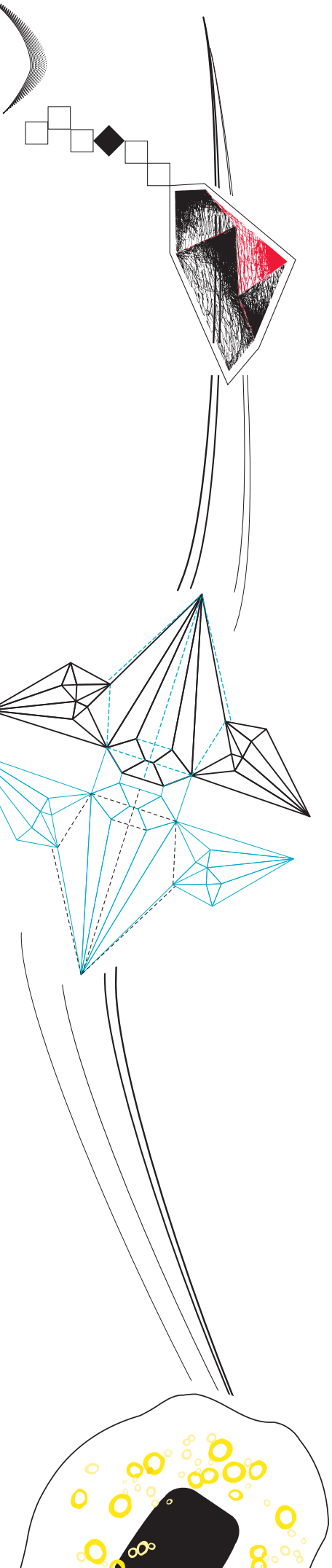

KRYLOV SUBSPACE TIME DOMAIN COMPUTATIONS OF MONOCHROMATIC SOURCES FOR MULTI-FREQUENCY OPTICAL RESPONSE

A.M. HANSE

MSc. Thesis
April 2017

Assesment Committee

Prof. dr. J.J.W. van der Vegt
Dr. R. Uppu
Dr. C. Brune
Dr. M.A. Botchev

Supervisor

Dr. M.A. Botchev

Chair Mathematics of Computational Science
Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

Abstract

The study of correlations of speckle patterns of scattering media from incident beams with different angle, frequency and spatial location provides insight in the behavior of light transport in scattering media. This insight can be exploited for numerous applications such as real-time imaging of biological tissue. Multi-frequency studies of speckle correlations can provide information about the nature and strength of disorder in a material. Currently the main method of calculating speckle patterns for multiple frequencies is broad band pulse excitation using the finite-difference-time-domain (FDTD) method. However, to achieve a reasonable accuracy multiple computations with narrower pulses of partially overlapping frequency spectra have to be done. In these computations the spatial part of the source is constant and only the temporal part of the source is different for every run. In Krylov subspace exponential time integration methods the system is projected on a smaller system of ODEs, this projection is made independent of the temporal part of the source.
In this research we present three different Krylov subspace methods to efficiently solve Maxwell's equations in a scattering medium for monochromatic sources to calculate a multi-frequency response. Two of these methods are shown to be efficient, they perform better than the implicit trapezoidal rule (ITR) scheme. The first of these methods simulates Maxwell's equations in one big time step and exploits the fact that the projection to a system of smaller ODEs is made independent of the temporal part of the source. The memory requirements of this method are very high but it the fastest method for finding multi-frequency solutions. In the second method, the solution is rewritten as a time periodic asymptotic solution plus a correction term. The latter is a solution to a homogeneous problem which can be solved by a Krylov subspace method efficiently.

