problem, the matrix M_k is block-diagonal and non-singular. Moreover, in the construction of the Brezzi-flux an explicit inverse of every block is needed. Since an explicit inverse of M_k is already available and only a block-diagonal matrix, it was chosen to solve the (standard) eigenvalue problem $M_k^{-1} S_k x = \lambda x$ instead. Usually this takes more work, but the extra work lies in constructing the matrix inverse, which is already known and results therefore in a faster algorithm.

Again write the eigenvalue problem as $Ax = \lambda x$ and note from Section 4.1 that it is also possible to write $AV_k y = \lambda V_k y$ as an approximate problem in the Krylov subspace. From the orthonormality of V_k it follows $V_k^T A V_k y = T_k y = \lambda y$. For stability reasons the symmetry of A is no longer used. This means T_k is no longer tri-diagonal, but extra (small) corrections are allowed in the orthogonalization process that appear in the upper triangular part of T_k. This has as a disadvantage that the v_j are all required explicitly, so the dimension k of the Krylov space is limited by memory capacity.

To compensate, some components of V_k that correspond to uninteresting eigenvalues are dropped. For this the Krylov-Schur method developed by Stewart [29] is used. Stewart slightly generalises the form $AV_k = V_k T_K + \beta_{k+1} v_{k+1} e_k^T$ and uses the freedom gained by this generalisation to move converged eigenvalues to the leftmost columns of T and undesired eigenvalues to the rightmost columns of T. Moreover, he shows that truncating T and V after this has been done removes the invariant spaces associated with the undesired eigenvalues from the eigenvalue-problem, even when V is expanded again by the usual expansion procedure.

This removes the memory bounds on k without reducing the convergence properties of the procedure so that, in the end only the desired eigenvalues and information about their invariant spaces are contained in V and T.

Chapter 5

Density of States and Local Density of States

In practical situations, instead of measuring the dispersion relation, it is easier to measure the (local) density of states. This chapter will present a derivation of this measure, and provides a way to compute it from the eigenvalues and eigenfunctions obtained from the discontinuous Galerkin finite element solution.

5.1 Derivation of the formulas

The local density of states is given in most physics texts; e.g. [23] as

$$\rho_p\left(x,\omega,n_p\right)=\frac{6\omega}{\pi}n_p^T Im\left[G\left(x,x,\omega\right)\right]n_p, \tag{5.1}$$

where G is the Green's tensor, n_p is used to focus on a specific wave polarisation and Im represents the imaginary part. With the eigenfunctions already available from computing the dispersion relation, it is natural to construct the Green's tensor by an eigenfunction expansion. This quickly gives rise to a naming conflict. Denote by ω the given scalar quantity as used in (4.1) and denote by $(\omega_{n,k},\psi_{n,k})$ the n-th eigenpair of the k-shifted eigenproblem.

The first goal is to construct the Greens function. For this purpose, we introduce the expansion of the Greens function in eigenfunctions

$$G\left(x,x',\omega\right)=\int\limits_{BZ}\sum_n A_{n,k}\left(x',\omega\right)\otimes\psi_{n,k}\left(x\right)\,\mathrm{d}k, \tag{5.2}$$

where the $A_{n,k}$ are the expansion coefficients, $\otimes$is the tensor product and BZ denotes the Brillouin Zone. Now use the definition of the Greens function for (4.1)

$$\nabla\times\mu_r^{-1}\nabla\times G\left(x,x',\omega\right)-\omega^2\epsilon_r G\left(x,x',\omega\right)=I\delta^3(x-x'), \tag{5.3}$$

where $\delta^3\left(x\right)$ is the three-dimensional Dirac delta distribution. Introduce the expansion for the Greens function given by (5.2) into (5.3)

$$\int\limits_{BZ}\sum_n A_{n,k}\left(x',\omega\right)\otimes\left(\nabla\times\mu_r^{-1}\nabla\times\psi_{n,k}\left(x\right)-\omega^2\epsilon_r\psi_{n,k}\left(x\right)\right)\,\mathrm{d}k=I\delta^3(x-x'), \tag{5.4}$$

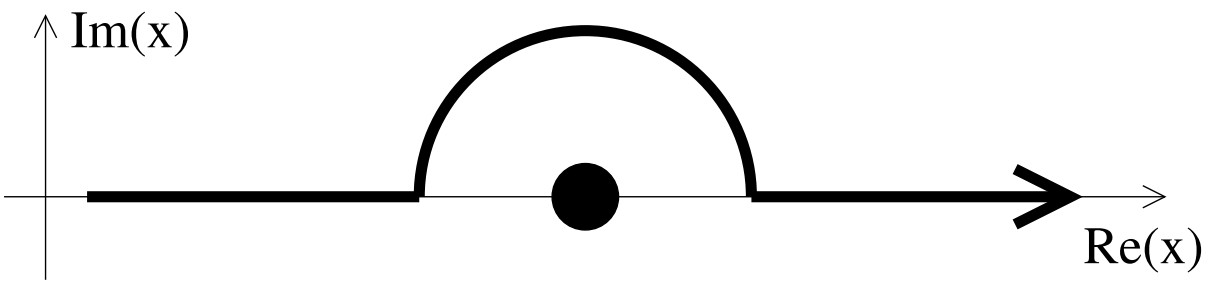


Figure 5.1: Integrating real functions with singularities is actually complex contour integration. The dot is the location of a singuarity, the line is the contour of the contour integral.

and use the fact that $\psi_{n,k}$ is the solution to the eigenvalue problem (2.5). Substitution in (5.4) then results in the expression

$$\int\limits_{BZ} \sum_n A_{n,k}\left(x',\omega\right) \otimes \left(\omega_{n,k}^2 - \omega^2\right) \epsilon_r \psi_{n,k}\left(x\right) \,\mathrm{d}k = I\delta^3(x-x').$$

Now take the L^2-inner product with $\psi_{n,k}^*$ and make sure that the eigenfunctions are orthonormal in the $L^2\left(\Omega\right)$- inner product to find

$$A_{n,k}\left(x',\omega\right) = \frac{\psi_{n,k}^*\left(x'\right)}{\left(2\pi\right)^3\left(\omega_{n,k}^2 - \omega^2\right)}.$$

So the Greens function can be represented as

$$G\left(x,x',\omega\right) = \frac{1}{\left(2\pi\right)^3} \int\limits_{BZ} \sum_n \frac{\psi_{n,k}\left(x'\right)\psi_{n,k}^*\left(x\right)}{\omega_{n,k}^2 - \omega^2} \,\mathrm{d}k.$$

The next step is to find the imaginary part of the Greens function. The only factor that can have an imaginary part is $\frac{1}{\omega_{n,k}^2-\omega^2}$. Whenever $\omega^2 \neq \omega_{n,k}^2$ this factor is also real, but from complex analysis it is known that

$$\int\limits_{\omega_{n,k}-\varepsilon}^{\omega_{n,k}+\varepsilon} \frac{1}{\omega_{n,k}^2 - \omega^2} \,\mathrm{d}\omega = \int\limits_{\omega_{n,k}-\varepsilon}^{\omega_{n,k}+\varepsilon} \frac{1}{\left(\omega_{n,k}-\omega\right)\left(\omega_{n,k}+\omega\right)} \,\mathrm{d}\omega = \frac{\pi \mathrm{i}}{2\omega_{n,k}},$$

with ε real and sufficiently small. The contour of this complex integral is visualised in Figure 5.1. The only solution is then that the imaginary part forms a Dirac distribution centred at the pole. A similar expression holds for the other pole, but one is generally only interested in the positive ω part of the Greens function. With this the local density of states can be rewritten in terms of the eigenfunctions as

$$\rho_p\left(x,\omega,n_p\right) = \frac{3}{\left(2\pi\right)^3} \int\limits_{BZ} \sum_n n_p^T \left(\psi_{n,k}\left(x\right)\psi_{n,k}^*\left(x\right)\right) n_p \delta\left(\omega - \omega_{n,k}\right) \,\mathrm{d}k. \tag{5.5}$$

If the orientation of an emitter is not fixed, it is possible to average the orientation out of this expression. In this case the local density of states is

$$\rho(x,\omega) = \frac{1}{(2\pi)^3} \int_{BZ} \sum_n |\psi_{n,k}(x)|^2 \, \delta(\omega - \omega_{n,k}) \, \mathrm{d}k. \tag{5.6}$$

For the density of states we also integrate over all positions

$$\rho(\omega) = \frac{1}{(2\pi)^3} \int_{BZ} \sum_n \delta(\omega - \omega_{n,k}) \, \mathrm{d}k. \tag{5.7}$$

5.2 Evaluating integrals involving a Dirac distribution

The equations for the (L)DOS given by (5.5), (5.6) and (5.7) all require computing integrals of the form $\int_{BZ} f(k) \, \delta(\omega - \omega(k)) \, \mathrm{d}k$. As a first approximation, we use linear interpolation on a tessellation of tetrahedra to evaluate this integral. Restrict the integral to an arbitrary tetrahedron inside the Brillouin zone and name the corners k_0, k_1, k_2 and k_3. Solve the k-shifted eigenvalue problem to find $\omega_i = \omega(k_i)$ and $f_i = f(k_i)$. Now

$$\omega(k) \approx \omega_0 + a \cdot (k - k_0), \tag{5.8}$$

where

$$\begin{aligned} a = &\frac{(\omega_1 - \omega_0)(k_2 - k_0) \times (k_3 - k_0)}{(k_1 - k_0) \cdot ((k_2 - k_0) \times (k_3 - k_0))} + \frac{(\omega_2 - \omega_0)(k_3 - k_0) \times (k_1 - k_0)}{(k_2 - k_0) \cdot ((k_3 - k_0) \times (k_1 - k_0))} \\ &+ \frac{(\omega_3 - \omega_0)(k_1 - k_0) \times (k_2 - k_0)}{(k_3 - k_0) \cdot ((k_1 - k_0) \times (k_2 - k_0))}. \end{aligned} \tag{5.9}$$

Note, (5.9) can be quickly verified by substituting the corners $k_0, \ldots, k_3$ into (5.8). Because of linearity there is always a plane k_ω such that $\omega(k_\omega) = \omega$ (except in the degenerate case). This means it is possible to apply a change of coordinates

$$\int_{BZ} f(k) \, \delta(\omega - \omega(k)) \, \mathrm{d}k = \int_{BZ} f(k) \, \delta(k_\omega - k(\omega)) \, \frac{1}{|\nabla_k \omega(k)|} \, \mathrm{d}k,$$

where in the linear case $\nabla_k \omega = a$.

Now find the intersection of the k_ω-plane and the tetrahedron. Assume $\omega_0 \leq \omega_1 \leq \omega_2 \leq \omega_3$. If this is not the case it is possible to generalise the following formulas, keeping as distinctive factor the number of corners where $\omega_{n,k} < \omega$, but this needlessly complicates the derivation. Now identify several cases. These cases are also illustrated in Figure 5.2

- If $\omega < \omega_0$ or $\omega_3 < \omega$ the intersection is empty.
- If $\omega_0 < \omega < \omega_1$ the intersection is a triangle with corners $\frac{\omega - \omega_0}{\omega_1 - \omega_0} k_1 + \frac{\omega_1 - \omega}{\omega_1 - \omega_0} k_0$, $\frac{\omega - \omega_0}{\omega_2 - \omega_0} k_2 + \frac{\omega_2 - \omega}{\omega_2 - \omega_0} k_0$ and $\frac{\omega - \omega_0}{\omega_3 - \omega_0} k_3 + \frac{\omega_3 - \omega}{\omega_3 - \omega_0} k_0$. By applying the midpoint integration rule it can be seen that

$$\begin{aligned}
&\int f(k)\,\delta(\omega-\omega(k))\,\mathrm{d}k\\
\approx&\frac{1}{6\,|a|}\sum_{i=1}^{3}\frac{(\omega-\omega_0)\,f_i+(\omega_i-\omega)\,f_0}{\omega_i-\omega_0}\\
&\left|\left(\frac{(\omega-\omega_0)\,k_2+(\omega_2-\omega)\,k_0}{\omega_2-\omega_0}-\frac{(\omega-\omega_0)\,k_1+(\omega_1-\omega)\,k_0}{\omega_1-\omega_0}\right)\times\right.\\
&\left.\left(\frac{(\omega-\omega_0)\,k_3+(\omega_3-\omega)\,k_0}{\omega_3-\omega_0}-\frac{(\omega-\omega_0)\,k_1+(\omega_1-\omega)\,k_0}{\omega_1-\omega_0}\right)\right|.
\end{aligned}$$

- If $\omega_2<\omega<\omega_3$ the intersection is also a triangle. This time it has corners $\frac{\omega_3-\omega}{\omega_3-\omega_0}k_0+\frac{\omega-\omega_0}{\omega_3-\omega_0}k_3$, $\frac{\omega_3-\omega}{\omega_3-\omega_1}k_1+\frac{\omega-\omega_1}{\omega_3-\omega_1}k_3$ and $\frac{\omega_3-\omega}{\omega_3-\omega_2}k_2+\frac{\omega-\omega_2}{\omega_3-\omega_2}k_3$. Again apply the midpoint-rule

$$\begin{aligned}
&\int f(k)\,\delta(\omega-\omega(k))\,\mathrm{d}k\\
\approx&\frac{1}{6\,|a|}\sum_{i=0}^{2}\frac{(\omega_3-\omega)\,f_i+(\omega-\omega_i)\,f_3}{\omega_3-\omega_i}\\
&\left|\left(\frac{(\omega_3-\omega)\,k_1+(\omega-\omega_1)\,k_3}{\omega_3-\omega_1}-\frac{(\omega_3-\omega)\,k_1+(\omega-\omega_1)\,k_3}{\omega_3-\omega_1}\right)\times\right.\\
&\left.\left(\frac{(\omega_3-\omega)\,k_2+(\omega-\omega_2)\,k_3}{\omega_3-\omega_2}-\frac{(\omega_3-\omega)\,k_1+(\omega-\omega_1)\,k_3}{\omega_3-\omega_1}\right)\right|.
\end{aligned}$$

- If $\omega_1<\omega<\omega_2$ the intersection is a quadrilateral. It has corners $\frac{\omega-\omega_0}{\omega_2-\omega_0}k_2+\frac{\omega_2-\omega}{\omega_2-\omega_0}k_0$, $\frac{\omega-\omega_0}{\omega_3-\omega_0}k_3+\frac{\omega_3-\omega}{\omega_3-\omega_0}k_0$, $\frac{\omega-\omega_1}{\omega_2-\omega_1}k_2+\frac{\omega_2-\omega}{\omega_2-\omega_1}k_1$ and $\frac{\omega-\omega_1}{\omega_3-\omega_1}k_3+\frac{\omega_3-\omega}{\omega_3-\omega_1}k_1$. Now cut the quadrilateral into two triangles and compute the integral on both triangles.

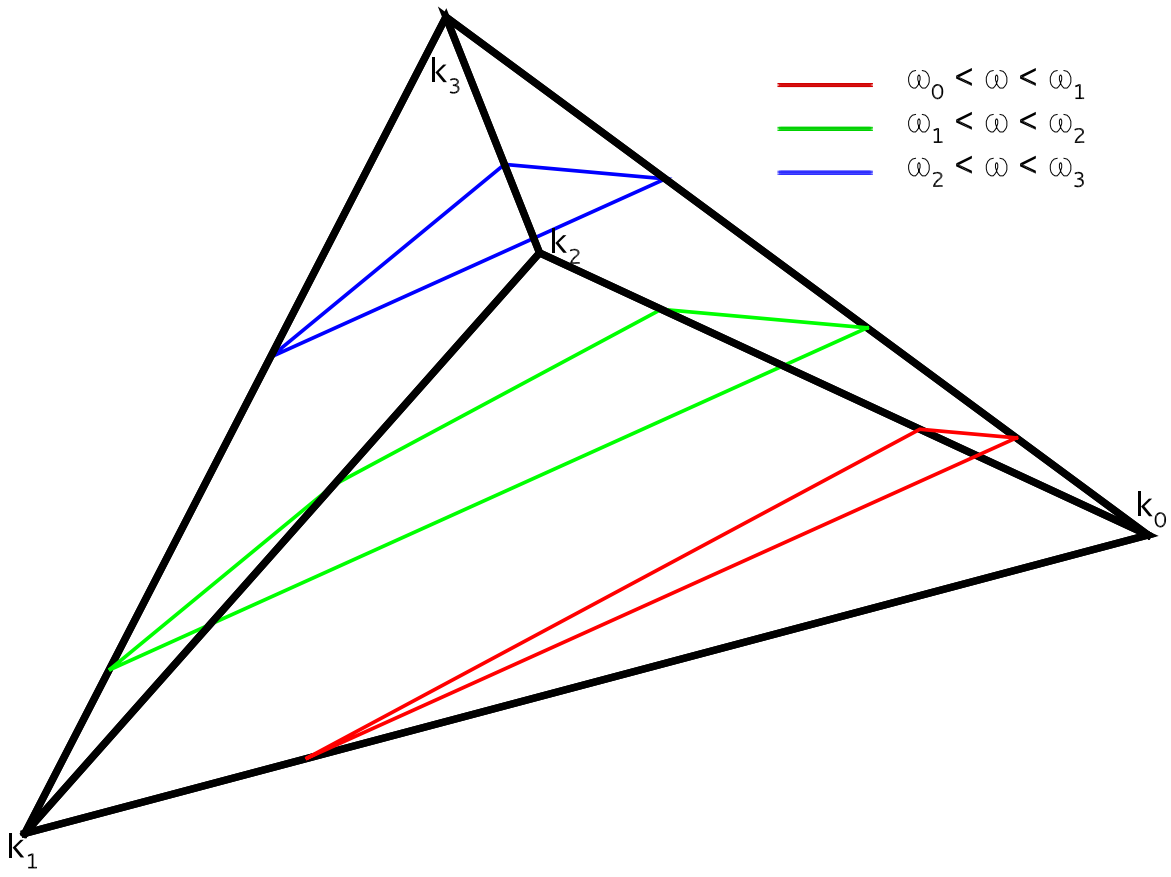


Figure 5.2: Example tetrahedron and constant ω planes for three different values of ω.

Chapter 6

Results

The discontinuous Galerkin finite element discretisation has been implemented in the C++ code DG-Max using hpGEM [30]. PETSc [6] was used to provide a linear system solver and SLEPc [15] was used as eigenvalue solver. In this chapter a series of tests cases will be presented to demonstrate that the DG-Max code works correctly.

6.1 Convergence tests

First of all the tests presented in [27] and [12] are repeated as a basic validation of DG-Max.

6.1.1 Homogenous boundary conditions

Before considering periodic domains it is important to know if the problem is solved correctly for the homogeneous boundary conditions $n \times E$ in a homogeneous medium. For this purpose the initial conditions and the source term are set such that the exact solution is

$$E(t,x,y,z) = \begin{pmatrix} \sin(\pi y)\sin(\pi z) \\ \sin(\pi z)\sin(\pi x) \\ \sin(\pi x)\sin(\pi y) \end{pmatrix} \cos\left(\sqrt{2}\pi t\right),$$

with domain dimensions $1 \times 1 \times 1$. Due to the nature of the solution the error of the time-dependent numerical solution fluctuates in time. The simulation is stopped at $t = 2.5$. Then the error shown is computed as

$$\max_{0<t\leq 2.5} \|E - E_h\|.$$

Based on the works of Houston et al. [17, 8], the expected order of convergence is $O\left(h^{p+1}\right)$ in the $\|\cdot\|_{L^2}$-norm and $O\left(h^p\right)$ in the $\|\cdot\|_{DG}$-norm. For the eigenvalue problem the expected order of convergence is $O\left(h^{2p}\right)$. Here h is the diameter of the largest element in the mesh and p the piece-wise polynomial order of the basis functions used. The eigenvalue solver is set to find the smallest positive eigenvalues by choosing a target with prior knowledge such that the 24 closest eigenvalues include at least the smallest positive eigenvalue. Any eigenvalues with value 0 are manually filtered out. The exact eigenvalues are taken from [27].

