UPPSALA
UNIVERSITET

Advanced Numerical Methods

# Numerical Analysis of Maxwell's Equations

**Authors**:

Matilda Tiberg

matilda.tiberg.5927@student.uu.se

Uppsala

September 21, 2022

# Contents

# 1 Introduction

Different phenomena that arise in the world can often be described by a partial differential equation (PDE). However, these PDE:s does not always have analytical solution. To get around this inconvenience, different numerical methods exists.

In this project, FDM are being applied on Maxwell's equation, which is a system of three PDE:s in 2D. Under the assumption that there exists no free charges or currents, the system of PDE:s can be seen in equation (1), where $\mathbf{u}$ is defined below. Everything is denoted in cartesian coordinates. The system consists of three components: the electric field $E$ has one $x$-component and one $y$-component. The magnetic field, $H$, spreads in $z$-direction.

$$\mathbf{C}\mathbf{u}_t = \mathbf{A}\mathbf{u}_x + \mathbf{B}\ \mathbf{u}_y \tag{1}$$

The coefficients are given by the following below, where $\epsilon$ is the permittivity and $\mu$ is the permeability. Generally, these depends on some time-factor as well as the spatial location, but in this assignment they are treated as constants.

$$\mathbf{u} = \begin{bmatrix} E^{(x)} \\ H^{(z)} \\ E^{(y)} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & \epsilon \end{bmatrix}$$

## 1.1 Project Description

The aim of the project is to derive and implement two stable numerical methods that solves Maxwell's equations, which is a system of PDE:s. The numerical methods SBP-Simultaneous Approximation Term (SAT) and SBP-Projection, where SBP is an acronym for *summation by parts*.

In the first assignment, the two methods are applied to the 1D version of Maxwell's equation, consisting of two components. Two different boundary conditions (BC:s) are imposed: characteristic BC and Dirichlet BC. The results are compared to the analytical solution, and the convergence rates are being discussed.

# 2 Theory

## 2.1 The Energy Method

In order to achieve a stable solution to a PDE, it is important to make sure that the problem is well-posed which means that there exists a unique solution and that the energy of the problem is bounded according to the energy estimate in (2). Here, $K$ and $\eta$ are some constants independent of $\mathbf{f}$ and $||\mathbf{u}||_{\mathbf{C}}$ is a norm. This energy estimate is independent of the boundary data and any (if existent) forcing function, and is derived for the continous problem.

$$||\mathbf{u}||_{\mathbf{C}} \leq Ke^{\eta t}||\mathbf{f}(\cdot)||_{\mathbf{C}} \tag{2}$$

In order to derive an stable energy estimate, the simplification in equation (3) can be used. If this equation is set to being equal to zero, energy conservation is achieved.

$$\frac{d}{dt}||\mathbf{u}||_{\mathbf{C}}^2 \leq 0 \tag{3}$$

Defining the inner product as equation (4), where $*$ denotes the conjugate transpose, the energy estimate can be written as equation (5), if $\mathbf{C}$ is a hermitian matrix.

$$(\mathbf{u}, \mathbf{w})_C = \int_{x_l}^{x_r} \mathbf{u}^* \mathbf{C} \mathbf{w} dx \tag{4}$$

$$\frac{d}{dt}||\mathbf{u}||_{\mathbf{C}} = (\mathbf{u}, \mathbf{u})_C + (\mathbf{u}, \mathbf{u})_C^* \leq 0 \tag{5}$$

## 2.2 SBP-operators

In order to implement a numerical solution, a continuous problem needs to be discretized. Since the energy method above is derived for the continuous problem, a discrete version is needed. SBP mimics integration by parts (IBP), by instead summating by parts. The energy estimate is then given by the following

$$||\mathbf{v}||_{\mathbf{HC}}^2 = \mathbf{v}^T \tilde{H} \mathbf{C} \mathbf{v} \leq 0 \tag{6}$$

where $v^T = [v^{(1)}, ..., v^{(k)}]$ is the discrete solution vector consisting of $k$ components, where the $i^{\text{it}}$ component is given by $v^{(i)T} = [v_1^{(i)},\ v_2^{(i)}, ..., v_m^{(i)}]$. $H$ is a positive and symmetric matrix, and $\tilde{H} = I_k \otimes H$ where $I_k$ defines is the identity matrix. The 1D domain $x \in [x_l,\ x_r]$ is discretized into $m$ equidistant points, giving a spatial step size $h = \frac{x_r - x_l}{m-1}$. The unit vectors are defined in equation (7).

$$\begin{aligned} e_1 &= [1, 0, ..., 0]^T \\ e_m &= [0, ..., 0, 1]^T \end{aligned} \tag{7}$$

Using the things stated above, the SBP-operator for the first derivative is given by equation (8), where Q is a symmetric matrix. SBP-operators can be created at different levels of accuracy, but in this project a $4^t h$ order accurate SBP-operator is used, called SBP4. The code for creating the SBP-operators was given to me by the teacher.

$$D_1 = H^{-1}(Q - \frac{1}{2}e_1 e_1^T + \frac{1}{2}e_m e_m^T) \tag{8}$$

## 2.3 SBP-SAT Method

SAT is, combined with SBP, a way to impose the BC:s in a way to preserve the SBP property and numerical stability. This method sets the BC:s weakly, meaning that they can somewhat fluctuate around the real value. Applying the SBP-SAT method to a PDE involving on the form of (14), the equation below follows, were the SAT-terms are defined in equation (10), where $L_{l,r}$ denotes the boundary operators on the left and right sides of the domain, and $g_{l,r}$ are the BC:s

$$\mathbf{C}v_t = (\mathbf{A} \otimes \mathbf{D}_1)v + SAT_l + SAT_r \tag{9}$$

$$\begin{aligned} SAT_l &= \tau_l \otimes \left(\mathbf{H}^{-1} e_1 (L_l^T v - g_l)\right) \\ SAT_r &= \tau_r \otimes \left(\mathbf{H}^{-1} e_m (L_r^T v - g_r)\right) \end{aligned} \tag{10}$$

$\tau_{l,r}$ are the tuning parameters at each boundary for the problem, and has $k$ components each, and these parameters are chosen such that the method is well-posed according to the semi-discrete energy method.

## 2.4 SBP-Projection Method

SBP-Projection is a method to strongly impose the BC:s, setting them to the exact intended value, while still maintaining stability of the numerical solution, via orthogonal projection. In contrast to the SBP-SAT method, there are no tuning parameters involved. Denoting the boundary operator as $L$, the projection operator $P$ can be defined as in equation (11), where $I$ is the identity matrix of size $k * m$ (where $k$ is the amount of components of $v$ and $m$ is the amount of grid points), and $\tilde{H}$ is defined as before in the SBP-section.

$$P = I - \tilde{H}^{-1} L (L^T \tilde{H}^{-1} L)^{-1} L^T \tag{11}$$

For a problem of the form (14), the following stable semi-discrete formulation is achieved, where $D_1$ is the SBP-operator for the first derivative.

$$\mathbf{C}v_t = P(A \otimes D_1)Pv \tag{12}$$

## 2.5 Time-Stepping

Both semi-discrete approximation methods were discretized in time using the $4^{\text{th}}$-order Runge-Kutta method. To ensure stability, the CFL-condition was calculated using the spectral radius $\rho(A_v) = max(|\lambda|)$, where $A_v$ comes from the expression $Cv_t = A_v v$ and it depends on which method and which BC:s that were used in the implementation. $\lambda$ is the eigenvalues to the matrix $A_v$. The CFL-condition for RK-4 is then calculated by the following

$$CFL = \frac{2.8}{\rho(A_v)} \tag{13}$$

From this, the time step $k$ can be calculated as $k = \frac{CFL}{5} h$, where $h$ is the spatial step size defined by $h = \frac{x_r - x_l}{m-1}$. The CFL-condition is divided by 5 to ensure that the temporal truncation error is negligible compared to the spatial truncation error.