Keywords: Krylov subspace exponential time integration, SAI, Scattering media, Speckle correlations

Contents

Chapter 1

Introduction

When light propagates through a material like paint or human tissue, it scatters due to nano scale inhomogeneities in the refractive index [13]. Media with an irregular spatial structure in the refractive index are therefore called scattering media. When sending a monochromatic beam of light through a scattering medium a wildly fluctuating speckle pattern occurs as a result of the multiple scattering the light undergoes while propagating through the medium. The speckle patterns vary with the angle, phase and frequency of the incident beam [20]. While the speckle pattern seems apparently random, it possesses a rich variety of characteristic correlations in the angle, frequency and spatial location of the light propagating in the disordered photonic media [20]. These correlation functions provide insight into the fundamental aspects of light transport that can be exploited for applications such as real-time imaging and spectrometry ([9], [17]). There is also an interdisciplinary interest in the study of the correlation of random wave fields that result from wave propagation through complex media, examples are acoustics, seismology and marine sciences [10].
One of the applications of the study of single frequency speckle patterns is for example to focus the light on the other side of a scattering medium. This controlled wavefront is constructed as a properly phased multiple of incident beams [13], as first demonstrated by Vellekoop and Mosk [21]. This has since been extended to shaping a wavefront through, for example, the living tissue in the ear of a mouse [11]. This field of study has a lot of other interesting applications such as body imaging [12]. While most of the wavefront shaping methods to-date have demonstrated the control of monochromatic light, multifrequency control of light is gaining momentum [14] which would enable applications such as multicolor biomedical imaging and light harvesting in solar cells.
If one is interested in the correlation of speckle patterns for different frequencies it is important to have sophisticated computing techniques to calculate these speckle patterns. Currently the main method of calculating speckle patterns for polychromatic sources is broadband pulse excitation in finite-difference-time-domain (FDTD) methods, which can for instance be done using programs like MEEP [16]. In broadband pulse excitation the source is a pulse containing a spectrum of different frequencies. The optical response of the computational structure, which is the electromagnetic field, is Fourier transformed to calculate the speckle pattern at different frequencies. A disadvantage of broadband pulse excitation is that it results in a trade-off between accuracy and speed. A wide frequency spectrum of the source lowers the accuracy, to compensate for this loss in accuracy multiple computations with narrower pulses of partially overlapping frequency spectra have to be done.
For all runs with sources of different frequency spectra the spatial part of the source is the same. If we solve such a system using Krylov subspace methods the system is projected on a smaller ODE system. This projection is made independent of the temporal part of the source. This suggests that we might be able to use Krylov methods for exponential time integration such that we can solve the same problem efficiently for multiple sources. In this research we investigate this and other features of the Krylov subspace exponential methods to solve this scattering problem.

1.1 Krylov subspace methods

Since it is necessary to perform multiple runs with the same spatial source to get a high accuracy it can be advantageous to look at Krylov subspace methods. We have to solve the following system for multiple inputs $\alpha(t)$.

$$\begin{aligned} \boldsymbol{y}' &= -\boldsymbol{A}\boldsymbol{y} + \alpha(t)\boldsymbol{g}_s, \\ \boldsymbol{y}(0) &= \boldsymbol{0}, \end{aligned}$$

where $\boldsymbol{A}$ is a sparse discretization matrix for Maxwell's equations that models a scattering medium, $\boldsymbol{y}$ is a vector containing the electric and magnetic fields and $\boldsymbol{g}_s$ is the spatial part of the source. This system has to be solved for different functions $\alpha(t)$, which can be pulses as in the previous section or just single frequencies. Using Krylov exponential time integration such a problem would be solved by searching for a solution in a Krylov subspace built up by $\boldsymbol{A}$ and $\boldsymbol{g}_s$. After an orthonormal basis for this space is built which often requires the biggest share of the work the solution can be approximated by solving a smaller ODE. The important thing here is that this base can be built up independent of the function $\alpha(t)$. If we want to find a solution of this problem for multiple functions $\alpha(t)$ we only have to solve this smaller ODE multiple times.
As suggested in [2] and [19] for shift-and-invert (SAI) Krylov subspace methods mesh-independent convergence can be expected. This means that as we refine our mesh we can expect the same amount of Krylov subspace iterations necessary for convergence. Apart from this Krylov exponential time integration methods have been shown to perform well in the simulation of photonic crystals [2].
In this research we propose several methods of performing time-integration using the Krylov subspace for multiple input functions. We compare the performance of these methods with the performance of the implicit trapezoidal rule (ITR) scheme.