It can be seen that the order of convergence is usually as good as expected. There are a few exceptions. In these cases the solution always is already highly converged and the error close to machine precision.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 5$	1.27152e-0	-	5.23950e-0	-
$N_{el} = 40$	6.57174e-1	0.95	2.98154e-0	0.81
$N_{el} = 320$	2.03360e-1	1.69	1.09302e-0	1.45
$N_{el} = 2560$	5.27515e-2	1.95	4.85793e-1	1.17
$N_{el} = 20480$	1.33414e-2	1.98	2.54416e-1	0.93
p=2				
$N_{el} = 5$	3.83875e-1	-	1.90489e-0	-
$N_{el} = 40$	5.53516e-2	2.79	5.40066e-1	1.82
$N_{el} = 320$	4.53141e-3	3.61	1.43667e-1	1.91
$N_{el} = 2560$	5.68881e-4	2.99	3.61636e-2	1.99
p=3				
$N_{el} = 5$	1.31993e-1	-	1.04337e-0	-
$N_{el} = 40$	4.12030e-3	5.00	1.10527e-1	3.24
$N_{el} = 320$	2.63932e-4	3.96	1.46722e-2	2.91
$N_{el} = 2560$	1.77995e-5	3.89	1.85421e-3	2.98
p=4				
$N_{el} = 5$	1.27279e-2	-	1.53443e-1	-
$N_{el} = 40$	5.87734e-4	4.44	1.88147e-2	3.03
$N_{el} = 320$	1.86281e-5	4.98	1.20593e-3	3.96
p=5				
$N_{el} = 5$	7.02176e-3	-	9.93113e-2	-
$N_{el} = 40$	7.90798e-5	6.47	2.61866e-3	5.25
$N_{el} = 320$	DNF	-	DNF	-

Table 6.1: Errors found for time dependent Maxwell calculations using the Brezzi numerical flux and perfectly conducting boundary conditions together with piecewise polynomial basisfunctions of order p and N_{el} elements. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also listed. DNF means the computer killed the simulation before it finished.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 5$	1.32366e-0	-	5.66014e-0	-
$N_{el} = 40$	9.17346e-1	0.53	4.44130e-0	0.35
$N_{el} = 320$	2.59439e-1	1.82	1.27989e-0	1.80
$N_{el} = 2560$	5.13186e-2	2.34	5.17806e-1	1.31
$N_{el} = 20480$	6.35779e-3	3.01	2.47697e-1	1.06
p=2				
$N_{el} = 5$	6.57622e-1	-	3.23391e-0	-
$N_{el} = 40$	6.89208e-2	3.25	6.17225e-1	2.39
$N_{el} = 320$	4.41895e-3	3.96	1.46087e-1	2.08
$N_{el} = 2560$	4.73919e-4	3.22	3.93327e-2	1.89
p=3				
$N_{el} = 5$	1.63392e-1	-	1.06601e-0	-
$N_{el} = 40$	4.40342e-3	5.21	1.15940e-1	3.20
$N_{el} = 320$	2.40996e-4	4.19	1.48420e-2	2.97
$N_{el} = 2560$	1.56225e-5	3.95	1.87562e-3	2.98
p=4				
$N_{el} = 5$	1.50866e-2	-	1.58541e-1	-
$N_{el} = 40$	5.10110e-4	4.89	1.81594e-2	3.13
$N_{el} = 320$	2.44701e-5	4.38	1.30421e-3	3.80
p=5				
$N_{el} = 5$	7.18309e-3	-	1.00695e-1	-
$N_{el} = 40$	8.31709e-5	6.43	2.73436e-3	5.20
$N_{el} = 320$	DNF	-	DNF	-

Table 6.2: Errors found for time dependent Maxwell calculations using the IP numerical flux and perfectly conducting boundary conditions together with piecewise polynomial basisfunctions of order p and N_{el} elements. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also listed. DNF means the computer killed the simulation before it finished.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 5$	2.59635e-1	-	4.00567e-0	-
$N_{el} = 40$	2.46101e-1	0.08	1.84197e-0	1.12
$N_{el} = 320$	5.07053e-2	2.28	9.45967e-1	0.96
$N_{el} = 2560$	1.18549e-2	2.10	4.75940e-1	0.99
$N_{el} = 20480$	2.91011e-3	2.03	2.38232e-1	1.00
p=2				
$N_{el} = 5$	2.63838e-1	-	1.26906e-0	-
$N_{el} = 40$	2.89828e-2	3.19	4.85614e-1	1.39
$N_{el} = 320$	3.25228e-3	3.16	1.26697e-1	1.94
$N_{el} = 2560$	3.92899e-4	3.05	3.20109e-2	1.98
p=3				
$N_{el} = 5$	5.40371e-2	-	8.61001e-1	-
$N_{el} = 40$	4.28163e-3	3.66	9.62480e-2	3.16
$N_{el} = 320$	2.13142e-4	4.33	1.25106e-2	2.94
$N_{el} = 2560$	1.86856e-5	3.51	1.57853e-3	2.99
p=4				
$N_{el} = 5$	2.25090e-2	-	1.07844e-1	-
$N_{el} = 40$	5.17862e-4	5.44	1.52262e-2	2.82
p=5				
$N_{el} = 5$	4.52047e-3	-	8.70284e-2	-

Table 6.3: Errors found for time harmonic Maxwell calculations using the Brezzi numerical flux and perfectly conducting boundary conditions together with piecewise polynomial basisfunctions of order p and N_{el} elements. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also listed.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 5$	5.17505e-1	-	4.25017e-0	-
$N_{el} = 40$	2.90662e-1	0.83	1.94746e-0	1.13
$N_{el} = 320$	6.04859e-2	2.26	9.87165e-1	0.98
$N_{el} = 2560$	1.15523e-2	2.39	4.82840e-1	1.03
$N_{el} = 20480$	1.98266e-3	2.54	2.36082e-1	1.03
p=2				
$N_{el} = 5$	3.03454e-1	-	1.47002e-0	-
$N_{el} = 40$	3.09558e-2	3.29	5.03171e-1	1.55
$N_{el} = 320$	3.25404e-3	3.25	1.31052e-1	1.94
$N_{el} = 2560$	5.02666e-4	2.69	3.57966e-2	1.87
p=3				
$N_{el} = 5$	5.82321e-2	-	8.81547e-1	-
$N_{el} = 40$	4.34719e-3	3.74	9.86913e-2	3.16
$N_{el} = 320$	2.13261e-4	4.35	1.28777e-2	2.94
$N_{el} = 2560$	1.05514e-5	1.34	1.65271e-3	2.96
p=4				
$N_{el} = 5$	2.32344e-2	-	1.20657e-1	-
$N_{el} = 40$	5.29809e-4	5.45	1.55773e-2	2.95
p=5				
$N_{el} = 5$	4.47974e-3	-	8.81404e-2	-

Table 6.4: Errors found for time harmonic Maxwell calculations using the IP numerical flux and perfectly conducting boundary conditions together with piecewise polynomial basisfunctions of order p and N_{el} elements. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also listed.

Eigenvalue	$N_{el} = 5$		$N_{el} = 320$		$N_{el} = 2560$		$N_{el} = 20480$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	9.9476e-1	-	4.6094e-2	1.82	1.1750e-2	1.97	2.9500e-3	1.99
$2\pi^2$	9.9476e-1	-	4.6094e-2	1.82	1.1750e-2	1.97	2.9500e-3	1.99
$2\pi^2$	9.9476e-1	-	4.6094e-2	1.82	1.1750e-2	1.97	2.9500e-3	1.99
$3\pi^2$	1.4492e-0	-	6.7934e-2	2.39	1.7298e-2	1.97	4.3403e-3	1.99
$3\pi^2$	1.4492e-0	-	6.7934e-2	2.39	1.7298e-2	1.97	4.3403e-3	1.99
$5\pi^2$	1.2963e-0	-	1.0883e-1	0.45	2.5891e-2	2.07	7.2114e-3	1.84
$5\pi^2$	1.2963e-0	-	1.0883e-1	0.45	2.5891e-2	2.07	7.2114e-3	1.84
$5\pi^2$	1.2963e-0	-	1.0883e-1	0.45	2.5891e-2	2.07	7.2114e-3	1.84
$5\pi^2$	1.4386e-0	-	1.0883e-1	1.86	2.5891e-2	2.07	7.2114e-3	1.84
$5\pi^2$	1.4386e-0	-	1.0883e-1	1.86	2.5891e-2	2.07	7.2114e-3	1.84
$5\pi^2$	1.4386e-0	-	1.0883e-1	1.86	2.5891e-2	2.07	7.2114e-3	1.84
$6\pi^2$	1.6212e-0	-	1.2708e-1	1.65	3.3645e-2	1.92	8.4942e-3	1.99
$6\pi^2$	1.6212e-0	-	1.2708e-1	1.65	3.3645e-2	1.92	8.4942e-3	1.99
$6\pi^2$	1.6212e-0	-	1.2708e-1	1.65	3.3645e-2	1.92	8.4942e-3	1.99
$6\pi^2$	not found		1.3173e-1	-	3.4426e-2	1.94	8.6713e-3	1.99
$6\pi^2$	not found		1.3173e-1	-	3.4426e-2	1.94	8.6713e-3	1.99
$6\pi^2$	not found		1.3173e-1	-	3.4426e-2	1.94	8.6713e-3	1.99

Table 6.5: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the Brezzi numerical flux. Piece-wise linear basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. For $N_{el} = 5$ also some spurious eigenvalues were found, these are not shown. The decision which eigenvalues are considered spurious was made based on the expected order of convergence of the eigenvalues. The order of convergence is also shown. Computations were also done on 40 elements, but due to space limitations these results are not shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	8.6580e-2	-	1.1690e-2	2.89	8.3800e-4	3.80	5.4500e-5	3.94
$2\pi^2$	8.6580e-2	-	1.1690e-2	2.89	8.3800e-4	3.80	5.4500e-5	3.94
$2\pi^2$	8.6580e-2	-	1.1690e-2	2.89	8.3800e-4	3.80	5.4500e-5	3.94
$3\pi^2$	3.3452e-1	-	8.0807e-3	5.37	1.8703e-3	2.11	1.2300e-4	3.93
$3\pi^2$	3.3452e-1	-	8.0807e-3	5.37	1.8703e-3	2.11	1.2300e-4	3.93
$5\pi^2$	2.8120e-1	-	3.2992e-2	3.09	4.6606e-3	2.82	3.1580e-4	3.88
$5\pi^2$	2.8120e-1	-	3.2992e-2	3.09	4.6606e-3	2.82	3.1580e-4	3.88
$5\pi^2$	2.8120e-1	-	3.2992e-2	3.09	4.6606e-3	2.82	3.1580e-4	3.88
$5\pi^2$	not found		6.0746e-2	-	4.6606e-3	3.70	3.1580e-4	3.88
$5\pi^2$	not found		6.0746e-2	-	4.6606e-3	3.70	3.1580e-4	3.88
$5\pi^2$	not found		6.0746e-2	-	not found		3.1580e-4	3.79
$6\pi^2$	4.8693e-1	-	6.2600e-2	2.96	6.6410e-3	3.24	4.5817e-4	3.86
$6\pi^2$	4.8693e-1	-	6.2600e-2	2.96	6.6410e-3	3.24	4.5817e-4	3.86
$6\pi^2$	4.8693e-1	-	6.2600e-2	2.96	6.6410e-3	3.24	4.5817e-4	3.86
$6\pi^2$	not found		8.1383e-2	-	6.8365e-3	3.57	4.6817e-4	3.87
$6\pi^2$	not found		8.1383e-2	-	6.8365e-3	3.57	4.6817e-4	3.87
$6\pi^2$	not found		8.1383e-2	-	6.8365e-3	3.57	4.6817e-4	3.87

Table 6.6: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the Brezzi numerical flux. Piece-wise quadratic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$		$N_{el} = 320$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	2.8411e-2	-	4.6700e-4	5.93	8.5000e-5	2.46
$2\pi^2$	2.8411e-2	-	4.6700e-4	5.93	8.5000e-5	2.46
$2\pi^2$	2.8411e-2	-	4.6700e-4	5.93	8.5000e-5	2.46
$3\pi^2$	7.6612e-2	-	2.6157e-3	4.87	2.7333e-4	3.26
$3\pi^2$	7.6612e-2	-	2.6157e-3	4.87	2.7333e-4	3.26
$5\pi^2$	5.3142e-2	-	5.3394e-3	3.32	1.1100e-4	5.59
$5\pi^2$	5.3142e-2	-	5.3394e-3	3.32	1.1100e-4	5.59
$5\pi^2$	5.3142e-2	-	5.3394e-3	3.32	1.1100e-4	5.59
$5\pi^2$	1.0647e-1	-	6.0208e-3	4.14	1.1100e-4	5.76
$5\pi^2$	1.0647e-1	-	6.0208e-3	4.14	1.1100e-4	5.76
$5\pi^2$	1.0647e-1	-	6.0208e-3	4.14	not found	
$6\pi^2$	1.2742e-1	-	8.8458e-3	3.85	1.9333e-4	5.52
$6\pi^2$	1.2742e-1	-	8.8458e-3	3.85	1.9333e-4	5.52
$6\pi^2$	1.2742e-1	-	8.8458e-3	3.85	1.9333e-4	5.52
$6\pi^2$	2.2783e-1	-	8.9212e-3	4.67	1.9950e-4	5.48
$6\pi^2$	2.2783e-1	-	8.9212e-3	4.67	1.9950e-4	5.48
$6\pi^2$	2.2783e-1	-	8.9212e-3	4.67	1.9950e-4	5.48

Table 6.7: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the Brezzi numerical flux. Piece-wise cubic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$		$N_{el} = 320$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$8\pi^2$	1.8003e-1	-	2.5213e-4	9.48	˜2.5e-7	9.98
$8\pi^2$	1.9039e-1	-	2.5213e-4	9.56	˜2.5e-7	9.98
$8\pi^2$	not found		2.5213e-4		˜2.5e-7	9.98
$9\pi^2$	1.9989e-1	-	1.6444e-4	10.25	˜3e-7	9.10
$9\pi^2$	2.1590e-1	-	1.6444e-4	10.36	˜3e-7	9.10
$9\pi^2$	not found		1.6444e-4		˜3e-7	9.10
$9\pi^2$	2.9890e-1	-	3.0589e-4	9.93	˜3e-7	9.99
$9\pi^2$	3.0582e-1	-	3.0589e-4	9.97	˜3e-7	9.99
$9\pi^2$	not found		3.0589e-4	-	not found	

Table 6.8: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the Brezzi numerical flux. Piece-wise quintic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown. These are not the first 17 eigenvalues. Instead they are hand-picked such that the required precision to show 10th order convergence is reached without having to dodge numerical accuracy issues unrelated to the method.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$		$N_{el} = 20480$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	2.3145e-1	-	5.7874e-2	2.00	1.1505e-2	2.33	1.5870e-3	2.86
$2\pi^2$	2.3145e-1	-	5.7874e-2	2.00	1.1505e-2	2.33	1.5870e-3	2.86
$2\pi^2$	2.3145e-1	-	5.7874e-2	2.00	1.1505e-2	2.33	1.5870e-3	2.86
$3\pi^2$	5.4758e-1	-	8.2680e-2	2.73	1.6688e-2	2.31	2.3720e-3	2.81
$3\pi^2$	5.4758e-1	-	8.2680e-2	2.73	1.6688e-2	2.31	2.3720e-3	2.81
$5\pi^2$	2.4867e-1	-	1.3602e-1	0.87	2.8019e-2	2.28	4.0442e-3	2.79
$5\pi^2$	2.4867e-1	-	1.3602e-1	0.87	2.8019e-2	2.28	4.0442e-3	2.79
$5\pi^2$	2.4867e-1	-	1.3602e-1	0.87	2.8019e-2	2.28	4.0442e-3	2.79
$5\pi^2$	not found		1.3606e-1	-	2.8032e-2	2.28	4.0444e-3	2.79
$5\pi^2$	not found		1.3606e-1	-	2.8032e-2	2.28	4.0444e-3	2.79
$5\pi^2$	not found		1.3606e-1	-	2.8032e-2	2.28	4.0444e-3	2.79
$6\pi^2$	5.6279e-1	1.45	1.5303e-1	1.88	3.2542e-2	2.23	4.7172e-3	2.79
$6\pi^2$	5.6279e-1	1.45	1.5303e-1	1.88	3.2542e-2	2.23	4.7172e-3	2.79
$6\pi^2$	5.6279e-1	1.45	1.5303e-1	1.88	3.2542e-2	2.23	not found	
$6\pi^2$	not found		1.6172e-1	-	3.3157e-2	2.29	4.8417e-3	2.78
$6\pi^2$	not found		1.6172e-1	-	3.3157e-2	2.29	4.8417e-3	2.78
$6\pi^2$	not found		1.6172e-1	-	3.3157e-2	2.29	not found	

Table 6.9: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the IP numerical flux. Piece-wise linear basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	1.4950e-1	-	1.4546e-2	3.36	8.6500e-4	4.07	3.4000e-5	4.67
$2\pi^2$	1.4950e-1	-	1.4546e-2	3.36	8.6500e-4	4.07	3.4000e-5	4.67
$2\pi^2$	1.4950e-1	-	1.4546e-2	3.36	8.6500e-4	4.07	3.4000e-5	4.67
$3\pi^2$	4.5604e-1	-	8.8907e-3	5.68	1.9133e-3	2.22	7.4000e-5	4.69
$3\pi^2$	4.5604e-1	-	8.8907e-3	5.68	1.9133e-3	2.22	7.4000e-5	4.69
$5\pi^2$	5.2441e-1	-	4.0556e-2	3.69	4.7756e-3	3.09	1.8800e-4	4.67
$5\pi^2$	5.2441e-1	-	4.0556e-2	3.69	4.7756e-3	3.09	1.8800e-4	4.67
$5\pi^2$	5.2441e-1	-	4.0556e-2	3.69	4.7756e-3	3.09	1.8800e-4	4.67
$5\pi^2$	not found		7.8506e-2	-	4.7760e-3	4.04	1.8860e-4	4.66
$5\pi^2$	not found		7.8506e-2	-	4.7760e-3	4.04	1.8860e-4	4.66
$5\pi^2$	not found		7.8506e-2	-	4.7760e-3	4.04	1.8860e-4	4.66
$6\pi^2$	not found		7.8721e-2	-	6.7672e-3	3.54	2.6467e-4	4.68
$6\pi^2$	not found		7.8721e-2	-	6.7672e-3	3.54	2.6467e-4	4.68
$6\pi^2$	not found		7.8721e-2	-	6.7672e-3	3.54	not found	
$6\pi^2$	not found		1.0012e-1	-	6.9013e-3	3.86	2.8817e-4	4.58
$6\pi^2$	not found		1.0012e-1	-	6.9013e-3	3.86	2.8817e-4	4.58
$6\pi^2$	not found		1.0012e-1	-	6.9013e-3	3.86	not found	

Table 6.10: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the IP numerical flux. Piece-wise quadratic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$		$N_{el} = 320$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	3.4835e-2	-	4.9600e-4	6.13	7.0000e-6	6.15
$2\pi^2$	3.4835e-2	-	4.9600e-4	6.13	7.0000e-6	6.15
$2\pi^2$	3.4835e-2	-	4.9600e-4	6.13	7.0000e-6	6.15
$3\pi^2$	1.0674e-1	-	2.8707e-3	5.22	2.3667e-5	6.92
$3\pi^2$	1.0674e-1	-	2.8707e-3	5.22	2.3667e-5	6.92
$5\pi^2$	6.7150e-2	-	5.7590e-3	3.54	9.4200e-5	5.93
$5\pi^2$	6.7150e-2	-	5.7590e-3	3.54	9.4200e-5	5.93
$5\pi^2$	6.7150e-2	-	5.7590e-3	3.54	9.4200e-5	5.93
$5\pi^2$	1.2554e-1	-	6.6128e-3	4.25	9.4400e-5	6.13
$5\pi^2$	1.2554e-1	-	6.6128e-3	4.25	9.4400e-5	6.13
$5\pi^2$	1.2554e-1	-	6.6128e-3	4.25	9.4400e-5	6.13
$6\pi^2$	1.5028e-1	-	9.2027e-3	4.03	1.6400e-4	5.81
$6\pi^2$	1.5028e-1	-	9.2027e-3	4.03	1.6400e-4	5.81
$6\pi^2$	1.5028e-1	-	9.2027e-3	4.03	1.6400e-4	5.81
$6\pi^2$	2.6798e-1	-	9.6335e-3	4.80	1.6667e-4	5.85
$6\pi^2$	2.6798e-1	-	9.6335e-3	4.80	1.6667e-4	5.85
$6\pi^2$	2.6798e-1	-	9.6335e-3	4.80	1.6667e-4	5.85

Table 6.11: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the IP numerical flux. Piece-wise cubic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$2\pi^2$	1.0210e-3	-	˜1.4e-5	6.1884
$2\pi^2$	1.0210e-3	-	˜1.4e-5	6.1884
$2\pi^2$	1.0210e-3	-	˜1.4e-5	6.1884
$3\pi^2$	4.7673e-3	-	˜8.3e-6	9.17
$3\pi^2$	4.7673e-3	-	˜8.3e-6	9.17
$5\pi^2$	5.0256e-2	-	3.4040e-4	7.21
$5\pi^2$	5.0256e-2	-	3.4040e-4	7.21
$5\pi^2$	5.0256e-2	-	3.4040e-4	7.21
$5\pi^2$	5.7411e-2	-	3.8720e-4	7.21
$5\pi^2$	5.7411e-2	-	3.8720e-4	7.21
$5\pi^2$	5.7411e-2	-	3.8720e-4	7.21
$6\pi^2$	1.0411e-1	-	7.6983e-4	7.07
$6\pi^2$	1.0411e-1	-	7.6983e-4	7.07
$6\pi^2$	1.0411e-1	-	7.6983e-4	7.07
$6\pi^2$	1.1543e-1	-	7.8333e-4	7.20
$6\pi^2$	1.1543e-1	-	7.8333e-4	7.20
$6\pi^2$	1.1543e-1	-	7.8333e-4	7.20

Table 6.12: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the IP numerical flux. Piece-wise quartic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.