# 3 Derivation of Stable SBP-SAT and SBP-Projection

## 3.1 Introduction

In this part, the considered PDE is Maxwell's equation in 1D given in (14). The coefficient matrices are given below. Here, $\mathbf{u} = \begin{bmatrix} E^{(y)} & H^{(z)} \end{bmatrix}^T$, but for simplicity the notation $\mathbf{u}(x,t) = \begin{bmatrix} u^{(1)}(x,t) & u^{(2)}(x,t) \end{bmatrix}^T$ is used instead. The permittivity and the permeability is set to $\epsilon = \mu = 1$. For this problem, the domain is set to span $x \in [x_l,\ x_r]$, where $x_l = -1$ and $x_r = 1$, and $t \in (0,\ T)$ where $T = 1.8$.

$$\begin{cases} \mathbf{C}\mathbf{u}_t = \mathbf{A}\mathbf{u}_x, & x \in [x_l, x_r]\ , t \in (0,\ T) \\ \mathbf{u}(x,0) = \mathbf{u}_0(x), & x \in [x_l, x_r] \end{cases} \tag{14}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \epsilon & 0 \\ 0 & \mu \end{bmatrix}$$

The analytical solution is given by equation (15), where $L$ denotes the length of the domain. The variable $\boldsymbol{\theta}$ is defined in equation (16) which are some Gaussian profiles with width $r$. In the calculations, $r = 0.1$ is used.

$$\mathbf{u} = \begin{bmatrix} -\theta^{(1)}(x, L-t) - \theta^{(2)}(x, L-t) \\ -\theta^{(1)}(x, L-t) + \theta^{(2)}(x, L-t) \end{bmatrix} \tag{15}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \exp\Big[ -\Big(\frac{x-t}{r}\Big)^2 \Big] \\ -\exp\Big[ -\Big(\frac{x+t}{r}\Big)^2 \Big] \end{bmatrix} \tag{16}$$

The initial condition, $\mathbf{u}_0(x)$ to the problem is given by the following equation

$$\mathbf{u}(x,0) = \begin{bmatrix} \theta^{(2)}(x,0) - \theta^{(1)}(x,0) \\ \theta^{(2)}(x,0) + \theta^{(2)}(x,0) \end{bmatrix} \tag{17}$$

## 3.2 Dirichlet Boundary Conditions

In order to derive the correct (minimal) amount of Dirichlet boundary conditions, the energy method needs to be introduced. Here, the norm is written as the inner product $(\cdot, \cdot)$. The different terms used in the energy method is seen in equation (18).

$$\begin{cases} (\mathbf{u},\ \mathbf{u}_t)_{\mathbf{C}} = (\mathbf{u},\ \mathbf{A}\mathbf{u}_x) = [\mathbf{u}^*\mathbf{A}\mathbf{u}]_{x_l}^{x_r} - (\mathbf{u}_x^*,\ \mathbf{A}\mathbf{u}) \\ (\mathbf{u}_t,\ \mathbf{u})_{\mathbf{C}} = (\mathbf{A}\mathbf{u}_x,\ \mathbf{u}) = (\mathbf{u}_x,\ \mathbf{A}^*\mathbf{u}) \end{cases} \tag{18}$$

Using the fact that $(\mathbf{u},\ \mathbf{u}_t)_{\mathbf{C}} + (\mathbf{u}_t,\ \mathbf{u})_{\mathbf{C}} = \frac{d}{dt}(\mathbf{u},\ \mathbf{u})_{\mathbf{C}} = \frac{d}{dt}||\mathbf{u}||_{\mathbf{C}}^2$, an energy estimate can be derived as the following:

$$\frac{d}{dt}||\mathbf{u}||_{\mathbf{C}}^2 = [\mathbf{u}^*\mathbf{A}\mathbf{u}]_{x_l}^{x_r} - (\mathbf{u}_x,\ \mathbf{A}\mathbf{u}) + (\mathbf{u}_x,\ \mathbf{A}^*\mathbf{u}) = [\mathbf{u}^*\mathbf{A}\mathbf{u}]_{x_l}^{x_r} - \Big(\mathbf{u}_x,\ (\mathbf{A} - \mathbf{A}^*)\mathbf{u}\Big) \tag{19}$$

Since $\mathbf{A}$ is a Hermitian matrix, $\mathbf{A}$ has the property $\mathbf{A}^* = \mathbf{A}$. Hence, equation (19) can be written as equation 20, where the inequality is introduced to establish numerical stability.

$$\frac{d}{dt}||\mathbf{u}||_{\mathbf{C}}^2 = [\mathbf{u}^*\mathbf{A}\mathbf{u}]_{x_l}^{x_r} \leq 0 \tag{20}$$

Since the problem states that $u$ has two components $\mathbf{u} = [u^{(1)},\ u^{(2)}]^T$, the following inequality is achieved:

$$2u^{(1)}u^{(2)}\Big|^{x_r} - 2u^{(1)}u^{(2)}\Big|^{x_l} \leq 0 \tag{21}$$

From equation (21), the correct amount of boundary conditions can be derived. In order to fulfill energy conservation, the energy estimate in said equation must equal to zero. This is achieved by making sure that the two terms equals to zero, which leads to the following well-posed set of Dirichlet boundary conditions:

$$\begin{cases} u^{(1)}(x) = 0, & x = x_l \\ u^{(2)}(x) = 0, & x = x_r \end{cases} \tag{22}$$

### 3.2.1 SBP-SAT

The semi-discretized approximation of the Dirichlet boundary conditions in equation (22) are given by equation (23). The following is defined as $e^{(1)} = [1,\ 0]$ and $e^{(2)} = [0,\ 1]$.

$$\begin{cases} L_l v = v_1^{(1)} = (e^{(1)} \otimes e_1^T)v = 0 \\ L_r v = v_m^{(2)} = (e^{(2)} \otimes e_m^T)v = 0 \end{cases} \tag{23}$$

When imposing these discrete Dirichlet boundary conditions together with a SBP-SAT semi-discretization of equation (14), equation (24) follows:

$$\mathbf{C}v_t = (\mathbf{A} \otimes \mathbf{D}_1)v + \tau_l \otimes \left(\mathbf{H}^{-1}e_1(e^{(1)} \otimes e_1^T)v\right) + \tau_r \otimes \left(\mathbf{H}^{-1}e_m(e^{(2)} \otimes e_m^T)v\right) \tag{24}$$

In order to find the tuning parameters $\tau_l = [\tau_l^{(1)}, \tau_l^{(2)}]^T$ and $\tau_r = [\tau_r^{(1)}, \tau_r^{(2)}]^T$, equation (24) is multiplied with $v^T I_2 \otimes H$ on both sides and then added to the transpose. This yields the energy estimate:

$$\frac{d}{dt}||v||^2_{\mathbf{HC}} = \tau_l^{(1)}v_1^{(1)}v_1^{(1)} + (\tau_l^{(2)} - 1)v_1^{(1)}v_1^{(2)} + (\tau_r^{(1)} + 1)v_m^{(1)}v_m^{(2)} + \tau_r^{(2)}v_m^{(2)}v_m^{(2)} \tag{25}$$

In order to get stability the tuning parameters need to be as the following, where equality yields energy conservation.

$$\begin{bmatrix} \tau_l^{(1)} \\ \tau_l^{(2)} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} \tau_r^{(1)} \\ \tau_r^{(2)} \end{bmatrix} \leq \begin{bmatrix} -1 \\ 0 \end{bmatrix} \tag{26}$$

### 3.2.2 SBP-Projection

The SBP-Projection approximation of (14) is

$$Cv_t = P(A \otimes D_1)Pv \tag{27}$$

where the projection operator is defined as $P = I - \tilde{H}L\tilde{B}L^T$. From the previous derivations, the boundary operator for the Dirichlet BC was found to be

$$L^T = \begin{bmatrix} e^{(1)} \otimes e_1 \\ e^{(1)} \otimes e_m \end{bmatrix} \tag{28}$$

where $I$ is the identity matrix of size $2m$ and $\tilde{H} = I_2 \otimes H$, where $H$ is the SBP-operator and $I_2$ is the identity matrix of size 2. The matrix $\tilde{B} = \left(L^T\tilde{H}^{-1}L\right)$.

### 3.3 Characteristic Boundary Conditions

In order to find the characteristic boundary conditions, equation (14) first needs to be diagonalized. To do this, the relation $\mathbf{A} = \mathbf{S\Lambda S^{-1}}$ is used, where $\mathbf{\Lambda}$ is the diagonal matrix containing the eigenvalues to $\mathbf{A}$, and $\mathbf{S}$ is an invertible matrix that diagonalizes $\mathbf{A}$. The eigenvalues to $\mathbf{A}$ is $\lambda_1 = -1,\ \lambda_2 = 1$, thus getting $\mathbf{\Lambda} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$, $\mathbf{S} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}$ and $\mathbf{S}^{-1} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$

Diagonalizing equation (14): $\mathbf{S^{-1}CSu}_t = \mathbf{S^{-1}AS u}_x$, and then multiplying by $\mathbf{S}^{-1}$ on both sides, thus having the expression

$$\mathbf{S^{-1}CSS^{-1}u}_t = \mathbf{\Lambda S^{-1}u}_x \tag{29}$$

Introducing the variable $\tilde{\mathbf{u}} = \mathbf{S^{-1}u}$ as well as $\mathbf{C}_S = \mathbf{S^{-1}CS}$ to simplify the calculations, equation (30) is achieved.

$$\mathbf{C}_S\tilde{\mathbf{u}}_t = \mathbf{\Lambda}\tilde{\mathbf{u}}_x \tag{30}$$