Chapter 2

Maxwell's equations and the Yee grid

In this chapter we discuss the Maxwell's equations in 2D and their spatial discretization as explained in [18]. Furthermore we discuss the implementation of PML boundaries. PML boundaries are absorbing boundaries we use to simulate an infinite domain. Furthermore we discuss how we implement the discontinuities in the dielectric constant using homogenization.
Light propagation is governed by Maxwell's equations:

$$\nabla \cdot \boldsymbol{B}(\boldsymbol{r},t) = 0, \qquad \frac{\partial \boldsymbol{B}(\boldsymbol{r},t)}{\partial t} = -\nabla \times \boldsymbol{E}(\boldsymbol{r},t) - \boldsymbol{M}'(\boldsymbol{r},t),$$
$$\nabla \cdot \boldsymbol{D}(\boldsymbol{r},t) = \rho, \qquad \frac{\partial \boldsymbol{D}(\boldsymbol{r},t)}{\partial t} = -\nabla \times \boldsymbol{H}(\boldsymbol{r},t) - \boldsymbol{J}'(\boldsymbol{r},t).$$

where $\boldsymbol{r}$ denotes a position, t is the time, $\boldsymbol{E}$ is the Electrical field, $\boldsymbol{H}$ is the magnetic field and $\boldsymbol{B}$ and $\boldsymbol{D}$ are the magnetic induction field and the electric displacement current. Furthermore $\boldsymbol{J}'$ is the electric current density and $\boldsymbol{M}'$ is the nonphysical magnetic current density. We adopt the following constitutive relationships between the field and material parameters:

$$\boldsymbol{D}(\boldsymbol{r},t) = \epsilon_0 \epsilon_r(\boldsymbol{r}) \boldsymbol{E}(\boldsymbol{r},t),$$
$$\boldsymbol{B}(\boldsymbol{r},t) = \mu_0 \mu_r(\boldsymbol{r}) \boldsymbol{H}(\boldsymbol{r},t),$$

where $\epsilon = \epsilon_r \epsilon_0$ is the electric permittivity, and $\mu = \mu_r \mu_0$ is the magnetic permeability. ϵ_0 and μ_0 are the electric permittivity and magnetic permeability of free space.
Furthermore we assume that our sources can be written as:

$$\boldsymbol{J}'(\boldsymbol{r},t) = \sigma(\boldsymbol{r}) \boldsymbol{E}(\boldsymbol{r},t) + \boldsymbol{J}(\boldsymbol{r},t),$$
$$\boldsymbol{M}'(\boldsymbol{r},t) = \sigma_m(\boldsymbol{r}) \boldsymbol{H}(\boldsymbol{r},t) + \boldsymbol{M}(\boldsymbol{r},t),$$

where σ and σ_m are parameters for electric and magnetic losses, $\boldsymbol{J}$ and $\boldsymbol{M}$ are independent source terms.
We are only interested in the electromagnetic field in two dimensions. Therefore we assume that our electromagnetic fields are homogeneous along the z-axis. Then we can reduce our equations into two sets of independent equations, either transverse magnetic (TM^z) mode or transverse electric (TE^z) mode. In the first one we only have the H_x, H_y and E_z field, and in the second one only the E_x, E_y and H_z field. We will use the TM^z mode in our simulation, which can be written as:

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu_0 \mu_r} \left(-\frac{\partial E_z}{\partial y} - M_x - \sigma_m H_x \right),$$
$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu_0 \mu_r} \left(\frac{\partial E_z}{\partial x} - M_y - \sigma_m H_y \right),$$
$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon_0 \epsilon_r} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - J_z - \sigma E_z \right).$$

In the absence of source terms and without absorption (i.e. $\sigma_m = 0, \sigma = 0$), we can write the field a bit different, let $\boldsymbol{H} = (H_y, H_x)$ and $E = E_z$. Now:

$$\frac{\partial \boldsymbol{H}}{\partial t} = \frac{1}{\mu_0 \mu_r} \nabla E,$$
$$\frac{\partial E}{\partial t} = \frac{1}{\epsilon_0 \epsilon_r} \nabla \cdot \boldsymbol{H},$$

which is a standard formulation of a wave equation. One can simplify this a bit more to arrive at:

$$\frac{\partial^2 E}{\partial t^2} = \frac{1}{\epsilon_0 \mu_0 \epsilon_r \mu_r} \Delta E = c^2 \Delta E,$$

where c is the wave speed. In free space (i.e. $\mu_r = \epsilon_r = 1$) we see that the wave speed is $c_0 = \frac{1}{\sqrt{\epsilon_0 \mu_0}}$.