Eigenvalue	$N_{el} = 5$		$N_{el} = 40$		$N_{el} = 320$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$9\pi^2$	2.1343e-1	-	1.7222e-4	10.28	˜1e-7	10.60
$9\pi^2$	2.3621e-1	-	1.7222e-4	10.42	˜1e-7	10.60
$9\pi^2$	not found		1.7222e-4	-	not found	
$9\pi^2$	3.1029e-1	-	3.0511e-4	9.99	˜2e-7	10.42
$9\pi^2$	not found		3.0511e-4	-	˜2e-7	10.42
$9\pi^2$	not found		3.0511e-4	-	not found	
$10\pi^2$	2.0607e-1	-	2.3450e-4	9.78	˜3e-7	9.61
$10\pi^2$	2.1994e-1	-	2.3450e-4	9.87	˜3e-7	9.61
$10\pi^2$	not found					
$10\pi^2$	2.4418e-1	-	2.9030e-4	9.72	˜3e-7	9.92
$10\pi^2$	not found		2.9030e-4		˜3e-7	9.92
$10\pi^2$	not found					

Table 6.13: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem with perfectly conducting boundary conditions and the IP numerical flux. Piece-wise quintic basisfunctions on N_{el} elements were used. The domain has dimensions $1 \times 1 \times 1$. The order of convergence is also shown.. These are not the first 17 eigenvalues. Instead they are hand-picked such that the required precision to show 10th order convergence is reached without having to dodge numerical accuracy issues unrelated to the method.

In particular for the IP-flux we do not consistently find the correct multiplicity of the eigenvalues. An important factor in this may be that the value used for a_F in the IP numerical flux is only a decent guess instead of a serious attempt at the optimal choice. This strategy was also used for the Brezzi numerical flux, but the optimal choice for η_F is always between 1 and 2, making it much harder to guess wrong.

It can be concluded that the DG-Max code works for this target solution. While this is in no way sufficient to show that the algorithm will always converge, it is at least sufficient to advance to more complicated test cases. Based on their accuracy the Brezzi numerical flux and the IP numerical flux show similar performance. The Brezzi numerical flux appears to be slightly more robust however.

6.1.2 Periodic domain

The next test series was run on a homogeneous periodic domain with a unit domain of size $1 \times 1 \times 1$. This test series attempts to verify that the boundaries are properly connected. This means that for this test series $k = 0$. For the time-harmonic Maxwell equations and the time dependent Maxwell equations the source term and initial conditions are set such that the exact solution is

$$E(t, x, y, z) = \begin{pmatrix} \sin(2\pi y)\sin(2\pi z) \\ \sin(2\pi z)\sin(2\pi x) \\ \sin(2\pi x)\sin(2\pi y) \end{pmatrix} \cos\left(2\sqrt{2}\pi t\right). \tag{6.1}$$

Due to the nature of the solutions the error of the time-dependent numerical solution fluctuates in time. The simulation is stopped at $t = 2.5$. The error shown is then computed as

$$\max_{0<t\leq 2.5} \|E - E_h\|.$$

The expected order of convergence is the same as in the non-periodic case: $O\left(h^{p+1}\right)$ in the $\|\cdot\|_{L^2}$-norm and $O\left(h^p\right)$ in the $\|\cdot\|_{DG}$-norm; and $O\left(h^{2p}\right)$ for the eigenvalues. Here h is the diameter of the largest element and p the piece-wise polynomial order of the basis functions used. This time most of the results show slightly higher convergence rates than expected. More tests are needed to check if this is actually the case or if this is some phantom result produced by the lack of data available. In either case the results presented so far serve as an indication that the code performs at least as good as expected in the case of periodic boundaries.

These tests reinforce the observation that the IP numerical flux and the Brezzi numerical flux have similar accuracy performance and that the Brezzi flux is slightly more robust.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 40$	1.3422e-0	-	1.1222e+1	-
$N_{el} = 320$	1.1953e-0	0.17	1.0664e+1	0.07
$N_{el} = 2560$	3.9579e-1	1.59	3.7129e+0	1.52
$N_{el} = 20480$	1.0343e-1	1.94	1.1006e+0	1.75
p=2				
$N_{el} = 40$	7.2591e-1	-	7.0742e-0	-
$N_{el} = 320$	1.0404e-1	2.80	1.1444e-0	2.63
$N_{el} = 2560$	7.7700e-3	3.74	2.9185e-1	1.97
p=3				
$N_{el} = 40$	2.4926e-1	-	2.7181e-0	-
$N_{el} = 320$	5.1728e-3	5.59	2.3178e-1	3.55
$N_{el} = 2560$	DNF	-	DNF	-
p=4				
$N_{el} = 40$	1.4173e-2	-	3.1003e-1	-
$N_{el} = 320$	5.6864e-4	4.64	3.7375e-2	3.05
p=5				
$N_{el} = 40$	6.74247e-3	-	1.9589e-1	-

Table 6.14: Errors found for time dependent Maxwell calculations using the Brezzi numerical flux on a periodic domain of size $1 \times 1 \times 1$. Piece-wise polynomials of order p on N_{el} elements were used. The order of convergence is also shown. DNF means the computer killed the simulation before it finished.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 40$	1.2528e-0	-	1.0707e+1	-
$N_{el} = 320$	1.5445e-0	-0.30	1.3612e+1	-0.35
$N_{el} = 2560$	4.0748e-1	1.92	3.8362e+0	1.83
$N_{el} = 20480$	5.9491e-2	2.78	1.0084e+0	1.93
p=2				
$N_{el} = 40$	1.1126e-0	-	1.0772e+1	-
$N_{el} = 320$	1.1522e-1	3.27	1.1854e+0	3.18
$N_{el} = 2560$	5.7437e-3	4.33	2.8806e-1	2.04
p=3				
$N_{el} = 40$	2.9124e-1	-	3.0617e-0	-
$N_{el} = 320$	4.8966e-3	5.89	2.2725e-1	3.75
$N_{el} = 2560$	DNF	-	DNF	-
p=4				
$N_{el} = 40$	1.3780e-2	-	3.2157e-1	-
$N_{el} = 320$	5.5115e-4	4.64	3.5923e-2	3.16

Table 6.15: Errors found for time dependent Maxwell calculations using the IP numerical flux on a periodic domain of size $1 \times 1 \times 1$. Piece-wise polynomials of order p on N_{el} elements were used. The order of convergence is also shown. DNF means the computer killed the simulation before it finished.

	$\|E - E_h\|_0$	Order	$\|E - E_h\|_{DG}$	Order
p=1				
$N_{el} = 40$	2.8315e-1	-	8.0153e-0	-
$N_{el} = 320$	7.8529e-1	-1.47	3.7502e-0	1.10
$N_{el} = 2560$	1.1001e-1	2.84	1.9003e-0	0.98
$N_{el} = 20480$	1.7225e-2	2.68	9.582e-1	0.99
p=2				
$N_{el} = 40$	8.7998e-1	-	2.6397e-0	-
$N_{el} = 320$	7.5250e-2	3.55	9.7377e-1	1.44
$N_{el} = 2560$	5.6153e-3	3.74	2.5441e-1	1.94
p=3				
$N_{el} = 40$	9.3049e-2	-	1.7236e-0	-
$N_{el} = 320$	1.2637e-2	2.88	1.9311e-1	3.16
$N_{el} = 2560$	4.4256e-4	4.84	2.5070e-2	2.95
p=4				
$N_{el} = 40$	8.4310e-2	-	2.2745e-1	-
$N_{el} = 320$	1.3479e-3	5.97	3.0483e-2	2.90

Table 6.16: Errors found for time harmonic Maxwell calculations using the Brezzi numerical flux on a periodic domain of size $1 \times 1 \times 1$. Piece-wise polynomials of order p on N_{el} elements were used. The order of convergence is also shown. DNF means the computer killed the simulation before it finished.

	$\|E-E_h\|_0$	Order	$\|E-E_h\|_{DG}$	Order
p=1				
$N_{el}=40$	5.1160e-1	-	8.4678e-0	-
$N_{el}=320$	7.9627e-1	-0.64	3.9248e-0	1.11
$N_{el}=2560$	1.1043e-1	2.85	1.9392e-0	1.02
$N_{el}=20480$	1.5009e-2	2.88	9.4445e-1	1.04
p=2				
$N_{el}=40$	8.9083e-1	-	3.0143e-0	-
$N_{el}=320$	7.5562e-2	3.56	1.0087e-0	1.58
$N_{el}=2560$	5.4341e-3	3.80	2.6719e-1	1.92
p=3				
$N_{el}=40$	9.4585e-2	-	1.7625e-0	-
$N_{el}=320$	1.2576e-2	2.91	1.9849e-1	3.15
$N_{el}=2560$	4.3435e-4	4.86	3.0089e-2	2.72
p=4				
$N_{el}=40$	8.4366e-2	-	2.4471e-1	-
$N_{el}=320$	1.2395e-3	6.09	3.2284e-2	2.92

Table 6.17: Errors found for time harmonic Maxwell calculations using the IP numerical flux on a periodic domain of size $1 \times 1 \times 1$. Piece-wise polynomials of order p on N_{el} elements were used. The order of convergence is also shown. DNF means the computer killed the simulation before it finished.

Eigenvalue	$N_{el}=40$		$N_{el}=320$		$N_{el}=2560$		$N_{el}=20480$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	2.0149e-1	-	1.0714e-1	0.91	2.2171e-2	2.27	3.4558e-3	2.68
$4\pi^2$	2.0149e-1	-	1.0714e-1	0.91	2.2171e-2	2.27	3.4558e-3	2.68
$4\pi^2$	2.0149e-1	-	1.0714e-1	0.91	2.2171e-2	2.27	3.4558e-3	2.68
$4\pi^2$	2.0149e-1	-	1.0714e-1	0.91	2.2171e-2	2.27	3.4558e-3	2.68
$4\pi^2$	2.0149e-1	-	1.0714e-1	0.91	2.2171e-2	2.27	3.4558e-3	2.68
$4\pi^2$	2.0149e-1	-	not found		not found		3.4558e-3	1.96
$4\pi^2$	5.6008e-1	-	1.0714e-1	2.39	2.2717e-2	2.27	3.4558e-3	2.68
$4\pi^2$	5.6008e-1	-	1.0714e-1	2.39	2.2717e-2	2.27	3.4558e-3	2.68
$4\pi^2$	5.6008e-1	-	1.0714e-1	2.39	2.2717e-2	2.27	3.4558e-3	2.68
$4\pi^2$	5.6008e-1	-	1.0714e-1	2.39	2.2717e-2	2.27	3.4558e-3	2.68
$4\pi^2$	5.6008e-1	-	1.0714e-1	2.39	2.2717e-2	2.27	3.4558e-3	2.68
$4\pi^2$	5.6008e-1	-	not found		not found		3.4558e-3	2.45

Table 6.18: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the IP numerical flux. Piece-wise linear basisfunctions on N_{el} elements were used. The order of convergence is also shown.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	1.1100e-2	-	2.9715e-3	1.90	1.1850e-4	4.65
$4\pi^2$	1.1100e-2	-	2.9715e-3	1.90	1.1850e-4	4.65
$4\pi^2$	1.1100e-2	-	2.9715e-3	1.90	1.1850e-4	4.65
$4\pi^2$	1.1100e-2	-	2.9715e-3	1.90	1.1850e-4	4.65
$4\pi^2$	1.1100e-2	-	2.9715e-3	1.90	1.1850e-4	4.65
$4\pi^2$	1.1100e-2	-	2.9715e-3	1.90	not found	
$4\pi^2$	8.5554e-2	-	2.9715e-3	4.85	1.1850e-4	4.65
$4\pi^2$	8.5554e-2	-	2.9715e-3	4.85	1.1850e-4	4.65
$4\pi^2$	8.5554e-2	-	2.9715e-3	4.85	1.1850e-4	4.65
$4\pi^2$	8.5554e-2	-	2.9715e-3	4.85	1.1850e-4	4.65
$4\pi^2$	8.5554e-2	-	2.9715e-3	4.85	1.1850e-4	4.65
$4\pi^2$	8.5554e-2	-	not found		not found	

Table 6.19: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the IP numerical flux. Piece-wise quadratic basisfunctions on N_{el} elements were used. The order of convergence is also shown.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	4.8825e-4	-	4.5000e-5	3.44	~5e-7	6.49
$4\pi^2$	4.8825e-4	-	4.5000e-5	3.44	~5e-7	6.49
$4\pi^2$	4.8825e-4	-	4.5000e-5	3.44	~5e-7	6.49
$4\pi^2$	4.8825e-4	-	4.5000e-5	3.44	~5e-7	6.49
$4\pi^2$	4.8825e-4	-	4.5000e-5	3.44	~5e-7	6.49
$4\pi^2$	4.8825e-4	-	4.5000e-5	3.44	~5e-7	6.49
$4\pi^2$	5.1875e-3	-	4.5000e-5	6.85	~5e-7	6.49
$4\pi^2$	5.1875e-3	-	4.5000e-5	6.85	~5e-7	6.49
$4\pi^2$	5.1875e-3	-	4.5000e-5	6.85	~5e-7	6.49
$4\pi^2$	5.1875e-3	-	4.5000e-5	6.85	~5e-7	6.49
$4\pi^2$	5.1875e-3	-	4.5000e-5	6.85	~5e-7	6.49
$4\pi^2$	5.1875e-3	-	4.5000e-5	6.85	~5e-7	6.49

Table 6.20: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the IP numerical flux. Piece-wise cubic basisfunctions on N_{el} elements were used. The order of convergence is also shown.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	~1.05e-5	-	~2.5e-7	5.39
$4\pi^2$	~1.05e-5	-	~2.5e-7	5.39
$4\pi^2$	~1.05e-5	-	~2.5e-7	5.39
$4\pi^2$	~1.05e-5	-	~2.5e-7	5.39
$4\pi^2$	~1.05e-5	-	~2.5e-7	5.39
$4\pi^2$	~1.05e-5	-	~2.5e-7	5.39
$4\pi^2$	~2.00e-4	-	~2.5e-7	9.64
$4\pi^2$	~2.00e-4	-	~2.5e-7	9.64
$4\pi^2$	~2.00e-4	-	~2.5e-7	9.64
$4\pi^2$	~2.00e-4	-	~2.5e-7	9.64
$4\pi^2$	~2.00e-4	-	~2.5e-7	9.64
$4\pi^2$	~2.00e-4	-	~2.5e-7	9.64

Table 6.21: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the IP numerical flux. Piece-wise quartic basisfunctions on N_{el} elements were used. The order of convergence is also shown.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$		$N_{el} = 20480$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	1.8443e-1	-	8.6770e-2	1.09	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	1.8443e-1	-	8.6770e-2	1.09	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	1.8443e-1	-	8.6770e-2	1.09	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	1.8443e-1	-	8.6770e-2	1.09	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	1.8443e-1	-	8.6770e-2	1.09	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	1.8443e-1	-	8.6770e-2	1.09	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	3.6352e-1	-	8.6770e-2	2.07	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	3.6352e-1	-	8.6770e-2	2.07	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	3.6352e-1	-	8.6770e-2	2.07	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	3.6352e-1	-	8.6770e-2	2.07	2.2168e-2	1.97	5.5603e-3	2.00
$4\pi^2$	3.6352e-1	-	8.6770e-2	2.07	2.2168e-2	1.97	not found	
$4\pi^2$	3.6352e-1	-	not found		not found		not found	

Table 6.22: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the Brezzi numerical flux. Piece-wise linear basisfunctions on N_{el} elements were used. The order of convergence is also shown.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$		$N_{el} = 2560$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	9.5790e-3	-	2.7493e-3	1.80	1.8225e-4	3.92
$4\pi^2$	9.5790e-3	-	2.7493e-3	1.80	1.8225e-4	3.92
$4\pi^2$	9.5790e-3	-	2.7493e-3	1.80	1.8225e-4	3.92
$4\pi^2$	9.5790e-3	-	2.7493e-3	1.80	1.8225e-4	3.92
$4\pi^2$	9.5790e-3	-	2.7493e-3	1.80	1.8225e-4	3.92
$4\pi^2$	9.5790e-3	-	2.7493e-3	1.80	1.8225e-4	3.92
$4\pi^2$	6.7044e-2	-	2.7493e-3	4.61	1.8225e-4	3.92
$4\pi^2$	6.7044e-2	-	2.7493e-3	4.61	1.8225e-4	3.92
$4\pi^2$	6.7044e-2	-	2.7493e-3	4.61	1.8225e-4	3.92
$4\pi^2$	6.7044e-2	-	2.7493e-3	4.61	1.8225e-4	3.92
$4\pi^2$	6.7044e-2	-	2.7493e-3	4.61	1.8225e-4	3.92
$4\pi^2$	6.7044e-2	-	2.7493e-3	4.61	1.8225e-4	3.92

Table 6.23: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the Brezzi numerical flux. Piece-wise quadratic basisfunctions on N_{el} elements were used. The order of convergence is also shown.

Eigenvalue	$N_{el} = 40$		$N_{el} = 320$	
	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order	$\frac{\|\lambda-\lambda_h\|}{\lambda}$	Order
$4\pi^2$	4.6575e-4	-	˜4.7e-5	3.31
$4\pi^2$	4.6575e-4	-	˜4.7e-5	3.31
$4\pi^2$	4.6575e-4	-	˜4.7e-5	3.31
$4\pi^2$	4.6575e-4	-	˜4.7e-5	3.31
$4\pi^2$	4.6575e-4	-	˜4.7e-5	3.31
$4\pi^2$	4.6575e-4	-	˜4.7e-5	3.31
$4\pi^2$	4.3643e-3	-	˜4.7e-5	6.54
$4\pi^2$	4.3643e-3	-	˜4.7e-5	6.54
$4\pi^2$	4.3643e-3	-	˜4.7e-5	6.54
$4\pi^2$	4.3643e-3	-	˜4.7e-5	6.54
$4\pi^2$	4.3643e-3	-	˜4.7e-5	6.54
$4\pi^2$	4.3643e-3	-	˜4.7e-5	6.54

Table 6.24: Exact eigenvalues and relative errors of the computed eigenvalues for the eigenproblem on a periodic domain of size $1 \times 1 \times 1$ using the Brezzi numerical flux. Piece-wise cubic basisfunctions on N_{el} elements were used. The order of convergence is also shown.

6.2 Wave-vector

The next test is to check if the DG-Max code works for all choices of k. The easiest test is a cube where ϵ_r is constant, modelling a homogeneous medium. In this case a subset of the possible eigenmodes can be computed exactly. Assume $\tilde{E}$ is constant and orthogonal to k. In this case (4.6) can be reduced to

$$\mathrm{i}k \times \mathrm{i}k \times \tilde{E} = \omega^2 \tilde{E}.$$

So $\|k\|^2 = \omega^2$. From this relation Bloch-Floquet theory can be used to show that at least $\|k + 2\pi l\| = \omega^2 \quad \forall l \in \mathbb{Z}^3$ are eigenvalues of (4.6). In [18] it is shown for the unbounded homogeneous case that $\|k\|^2 = \omega^2$ are the only eigenvalues, but their translation argument breaks in the case of infinite repetitions of a bounded domain. This makes it all the more important to check that there are no spurious modes introduced by the boundary.

Figure 6.1 shows that all eigenvalues are found and that no spurious eigenvalues are found. This shows that the DG-Max code handles the wave-vector correctly.

6.3 Variable ϵ_r

The next goal is to be able to perform simulations for cases where ϵ_r is only piece-wise constant. There are some results on how the band structure should look like available in [18] and [7]. Three of these results are tested using DG-Max. They have in common that their symmetry properties are such that in three dimensions the cube is the most intuitive unit cell.

The first domain consists of an alternating stack of two different materials of thicknes d_1 and $d_2 = 1 - d_1$. Figure 6.3 shows the band structure for a stack where $d_1 = d_2 = 0.5$, $\epsilon_1 = 1$ and $\epsilon_2 = 13$. This domain is such that the interface between ϵ_1 and ϵ_2 is smooth and it is easy to model with the default grid generation strategy.

The second domain consists of a square lattice of square columns. The columns have width w_1. Boffi [7] documents the case with $w_1 = 0.4$, $\epsilon_{column} = 1$ and $\epsilon_{bulk} = 8.9$ (alumina for the desired frequency ranges). This domain can also be modelled by the default grid, but it has sharp corners where singularities may appear.

The third domain consists of a square lattice of cylinders. The cylinders have radius r_1. Joanopoulos [18] documents the case with $r_1 = 0.2$, $\epsilon_{column} = 8.9$ and $\epsilon_{bulk} = 1$. This domain again features a smooth bi-material interface, but it requires a grid specially designed to have element boundaries at the bi-material interface.

In all cases the unit domain is chosen such that the symmetry properties of the domain are optimised, so the centres of the cylinders are in the centre of the unit cell and for the alternating stack the zone with high permittivity is split into two zones, each with width $d_2/2$ on opposite ends of the domain.