By applying the energy method on equation (30) in similar way as in equation (18)-(19), the energy estimate can be expressed as (31). Inserting the values of $\lambda$, the inequality in equation (32) is achieved.

$$\frac{d}{dt}||\tilde{\mathbf{u}}||^2_{C_S} = \Big[\tilde{\mathbf{u}}^*\mathbf{\Lambda u}\Big]_{x_l}^{x_r} = \left[\sum_{i=1}^{k}\lambda^{(i)}|\tilde{u}^{(i)}|^2\right]_{x_l}^{x_r} \leq 0 \tag{31}$$

$$-|\tilde{u}^{(1)}|^2\Big|^{x_r} + |\tilde{u}^{(2)}|^2\Big|^{x_r} + |\tilde{u}^{(1)}|^2\Big|^{x_l} - |\tilde{u}^{(2)}|^2\Big|^{x_l} \leq 0 \tag{32}$$

Since $|\tilde{u}^{(i)}|^2$ always is positive, $\tilde{u}^{(1)}$ does not need to be specified at the right boundary, and $|\tilde{u}^{(2)}|$ does not need to be specified at the left boundary. This due to the minus sign in front of the terms. In order for the energy estimate to hold the following needs to be satisfied:

$$\begin{cases} \tilde{u}^{(1)} = 0 & \text{at } x = x_l \\ \tilde{u}^{(2)} = 0 & \text{at } x = x_r \end{cases} \tag{33}$$

Rewriting the characteristic variable as $\begin{bmatrix} \tilde{u}^{(1)} \\ \tilde{u}^{(2)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}u^{(1)} - \frac{1}{2}u^{(2)} \\ \frac{1}{2}u^{(1)} + \frac{1}{2}u^{(2)} \end{bmatrix}$ and inserting in the above equation, thus getting the boundary conditions in equation (34).

$$\begin{cases} u^{(1)} - u^{(2)} = 0 & \text{at } x = x_l \\ u^{(1)} + u^{(2)} = 0 & \text{at } x = x_r \end{cases} \tag{34}$$

#### 3.3.1 SBP-SAT

The semi-discretized approximation of the characteristic boundary conditions in equation (34) are given by equation (35).

$$\begin{cases} L_l v = v_1^{(1)} - v_1^{(2)} = \Big[(e^{(1)} \otimes e_1^T) - (e^{(2)} \otimes e_1^T)\Big]v = 0 \\ L_r v = v_m^{(2)} + v_m^{(2)} = \Big[(e^{(1)} \otimes e_m^T) + (e^{(2)} \otimes e_m^T)\Big]v = 0 \end{cases} \tag{35}$$

From this, we get the semi-discrete SBP-SAT formulation of equation (14):

$$\mathbf{C}v_t = (\mathbf{A}\otimes\mathbf{D}_1)+\tau_l\otimes\mathbf{H}^{-1}e_1\Big(\Big[(e^{(1)}\otimes e_1^T)-(e^{(2)}\otimes e_1^T)\Big]v\Big)+\tau_r\otimes\mathbf{H}^{-1}e_m\Big(\Big[(e^{(1)}\otimes e_m^T)+(e^{(2)}\otimes e_m^T)\Big]v\Big) \quad (36)$$

In order to find the tuning parameters $\tau_l = [\tau_l^{(1)}, \tau_l^{(2)}]^T$ and $\tau_r = [\tau_r^{(1)}, \tau_r^{(2)}]^T$, equation (36) is multiplied with $v^T I_2 \otimes H$ on both sides and then added to the transpose. This yields the energy estimate:

$$\frac{d}{dt}||v||_C^2 = \Big[1+\tau_r^{(1)}+\tau_r^{(2)}\Big]v_m^{(1)}v_m^{(2)}+\Big[\tau_l^{(2)}-\tau_l^{(1)}-1\Big]v_1^{(1)}v_1^{(2)}+\tau_l^{(1)}v_1^{(1)}v_1^{(1)}-\tau_l v_1^{(2)}v_1^{(2)}+\tau_r^{(1)}v_m^{(1)}v_m^{(1)}+\tau_r v_m^{(2)}v_m^{(2)} \leq 0 \quad (37)$$

To fulfill the inequality in the equation above, we have the following constraints:

$$\begin{cases} 1+\tau_r^{(1)}+\tau_r^{(2)} \leq 0 \\ \tau_l^{(2)}-\tau_l^{(1)}-1 \leq 0 \\ \tau_l^{(1)} \leq 0, \qquad \tau_l^{(2)} \geq 0 \\ \tau_r^{(1)} \leq 0, \qquad \tau_r^{(2)} \leq 0 \end{cases} \quad (38)$$

Finding tuning parameters that fulfills the constraints results in the following values:

$$\begin{bmatrix} \tau_l^{(1)} \\ \tau_l^{(2)} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \qquad \begin{bmatrix} \tau_r^{(1)} \\ \tau_r^{(2)} \end{bmatrix} \leq \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \quad (39)$$

### 3.3.2 SBP-Projection

The SBP-Projection approximation of (14) is

$$Cv_t = P(A\otimes D_1)Pv \quad (40)$$

where the projection operator is defined as $P = I - \tilde{H}L\tilde{B}L^T$. From the previous derivations, the boundary operator for the Dirichlet BC was found to be

$$L^T = \begin{bmatrix} e^{(1)}\otimes e_1 - e^{(2)}\otimes e_1 \\ e^{(1)}\otimes e_m + e^{(2)}\otimes e_m \end{bmatrix} \quad (41)$$

where $I$ is the identity matrix of size $2m$ and $\tilde{H} = I_2 \otimes H$, where $H$ is the SBP-operator and $I_2$ is the identity matrix of size 2. The matrix $\tilde{B} = \Big(L^T\tilde{H}^{-1}L\Big)$.

## 3.4 Results

### 3.4.1 Comparing with Analytical Solution

Running the calculations with $m = 201$ grid points, to $T = 1.8$ gave the results presented in Figure 1-4. Here, it can clearly be seen that the solution has dissipated for the characteristic BC, and there are only some small oscillations left. As for the Dirichlet BC, the calculated solution matches the analytical solution well for both SBP-SAT and SBP-Projection.

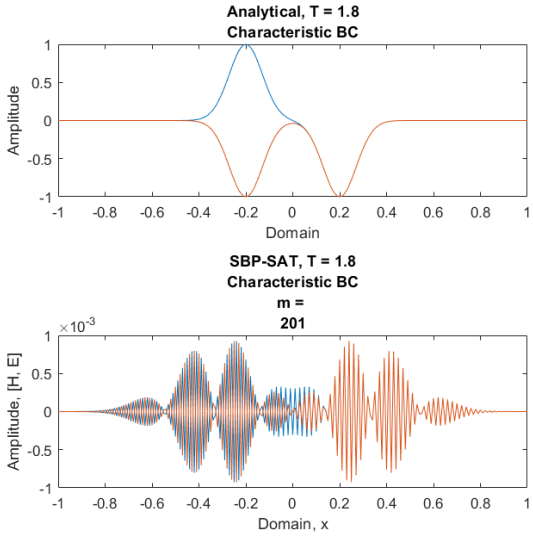


**Figure 1:** *The results for SBP-SAT together with Characteristic BC, simulated for $m = 201$ grid points. Compared to analytical solution.*

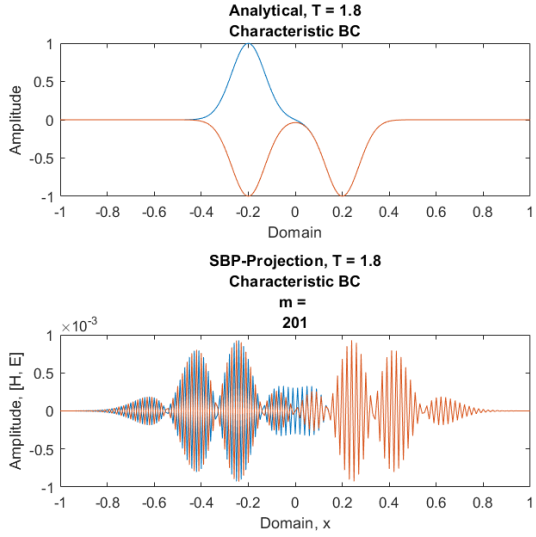


**Figure 2:** *The results for SBP-Projection together with Characteristic BC, simulated for $m = 201$ grid points. Compared to analytical solution.*