2.1 Discretization of Maxwell's equations

Before we discretize our set of equations we make them dimensionless. Currently there are a lot of very small numbers involved in the equations and there is a big difference in the magnitude of the electrical and magnetic field. This can cause unnecessary numerical errors, hence we want the fields to be of about the same strength. We transform the equations to a dimensionless set of equations by using the scale invariance of the equations ([7], [16]), which basically amounts to setting $c_0 = \mu_0 = \epsilon_0 = 1$.
In the following part we write our old variables with subscript s, hence we denote our new dimensionless electrical field with $\boldsymbol{E}_z$ and the old one with $\boldsymbol{E}_{zs}$. We introduce the following variables: let L be our typical length in meters, one unit length in the dimensionless equations is L in physical parameters. Let us choose a typical magnetic field strength H_0 in $\frac{A}{m}$. The other parameters in the dimensionless equations are scaled along our choices of these two parameters. furthermore we introduce the constants $Z_0 = \sqrt{\frac{\mu_0}{\epsilon_0}}$ which is the impedance of vacuum, and $c_0 = \frac{1}{\sqrt{\mu_0 \epsilon_0}}$ the speed of light in vacuum.
The variables are transformed in the following way:

$$x = \frac{1}{L} x_s, \quad y = \frac{1}{L} y_s, \quad t = \frac{c_0}{L} t_s,$$

$$\sigma(x,y) = Z_0 L \sigma_s(x_s, y_s), \quad \sigma_m(x,y) = Z_0 L \sigma_{ms}(x_s, y_s), \quad \mu_r(x,y) = \mu_{rs}(x_s, y_s), \quad \epsilon_r(x,y) = \epsilon_{rs}(x_s, y_s),$$

$$E_z(x,y,t) = \frac{1}{H_0 Z_0} E_{zs}(x_s, y_s, t_s), \qquad H_x(x,y,t) = \frac{1}{H_0} H_{xs}(x_s, y_s, t_s),$$
$$H_y(x,y,t) = \frac{1}{H_0} H_{ys}(x_s, y_s, t_s), \qquad J_z(x,y,t) = \frac{L}{H_0} J_{zs}(x_s, y_s, t_s),$$
$$M_x(x,y,t) = \frac{L}{H_0 Z_0} M_{xs}(x_s, y_s, t_s), \qquad M_y(x,y,t) = \frac{L}{H_0 Z_0} H_{ys}(x_s, y_s, t_s).$$

This transformation gives the new set of equations:

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu_r} \left(-\frac{\partial E_z}{\partial y} - M_x - \sigma_m H_x \right), \tag{2.1}$$
$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu_r} \left(\frac{\partial E_z}{\partial x} - M_y - \sigma_m H_y \right), \tag{2.2}$$
$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon_r} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - J_z - \sigma E_z \right). \tag{2.3}$$

Now the electric and magnetic field are scaled such that their magnitude is of the same order which increases numerical stability. The equations are discretized on a Yee grid [23]. This is a staggered grid that is designed for Maxwell's equations. Let our domain be $[x_a, x_b] \times [y_a, y_b]$, for simplicity suppose that $x_a, x_b, y_a, y_b \in \mathrm{Z}$. We choose $\Delta x = \Delta y = \frac{1}{\text{resolution}}$, let $n_x = (x_b - x_a) \cdot \text{resolution} - 1$ and $n_y = (y_b - y_a) \cdot \text{resolution} - 1$. We assume that we have Perfect Electrical Conductor (PEC) boundaries, hence $E_z = 0$ on the boundaries. With these boundaries all incoming fields are reflected. Our discrete fields are approximated on the following points:

$$H_{x,i,j} \approx H_x(x_a + i\Delta x, y_a + (j - \tfrac{1}{2})\Delta y, t),$$

for $i = 1, \ldots, nx$ and $j = 1, \ldots, ny + 1$,

$$H_{y,i,j} \approx H_x(x_a + (i - \tfrac{1}{2})\Delta x, y_a + j\Delta y, t),$$

for $i = 1, \ldots, n_x + 1$ and $j = 1, \ldots, n_y$,

$$E_{z,i,j} \approx E_z(x_a + i\Delta x, y_a + j\Delta y, t),$$

for $i = 1, \ldots, n_x$ and $j = 1, \ldots, n_y$.
With this configuration the E_z field point lie exactly between two H_x field points in the y direction and between two H_y field points in the x direction. Also the H_x field points lie exactly between two E_z field points in the y direction and the H_y field points lie exactly between two E_z field points in the x direction. This means that we can calculate all the required derivatives with second order accuracy.
The original Yee-algorithm also uses a staggered grid in time. The time derivatives of H_x and H_y depend only on the E_z derivatives and vice versa. Hence they are updated independently of each other. Krylov methods require us to write our equations as a matrix vector product hence we approximate all field variables on the same points in time.
Now we transform this discretization into a matrix vector equation.
Let us first define our field vectors:

$$\begin{aligned}
\boldsymbol{h}_{x,i+(j-1)n_x} &= H_{x,i,j}, && \text{for } i = 1, \ldots, n_x \text{ and } j = 1, \ldots, n_y + 1,\\
\boldsymbol{h}_{y,i+(j-1)(n_x+1)} &= H_{y,i,j} && \text{for } i = 1, \ldots, n_x + 1 \text{ and } j = 1, \ldots, n_y,\\
\boldsymbol{e}_{z,i+(j-1)n_x} &= E_{z,i,j} && \text{for } i = 1, \ldots, n_x \text{ and } j = 1, \ldots, n_y,
\end{aligned}$$

where $\boldsymbol{h}_x \in \mathrm{R}^{n_x(n_y+1)}$, $\boldsymbol{h}_y \in \mathrm{R}^{(n_x+1)n_y}$ and $\boldsymbol{e}_z \in \mathrm{R}^{n_x n_y}$. And let us define the vector $\boldsymbol{y}$ as:

$$\boldsymbol{y} := \begin{bmatrix} \boldsymbol{h}_x \\ \boldsymbol{h}_y \\ \boldsymbol{e}_z \end{bmatrix}.$$

Furthermore let $\boldsymbol{m}_x, \boldsymbol{m}_y$ and $\boldsymbol{j}_z$ be the vectors with the values of M_x, M_y and J_z on the corresponding field points.
In the following part we use the superscript x, y or z to specify that a vector or matrix is defined on respectively the H_x, H_y or E_z field points. We will now define matrices that calculate our discrete derivatives, let us start with equation (2.1). In the discrete version this equation can be written as:

$$\begin{aligned}
\frac{\partial \boldsymbol{h}_{x,i+(j-1)n_x}}{\partial t} &= \frac{1}{\boldsymbol{\mu}^x_{r,i+(j-1)n_x}} \left(-\frac{\boldsymbol{e}_{z,i+(j-1)n_x} - \boldsymbol{e}_{z,i+(j-2)n_x}}{\Delta y} - \boldsymbol{\sigma}^x_{m,i+(j-1)n_x} \boldsymbol{h}_{x,i+(j-1)n_x} - \boldsymbol{m}_{x,i+(j-1)n_x} \right),\\
&\text{for } j = 2, \ldots, n_y,\\
\frac{\partial \boldsymbol{h}_{x,i}}{\partial t} &= \frac{1}{\boldsymbol{\mu}^x_{r,i}} \left(-\frac{\boldsymbol{e}_{z,i}}{\Delta y} - \boldsymbol{\sigma}^x_{m,i} \boldsymbol{h}_{x,i} - \boldsymbol{m}_{x,i} \right),\\
\frac{\partial \boldsymbol{h}_{x,i+n_y n_x}}{\partial t} &= \frac{1}{\boldsymbol{\mu}^x_{r,i+n_y n_x}} \left(-\frac{-\boldsymbol{e}_{z,i+(n_y-1)n_x}}{\Delta y} - \boldsymbol{\sigma}^x_{m,i+n_y n_x} \boldsymbol{h}_{x,i+n_y n_x} - \boldsymbol{m}_{x,i+n_y n_x} \right),
\end{aligned}$$