Figures 6.3 and 6.4 compare the DGFEM results for the multilayer stack with established numerical results. Figure 6.4 is generated using MPB [18]. Figure 6.3 is made using DG-Max. A small quantitative difference can be seen for higher frequency modes. This is likely because the finite element solution is not sufficiently converged yet. Other than this the two computational results match well.

For the case of the square columns it should be noted that the reported domain configurations do not match the domain configurations used for the creation of the shown data (it is expected that the graph shown in [7] is constructed using $w_1 = 0.8$). For the purpose of comparison

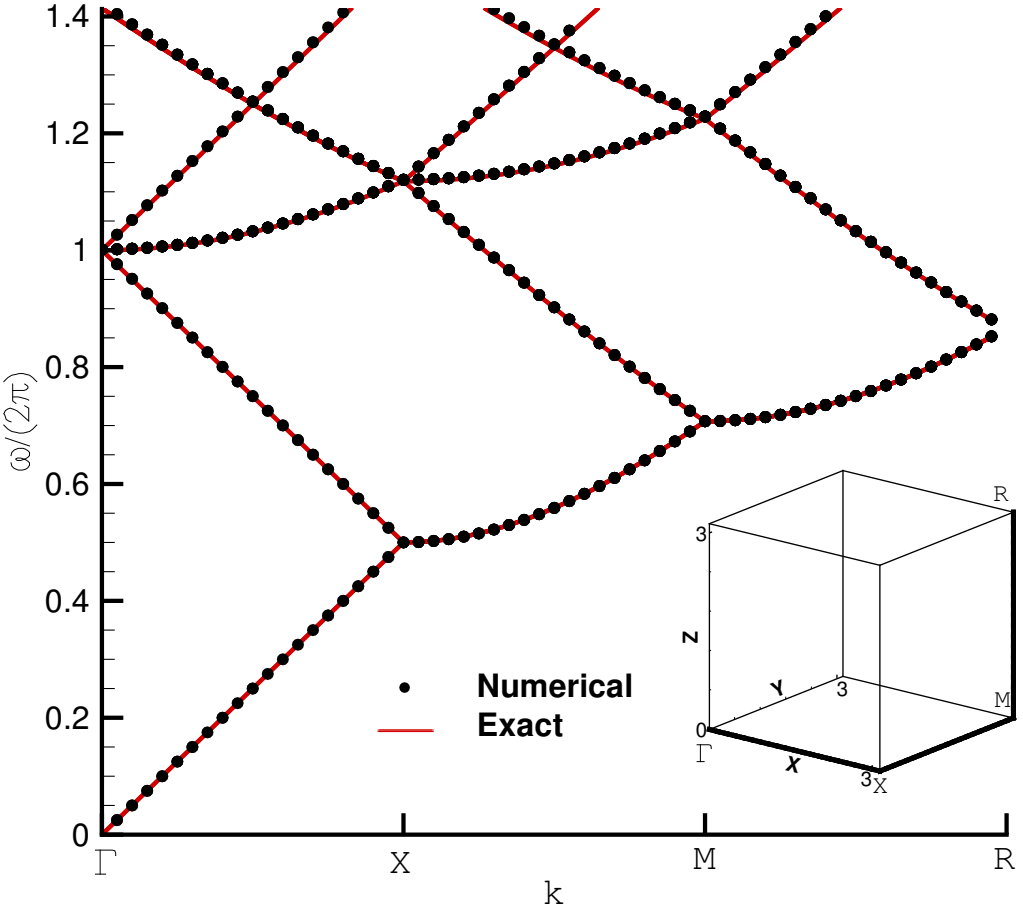


Figure 6.1: First few eigenfrequencies of vacuum in a periodic cube as a function of the wavevector. Computed using the Brezzi numerical flux on 320 quadratic elements. Also plotted are the predicted eigenvalues based on the guess that $\tilde{E}$ is constant and orthogonal to k.

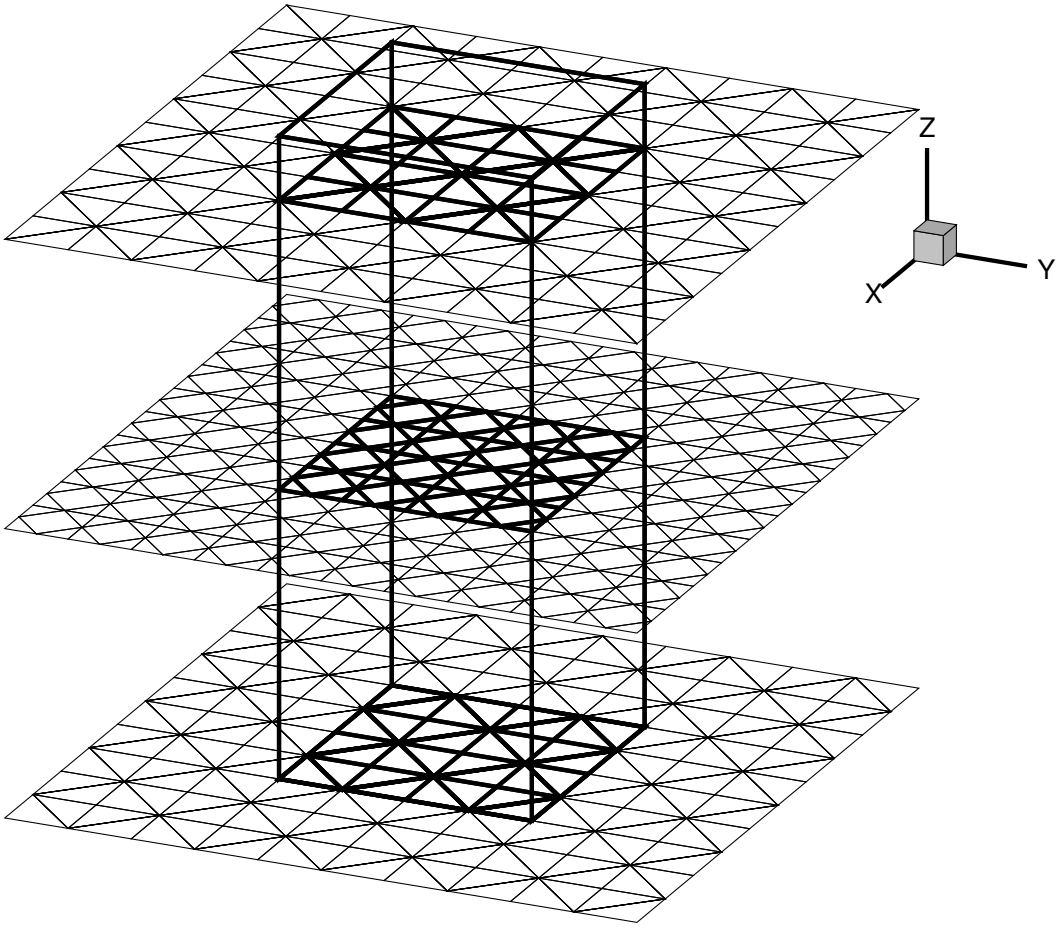


Figure 6.2: Mesh used for computations of the square grid of square columns. For clarity, bold lines were used in the region of lower dielectric permittivity. The three cutting plains show the mesh at the bottom of a layer (botton), at the middle of a layer (middle) and at the top of a layer (top). These layers are rotated and stacked to fill the domain using a total of 10 layers.

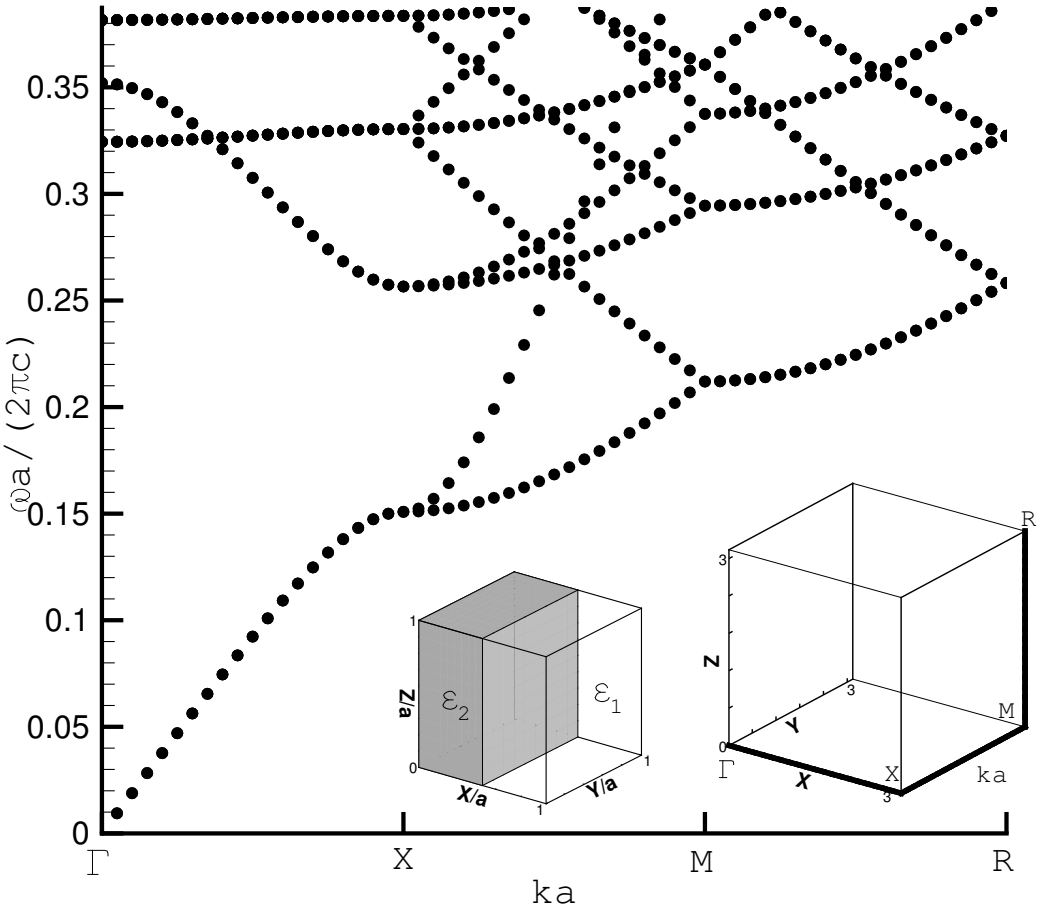


Figure 6.3: First few eigenfrequencies of a multi-layer stack. Computed using the IP numerical flux on a mesh with 320 cubic elements. A unit domain in the periodic medium is also presented. The grey area has a dielectric permittivity of 13. The white area has a dielectric permittivity of 1.

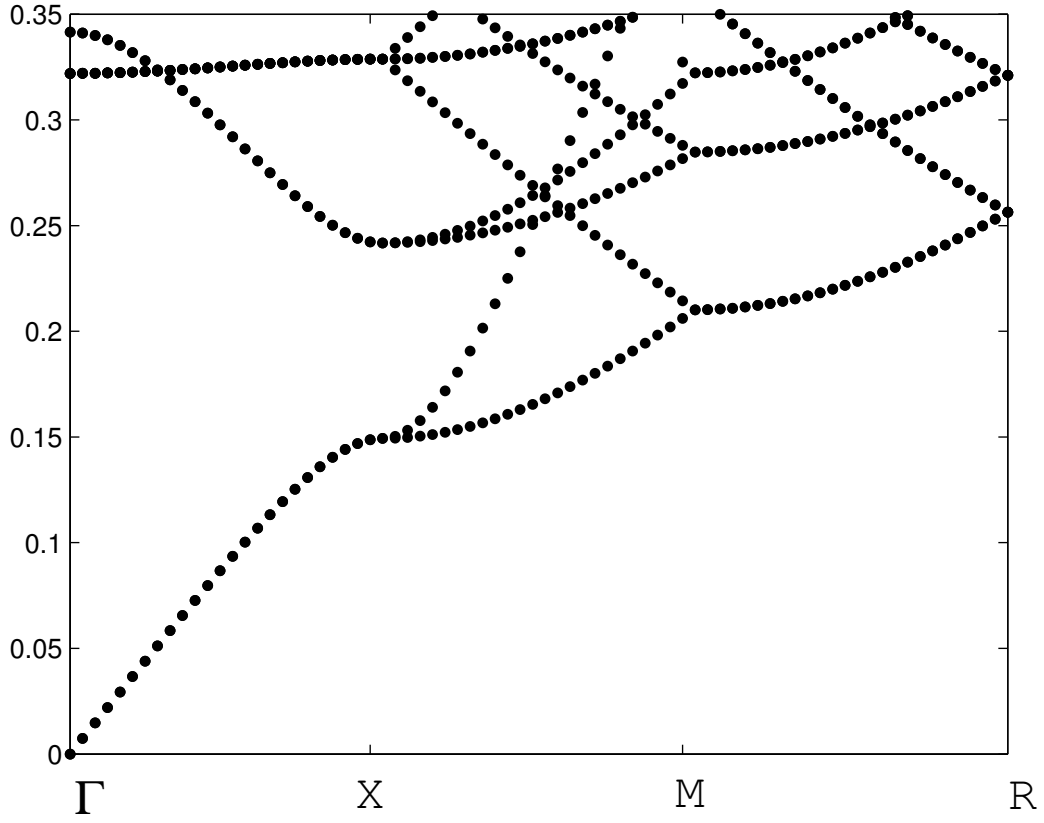


Figure 6.4: First few eigenfrequencies of a multi-layer stack. Reference picture computed with MPB using default settings. This picture is meant to display solutions of the same problem as the one described in Figure 6.3.

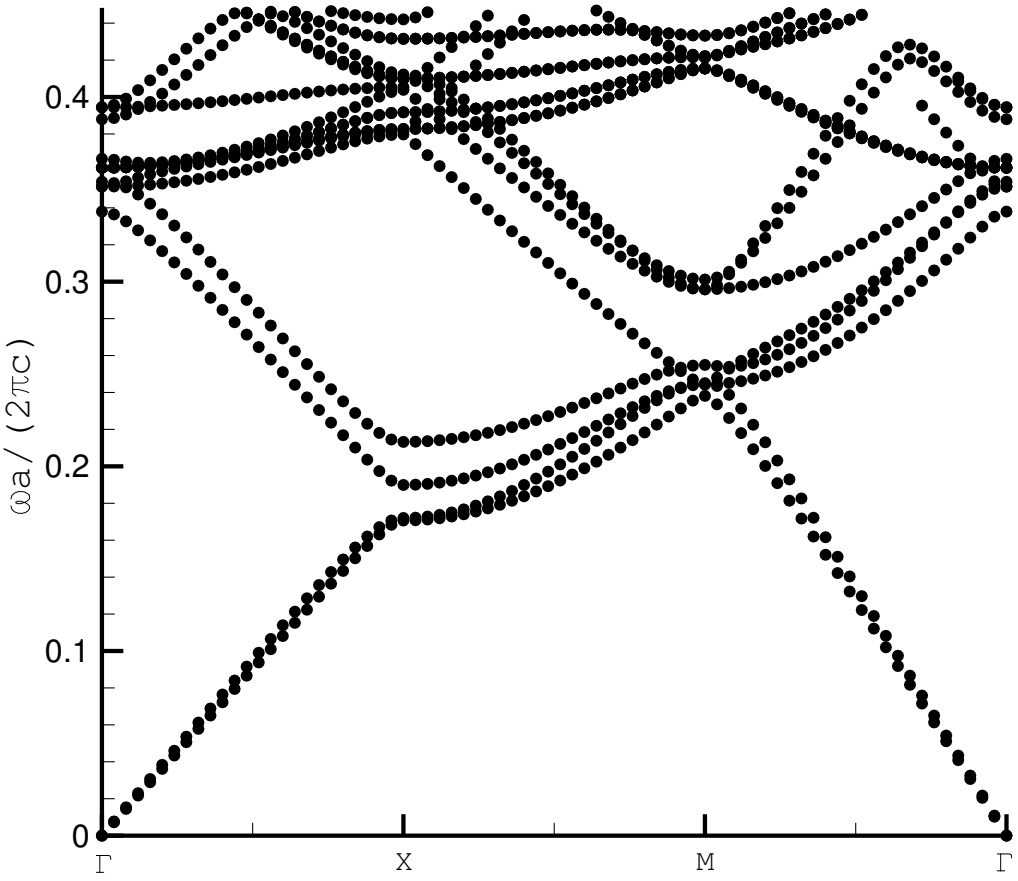


Figure 6.5: Dispersion relation computed using MPB that matches the parameters given in Boffi et al. [7]. Note that the distinction between TE and TM modes was dropped as it is not relevant for comparison purposes.

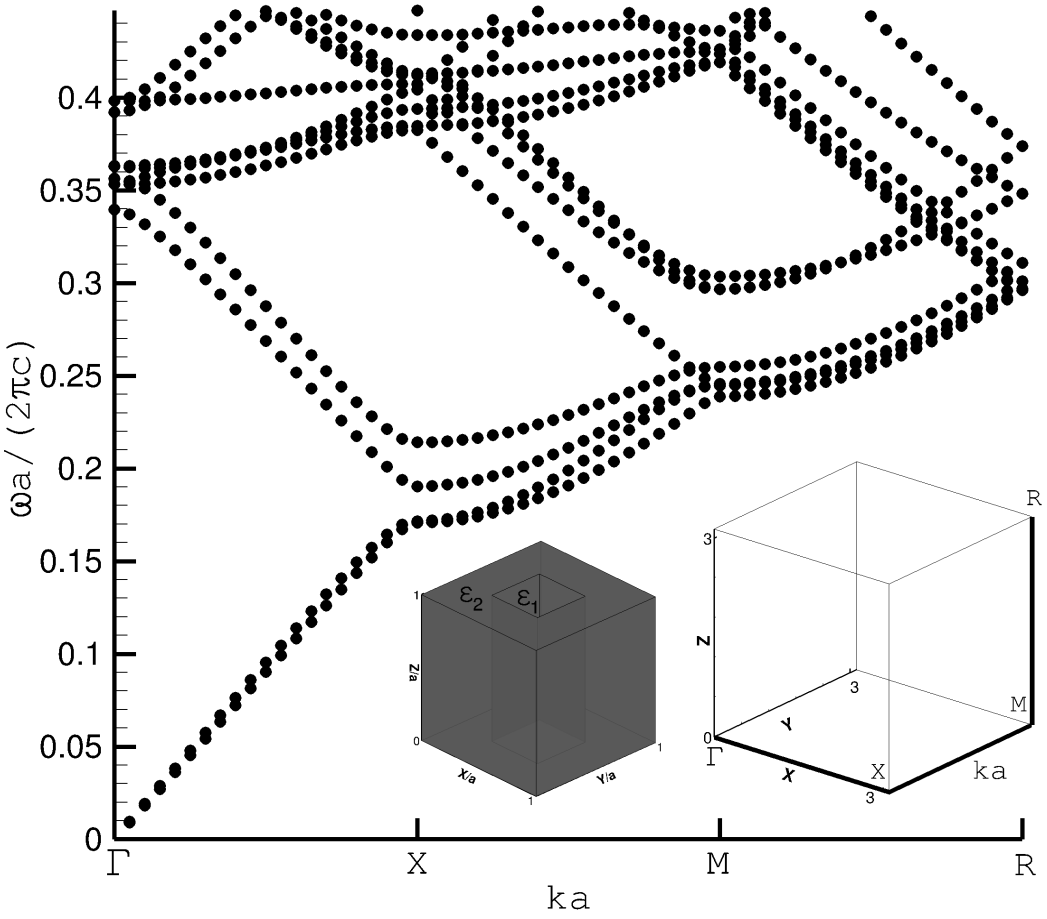


Figure 6.6: First few eigenfrequencies of a square grid of square columns. Computed using the Brezzi numerical flux on a mesh with 5000 linear elements. A unit domain in the periodic medium is also presented. The grey area has a dielectric permittivity of 8.9. The white, transparent area has a dielectric permittivity of 1.