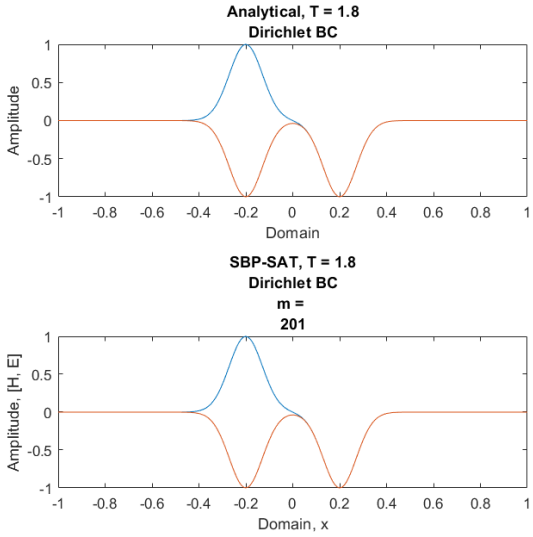


**Figure 3:** *The results for SBP-SAT together with Dirichlet BC, simulated for* $m = 201$ *grid points. Compared to analytical solution.*

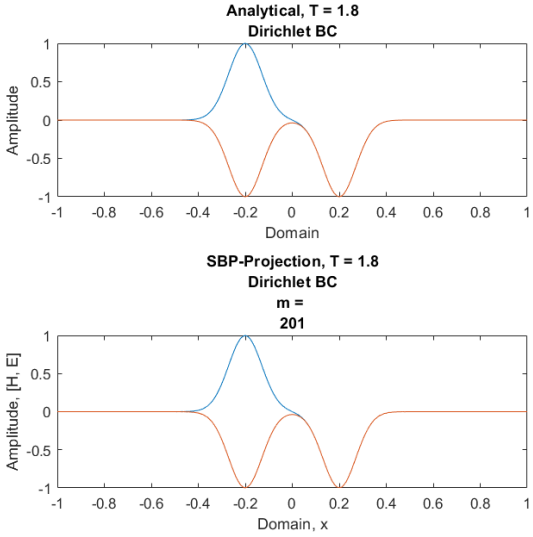


**Figure 4:** *The results for SBP-SAT together with Dirichlet BC, simulated for* $m = 201$ *grid points. Compared to analytical solution.*

### 3.4.2 Boundary Interaction

When $T = 1$, the peaks interacts with the boundaries. The results are presented in Figure 5-8. It can be seen that the interaction with the boundaries are highly dependent on how the BC:s are imposed. The charac. BC:s are more similar to the analytical solution than the Dirichlet BS, however, the sides are flipped compared to the analytical solution. This could be due to how the boundary conditions were chosen when deriving the stable energy estimates. The Dirichlet BC is, as derived, zero for one of the components at the left side, and zero for the other component at the right side. This differs more from the analytical solution, however, according to the results for $T = 1.8$, the Dirichlet conditions still ends up being more correct. There seem to be no significant difference between the two different methods.

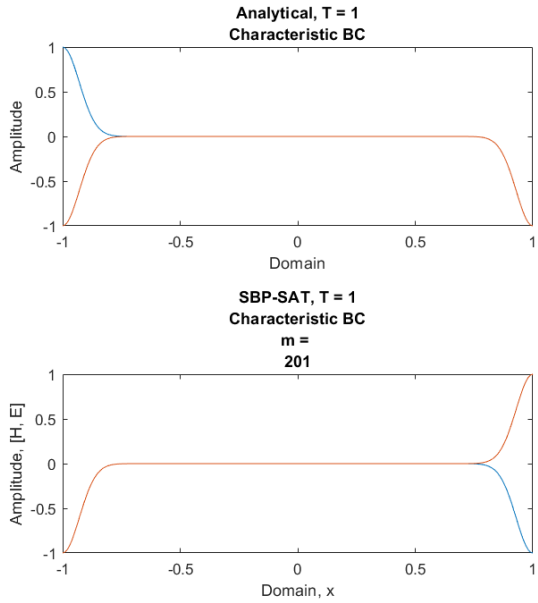


**Figure 5:** *The results for SBP-SAT together with charac. BC, simulated for* $m = 201$ *grid points,* $T = 1$*. Compared to analytical solution.*

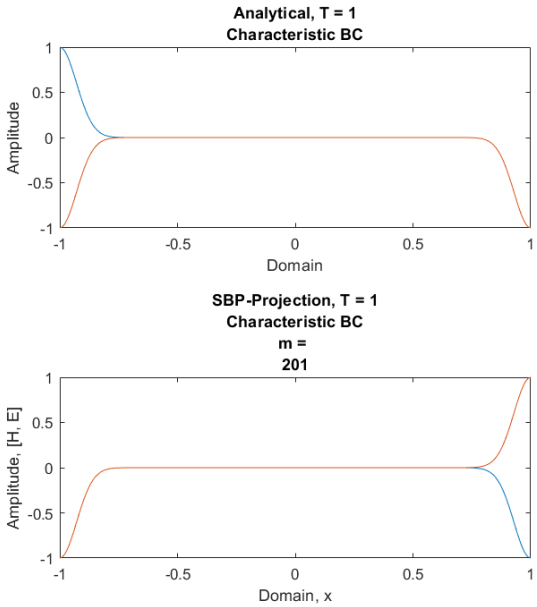


**Figure 6:** *The results for SBP-Projection together with charac. BC, simulated for* $m = 201$ *grid points,* $T = 1$*. Compared to analytical solution.*

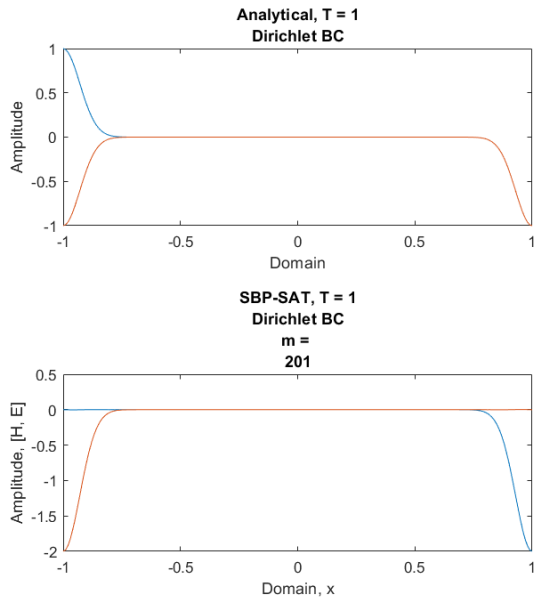


**Figure 7:** *The results for SBP-SAT together with Dirichlet BC, simulated for $m = 201$ grid points, $T = 1$. Compared to analytical solution.*

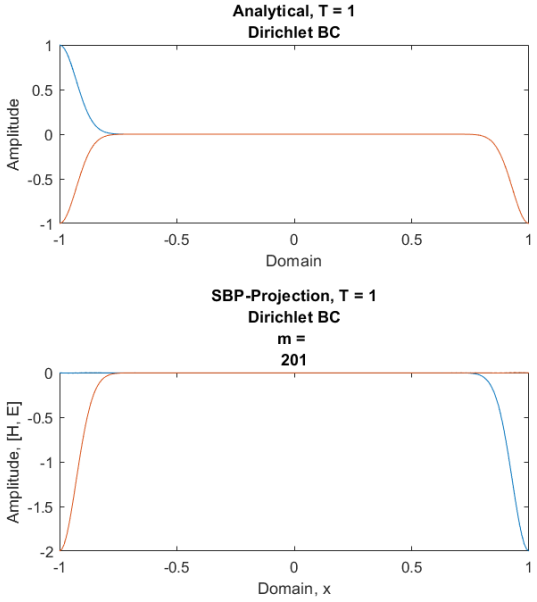


**Figure 8:** *The results for SBP-Projection together with Dirichlet BC, simulated for $m = 201$ grid points, $T = 1$. Compared to analytical solution.*

### 3.4.3 Time Stepping

In Table 1, the CFL-condition is calculated at $m = 101$. Here it can be seen that the there is no difference between the SAT and Projection methods for characteristic BC. However, for the Dirichlet BC it can be seen that the CFL-condition is significantly smaller for the SAT-method. This means that for Dirichlet BC, SAT allows a bigger time step without compromising the stability of the calculations.

**Table 1:** *Presenting the CFL-condition for the two different methods for differently imposed BC:s. Here calculated for $m = 101$.*

| | **SAT** | **Projection** |
|---|---|---|
| **Characteristic** | 0.0408 | 0.0408 |
| **Dirichlet** | 0.0289 | 0.0408 |

## 3.5 Eigenvalues

Writing the problem on the form $Cv_t = A_v v$, where $A_v$ is a matrix depending on what method and BC:s that were used, the eigenvalues can be calculated. The eigenvalues are important since they determine the stability of the problem. In Figure 9-12, the eigenvalues are plotted. Here, it can be seen that the eigenvalues are vastly different between the charac. BC and the Dirichlet BC:s, but only some small differences between the two methods.