all equations hold for $i = 1, \ldots, n_x$.
Furthermore $\boldsymbol{\mu}^x_{r,i+(j-1)n_x}$ is the vector with values of μ_r on the points where H_x is defined, and $\boldsymbol{\sigma}^x_{m,i+(j-1)n_x}$ is defined in the same way for σ_m. The last two equations different because of the PEC boundary conditions.
Let us define the following matrix:

$$\boldsymbol{K}_n := \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & -1 \end{bmatrix},$$

where $\boldsymbol{K}_n \in \mathrm{R}^{n+1 \times n}$. Now define the Matrix $\boldsymbol{D}^x_y$ (the discrete derivative of the H_x-field with respect to y) using the Kronecker product as:

$$\boldsymbol{D}^x_y := \frac{1}{\Delta y} \boldsymbol{K}_{n_y} \otimes \boldsymbol{I}_{n_x},$$

where $\boldsymbol{I}_n$ is the $n \times n$ identity Matrix. Furthermore let M^x be the diagonal matrix with μ^x_r on its diagonal and Σ^x_m the diagonal matrix with $\boldsymbol{\sigma}^x_m$ on its diagonal. Now we can write the discretization of equation (2.1) in matrix vector form as:

$$\frac{\partial \boldsymbol{h}_x}{\partial t} = \boldsymbol{M}^{x,-1} \left(-\boldsymbol{D}^x_y \boldsymbol{e}_z - \boldsymbol{\Sigma}^x_m \boldsymbol{h}_x - \boldsymbol{m}_x \right).$$

Let us progress to equation (2.2) now:

$$\frac{\partial \boldsymbol{h}_{y,i+(j-1)(n_x+1)}}{\partial t} = \frac{1}{\boldsymbol{\mu}^y_{r,i+(j-1)(n_x+1)}} \left(\frac{\boldsymbol{e}_{z,i+(j-1)(n_x+1)} - \boldsymbol{e}_{z,i-1+(j-1)(n_x+1)}}{\Delta x} - \boldsymbol{\sigma}^y_{m,i+(j-1)(n_x+1)} \boldsymbol{h}_{y,i+(j-1)(n_x+1)} - \boldsymbol{m}_{y,i+(j-1)(n_x+1)} \right), \text{ for } i = 2, \ldots, n_x,$$

$$\frac{\partial \boldsymbol{h}_{y,1+(j-1)(n_x+1)}}{\partial t} = \frac{1}{\boldsymbol{\mu}^y_{r,1+(j-1)(n_x+1)}} \left(\frac{\boldsymbol{e}_{z,1+(j-1)(n_x+1)}}{\Delta y} - \boldsymbol{\sigma}^y_{m,1+(j-1)(n_x+1)} \boldsymbol{h}_{y,1+(j-1)(n_x+1)} - \boldsymbol{m}_{y,1+(j-1)(n_x+1)} \right),$$

$$\frac{\partial \boldsymbol{h}_{y,j(n_x+1)}}{\partial t} = \frac{1}{\boldsymbol{\mu}^y_{r,j(n_x+1)}} \left(\frac{-\boldsymbol{e}_{z,j(n_x+1)}}{\Delta y} - \boldsymbol{\sigma}^y_{m,j(n_x+1)} \boldsymbol{h}_{x,j(n_x+1)} - \boldsymbol{m}_{y,j(n_x+1)} \right).$$