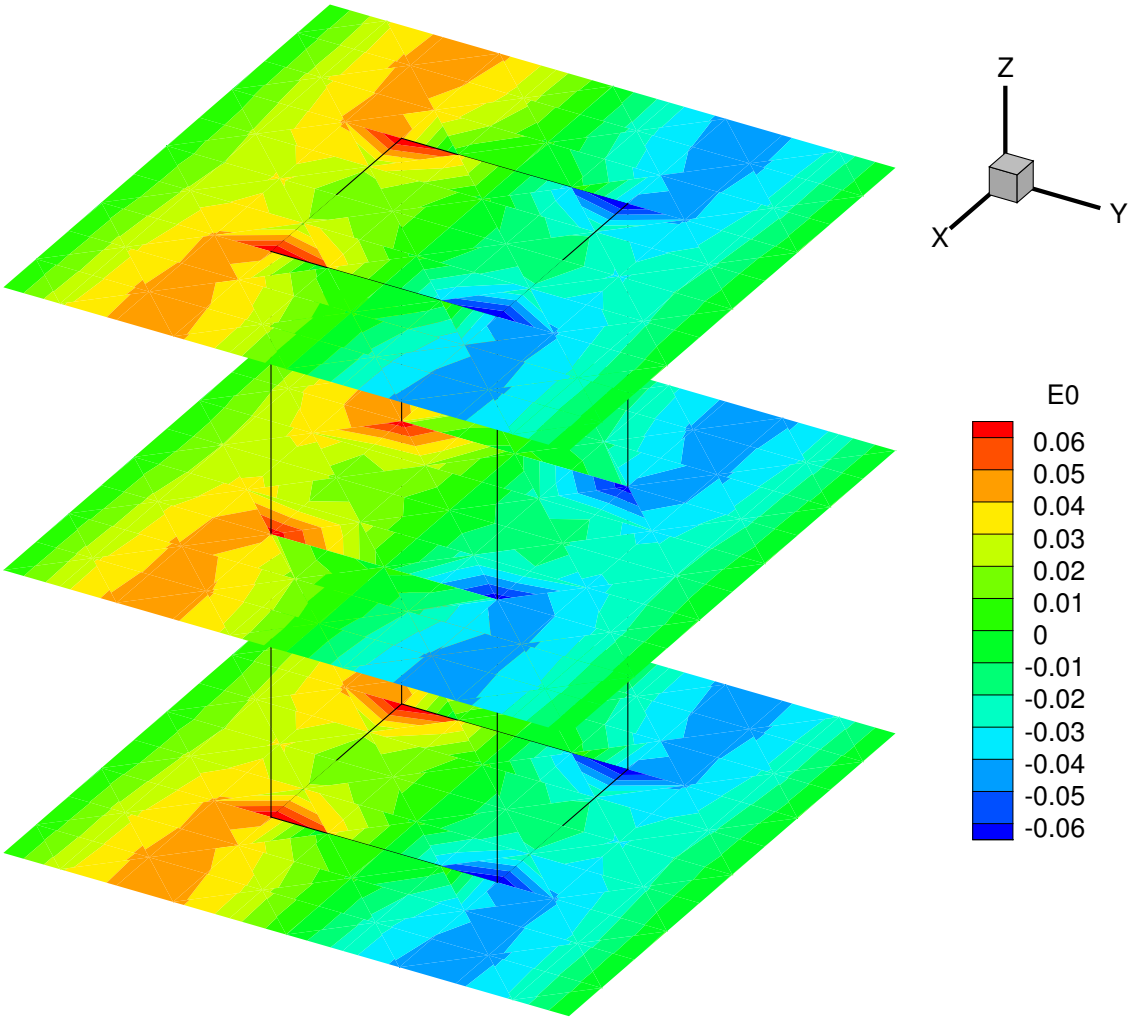


Figure 6.7: One of the eigenfunctions of the square grid of square columns. This is only one component of the eigenfunction. The full eigenfunction is of the form $(E0\,(x,y,z)\,, E0\,(y,x,z)\,, 0)^T$. The solution is constant in the z-direction.

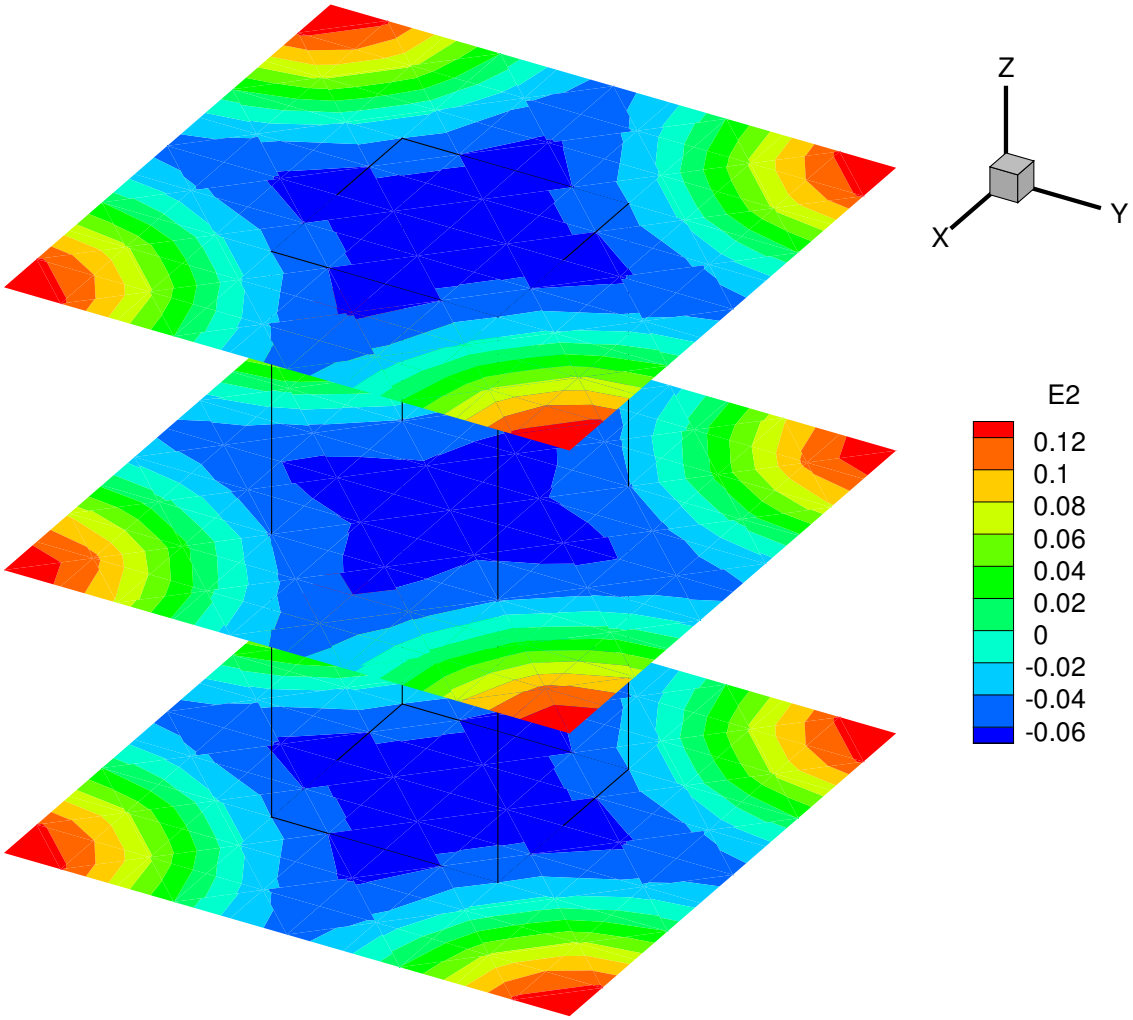


Figure 6.8: One of the eigenfunctions of the square grid of square columns. This is only one component of the eigenfunction. The full eigenfunction is of the form $(0, 0, E2\,(x, y, z))^T$. The solution is constant in the z-direction.

the numerical data were recreated using MPB. This comparison is in excellent agreement with the numerical data presented in Figure 6.6. Some extra pictures are included to show a few eigenfunctions at the point Γ.

For the case with the cylinder a mesh was generated using the commercial external tools Rhinoceros [14] and Centaur [1]. It turns out that solving the full eigenvalue-problem to find the dispersion relation is too time-consuming with the available software implementation on the current hardware. Finding one harmonic response is much faster so a picture of a harmonic response is included to give an idea how the solutions on this domain look like.

From the results obtained so far no discrepancies with reference solutions were found, so it can be concluded that the DG-Max code works satisfactory, but is slow for the cases considered so far. More thorough testing should to done before DG-Max can be trusted to work on unstructured grids, but on structured grids it is reasonably safe to assume it works satisfactory.

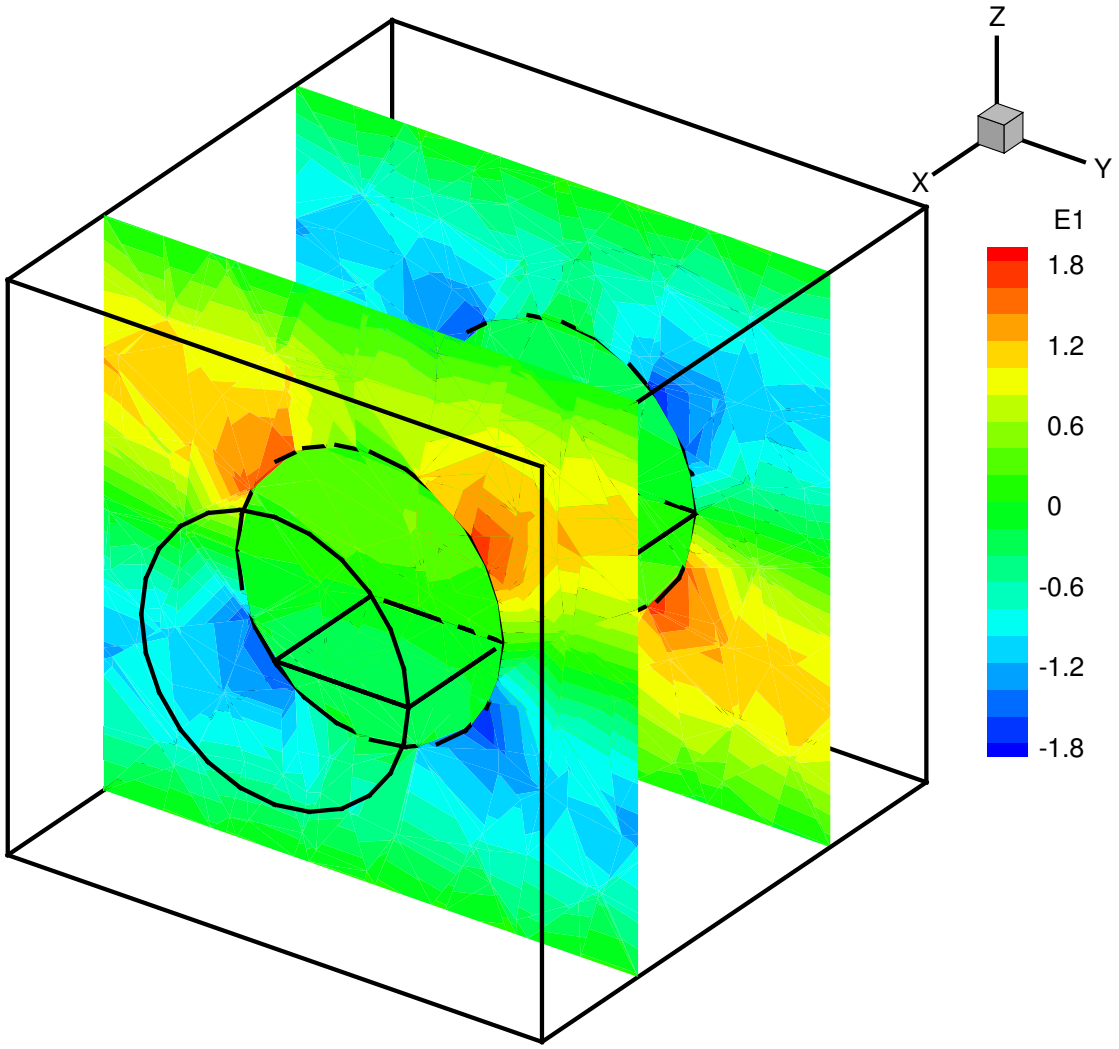


Figure 6.9: One component of the harmonic response with the source $i\omega J$ given by (6.1) of a domain with a cylinder of radius $r_1 = 0.25$. The simulation was done using 3257 linear elements. The cylinder was drawn with a radial plane to emphasize its 3 dimensional nature. The solution is harmonic in the x-direction. The planes drawn show the extremes of the solution.

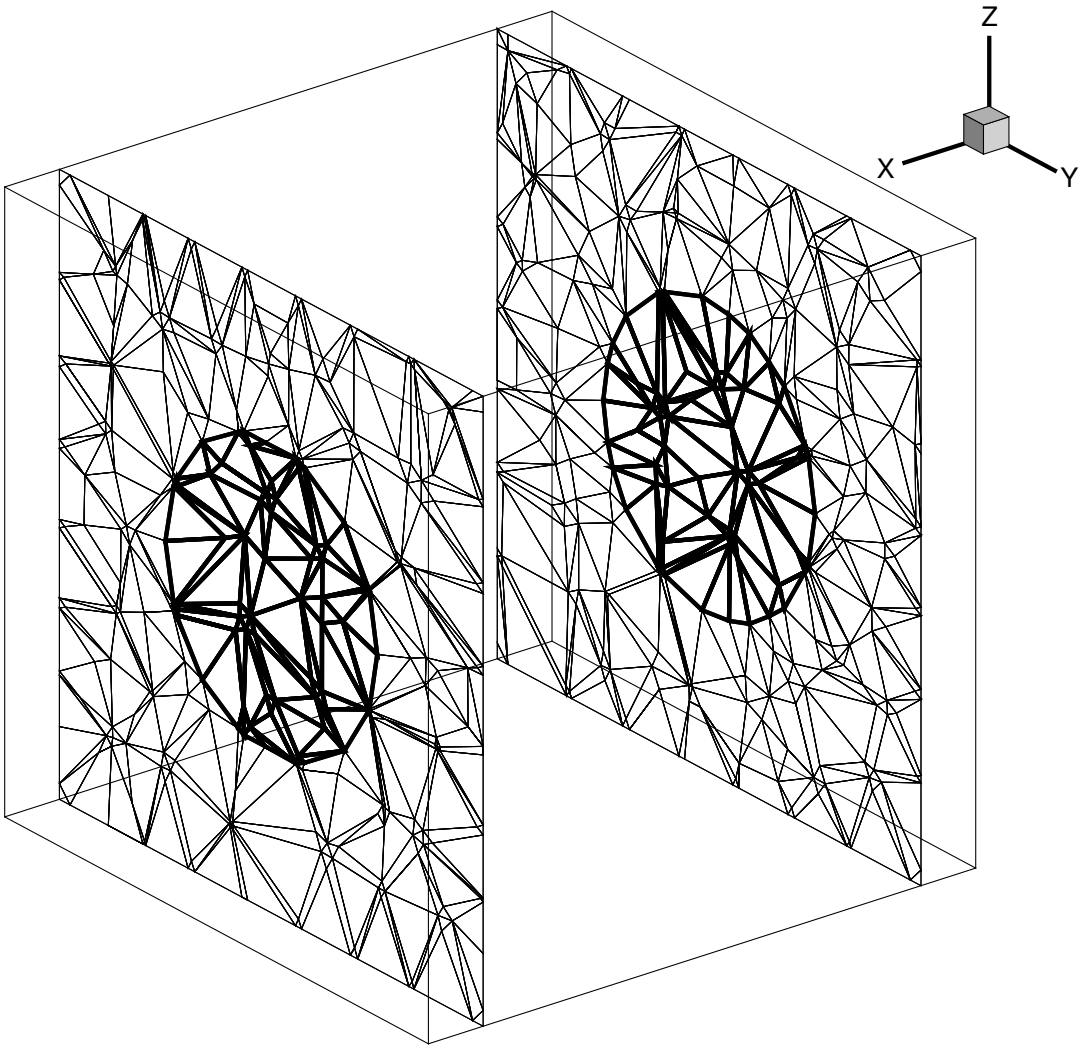


Figure 6.10: Mesh used for computations of the grid of cylinders. For clarity, bold lines were used in the region of higher dielectric permittivity. The mesh is unstructured.

6.4 LDOS

As a first test some simulations were performed with the goal of computing the (L)DOS of the uniform periodic domain. This domain models an infinite uniform medium. This means the domain and the solution should be completely invariant under translations or rotations and the expressions $n_p^T \left(\psi\psi^*\right) n_p$ and $|\psi|^2$ both reduce to constants. The domain of the integral $\int \delta(\omega - \omega_{n,k})\,\mathrm{d}k$ then can be reduced to the surface area of a sphere with radius ω. So $\rho \propto \omega^2$. For the purposes of finding the LDOS the eigenvalue solver is set to target the smallest positive eigenvalues using the standard strategy. The two physical constant solutions are manually added.

These simulations appear to show a converging behaviour towards a homogeneous quadratic polynomial. This polynomial is not the exact solution, however. This is caused by a very heavy reliance on having found all the eigenmodes in the spectral interval of interest. As shown in earlier tables the eigenvalues solver currently in use occasionally underestimates the multiplicity of some eigenvalues by 1 or 2. This effect usually disappears when sufficient elements are used and every physical eigenvalue is always found at least once, so it is not considered to be a very large problem when computing the dispersion relation. For the (L)DOS, however, if even one or two eigenvalues are missing, this breaks the numbering of the $\omega_{n,k}$ for some k-points, introducing a large error term that is not representative for the error of the LDOS computation.

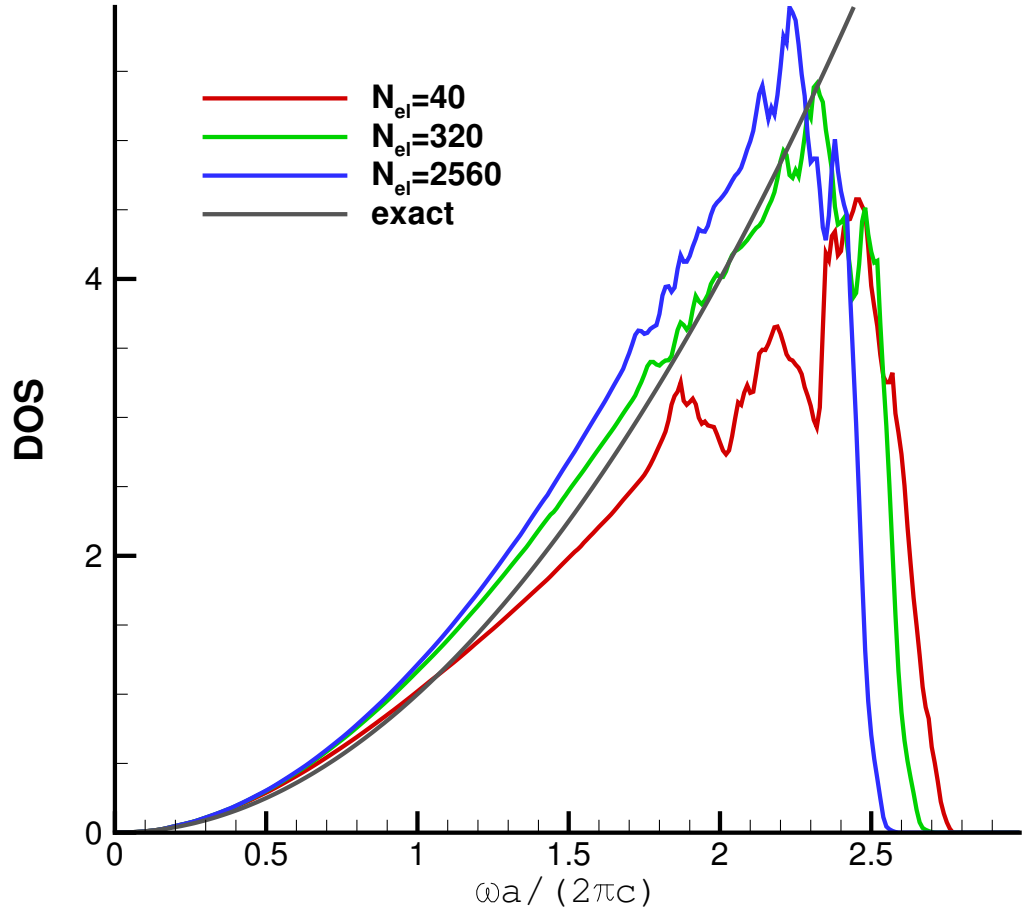


Figure 6.11: Density of states (DOS) with N_{el} elements and linear basis functions in a uniform periodic domain using 165 k-points. The graph converges to a slight overestimation of the DOS. This is probably caused by the missing eigenvalues.