As seen for the charac. BC:s, the eigenvalues tends towards a value where $Re(\lambda) << 0$, and $Im(\lambda) \approx 0$. This leads to a dampened system, which is consistent with the results presented earlier. As for the Dirichlet

BC:s, the eigenvalues are really small and close to zero. However, some of them have a positive value of $Re(\lambda)$, but these are on the scale of $10^{-14}$, and thus not causing any devastating instabilities.

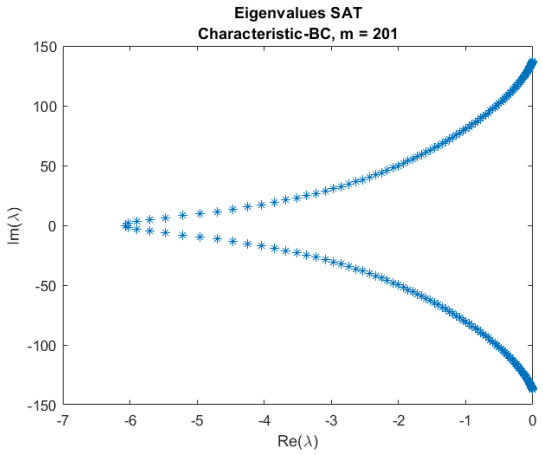


**Figure 9:** *The eigenvalues for SBP-SAT with charac. BC.*

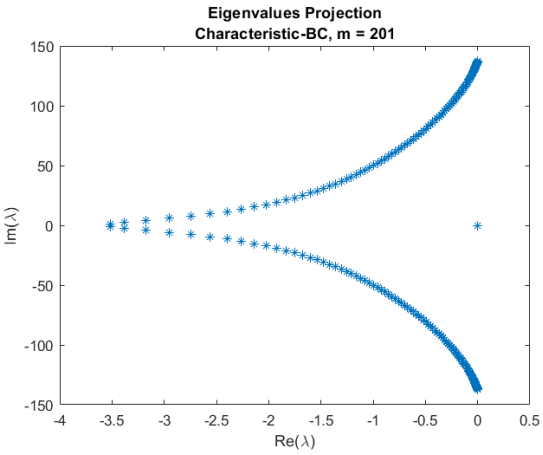


**Figure 10:** *The eigenvalues for SBP-Projection with charac. BC.*

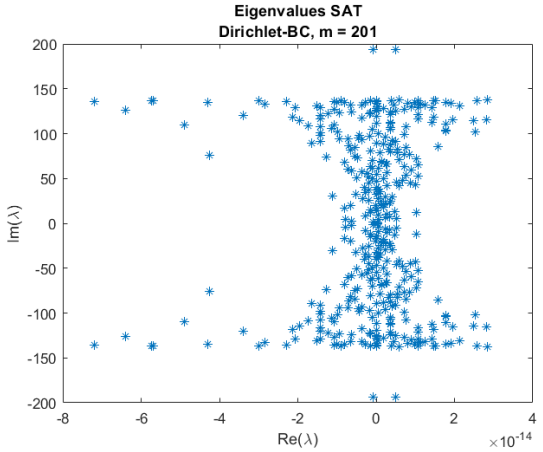


**Figure 11:** *The eigenvalues for SBP-SAT with Dirichlet BC.*

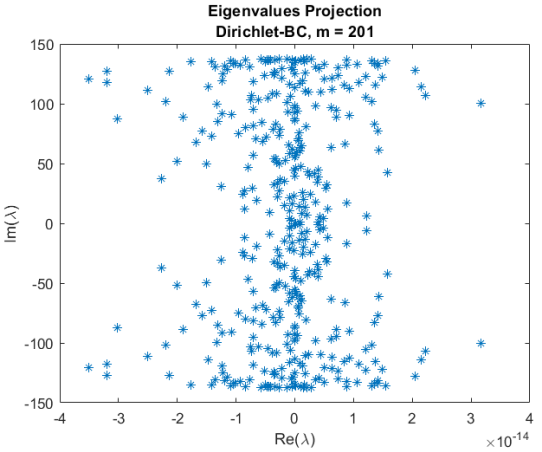


**Figure 12:** *The eigenvalues for SBP-Projection with Dirichlet BC.*

### 3.5.1 Convergence

When studying the convergence, the error was calculated by taking the discrete $l^2$-norm by using $\mathbf{H}$, defined as $||v||_H = \sqrt{v^* \mathbf{H} v}$. The calculated results for SAT and Projection together with the two BC:s can be seen in Table 2. Here, it can clearly be seen that the characteristic BC does not converge towards the analytical solution independent of the chosen method. As for the Dirichlet BC, the two methods performs rather similar to each other, and the error is decreasing steadily.

When using MATLABs function `Polyfit` to fit a linear polynom to the error, it can be seen that the SAT-method reaches a convergence of around 3.8 for Dirichlet BC, while Projection reaches a convergence rate of around 3.2. This is demonstrated in Figure 13 and 14. The convergence rates are presented in Table 3 were the fitted slope for the characteristic BC:s also can be seen. However, these performed really poorly.

**Table 2:** *The error calculated in the $l_2$-norm for different grid sizes $m$.*

| | Charac. | | Dirichlet | |
|---|---|---|---|---|
| **m** | **SAT** | **Projection** | **SAT** | **Projection** |
| **61** | 0.7087 | 0.7087 | 0.1185 | 0.1122 |
| **101** | 0.7081 | 0.7081 | 2.135E(-2) | 2.241 E(-2) |
| **161** | 0.7080 | 0.7080 | 3.283E(-3) | 4.134 E(-3) |
| **201** | 0.7080 | 0.7080 | 1.401E(-3) | 1.992 E(-3) |
| **401** | 0.7080 | 0.7080 | 7.843E(-5) | 2.101 E(-4) |
| **601** | 0.7080 | 0.7080 | 2.133E(-5) | 6.406 E(-5) |

**Table 3:** *The convergence calculated by fitting a linear polynom for the $l^2$-errors.*

| | Charac. | | Dirichlet | |
|---|---|---|---|---|
| | **SAT** | **Projection** | **SAT** | **Projection** |
| **P** | 2.844E(-4) | 2.844E(-4) | 3.827 | 3.274 |

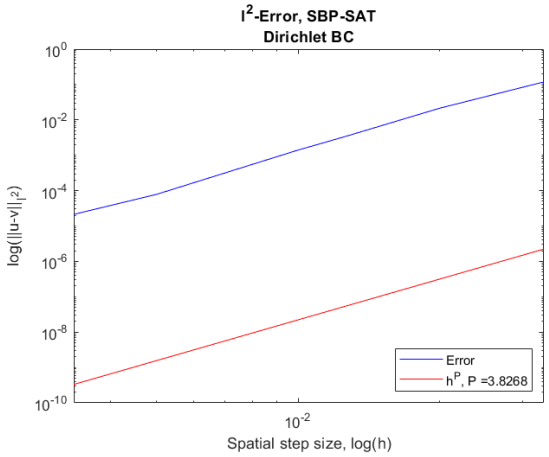


**Figure 13:** *hejehj*

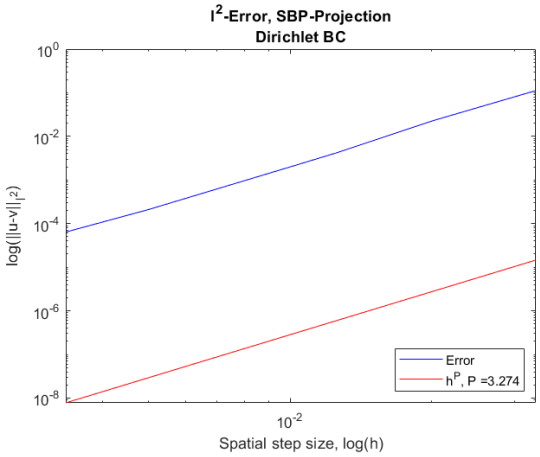


**Figure 14:** *hejehj*

The convergence rate between each size of $m$ can be calculated as the following:

$$q = \log_{10}\left(\frac{||u - v^{m_1}||_h}{||u - v^{m_2}||_h}\right) \Big/ \log_{10}\left(\frac{m_1}{m_2}\right) \tag{42}$$

This gives an indication on how the convergence increases or decreases with respect to the amount of grid points that are being used. The results are presented in Table 4. Here it can be seen, as previously stated, that the characteristic boundaries does not converge at all. The convergence rate instead decreases with an increasing amount of grid points. As for the Dirichlet BC, it can be seen that the SAT method consistently performs better than the Projection method. Since both methods are using $4^{th}$-order SBP-operators, the results of the SAT-method corresponds better to what was expected.