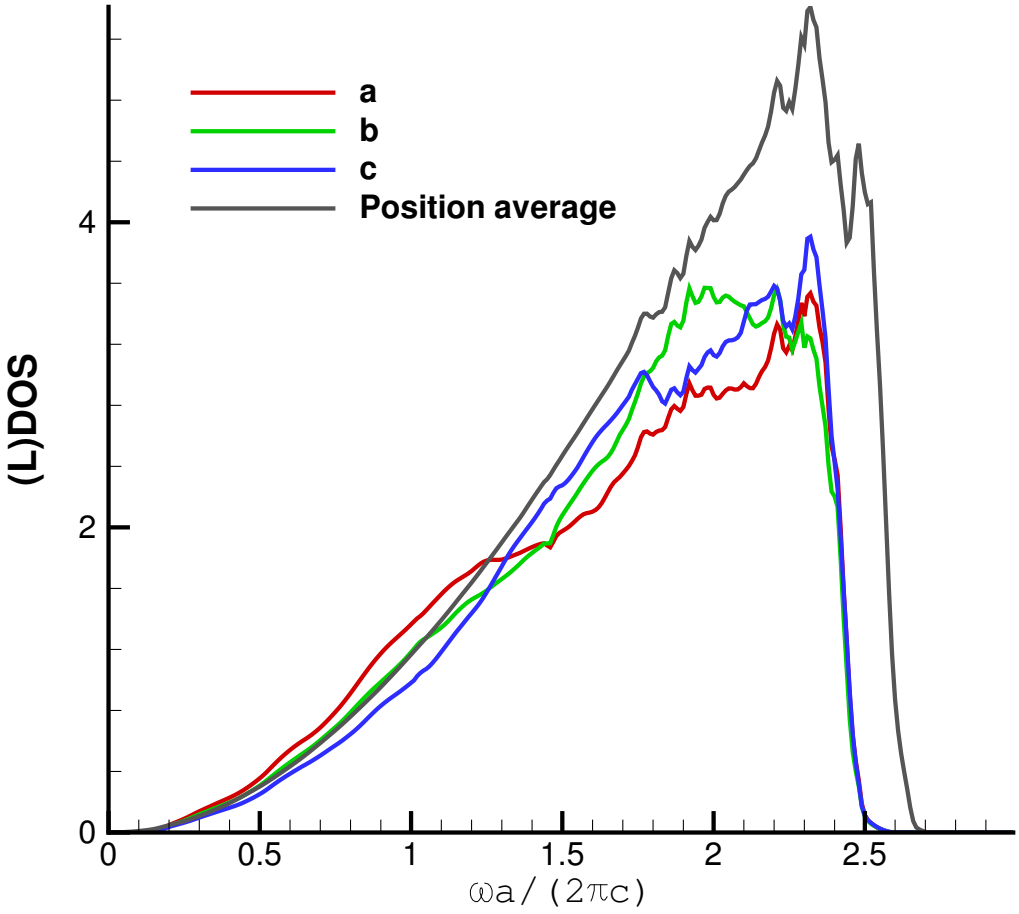


Figure 6.12: Local densitiy of states (LDOS) at three arbitrary locations a, b and c inside the unit domain. Piecewise linear basis functions on 320 elements and 165 k-points were used. The position averaged LDOS (actually the DOS) is also presented. The convergence of the LDOS is necessarily worse than the convergence of the DOS since the LDOS not only depends on the eigenvalues, but also on the eigenfunctions.

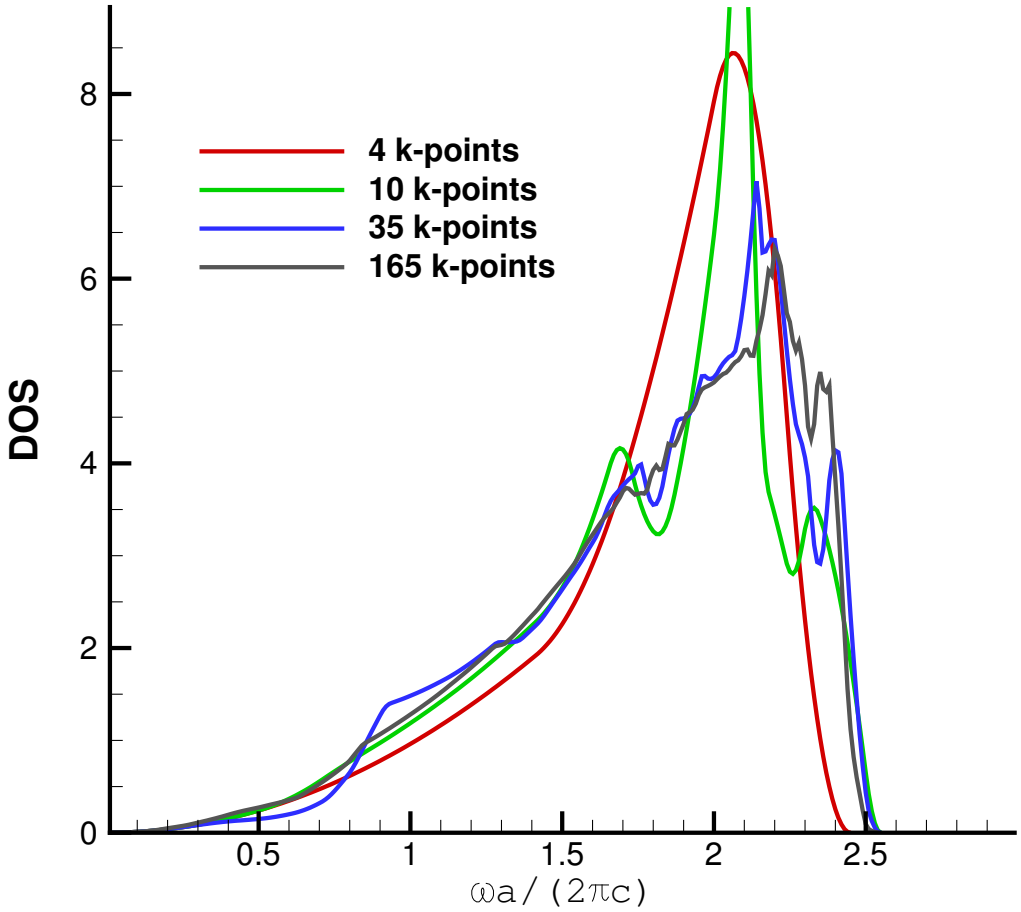


Figure 6.13: DOS with 320 cubic elements in a uniform periodic domain. Each increment in the number of k-points doubles the number of intervals in every cardinal direction.

Chapter 7

Notes on implementation

The previous chapters present, from a mathematical point of view, a complete description of the steps that need to be taken to solve the Maxwell equations numerically with a discontinuous Galerkin finite element method. When attempting to implement the algorithms described above it turns out there are some essential details hidden in the formulations. This chapter serves to highlight those details and to present strategies for dealing with them.

7.1 Quadrature

In order to evaluate integrals numerically one needs a quadrature rule. That is, a rule of the form

$$\int_K f(\xi)\ \mathrm{d}\xi \approx \sum_{i=1}^{N} w_i f(x_i),$$

where N is the number of mantissa used. It is possible to construct this approximation to be exact up to polynomial order p by taking N sufficiently large and using basis functions in $\mathcal{P}_p(K)$ that are of sufficiently high polynomial order. For example take $f(x) \in \{1, x, y, z, xy, \ldots, z^p\}$, then solve the resulting system against w_i and x_i. This is not easy, but has already been extensively done. See for an overview [10, 11].

Since the basis functions appear quadratically in the weak formulation the quadrature rule should be exact up to at least a polynomial order of $2p$. It turns out that quadrature rules for odd polynomial order are better known, so we use a quadrature rule that is exact up to order $2p+1$. This has the added benefit that this extra accuracy gives a better approximation of the error when an exact solution is known, especially for the time-dependent case near $t = 0$. In the DG-Max code f is evaluated for an entire element matrix in one go at every mantissa. The weights and additions are then applied using vectorised routines.

7.2 Basisfunctions

The basisfunctions themselves can be implemented as they are presented. Their curl needs, however, an application of the chain rule. The following relations were used in the implementation

$$L'_p = \begin{cases} 0 & p = 0, \\ 1 & p = 1, \\ \frac{2p-1}{p}\left(L_{p-1} + xL'_{p-1}\right) - \frac{p-1}{p}L'_{p-2} & p \geq 2, \end{cases}$$

$$\phi'_{p,AB} = \begin{cases} \nabla\lambda_B \times \nabla\lambda_A & p = 0, \\ 0 & p = 1, \\ \frac{2p-1}{p}L'_{p-1}(\lambda_B - \lambda_A)\nabla(\lambda_B - \lambda_A) \times \phi_{1,AB} & \\ -\frac{p-1}{p}L'_{p-2}(\lambda_B - \lambda_A)\nabla(\lambda_B - \lambda_A) \times \phi_{0,AB} & p \geq 2, \\ -\frac{p-1}{p}L_{p-2}(\lambda_B - \lambda_A)\nabla\lambda_B \times \nabla\lambda_A & \end{cases}$$

$$\begin{aligned}
\phi'_{p,e,D,AB} =& \left(\lambda_B L_p(\lambda_B - \lambda_A) - \lambda_A\lambda_B L'_p(\lambda_B - \lambda_A)\right)\nabla\lambda_A \times \nabla\lambda_C \\
&+ \left(\lambda_A L_p(\lambda_B - \lambda_A) + \lambda_A\lambda_B L'_p(\lambda_B - \lambda_A)\right)\nabla\lambda_B \times \nabla\lambda_C, \\
\phi'_{0,D,l,m} =& (\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) - \lambda_A\lambda_B\lambda_C L'_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) \\
&- \lambda_A\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L'_m(\lambda_C - \lambda_A))\nabla\lambda_A \times (v_B - v_A) \\
&+ (\lambda_A\lambda_C L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) + \lambda_A\lambda_B\lambda_C L'_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A))\nabla\lambda_B \times (v_B - v_A) \\
&+ (\lambda_A\lambda_B L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) + \lambda_A\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L'_m(\lambda_C - \lambda_A))\nabla\lambda_C \times (v_B - v_A), \\
\phi'_{1,D,l,m} =& (\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) - \lambda_A\lambda_B\lambda_C L'_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) \\
&- \lambda_A\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L'_m(\lambda_C - \lambda_A))\nabla\lambda_A \times (v_C - v_A) \\
&+ (\lambda_A\lambda_C L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) + \lambda_A\lambda_B\lambda_C L'_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A))\nabla\lambda_B \times (v_C - v_A) \\
&+ (\lambda_A\lambda_B L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) + \lambda_A\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L'_m(\lambda_C - \lambda_A))\nabla\lambda_C \times (v_C - v_A), \\
\phi'_{F,D,l,m} =& (\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) - \lambda_A\lambda_B\lambda_C L'_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) \\
&- \lambda_A\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L'_m(\lambda_C - \lambda_A))\nabla\lambda_A \times \nabla\lambda_D \\
&+ (\lambda_A\lambda_C L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) + \lambda_A\lambda_B\lambda_C L'_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A))\nabla\lambda_B \times \nabla\lambda_D \\
&+ (\lambda_A\lambda_B L_l(\lambda_B - \lambda_A) L_m(\lambda_C - \lambda_A) + \lambda_A\lambda_B\lambda_C L_l(\lambda_B - \lambda_A) L'_m(\lambda_C - \lambda_A))\nabla\lambda_C \times \nabla\lambda_D, \\
\phi'_{K,k,l,m,n} =& (\lambda_1\lambda_2\lambda_3 L_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0) \\
&-\lambda_0\lambda_1\lambda_2\lambda_3 L'_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0) \\
&-\lambda_0\lambda_1\lambda_2\lambda_3 L_l(\lambda_1 - \lambda_0) L'_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0) \\
&- \lambda_0\lambda_1\lambda_2\lambda_3 L_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L'_n(\lambda_3 - \lambda_0))\nabla\lambda_0 \times v_k \\
&+ (\lambda_0\lambda_2\lambda_3 L_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0) \\
&+ \lambda_0\lambda_1\lambda_2\lambda_3 L'_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0))\nabla\lambda_1 \times v_k \\
&+ (\lambda_0\lambda_1\lambda_3 L_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0) \\
&+ \lambda_0\lambda_1\lambda_2\lambda_3 L_l(\lambda_1 - \lambda_0) L'_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0))\nabla\lambda_2 \times v_k \\
&+ (\lambda_0\lambda_1\lambda_2 L_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L_n(\lambda_3 - \lambda_0) \\
&+ \lambda_0\lambda_1\lambda_2\lambda_3 L_l(\lambda_1 - \lambda_0) L_m(\lambda_2 - \lambda_0) L'_n(\lambda_3 - \lambda_0))\nabla\lambda_3 \times v_k,
\end{aligned}$$

where functions of x have lost their explicit x dependance to save space and, for vector valued functions, $\cdot' = \nabla \times \cdot$ is used to denote the curl.

7.3 Weak formulation

The semi-discrete formulations presented in this report are still a bit too complicated for direct implementation. To help alleviate this problem the equations were split into several terms that add up to the full equations. First of all consider the elements (or faces) one by one and remove the basisfunctions that are zero on that element from the summation over i. This will lead to several much smaller matrices, the element matrices, that will still add up to the full linear system.

For the mass matrix $\mathcal{M}$ there is only one term in the bilinear form: $(\epsilon_r \phi_i, \phi_j)_K$, which can be straightforwardly computed with the available quadrature rules.

Next, for the element contributions to the stiffness matrix $\mathcal{S}$ there is also only one term: $(\nabla \times \phi_i, \nabla \times \phi_j)_K$. While this term is not quite as easy as the contribution to the mass matrix it does not present any serious problems.

Now only the face contributions to the stiffness matrix are left. It pays to split these into a part that is common to both the IP-flux and the Brezzi-flux and a part that is unique to only one of them. The common part is

$$\begin{aligned}
&\int_F -[\![\phi_i(x)]\!]_T \cdot \{\!\{\nabla \times \phi_j(x)\}\!\} - \{\!\{\nabla \times \phi_i(x)\}\!\} \cdot [\![\phi_j(x)]\!]_T \,\mathrm{d}s \\
=&\int_F -\frac{1}{2}\left(n^L \times \phi_i^L + n^R \times \phi_i^R\right) \cdot \left(\nabla \times \phi_j^L + \nabla \times \phi_j^R\right) \\
&\quad - \frac{1}{2}\left(n^L \times \phi_j^L + n^R \times \phi_j^R\right) \cdot \left(\nabla \times \phi_i^L + \nabla \times \phi_i^R\right) \mathrm{d}s.
\end{aligned}$$

At most one of ϕ^L and ϕ^R will be nonzero for any i or j so remove the zero contributions and put the rest in a block structured matrix that looks like

$$\begin{pmatrix} S^{LL} & S^{LR} \\ S^{RL} & S^{RR} \end{pmatrix},$$

whose entries can be computed as

$$\begin{aligned}
S_{ij}^{LL} &= \int_F -\frac{1}{2}\left(\left(n \times \phi_i^L\right) \cdot \left(\nabla \times \phi_j^L\right) + \left(n \times \phi_j^L\right) \cdot \left(\nabla \times \phi_i^L\right)\right) \mathrm{d}s \\
S_{ij}^{RL} &= \int_F -\frac{1}{2}\left(\left(-n \times \phi_i^R\right) \cdot \left(\nabla \times \phi_j^L\right) + \left(n \times \phi_j^L\right) \cdot \left(\nabla \times \phi_i^R\right)\right) \mathrm{d}s \\
S_{ij}^{LR} &= \int_F -\frac{1}{2}\left(\left(n \times \phi_i^L\right) \cdot \left(\nabla \times \phi_j^R\right) + \left(-n \times \phi_j^R\right) \cdot \left(\nabla \times \phi_i^L\right)\right) \mathrm{d}s \\
S_{ij}^{RR} &= \int_F -\frac{1}{2}\left(\left(-n \times \phi_i^R\right) \cdot \left(\nabla \times \phi_j^R\right) + \left(-n \times \phi_j^R\right) \cdot \left(\nabla \times \phi_i^R\right)\right) \mathrm{d}s.
\end{aligned}$$

Take care that the right element requires $-n$ instead of n because the normal vector must point outward, while hpGEM provides a normal vector that point towards the right element. Also note that hpGEM provides a normal vector scaled by the size of the face because this is more

beneficial for most differential equations. This vector is scaled to be of unit length before it is used in these equations.

The part that is used only by the IP numerical flux can receive much the same treatment as the common part

$$\begin{aligned}
&\int_F a_F [\![\phi_i(x)]\!]_T \cdot [\![\phi_j(x)]\!]_T \,\mathrm{d}s \\
&= \int_F a_F \left(n^L \times \phi_i^L + n^R \times \phi_i^R\right) \cdot \left(n^L \times \phi_j^L + n^R \times \phi_j^R\right) \mathrm{d}s \\
&= \int_F a_F (n \times \phi_i) \cdot (n \times \phi_j) \,\mathrm{d}s. \qquad (7.1)
\end{aligned}$$

In (7.1) the labels $\cdot^L$ and $\cdot^R$ were dropped to prevent having to write down what is essentially the same equation four times.

The part that is used only by the Brezzi numerical flux is the hardest part. First, define

$$M_{ij} = (\phi_i, \phi_j)_K .$$

Then, project the lifting operator onto the space of basis functions and again drop the zero terms

$$\begin{aligned}
&(n_F + \eta_F) \left(\mathcal{R}_F \left([\![\phi_i]\!]_T\right), \mathcal{R}_F \left([\![\phi_j]\!]_T\right)\right)_F \\
&= (n_F + \eta_F) \left(\mathcal{R}_F \left([\![\phi_i]\!]_T\right), \phi_n\right)_\Omega M^{-1} \left(\phi_m, \mathcal{R}_F \left([\![\phi_j]\!]_T\right)\right)_F \\
&= (n_F + \eta_F) \left([\![\phi_i]\!]_T, \{\!\{\phi_n\}\!\}\right)_F M^{-1} \left(\{\!\{\phi_m\}\!\}, [\![\phi_j]\!]_T\right)_F \\
&= \frac{n_F + \eta_F}{4} \left(n^L \times \phi_i^L + n^R \times \phi_i^R, \phi_n^L + \phi_n^R\right)_F M^{-1} \left(\phi_m^L + \phi_m^R, n^L \times \phi_j^L + n^R \times \phi_j^R\right)_F \\
&= \frac{n_F + \eta_F}{4} (n \times \phi_i, \phi_n)_F M^{-1} (\phi_m, n \times \phi_j)_F .
\end{aligned}$$

This term is then computed using a matrix-matrix-matrix product. Here the most apparent disadvantage of using the Brezzi-flux can be seen: in order to compute the stiffness matrix one needs an explicit matrix inverse of the element mass matrix.

It may appear to the alert reader that terms and factors involving μ_r^{-1} went missing. This is correct; the current implementation is build with the extra assumption that $\mu_r^{-1} = 1$ everywhere, so these extra terms and factors are deemed irrelevant for the description of the implementation.

7.4 Normalisation of the eigenfunctions

In Section 5.1 the orthonormality of the eigenfunctions was used. In practice this requires some extra scaling. The used eigensolver guarantees the eigenvectors it finds are orthonormal in the L^2-(vector)norm, so use this as a starting point and project back to $L^2(\Omega)$

$$\delta_{ij} = \psi_i^* \psi_j = \sum_{n,m} \psi_{i,n}^* \phi_n^* M^{-1} \phi_m \psi_{j,m}, \qquad (7.2)$$

The matrix M is a positive definite real matrix, so its square root is properly defined and also positive definite and real. Denote by $M^{-\frac{1}{2}}$ the inverse of the square root of M. Then it follows

from (7.2) that the eigenfunctions $\sum_m M^{-\frac{1}{2}}\phi_m\psi_{i,m}$ are orthonormal. While the orthonormal eigenfunctions are a nice theoretical tool, it is often possible to use (7.2) to evaluate products of eigenfunctions. This circumvents the explicit construction of $M^{-\frac{1}{2}}$.

7.5 Coordinate transformations

The theory presented so far allows one in principle to compute all the integrals and solve the system. It is, however, more convenient to perform the computations on a reference element and use mappings to find the element and face matrices for the physical elements.

7.5.1 Transformation of the basisfunctions

The first consideration is the transformation of functions. For this follow the derivation presented in [21]. Let $\hat{K}$ and K be two bounded domains in $\mathbb{R}^3$ (for example the reference tetrahedron and a tetrahedron in the mesh) and let $F_k : \hat{K} \to K$ be a continuously differentiable bijection where $|\mathrm{d}F_k|$ does not change sign on $\hat{K}$. If for a scalar function p on K and $\hat{p} \in H^1(\hat{K})$ such that

$$p \circ F_K = \hat{p},$$

then the gradient operator transforms as

$$\nabla p = (\mathrm{d}F_K)^{-T}\,\hat{\nabla}\hat{p},$$

where $\hat{\nabla}$ acts with respect to the coordinate system for $\hat{K}$. By taking the curl of this equation it is immediately obvious that $\nabla p \in H(\mathrm{curl}; K)$ and $\hat{\nabla}\hat{p} \in H(\mathrm{curl}; \hat{K})$. So the transformation

$$u \circ F_K = (\mathrm{d}F_K)^{-T}\,\hat{u}. \tag{7.3}$$

is $H(\mathrm{curl})$ conforming.