**Table 4:** *Compares the q-value for SBP-SAT and SBP-Projection for different grid-sizes m. Two different BC:s used.*

| | Charac. | | Dirichlet | |
|---|---|---|---|---|
| **m** | **SAT** | **Projection** | **SAT** | **Projection** |
| **61** | - | - | - | - |
| **101** | 1.643E(-3) | 1.643E(-3) | 3.354 | 3.153 |
| **161** | 8.160E(-5) | 8.160E(-5) | 3.984 | 3.596 |
| **201** | 7.846E(-6) | 7.846(E-6) | 3.815 | 3.272 |
| **401** | 8.753E(-7) | 8.753E(-7) | 4.159 | 3.245 |
| **601** | 2.146E(-8) | 2.146(E-8) | 3.212 | 2.930 |

## 3.6 Discussion

From the results presented, it can be seen that there was a major difference between the two different BC:s. The Dirichlet BC corresponded better to the analytical solution after the interaction with the boundaries, while the characteristic BC:s had significant damping, which is consistent with the eigenvalues of the problems. However, the characteristic BC:s corresponded better to the analytical solution at the time when the peaks interacted with the boundaries (although flipped). This could probably mean that depending on what is being investigated, the choice of imposing the BC:s should be given some thought.

The difference between the two methods were not very big. SBP-SAT had a little higher convergence rate compared to SBP-Projection, 3.8 compared to 3.3, and thus performed somewhat better. However, the SBP-Projection method was significantly easier to implement, since it did not have any tuning parameters that needed to be determined.

# 4 Discontinuous Domain

In this part of the assignment, a discontinuous domain was introduced to Maxwell's equations in 1D. This discontinuity was in the form of different relative permeabilities, $\varepsilon_{l,r}$ on different sides of the domain. This leads to two different $\mathbf{C}$-matrices:

$$\mathbf{C}_l = \begin{bmatrix} \varepsilon_l & 0 \\ 0 & \mu \end{bmatrix} \qquad \mathbf{C}_r = \begin{bmatrix} \varepsilon_r & 0 \\ 0 & \mu \end{bmatrix}$$

Since the solution to equation (14) is dependent on $\mathbf{C}$, this results in two different solutions in the two blocks of the domain:

$$\begin{cases} \mathbf{C}_l \mathbf{u}_t^{(l)} = \mathbf{A}\mathbf{u}_x^{(l)}, & x \in [x_l, x_I]\,, t \in (0,\ T) \\ \mathbf{C}_r \mathbf{u}^{(\mathbf{r})}{}_t = \mathbf{A}\mathbf{u}_x^{(r)}, & x \in [x_I, x_r]\,, t \in (0,\ T) \end{cases} \tag{43}$$

In order for the solution to be continuous over the whole domain, the two equations need to be set to equal each other at the shared points $x_I$: $\mathbf{C}_l \mathbf{u}^{(\mathbf{l})}{}_t = \mathbf{C}_r \mathbf{u}^{(\mathbf{r})}{}_t$. This point is called an internal boundary, or interface, because of the need of extra boundary conditions which are not actually situated at the boundaries of the real domain.

Defining the following vectors as shown shown below. These are later used to choose component from the solution vector, which is defined as $v^T = [v_{l,1}^{(1)},\ ..., v_{l,m}^{(1)},\ v_{l,1}^{(2)},\ ..., v_{l,m}^{(2)},\ v_{r,1}^{(1)},\ ..., v_{r,m}^{(1)},\ v_{r,1}^{(2)},\ ..., v_{r,m}^{(2)}]$.

$$\begin{aligned} e^{(1,l)} &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \\ e^{(2,l)} &= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\ e^{(1,r)} &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \\ e^{(2,r)} &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{44}$$

## 4.1 Deriving Internal BC:s

In order to find a stable solution, the energy method are applied to the both equations in (43):

$$\begin{cases} (u^{(l)},\ \mathbf{C}_l u^{(l)}) = (u^{(l)},\ \mathbf{A}u_x^{(l)}) = \left[u^{(l)*}\mathbf{A}u^{(l)}\right]_{x_l}^{x_I} - (u_x^{(l)},\ \mathbf{A}u^{(l)}) \\ (u^{(r)},\ \mathbf{C}_r u^{(r)}) = (u^{(l)},\ \mathbf{A}u_x^{(r)}) = \left[u^{(r)*}\mathbf{A}u^{(r)}\right]_{x_I}^{x_r} - (u_x^{(r)},\ \mathbf{A}u^{(r)}) \end{cases} \implies \begin{cases} \frac{d}{dt}||u^{(l)}||_{\mathbf{C}_l}^2 = \left[u^{(l)*}\mathbf{A}u^{(l)}\right]_{x_l}^{x_I} \\ \frac{d}{dt}||u^{(r)}||_{\mathbf{C}_r}^2 = \left[u^{(r)*}\mathbf{A}u^{(r)}\right]_{x_I}^{x_r} \end{cases} \tag{45}$$

This procedure gives the following energy estimate:

$$\frac{d}{dt}\left(||u^{(l)}||_{\mathbf{C}_l}^2 + ||u^{(r)}||_{\mathbf{C}_r}^2\right) = \left(u^{(l)*}\mathbf{A}u^{(l)} - u^{(r)*}\mathbf{A}u^{(r)}\right)\Big|^{x_I} - u^{(l)*}\mathbf{A}u^{(l)}\Big|^{x_l} + u^{(r)*}\mathbf{A}u^{(r)}\Big|^{x_r} \leq 0 \tag{46}$$

From this, we have the internal boundary terms IT $= u^{(l)*}\mathbf{A}u^{(l)} - u^{(r)*}\mathbf{A}u^{(r)}$ at $x = x_I$. For the energy estimate to hold, this should equal to zero: $IT = 2u^{(1,l)}u^{(2,l)} - 2u^{(1,r)}u^{(2,r)} = 0$. This condition is fulfilled if the last point of the components in the left domain are equal to the first point of the components in the right domain. This can be written as:

$$\begin{cases} (e^{(1,l)} \otimes e_m^T)v = (e^{(1,r)} \otimes e_1^T)v \\ (e^{(2,l)} \otimes e_m^T)v = (e^{(2,l)} \otimes e_1^T)v \end{cases} \tag{47}$$

The boundary operator thus becomes the following, where $L_{l,r}$ depends on the chosen BC:s at the external boundaries (i.e. Dirichlet or characteristic).

$$\mathbf{L} = \begin{bmatrix} L_l \\ e^{(1,l)} \otimes e_m^T - e^{(1,r)} \otimes e_1^T \\ e^{(2,l)} \otimes e_m^T - e^{(2,l)} \otimes e_1^T \\ L_r \end{bmatrix} \tag{48}$$

## 4.2 Implementing the Projection Method

In order to enable the implementation of the Projection method, some of the matrices needs to be manipulated to match the dimensions (since we have four different components of $v$)and the different domain. Most important is the $\mathbf{C}$-matrix, since this decides how the wave should be have in the different domains. This diagonal $4m \times 4m$-matrix is constructed as below, where $\mathbf{C}_{l,r}$ contains the components $\varepsilon$ and $\mu$ for each side.

$$\left[\begin{array}{c|c} \mathbf{C}_l & 0 \\ \hline 0 & \mathbf{C}_r \end{array}\right]$$

Constructing the projection operator $P$ as $P = I_{4m} - \tilde{H}^{-1}L(L^T\tilde{H}^{-1}L)^{-1}L^T$, where the boundary operator is defined as in equation (48). The problem can thus be written as $v_t = A_v v$, where the following definitions are used:

$$\begin{cases} A_v = P\tilde{D}P \\ \tilde{D} = C^{-1}[I_2 \otimes (A \otimes D_1)] \end{cases} \tag{49}$$

## 4.3 Results

The calculations were run using Dirichlet conditions on the domain $x \in [-1, 1]$, with the interface situated at $x_I = 0$. The initial condition was manipulated to start at $x_0 = -0.5$ in order to not collide with the boundary immediately. The relative permeabilites were set to be $\varepsilon_l = 2$ and $\varepsilon_r = 1$. The end time was set to $T = 1.1$ in order for the reflection/transmission to be fully done, but without interfering with the reflected wave from the external boundary.

The amount of grid points used were $m = [61,\ 101,\ 161,\ 201,\ 501,\ 601]$ and where split evenly over the two domains. In Figure 15, the result at $T = 1.1$ for $m = 201$ is presented. Here, it can be seen that a large part of both the electric and magnetic waves were transmitted through the interface, which was expected since the right domain had a lower value of $\varepsilon$. Both components also had a small fraction of the wave reflected, seen as the small bumps on the left side of the vertical line.

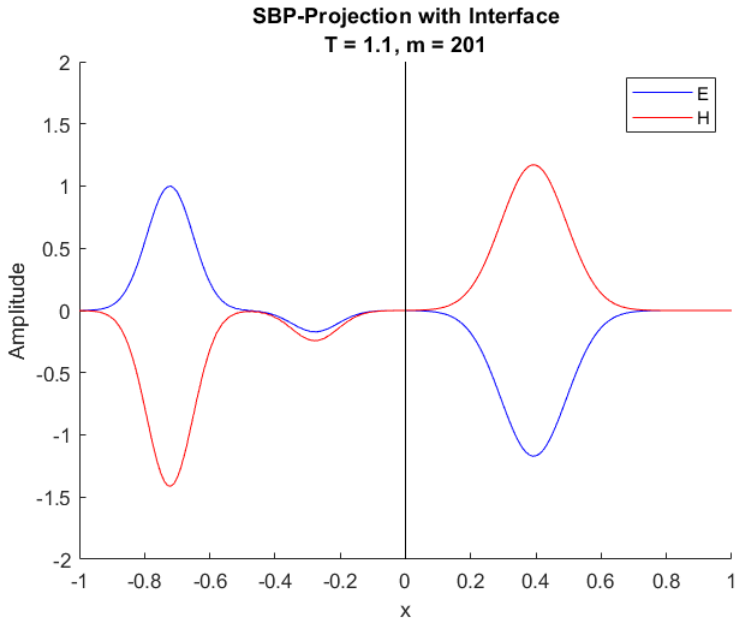


**Figure 15:** *Plot that show the reflected and transmitted waves at* $T = 1.1$ *for* $m = 201$*. The vertical lines denotes where the internal boundary is situated.*

In order to calculate the error, the theoretical height of the transmitted electrical wave, $E_T$, was needed to be calculated according to equation (50). Here, $\eta^{(l,r)} = \sqrt{\varepsilon_{l,r}}$ is the refractive index and $E_I = 1$ is the incident height of the wave right before it hits the interface.

$$\begin{cases} T = \frac{\eta^{(l)}}{\eta^{(l)}+\eta^{(r)}} \\ E_T = T \cdot E_I \end{cases} \tag{50}$$

The numerical height of the transmitted electrical wave, $E_{T,num}$ was obtained by finding the minimal value in the right domain. The fraction that was transmitted could then be calculated as $T_{num} = E_{T,num}/E_I$, where $E_I = 1$. The error could thus be calculated as:

$$e_T = |T - T_{num}| \tag{51}$$

The errors and convergence rates $q$ are presented in Table 5. Here it is shown that the error is steadily decreasing, however, the values of $q$ looks a bit funny since a maximal value of $q = 7.3$ is reached when $4^{\text{th}}$-order SBP-operators are being used. However, when fitting a line to the errors, shown in Figure 16, it can be seen that the average convergence rate is around $P = 3.3$. This is a reasonable value for the chosen method. The high fluctuations of $q$ is most likely due to the approximate way that the error is being calculated. When using different values of $m$, the convergence varied greatly, which means that this error calculation is highly sensitive of different disturbances.

**Table 5:** *The measured error and convergences for the implemented internal boundary, with* $\varepsilon_l = 2$ *and* $\varepsilon_r = 1$. *Dirichlet BC used at the external boundaries.*

| **m** | $\mathbf{e}_t$ | **q** |
|---|---|---|
| 61 | 3.300E(-2) | - |
| 101 | 7.807E(-3) | 2.8220 |
| 161 | 1.545E(-3) | 3.4474 |
| 201 | 4.520E(-4) | 5.5070 |
| 501 | 5.203E(-5) | 2.2593 |
| 601 | 1.382E(-5) | 7.2734 |

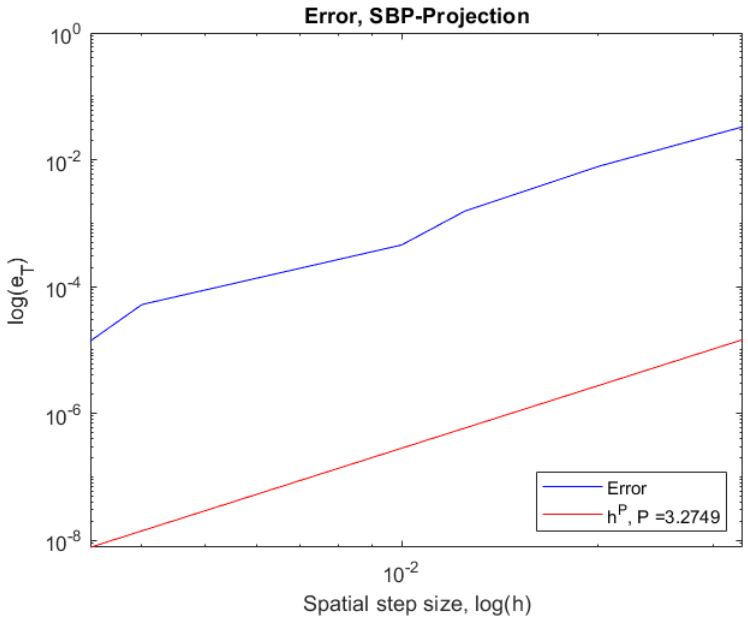


**Figure 16:** *The error plotted logarithmically against the spatial step size* $h$. *Plotted together with* $h^P$ *to compare convergence rate.*

## 4.4 Discussion

When implementing an interface due to the discontinuity in the relative permeabilities on the domain, the convergences of the Projection method reached a value of around $P = 3.3$ when using a approximate error estimation of the transmitted wave. This is a rather satisfying result, albeit a bit low since a $4^{\text{th}}$ SBP was being used. This reduced convergence is very likely to be due to the amount of grid points that were chosen for evaluation, since the convergence varied greatly with chosen values.

# Appendix A - Code Assignment 1

```
1
2  %%% SBP-SAT and SBP-Projection solvers for simplification
3  %%% of Maxwell's equations in 1D, imposing two different
4  %%% BC:s: characterisitic and Dirichlet.
5  %%% Calls the function RK4.m
6
7
8  clear all; close all;
9  % From project description
10 eps = 1;
11 mu = 1;
12 r = 0.1;
13 g_l = 0;
14 g_r = 0;
15 A = [0 1; 1 0];
16 C = [eps 0; 0 mu];
17 T = 1.8;
18
19 %Domain properties
20 %m = 201;
21 m = [61 101 161 201 401 601];
22 x_l = -1;    x_r = 1;
23 len = x_r-x_l;
24 h = (x_r-x_l)./(m-1);
25
26 n = input('1 for Characteristic \n2 for Dirichlet:          ');
27 disp(" ");
28 nn = input('1 for SAT \n2 for Projection:          ');
29 disp(" ");
30
31 for i=1:length(m)
32
33     x = x_l:h(i):x_r;
34
35     [theta_1, theta_end2] = theta(x,0,r);
36     u_initial = [(theta_end2-theta_1) (theta_1+theta_end2)];
37
38     [H, HI, D1, e_1, e_m] = SPB4_BV3(m(i),h(i));
39
40     % chooses variable
41     e_first = [1 0];    %v^(1)
42     e_second = [0 1];   %v^(2)
43
44
45     switch n
46         % Characteristic BC
47         case 1
48             BC = "Characteristic";
49             tau_l = [-1/2; 1/2];
50             tau_r = [-1/2; -1/2];
51             L_l = kron(e_first, e_1') - kron(e_second, e_1')-g_l;
52             L_r = kron(e_first, e_m') + kron(e_second, e_m')-g_r;
53
54         % Dirichlet BC
55         case 2
56             BC = "Dirichlet";
57             tau_l = [0; 1];
58             tau_r = [-1; 0];
59             L_l = kron(e_first, e_1');
60             L_r = kron(e_second, e_m');
61     end
```

```
62
63      switch nn
64          % Set SAT-terms
65          case 1
66              method = "SAT";
67              SAT_l = kron(tau_l, HI*e_1*L_l);
68              SAT_r = kron(tau_r, HI*e_m*L_r);
69              SBPx = kron(A, D1);
70              B = SBPx + SAT_l + SAT_r;
71
72              % Projection bby
73          case 2
74              method = "Projection";
75              L_P = [L_l; L_r]';
76              HI_P = kron(eye(2), HI);
77              temp = inv(L_P'*HI_P*L_P);
78              P = eye(2*m(i))-HI_P*L_P*temp*L_P';
79              B = P*kron(A, D1)*P;
80      end
81
82
83      % Time stepping
84      eigB = eig(B);
85      CFL = 2.8/max(abs(eigB));
86      k = (CFL/5)*h(i);
87
88      %k = 0.023*h(i);
89      v0 = u_initial';
90      v = RK4(B,v0,x,T,k);
91
92      % Calculating analytical solution
93      [theta_end1, theta_end2]= theta(x,len-T,r);
94      u_exact = [-(theta_end1+theta_end2) -(theta_end1-theta_end2)]';
95
96
97      % Calculating error
98      err = u_exact-v;
99      L2E(i)= sqrt(err'*kron(eye(2),H)*err);
100
101     % Plotting
102     figure(i)
103     subplot(2,1,1)
104     plot(x,u_exact(1:m(i)), x, u_exact(m(i)+1:2*m(i)));
105     title(["Analytical, T = " + T, BC + " BC"]);
106     xlabel("Domain");
107     ylabel("Amplitude");
108
109     subplot(2,1,2)
110     plot(x,v(1:m(i)), x, v(m(i)+1:2*m(i)));
111     title( ["SBP-" + method + ", T = " + T, BC + " BC", "m = " m(i)]);
112     xlabel("Domain, x");
113     ylabel("Amplitude, [H, E]");
114
115     figure(i+length(m))
116     plot(eigB, '*');
117     title(["Eigenvalues " + method, BC + "-BC, m = " + m(i)]);
118     xlabel("Re(\lambda)");
119     ylabel("Im(\lambda)");
120 end
121
122
123 %Convergence stuff
124 q = zeros(1,length(m));
125 for j=1:length(m)-1
126     q(j+1) = (log(L2E(j+1))-log(L2E(j))) / (log(h(j+1))-log(h(j)));
```

```
end

slope = polyfit(log(h), log(L2E),1);

figure(length(m)+i+1)
loglog(h, L2E, 'b', h, h.^(slope(1)),'r');
title(["l^2-Error, SBP-" + method, BC + " BC"]);
xlabel("Spatial step size, log(h)");
ylabel("log(||u-v||_{l^2})");
legend("Error", "h^P, P =" + slope(1),'Location', 'southeast');
```

```

%%% Runge-Kutta 4 for timestepping the solvers
%%% to Maxwell's equations.

function v = RK4(A, v0, x, T, k)
m = length(x);
temp = v0;

%figure(1);
%plot(x,temp(1:m),x,temp(m+1:2*m));

k = T/ceil(T/k);    %Step all the way to T
A = k.*(sparse(A));
t=0;
while t<T
    w1 = A*temp;
    w2 = A*(temp+0.5*w1);
    w3 = A*(temp+0.5*w2);
    w4 = A*(temp+w3);
    w = (w1+2*w2+2*w3+w4)/6;
    temp = temp+w;
%     plot(x,temp(1:m),x,temp(m+1:2*m));
%     drawnow;
    t=t+k;
end
v = temp;
end
```