For the computation of $\nabla \times u \circ F_K$ first write

$$[\nabla \times u]_{i,j} = \frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i},$$

for some $3 * 3$ matrix $[\nabla \times u]_{i,j}$. Note that (7.3) can be rewritten as

$$u_i = \sum_{k=1}^{3} \frac{\partial \hat{x}_k}{\partial x_i}\hat{u}_k.$$

Carry out the differentiations required for the curl-matrix

$$\frac{\partial u_i}{\partial x_j} = \frac{\partial}{\partial x_j}\sum_{k=1}^{3}\frac{\partial \hat{x}_k}{\partial x_i}\hat{u}_k = \sum_{k=1}^{3}\frac{\partial^2 \hat{x}_k}{\partial x_i \partial x_j}\hat{u}_k + \sum_{l=1}^{3}\frac{\partial \hat{x}_k}{\partial x_i}\frac{\partial \hat{x}_l}{\partial x_j}\frac{\partial \hat{u}_k}{\partial \hat{x}_l}.$$

and

$$\frac{\partial u_j}{\partial x_i} = \frac{\partial}{\partial x_i}\sum_{k=1}^{3}\frac{\partial \hat{x}_k}{\partial x_j}\hat{u}_k = \sum_{k=1}^{3}\frac{\partial^2 \hat{x}_k}{\partial x_j \partial x_i}\hat{u}_k + \sum_{l=1}^{3}\frac{\partial \hat{x}_k}{\partial x_j}\frac{\partial \hat{x}_l}{\partial x_i}\frac{\partial \hat{u}_k}{\partial \hat{x}_l}.$$

So

$$([\nabla \times u] \circ F_K)_{i,j} = \sum_{k=1}^{3}\sum_{l=1}^{3} \frac{\partial \hat{x}_k}{\partial x_i}\left(\frac{\partial \hat{u}_k}{\partial \hat{x}_l} - \frac{\partial \hat{u}_k}{\partial \hat{x}_l}\right)\frac{\partial \hat{x}_l}{\partial x_j}.$$

or equivalently

$$[\nabla \times u] \circ F_K = \mathrm{d}F_K^{-T}\left[\hat{\nabla} \times \hat{u}\right]\mathrm{d}F_K^{-1}.$$

The matrix $[\nabla \times u] \circ F_K$ still contains information about $\nabla \times u$. After some tedious, but straightforward linear algebra it is possible to show that

$$(\nabla \times u) \circ F_K = \frac{1}{|\mathrm{d}F_K|}\mathrm{d}F_K\hat{\nabla} \times \hat{u}.$$

7.5.2 Transformation of integrals

Use the standard identity

$$\int_K u \,\mathrm{d}x = \int_{\hat{K}} (u \circ F_K)\,|\mathrm{d}F_K|\,\mathrm{d}x \tag{7.4}$$

to find the integral transformed to the reference element. In the case of face integrals F_K is usually a composite map, so despite its simple appearance (7.4) can get quite involved. A finite element package able to resolve these mappings, such as hpGEM is recommended. In the case of hpGEM a user is only required to provide $u \circ F_K$.

7.6 Boundary conditions for the k-shifted formulation

In the k-shifted formulation there are still some factors of the form $\mathrm{e}^{-\mathrm{i}k\cdot x}$ left inside the jump and average operators. This is because k was introduced on purpose to allow for periodic behaviour with a period not conforming to the size of the domain. As a consequence $\mathrm{e}^{-\mathrm{i}k\cdot x}$ will not be continuous near domain boundaries and will need special a treatment there. Remember that ϕ_i and ϕ_j are nonzero only on one element. There are now two options: Either they are both active on the same element. In this case the contributions to the jump and average operators come only from that element and the factors $\mathrm{e}^{-\mathrm{i}k\cdot x}$ cancel. Or they are both active on different elements. In this case their contributions to the jump and average operators come from elements on opposite sides of Ω. This means the factors $\mathrm{e}^{-\mathrm{i}k\cdot x}$ don't cancel, but instead leave a shift of about the size of the domain.

There are also the undefined shifts that were introduced with the basisfunctions. To provide a definition the matrix formulation can be rewritten into

$$\omega^2 \mathrm{diag}\left(\mathrm{e}^{ik\cdot x_i}\right)\mathcal{M}\mathrm{diag}\left(\mathrm{e}^{-ik\cdot x_i}\right)\vec{E} = \mathrm{diag}\left(\mathrm{e}^{ik\cdot x_i}\right)\mathcal{S}\mathrm{diag}\left(\mathrm{e}^{-ik\cdot x_i}\right)\vec{E}.$$

From this it can be seen that these shifts will have no effect on the eigenvalues. The eigenvectors, however, are, for good choices of x_i, shifted such that the effects of introducing k are approximately cancelled. This means that, once found, the eigenvectors can be reused as pré-converged initial vectors for a next choice of k resulting in great speedups.

In practice, when performing computations for many different values of k at the same time the changes in k are small enough that even changing x_i into $-x_i$ has no significant impact on the simulation time. The requirement that the target eigenvalues are nonzero may have some impact on this since it forces SLEPc [15] to reconsider the 'desired' ordering of the eigenvalues every few steps. Even so, setting initial eigenvectors improves the simulation time significantly. Implementing the divergence condition allows a zero target; with this setting, the desired ordering of the eigenvectors won't change as often, potentially resulting in an even larger speed improvement and much easier (L)DOS computation.

7.7 Symmetry considerations and the Irreducible Brillouin Zone

In Section 7.5.1 transformation rules for a curl-conforming change of coordinates were set up. Nothing in that section is specific for basis-functions or single elements, so these rules can be applied also for solutions on the whole of Ω. Consider for example the mapping $F = (x, y, z) \mapsto (1 - x, y, z)$. This maps Ω onto itself. Moreover,

$$\int_{\Omega} \left(\mu^{-1} (\nabla \times u) \cdot (\nabla \times \phi) + \omega^2 \epsilon_r u \cdot \phi\right) \circ F \, \mathrm{d}x = \tag{7.5}$$

$$\int_{\hat{\Omega}} \mu^{-1} \frac{1}{|\mathrm{d}F|} \left(\hat{\nabla} \times \hat{u}\right)^T \mathrm{d}F^{-1} \mathrm{d}F^{-T} \left(\hat{\nabla} \times \hat{\phi}\right) + |\mathrm{d}F| \, \omega^2 \epsilon_r \hat{u}^T \mathrm{d}F^T \mathrm{d}F \hat{\phi} \, \mathrm{d}x, \tag{7.6}$$

and since $\mathrm{d}F = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ the matrices in (7.6) cancel. Flip the direction of integration in the x-direction so the integral is from 0 to 1 again and notice that $|\mathrm{d}F| = -1$ to see that (7.6) is equivalent to

$$\int_{\Omega} \mu^{-1} \left(\hat{\nabla} \times \hat{u}\right) \cdot \left(\hat{\nabla} \times \hat{\phi}\right) + \omega^2 \epsilon_r \hat{u} \cdot \hat{\phi} \, \mathrm{d}x.$$

So the only effect of applying F is that every term picked up a hat. This equivalence can also be understood from a physical point of view: An electromagnetic wave propagating from $x = 0$ to $x = 1$ 'sees' exactly the same domain as a wave propagating from $x = 1$ to $x = 0$. From a more practical perspective this means there is no need to consider wave-vectors with a negative x-component, because this mapping shows they have the same eigenvalues as wave-vectors with the same, but positive x-component. One has to be carefull, however, when computing the orientation-dependent local density of states to compute $\rho_p(r, \omega, n_p) + \rho_p(r, \omega, n_p \circ F)$ when integrating over one half of the Brillouin zone.

A similar argument can be applied to mirroring in the other coordinate directions $(x, y, z) \mapsto (x, 1 - y, z)$ and $(x, y, z) \mapsto (x, y, 1 - z)$ and to changing the ordering of the coordinates $(x, y, z) \mapsto (y, x, z)$. In the case of a homogeneous medium this can be an arbitrary reordering. In the three other test-cases ϵ_r treats one of the coordinate directions differently from the other two, so in that case only those two directions can be swapped.

By applying all four of these symmetries the domain of integration in the Brillouin zone can be reduced to a prism of size only $\frac{1}{16}$ of the full size of the Brillouin zone for the non-homogeneous cases. In the case of the homogeneous medium a further reduction to a tetrahedron of only $\frac{1}{48}$

of the size of the Brillouin zone is possible. Because of its importance for the computations this smaller wedge is named the irreducible Brillouin zone(IBZ).

7.8 Used packages

The finite element toolkit hpGEM [30] has been used to construct the matrices in the DG discretisation. It is a DGFEM package developed by the MACS chair at the University of Twente. It provides routines for mesh generation and numerical quadrature and can provide iterated access to all elements or all faces. PETSc [6] has been used for time integration and solving linear systems of equations. It is a computer algebra package that is intended as MATLAB for distributed memory systems. In principle hpGEM could also be used for the time integration, but time integration is mostly implemented to test the correctness of the code, so a choice has been made to use PETSc instead in order to prevent having to reïmplement the construction of the matrices. For the solution of the eigenvalue problems SLEPc [15] has been used. This is a specialised eigenvalue solver that is constructed to be used with the data structures and routines provided in PETSc wherever this is appropriate, minimising the amount of extra work needed to get it working with the rest of the code.

To construct complicated domains, such as the cylinder inside the cube, the CAD package Rhinoceros [14] was used. The domain descriptions generated by Rhinoceros were converted to tetrahedral meshes using the mesh-generator Centaur [1]. These meshes were used instead of the regular meshes described earlier. Both Rhinoceros and Centaur are commercial packages.

Chapter 8

Conclusions

For the cases presented so far the DG-Max code works as intended. This is in agreement with the conclusions by other authors so there is hope that the code also works as intended for cases not yet studied.

8.1 Outlook

This research concerns multiple different disciplines that will also have different opinions about what can be considered an improvement and what is irrelevant. Therefore this section is split into multiple subsections, so all possibilities for improvement can be highlighted for the areas that consider them relevant.

8.1.1 Mathematical aspects

A very important point of improvement would be to implement the extra condition $\nabla \cdot \epsilon_r E = 0$ to remove the null space of the curl-curl operation from the space of allowable solutions. It is known that the presence of a null space is detrimental for the convergence speed of Krylov subspace methods. So it is expected that removing the null-space will greatly improve the speed of the code.

As an alternative strategy to achieve speedups a specialised preconditioner or linear solver for matrices arising from Finite Element Methods can be developed. This will also be of use in other fields where Finite Element Methods are used to discretize (partial) differential equations.

If this is still insufficient to achieve the desired speed it is also possible to adapt the code for parallel computations.

A more thorough theoretical framework is useful for the construction of more accurate error estimates. In particular, a good a-posteriory error estimator is useful for automated refinement of the mesh in zones where a better accuracy is required.

At the moment basis functions are provided only for tetrahedral elements. If a basis is also provided for elements of other shapes this will allow for greater flexibility in constructing a mesh. In particular, if a basis can be constructed for elements with a curved boundary this allows for exact matching of material interfaces, even when the unit cell contains spherical or cylindrical structures.

8.1.2 Photonic crystals

For users of DG-Max attempting to design photonic crystals the code should be able to handle defects in the perfect infinite repetition of a unit cell. In particular, it should be possible to incorporate point or line defects, where the perfect translational symmetry of a crystal is broken in one unit cell or a line of unit cells. This adaption will also have to be able to handle finite crystals or more general defect shapes.

It is also important to be able to handle more general ϵ_r. It is a relatively easy change to make it a matrix, allowing the simulation of anisotropic media. Other options include a dependence of ϵ_r on frequency or field strength.

Another requirement will be an option to change the unit cell for periodic structures where the unit cell is not a cube.

Most important perhaps is a sensitivity analysis. Using the extra accuracy provided by the mathematical model, it is possible to analyse what happens when a crystal is not produced to mathematical perfection by artificially introducing small defects in a bi-material interface or in the given parameters.

8.1.3 Local Density of States

The attention given in this work to the Local Density of States can be interpreted as a preliminary exploration at best. Most importantly the order of accuracy of the LDOS computation should be determined. The order of accuracy should probably also be improved to match the accuracy achieved by the rest of the numerical model. Moreover, more testing needs to be done to verify that the computation is also correct in realistic photonic crystals.

An eigenvalue solver -ideally an iterative one- should be found that can reliably find all eigenvalues in a frequency range.

Ideally the eigenvalue problem has to be rewritten such that ω can be used as a given parameter. This will greatly reduce the amount of work that needs to be done when computing the LDOS. This probably also means that it is much easier to achieve a good order of convergence.

Note that at the moment the accuracy loss in the LDOS computation can be set independently from the accuracy of the eigenvalue problem, so if no improvements can be made in this area, it is not detrimental to the quality of the program or the usefulness of high accuracy in the rest of the program.

Simulations in this work have been limited to low ω and to the irreducible Brillouin zone. In this case the problem of crossing bands is avoided. For simulations with higher ω or over a larger part of the Brillouin zone band crossings can occur. In this case the naive lowest first sorting strategy will cause $\omega_n(k)$ to become (locally) nondifferentiable. This breaks the integration procedure presented in this report, so some other sorting strategy needs to be used in these cases.

Appendix A

Detailed description of the code

This Appendix will aim to present a description of the DG-Max code in a format where formulas are more readily presented than the online documentation. It will also include a comparison with the hpGEM 1 code written by D. Sármány. Numbered equations will denote steps in the algorithm the code executes. For the code written by Sármány, line numbers without a file name refer to line numbers in Max3D.cc and are based on the dgFourier3D sample code from hpGEM 1.

A.1 Preamble

At the beginning of the code all data is initialised. A timer is also started to measure performance. Sármány inputs only the desired polynomial order and the desired number of elements per direction n. The other options are hard-coded and need a recompilation. This is also the current, temporary set-up. Note that n takes a slightly more generalised meaning if using a centaur mesh with the old code.

`Old code found on lines 1-59 and in IncludeAll.hh. New code to be found in DomokosProblem::initialise() (DomokosProblem.cpp)`

A.2 Generating the mesh

The mesh is either read from a pre-generated Centaur file or constructed by assuming that $\Omega = (0,1)^3$ is subdivided into $n*n*n$ sub-cubes and then splitting the cubes into five tetrahedra, four of which are congruent. The exact subdivision can be seen in Figure A.1. Name the total number of elements N_τ and the number of local degrees of freedom N_p. Mesh generation is handled by the hpGEM kernel. For the new code this is so easy that it is considered part of the preamble. For the old code this is a separate stage. Separate function calls need to be made if you want a periodic mesh instead of a non-periodic one.

`Old code found on lines 63-133`

A.3 Setting up of the matrices

First PETSc is initialised. Then the global matrices $\mathcal{M} \in \mathbb{R}^{N_p N_\mathcal{T} \times N_p N_\mathcal{T}}$ or $\mathcal{M}^{-1} \in \mathbb{R}^{N_p N_\mathcal{T} \times N_p N_\mathcal{T}}$, the (inverse) mass matrix, and $\mathcal{S} \in \mathbb{R}^{N_p N_\mathcal{T} \times N_p N_\mathcal{T}}$, the stiffness matrix, are initialised. The RHS vectors $e \in \mathbb{R}^{N_p N_\mathcal{T}}$(the (exact) initial solution), $u \in \mathbb{R}^{N_p N_\mathcal{T}}$ (the derivative of the (exact)

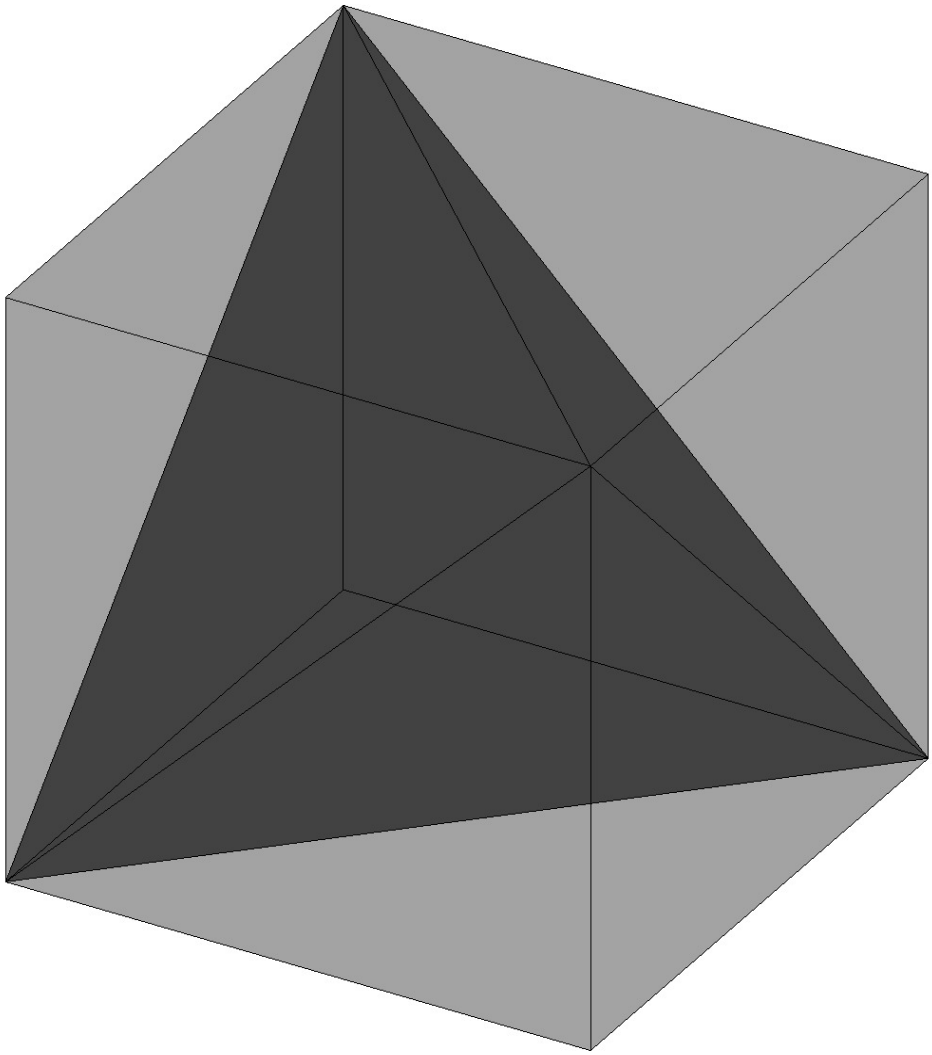


Figure A.1: Cube split into 5 tetrahedra. This is the subdivision used for generating a regular structured mesh in hpGEM. For a domain with more than one cube per direction some cubes are rotated such that the faces fit together.

initial solution), $j_0 \in \mathbb{R}^{N_p N_\mathcal{T}}$(the initial source term) and j(used to store the source term) and the auxiliary vectors X and Y are also initiated. Moreover space is allocated for some of the temporary data. This should be done in the constructor of BaseExtended. However, not all data is known during construction. So part of the initialisation is postponed to a more convenient moment. The old code is not object oriented so declaration of the matrices is postponed until all the required information is known.

The mass matrix $\mathcal{M}$ is a block-diagonal matrix with $N_\mathcal{T}$ nonzero blocks $[M] \in \mathbb{R}^{N_p \times N_p}$, so $\mathcal{M}^{-1}$ is also a block-diagonal matrix, but with blocks $\left[M^{-1}\right]$. Note that depending on the problem either $\mathcal{M}$ or $\mathcal{M}^{-1}$ is not needed explicitly and is therefore not initialised in the code. If the problem in principle allows both, $\mathcal{M}^{-1}$ is initialised because the Brezzi numerical flux requires $\left[M^{-1}\right]$ in either case. $\mathcal{S}$ has $N_\mathcal{T}^2$ blocks $[S] \in \mathbb{R}^{N_p \times N_p}$. The vectors j_0, e and u have $N_\mathcal{T}$ sub-vectors, each with length N_p.

It is the intention that this part of the code becomes a part of the hpGEM core in the form of a Global matrix assembly interface.

`Old code found on lines 146-234. New code found in hpGemUIExtentions::hpGemUIExtentions() (BaseExtended.cpp) and the first 20 lines of both MatrixFillerIP::fillMatrixes() and MatrixFillerBR::fillMatrixes() (fillMatrices.cpp)`

$$\begin{cases} \text{initialise } \mathcal{M} \leftarrow 0 & \text{for the time-harmonic problem} \quad \text{(A.1a)} \\ \text{initialise } \mathcal{M}^{-1} \leftarrow 0 & \text{else} \quad \text{(A.1b)} \end{cases}$$

$$\text{initialise } \mathcal{S} \leftarrow 0 \quad \text{(A.2)}$$

$$\text{initialise } j_0 \leftarrow 0 \quad \text{(A.3)}$$

$$\text{initialise } e \leftarrow 0 \quad \text{(A.4)}$$

$$\text{initialise } u \leftarrow 0 \quad \text{(A.5)}$$

A.4 Filling of the matrices

The next step in the algorithm is computing the element matrices. Before this can be done, some extra information is needed. The code needed to obtain this extra information is described first.

A.4.1 Meta-data

For each element the Jacobian and its determinant are computed and stored. Then the transpose of the inverse of the Jacobian is computed exactly. For example

$$\mathbb{J}^{-1}_{0,0} = \frac{\mathbb{J}_{1,1}\mathbb{J}_{2,2} - \mathbb{J}_{2,1}\mathbb{J}_{1,2}}{|\mathbb{J}|}.$$

In the old code the local numbering of the vertices is manually made consistent with the global numbering of the vertices. In the DG-Max code this is not needed because hpGEM 2 handles face to face mappings correctly. The old code also sets up integration rules for all polynomial orders, but then only uses the appropriate one.

The basis-functions at the quadrature points also have to be computed. They are already written out in Sections 3.4 and 7.2 so they are not repeated here. In the old code they are

computed preemptively for the elements and inefficiently for the faces. In the DG-Max code they are computed on the fly whenever they are requested for a reference point where they are not computed yet. The old code has all the basisfunctions explicitly coded. The new code uses some parameters to reuse code for basisfunctions with large similarities and to be able to increase the polynomial order of the basis without a lot of extra work.