# Appendix B - Code Interface

```
%%% SBP-Projection for Part 2
%%% Having some interface

% Forming vector as
% u^(1,l)_1... u^(1,l)_m u^(2,l)_1... u^(2,l)_m u^(1,r)_1... u^(1,r)_m etc

clear all; close all;

% From project description
eps_l = 2;
eps_r = 1;
mu = 1;
r = 0.1;
g_l = 0;
g_r = 0;
A = [0 1; 1 0];
C_l = [eps_l 0; 0 mu];
C_r = [eps_r 0; 0 mu];
T_end = 1.1;

% Setting the domain
x_l = -1;  x_I = 0;  x_r = 1;
x_0 = -0.5;
len = x_I-x_l;
gridpnts = ([61 101 161 201 501 601]-1)./2+1;
h = len./(gridpnts-1);

% Chooses component from vector
e_1_left = [1 0 0 0];       %u^(1) left domain
e_2_left = [0 1 0 0];       %u^(2) left domain
e_1_right = [0 0 1 0];      %u^(1) right domain
e_2_right = [0 0 0 1];      %u^(2) right domain

n = input('1 for Characteristic \n2 for Dirichlet:          ');
disp(" ");

for i=1:length(gridpnts)
    m = gridpnts(i);
    xl = x_l:h(i):x_I;
    xr = x_I:h(i):x_r;
    x = [xl xr];

    [H, HI, D1, e_1, e_m] = SPB4_BV3(m,h(i));
    [theta_1, theta_end2] = theta(xl,x_0,0,r);
    u_initial = [theta_end2-theta_1 theta_1+theta_end2 zeros(1,2*m)];

    % Constructs boundary-operators dep. on BC.
    switch n
        case 1
            BC = "Characteristic";
            L_left = kron(e_1_left, e_1') - kron(e_2_left, e_1');
            L_int_1 = kron(e_1_left, e_m') - kron(e_1_right, e_1');
            L_int_2 = kron(e_2_right, e_1')-kron(e_2_left, e_m');
            L_right = kron(e_2_right, e_m') + kron(e_2_right, e_m');

        case 2
            BC = "Dirichlet";
            L_left = kron(e_1_left, e_1');
            L_int_1 = kron(e_1_left, e_m') - kron(e_1_right, e_1');
            L_int_2 = kron(e_2_left, e_m') - kron(e_2_right, e_1');
            L_right = kron(e_2_right, e_m');
```

```
    end

    L = [L_left; L_int_1; L_int_2; L_right]';

    % Constructs C-matrix
    C = [kron(C_l, eye(m)) zeros(size(kron(C_l, eye(m))));
        zeros(size(kron(C_r, eye(m)))) kron(C_r, eye(m))];

    % Projection method
    HI_P = kron(eye(4), HI);
    temp = (L'*HI_P*L)\eye(4);
    P = eye(4*m)-HI_P*L*temp*L';
    D_bar = C\(kron(eye(2), kron(A, D1)));
    Av = P*D_bar*P;

    % Time stepping
    eigAv = eig(Av);
    CFL = 2.8/max(abs(eigAv));
    k = (CFL/5)*h(i);
    v0 = u_initial';
    v = RK4(Av, v0, x, T_end, k, m);

    % Analytical values
    eta_l = sqrt(eps_l);
    eta_r = sqrt(eps_r);
    T = 2*eta_l/(eta_l+eta_r);       %transmission
    R = (eta_l-eta_r)/(eta_l+eta_r); %reflection
    E_I = 1;                         %height incident wave
    E_T = T*E_I;                     %       transmitted
    E_R = R*E_I;                     %       reflected

    % Numerical values
    E_R_num = abs(min(v(1:m)));
    E_T_num = abs(min(v(2*m+1:3*m)));
    T_num = E_T_num/E_I;
    R_num = E_R_num/E_I;

    err_T(i) = abs(T - T_num);
    err_R(i) = abs(R - R_num);
    err(i) = mean(err_T+err_R);

    % Plotting stuff
    figure(i)
    %plot(xl, v(1:m), 'b', xr, v(2*m+1:3*m), 'b');
    hold on;
    plot(xl, v(1:m), 'b', xl, v(m+1:2*m), 'r', ...
        xr, v(2*m+1:3*m), 'b', xr, v(3*m+1:end), 'r');
    plot([x_I x_I], [-2 2],'k');
    title(["SBP-Projection with Interface", "T = " + T_end + ", m = "+ (2*m-1)]);
    legend("E", "H");
    xlabel("x");
    ylabel("Amplitude");
end

%Convergence stuff
q = zeros(1,length(m));
for j=1:length(gridpnts)-1
    q(j+1) = (log(err_T(j+1))-log(err_T(j))) / (log(h(j+1))-log(h(j)));
end

slope = polyfit(log(h), log(err_T),1);

figure(length(gridpnts)+i+1)
loglog(h, err_T, '-b', h, h.^(slope(1)),'r');
title(["Error, SBP-Projection"]);
```

```
xlabel("Spatial step size, log(h)");
ylabel("log(e_T)");
legend("Error", "h^P, P =" + slope(1),'Location', 'southeast');
```

```
%%% Runge-Kutta 4 for timestepping the solvers
%%% to Maxwell's equations with an Interface.
%%% Uncomment stuff in order to plot solution

function v = RK4(A, v0, x, T, k, m)
x_l = x(1:m);
x_r = x(m+1:end);
temp = v0;

%figure(1);
%plot(x(1:m),temp(1:m),x(m:2*m-1),temp(2*m+1:3*m));

k = T/ceil(T/k);    %Step all the way to T
A = k.*(sparse(A));
t=0;
i = 1;
while t<T
    w1 = A*temp;
    w2 = A*(temp+0.5*w1);
    w3 = A*(temp+0.5*w2);
    w4 = A*(temp+w3);
    w = (w1+2*w2+2*w3+w4)/6;
    temp = (temp+w);
    %plot(x(1:m),temp(1:m), x(m:2*m-1),temp(2*m+1:3*m));
    %plot([temp(1:m);
    %      temp(2*m+1:3*m)]);

    %if (mod(i,15) == 0)
    %    plot(x_l, temp(1:m), 'b', x_l, temp(m+1:2*m), 'r', ...
    %         x_r, temp(2*m+1:3*m), 'b', x_r, temp(3*m+1:end), 'r')
    %    ylim([-2 2])
    %    drawnow;
    %end
    i = i+1;
    t=t+k;
end
v = temp;
end
```