`Old code found on lines 234-262 and in files GetOrientation.hh, GetJacobians.hh, InvertJacobian.hh, TranspMatrix.hh GetfuncTables.hh, VectorCurl3D.hh, InitialiseVectorBasis.hh, edge.hh, edgeface.hh, facebubble.hh, faceinterior.hh, interiorbubble.hh, InitialiseScalarBasis.hh, Bary3D.hh, legend.hh and cross.hh. New code found in file ElementInfos.cpp and in file BasisFunctionCollection_Curl.cpp`

A.4.2 Compute element matrices

The following is done for every element E separately:

The local mass and stiffness matrices and the inverse of the mass matrix are initialized by resizing them to the correct size and clearing them. Now, for every $0 \leq i \leq j < N_p$ the actual values of the local mass matrix $[M]^E$ and the local stiffness matrix $[S]^{EE}$ are computed as

$$[M_{i,j}]^E \leftarrow [M_{j,i}]^E \leftarrow \int_E \mathbb{J}_E^{-1,T} \phi_i(x) \cdot \mathbb{J}_E^{-1,T} \phi_j(x) \, \mathrm{d}x \tag{A.6}$$

$$\left[M^{-1}\right]^E \leftarrow \left([M]^E\right)^{-1} \tag{A.7}$$

$$[S_{i,j}]^{EE} \leftarrow [S_{j,i}]^{EE} \leftarrow \int_E \frac{\mathbb{J}_E}{|\mathbb{J}_E|} (\nabla \times \phi_i(x)) \cdot \frac{\mathbb{J}_E}{|\mathbb{J}_E|} (\nabla \times \phi_j(x)) \, \mathrm{d}x \tag{A.8}$$

using integration routines provided by hpGEM. Here $\mathbb{J}_E$ is the Jacobian of the transformation from the reference element to E. Next, the mass matrix is inverted using Lapack's gesv (if this is desired). The old code also exports all entries larger then 10^{-12} into a sparse matrix structure by writing its global index $(localnoDOF * elID + i, localnoDOF * elID + j)$ and its value to a white-space separated text file for later use in external programs, such as Matlab.

The initial source term j, the exact solution e and its derivative u are initialised by resizing them to the correct size and clearing them. Now for every $0 \leq i < locNrDOF$ their entries are computed as

$$[j_{0,i}]^K \leftarrow \int_K \mathbb{J}_K^{-1,T} \phi_i(x) \cdot f_0(x) \, \mathrm{d}x \tag{A.9}$$

$$[e_i]^K \leftarrow \int_K \mathbb{J}_K^{-1,T} \phi_i(x) \cdot u_0(x) \, \mathrm{d}x \tag{A.10}$$

$$[u_i]^K \leftarrow \int_K \mathbb{J}_K^{-1,T} \phi_i(x) \cdot \frac{\mathrm{d}u_0(x)}{\mathrm{d}t} \, \mathrm{d}x \tag{A.11}$$

Where f_0 is the exact source term at $t = 0$ and u_0 is the exact solution at $t = 0$.

In the DG-Max code this is combined with the computations of the face contributions so PETSc can already begin redistributing the stiffness matrix, while the vector entries are still being computed.

It is the intention that the sorting of element contributions into the global matrix becomes part of the hpGEM core in the form of a Global matrix assembly interface.

`Old code can be found on lines 311-433 and in file calcElCont.hh, ExactSolution.hh ExactSolutionDeriv.hh, SourceTermData.hh. New code can be`

found in MatrixFillerIP::fillMatrixes(), MatrixFillerBR::fillMatrixes() (fillMatrices.cpp), DomokosProblem::elementMassIntegrand(), DomokosProblem::elementStiffnessIntegrand(), DomokosProblem::sourceTerm(), DomokosProblem::initialConditions() and DomokosProblem::initialConditionsDeriv() (DomokosProblem.cpp)

A.4.3 Compute face matrices

For the face contributions the difference between the old code and the DG-Max code is larger. The old code computes the block matrices mentioned in Section 7.3 as separate matrices while the DG-Max code computes the entire block structured matrix in one go. The old code splits the part that is common to both numerical fluxes into matrices F and G, it names the part used only by the IP numerical flux H and names the matrix $(\phi_i, n \times \phi_j)$ used for the Brezzi numerical flux D. The matrix D^T is named C. Beyond this internal faces require different treatment than boundary faces.

Internal face

If the face is an internal face it will have two neighboring elements. Name them l and r. First for $A, B \in \{l, r\}$ some auxiliary matrices F^{AB}, $G^{AB} = (F^{BA})^T$, H^{AB}, D^{AB} and $C^{AB} = (D^{BA})^T$ are first initialised and then computed as

$$\begin{aligned} F_{i,j}^{AB} &= \int_{\delta E} \left(n_A \times \mathbb{J}_A^{-1,T} \phi_i^A \right) \cdot \frac{\mathbb{J}_B}{|\mathbb{J}_B|} \left(\nabla \times \phi_j^B \right) \mathrm{d}x \\ H_{i,j}^{AB} &= \int_{\delta E} \left(n_A \times \mathbb{J}_A^{-1,T} \phi_i^A \right) \cdot \left(n_B \times \mathbb{J}_B^{-1,T} \phi_j^B \right) \mathrm{d}x \\ D_{i,j}^{AB} &= \int_{\delta E} \mathbb{J}_A^{-1,T} \phi_i^A \cdot \left(n_B \times \mathbb{J}_B^{-1,T} \phi_j^B \right) \mathrm{d}x \end{aligned}$$

Here ∂E is the face under consideration, $\mathbb{J}_A$ is the Jacobian of the transformation from the reference element to element A, ϕ_i^A is the i^{th} basis function of element A and n_A is the outward pointing normal vector.

Now compute the contributions to the stiffness matrix originating from the internal faces. They are, again for $\forall A, B \in \{l, r\}$

$$[S]^{AB} \leftarrow [S]^{AB} - \frac{1}{2} \left(F^{AB} + G^{AB} \right) + \left(1 + \frac{\eta_F}{4} \right) \left(C^{Al} \left[M^{-1} \right]^l D^{lB} + C^{Ar} \left[M^{-1} \right]^r D^{rB} \right) \quad \text{(A.12)}$$

with η_F a stability coefficient for the used numerical method.

Boundary face

If the face is a boundary face it will have only one neighboring element. This is stored as the left element. If F^{ll}, G^{ll}, H^{ll}, C^{ll} and D^{ll} are as defined in the previous Section, the corresponding contribution to the stiffness matrix can be computed as

$$[S]^{ll} \leftarrow [S]^{ll} - \left(F^{ll} + G^{ll} \right) + (4 + \eta_F) C^{ll} \left[M^{-1} \right]^l D^{ll} \quad \text{(A.13)}$$

Now to compute the right hand side contributions of the boundary faces, first set up some auxiliary vectors G, H and D as

$$G_i = \int_{\partial E} \frac{\mathbb{J}_l}{|\mathbb{J}_l|} (\nabla \times \phi_i) \cdot (n \times u_0) \, \mathrm{d}s$$
$$H_i = \int_{\partial E} \left(n \times \mathbb{J}_l^{-1,T} \phi_i \right) \cdot (n \times u_0) \, \mathrm{d}s$$
$$D_i = \int_{\partial E} \mathbb{J}_l^{-1,T} \phi_i \cdot (n \times u_0) \, \mathrm{d}s.$$

Here u_0 is the initial solution at $t = 0$.

Now the local contribution to the right hand side is

$$[J]^l \leftarrow [J]^l - G + (4 + \eta_F) \, C^{ll} \left[M^{-1} \right]^l D \tag{A.14}$$

`Old code can be found on lines 452-850 and in file calcFaceCont.hh. New code can be found in MatrixFillerIP::fillMatrixes(), MatixFillerBR::fillMatrixes() (fillMatrices.cpp), DomokosProblem::faceIntegrand(...), DomokosProblem::faceIntegrandIPPart(...) and DomokosProblem::faceIntegrandBRPart(...) (DomokosProblem.cpp)`

A.5 Solver

A.5.1 Time integration

For the time dependent Maxwell equations everything is ready to perform the time integration. The DG-Max code uses a second order leapfrogging scheme. The old code focuses more on time integration and offers 3 alternatives that have fourth order convergence.

In the second order scheme the basis coefficients are obtained from the interpolant first. Then, set the time step $\tau \leftarrow \lfloor \frac{2.4}{(2p+1)n} \rfloor$ and number of time steps $Nsteps \leftarrow \lceil \frac{t_{end}-t_0}{\tau} \rceil$. Update $\tau \leftarrow \lfloor \frac{t_{end}-t_0}{Nsteps} \rfloor$ and $Nsteps \leftarrow \lfloor \frac{t_{end}-t_0}{\tau} \rfloor$ a few times to ensure $Nsteps\tau$ is as close to $t_{end} - t_0$ as possible. Note that this choice of τ may be unstable for low p, so potentially τ will have to be

set slightly lower. Finally, the time integration is performed using the following steps:

$$u \leftarrow \mathcal{M}^{-1}u \tag{A.15}$$
$$e \leftarrow \mathcal{M}^{-1}e \tag{A.16}$$
$$t \leftarrow t_0 \tag{A.17}$$
$$\text{while } t < t_{end} \text{ do} \tag{A.18}$$
$$e \leftarrow e + \frac{\tau}{2}u \tag{A.19}$$
$$j_t = j_0 \frac{\eta(t)}{\eta(0)} \tag{A.20}$$
$$j_{t+\tau} = j_0 \frac{\eta(t+\tau)}{\eta(0)} \tag{A.21}$$
$$u \leftarrow \frac{1-\frac{\tau}{2}\sigma}{1+\frac{\tau}{2}\sigma}u + \mathcal{M}^{-1}\left(\tau \frac{\frac{(j_t+j_{t+\tau})}{2} - \mathcal{S}e}{1+\frac{\tau}{2}\sigma}\right) \tag{A.22}$$
$$e \leftarrow e + \frac{\tau}{2}u \tag{A.23}$$
$$t \leftarrow t + \tau \tag{A.24}$$
$$\text{end of while loop} \tag{A.25}$$

Note that this assumes the source term can be split into a spatial contribution and a temporal contribution $\eta(t)$. It also scales the boundary terms with this time contribution. For the time steps where Tecplot output is desired the entire solution vector is copied into a local data structure and stored for later post-processing.

`code can be found on line 929-959 and in file TimeIntegrateCO2.hh. New code found in BaseExtended::solveTimedependent()`

A.5.2 Solve the time-harmonic system

Solving the time-harmonic system is much easier. In this case the steps in the algorithm will be

$$\text{solve } \left(\mathcal{S} - \omega^2\mathcal{M}\right)x = \vec{J} \tag{A.26}$$

Some extra work is done to make sure the default settings for the sparse matrix solver will always work and to store the solution locally for postprocessing.

`New code found in BaseExtended::solveHarmonic()`

A.5.3 Find eigenvalues

Finding eigenvalues is more involved, because of the wave-vector dependence. To find all the eigenvalues do the following

$$\text{for all requested wave numbers} \tag{A.27}$$
$$\mathcal{S} \leftarrow \mathrm{e}^{\mathrm{i}\Delta k\cdot x_i}\mathcal{S}\mathrm{e}^{-\mathrm{i}\Delta k\cdot x_i} \tag{A.28}$$
$$\text{scale off-diagonal entries of } \mathcal{S} \text{ by } \mathrm{e}^{\mathrm{i}\Delta k\cdot\Delta x} \text{ (only actually done for boundary faces)} \tag{A.29}$$
$$\text{find eigenvalues of } \mathcal{M}^{-1}\mathcal{S} \tag{A.30}$$
$$\text{end of for loop} \tag{A.31}$$

Note that explicitly inverting the right hand side matrix is generally not the best way to solve a generalised eigenproblem, but in this case $\mathcal{M}^{-1}$ is already explicitly available from computing the Brezzi flux. To make it easier to mix and match pieces of code depending on the desired solution strategy it was opted to always explicitly invert $\mathcal{M}$. It is a bit unclear what information is relevant for post-processing, so for now this part is not present.

`New code found in BaseExtended::solveEigenvalues(), BaseExtended::makeShiftMatrix() and BaseExtended::findBoundaryBlocks() (BaseExtended.hpp)`

A.5.4 Computing the (L)DOS

The LDOS computation starts with computing eigenvalues for all needed k-points using the algorithm described in Section A.5.3. Extra care is taken that the sorting of the eigenvalues is consistent and manually add the eigenvalue 0 (multiplicity 2) to the set of eigenvalues belonging to $|k| = 0$. Once this is done, given a value of ω and for all tetrahedrons in the irreducible Brillouin Zone, compute a contribution to the LDOS from this tetrahedron using the following steps.

First compare ω against $\omega_0, \ldots, \omega_4$ and sort the ω_i, k_i and f_i into one of two lists based on whether ω_i is larger or smaller. The rest of the algorithm depends on the number of ω_i that are smaller than ω. If only one ω_i is smaller this is equivalent to the $\omega_0 < \omega < \omega_1$ case. The expression presented in Section 5.2 does not depend on the ordering of $\omega_1, ..., \omega_3$, so they are evaluated using the arbitrary ordering in which they were put into the list with larger ω_i.

for all needed wave numbers (A.32)
  $\mathcal{S} \leftarrow \mathrm{e}^{\mathrm{i}\Delta k \cdot x_i} \mathcal{S} \mathrm{e}^{-\mathrm{i}\Delta k \cdot x_i}$ (A.33)
  scale off-diagonal entries of $\mathcal{S}$ by $\mathrm{e}^{\mathrm{i}\Delta k \cdot \Delta x}$ (only actually done for boundary faces) (A.34)
  find eigenvalues of $\mathcal{M}^{-1}\mathcal{S}$ (A.35)
end of for loop (A.36)
for all tetrahedra in the IBZ (A.37)
  count the number of smaller ω_i (A.38)
  evaluate the integral using the expression presented in Section 5.2 (A.39)
  add result to DOS (A.40)
end of for loop (A.41)

`New code found in BaseExtended::solveDOS(), BaseExtended::makeShiftMatrix(), BaseExtended::findBoundaryBlocks() (BaseExtended.hpp) and in KspaceData::getIntegral() (kspacedata.cpp)`

A.6 Post-processing

During post-processing the discontinuous Tecplot writer is called to enable visualisation. After this some errors are computed by interpolating the numerical solution and its curl from the expansion coefficients and computing the L^2 norm of their difference. For the $H(\mathrm{curl})$-error of the numerical method the error in the tangential jump is also computed. Now the L^2 error,

the $H(\text{curl})$-error and the DG-error are printed to the screen. This process will be repeated if multiple time stages or multiple eigenfunctions are available for post-processing.

The old code also provides an estimate of the L^∞ error by evaluating the error of the numerical solution at all quadrature points. This has not been ported to the DG-Max code because the quadrature points may give not a very accurate error measure, especially if some of them lie outside the element.

`Old code found on lines 992-1076 and in files calcError.hh and CalcFaError.hh. New code found in BaseExtended::makeOutput(), BaseExtended::computeErrors(...), BaseExtended::faceErrorIntegrand(...), BaseExtended::elementErrorIntegrand(...), BaseExtended::writeFieldValues() (BaseExtended.hpp), DomokosProblem::exactSolution(...) and DomokosProblem::exactSolutionCurl(...) (DomokosProblem.cpp)`

Note that if the exact solution for the problem you are trying to solve is unknown, you can just enter an arbitrary 'exact' solution. Of course you should not expect the error estimates to be very accurate.

A.7 Used features of hpGEM

For the correct functioning of the code hpGEM provides many features that can be summarized as

- automated mesh generation. It was explained that some parameters need to be provided for the mesh generation. hpGEM uses these parameters to generate a regular mesh based on unit cells that look like Figure A.1.
- Computing Jacobians and coordinate transformations. Both in the case where the mesh is provided by an external mesh generator and in the case where the mesh is generated internally hpGEM sets up coordinate transformations from the reference element to the physical element and provides the Jacobian of this transformation so it can be used where needed for correctly transforming vector functions.
- Computing integrals. If an integrand is provided hpGEM can automatically compute integrals. This is useful both when filling the matrices and computing errors. Note that hpGEM assumes the integral will have to be transformed to the physical element and applies an appropriate coordinate transformation. This holds only for the intregral, if an individual function that forms the integrand also needs to be transformed this needs to be taken care of manually.
- Post-processing. The hpGEM package provides automated routines to write a computed solution to a Tecplot file for visualisation.
- Future work: Constructing the global system. A planned feature of hpGEM is the automated construction of a global system given a strategy to construct element contributions. This feature will include resorting a solution to the individual elements so post-processing can be done on a per-element basis. If this part is constructed to work in parallel then the entire code will be parallel.
- Future work: A generic way to set parameters of a code without having to recompile should be introduced. Initial efforts were made in this direction, but are not finalized yet. The goal is to have something like what is used for MPB.

Bibliography

[1] 2013. centaursoft.com.

[2] M. Ainsworth and J. Coyle. Hierarchic finite element bases on unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 58:2103–2130, 2003.

[3] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[4] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[5] S. Balay, J. Brown, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc Web page, 2013. http://www.mcs.anl.gov/petsc.

[6] S. Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A.M. Bruaset, and H.P. Langtangen, editors, *Modern software tools in scientific computing*, pages 163–202. Birkhäuser Press, 1997.

[7] D. Boffi, M. Conforti, and L. Gastaldi. Modified edge finite elements for photonic crystals. *Numerische Mathematik*, 105(2):249–266, 2006.

[8] A. Buffa, P. Houston, and I. Perugia. Discontinuous Galerkin computation of the Maxwell eigenvalues on simplicial meshes. *Journal of Computational and Applied Mathematics*, 204(2):317–333, 2007.

[9] A. Bulovyatov. *A parallel multigrid method for band structure computation of 3D photonic crystals with higher order finite elements*. PhD thesis, Universität Karlsruhe, 2010.

[10] R. Cools. An encyclopaedia of cubature formulas. *Journal of Complexity*, 19(3):445–453, 2003.

[11] R. Cools. Encyclopaedia of cubature formulas, 2013. http://nines.cs.kuleuven.be/research/ecf/ecf.html.

[12] I.F.C. Denissen. Discontinuous Galerkin finite element methods for the time-harmonic Maxwell equations in periodic media. Master's thesis, University of Twente, 2013.

[13] D.C. Dobson and J.E. Pasciak. Analysis of an algorithm for computing electromagnetic Bloch modes using Nedelec spaces. *Computational Methods in Applied Mathematics*, 1(2):138–153, 2001.

[14] R. McNeel et al., 2013. rhino3d.com.

[15] V. Hernandez, J.E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions of Mathematical Software*, 31(3):351–362, 2005.

[16] J.S. Hesthaven and T. Warburton. High-order nodal discontinuous Galerkin methods for the Maxwell eigenvalue problem. *Philosophical Transactions of the Royal Society of London A*, 362:493–524, 2004.

[17] P. Houston, I. Perugia, A. Schneebeli, and D. Schötzau. Interior penalty method for the indefinite time-harmonic Maxwell equations. *Numerische Mathematik*, 100(3):485–518, 2005.

[18] J.D. Joannopoulos. *Photonic crystals: molding the flow of light.* Princeton University Press, second edition, 2008.

[19] J.C. Knight. Photonic crystal fibres. *Nature*, 424(6950):847–851, 2003.

[20] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.* United States Governm. Press Office, 1950.

[21] P. Monk. *Finite element methods for Maxwell's equations.* Oxford University Press, 2003.

[22] Houston museum of natural science. A picture of an opal. commons.wikimedia.org/wiki/File:WLA_hmns_Fiery_Opal_Opalville_Mine.jpg, February 2009.

[23] L. Novotny and B. Hecht. *Principles of nano-optics.* Cambridge University Press, 2012.

[24] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[25] D. Sármány. Description of the three-dimensional Maxwell code implemented in hpGEM. Technical report, Universiteit Twente, 2009.

[26] D. Sármány. *High-order finite element approximations of the Maxwell equations.* PhD thesis, University of Twente, 2010.

[27] D. Sármány, F. Izsák, and J.J.W. van der Vegt. Optimal penalty parameters for symmetric discontinuous Galerkin discretisations of the time-harmonic Maxwell equations. *Journal of Scientific Computing*, 44:219–254, 2010.

[28] P. Solin, K. Segeth, and I. Dolezel. *Higher-order finite element methods.* CRC Press, 2003.

[29] G.W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002.

[30] J.J.W. van der Vegt, A.R. Thornton, V.R. Ambati, and A. Bell et al. Software library for discontinuous Galerkin methods, 2013. einder.ewi.utwente.nl/hpGEM.

[31] T. Warburton and M. Embree. The role of the penalty in the local discontinuous Galerkin method for the Maxwell's eignenvalue problem. *Computational Methods in the Applied Mechanical Engeneering*, 195:3205–3223, 2006.

[32] K. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *Antennas and Propagation, IEEE Transactions on*, 14(3):302–307, 1966.