All equations all hold for $j = 1, \ldots, n_y + 1$.
Again $\boldsymbol{\mu}^y_{r,i+(j-1)(n_x+1)}$ is the vector with values of μ_r but now on the H_y field points, the same holds for $\boldsymbol{\sigma}^y_m$ and σ_m.
As for the previous equation we write this as a matrix vector equation using the matrix $\boldsymbol{K}_n$ and a Kronecker product. Let us define $\boldsymbol{D}^y_x$ as:

$$\boldsymbol{D}^y_x := \frac{1}{\Delta x} \boldsymbol{I}_{n_y} \otimes \boldsymbol{K}_{n_x},$$

to arrive at:

$$\frac{\partial \boldsymbol{h}_y}{\partial t} = \boldsymbol{M}^{y,-1} \left(\boldsymbol{D}^x_y \boldsymbol{e}_z - \boldsymbol{\Sigma}^y_m \boldsymbol{h}_y - \boldsymbol{m}_y \right),$$

where again $\boldsymbol{\Sigma}^y_m$ and $\boldsymbol{M}^y$ are diagonal matrices with respectively $\boldsymbol{\sigma}^y_m$ and $\boldsymbol{\mu}^y$ on their diagonals.
Finally we discretize the equation for the electrical field (2.3).

$$\frac{\partial \boldsymbol{e}_{z,i+(j-1)n_x}}{\partial t} = \frac{1}{\boldsymbol{\epsilon}^z_{r,i+(j-1)n_x}} \left(\frac{\boldsymbol{h}_{y,i+1+(j-1)(n_x+1)} - \boldsymbol{h}_{y,i+(j-1)(n_x+1)}}{\Delta x} + \frac{\boldsymbol{h}_{y,i+jn_x} - \boldsymbol{h}_{y,i+(j-1)n_x}}{\Delta y} - \boldsymbol{\sigma}^z_{i+(j-1)n_x} - \boldsymbol{j}_{z,i+hn_x} \right)$$

Again we can define the derivatives using the Kronecker product, only this time we need a slightly different matrix:

$$\tilde{\boldsymbol{K}}_n := \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix},$$

where $\tilde{\boldsymbol{K}}_n \in \mathrm{R}^{n \times n+1}$. It is important to notice here that $\boldsymbol{K}_n = -\tilde{\boldsymbol{K}}_n^T$. Now we define two new matrices for the derivatives:

$$\boldsymbol{D}_x^z := \frac{1}{\Delta x} \boldsymbol{I}_{n_y} \otimes \tilde{\boldsymbol{K}}_{n_x},$$

and

$$\boldsymbol{D}_y^z := \frac{1}{\Delta y} \tilde{\boldsymbol{K}}_{n_y} \otimes \boldsymbol{I}_{n_x}.$$

We can see here that:

$$\boldsymbol{D}_x^{zT} = \frac{1}{\Delta x} \left(\boldsymbol{I}_{n_y} \otimes \tilde{\boldsymbol{K}}_{n_x} \right)^T = \frac{1}{\Delta x} \boldsymbol{I}_{n_y}^T \otimes \tilde{\boldsymbol{K}}_{n_x}^T = \frac{1}{\Delta x} \boldsymbol{I}_{n_y} \otimes -\boldsymbol{K}_{n_x} = -\boldsymbol{D}_x^y,$$

and similarly that:

$$\boldsymbol{D}_y^{zT} = -\boldsymbol{D}_y^x.$$

We will see later that an important property of our final Matrix follows from this. Now we write equation (2.3) in matrix vector form:

$$\frac{\partial \boldsymbol{e}_z}{\partial t} = \boldsymbol{E}^{z,-1} \left(\boldsymbol{D}_x^z \boldsymbol{h}_y - \boldsymbol{D}_y^z \boldsymbol{h}_x - \boldsymbol{\Sigma}^z \boldsymbol{e}_z - \boldsymbol{j}_z \right),$$

where E^z and Σ^z are diagonal matrices with respectively $\boldsymbol{\epsilon}_r^z$ and $\boldsymbol{\sigma}^z$ on their diagonal.
Now that we have all of our equations in Matrix vector format we can couple them as one big system:

$$\frac{\partial \boldsymbol{y}}{\partial t} = \begin{bmatrix} \boldsymbol{M}^x & & \\ & \boldsymbol{M}^y & \\ & & \boldsymbol{E}^z \end{bmatrix}^{-1} \left(\begin{bmatrix} -\boldsymbol{\Sigma}_m^x & & -\boldsymbol{D}_y^x \\ & -\boldsymbol{\Sigma}_m^y & \boldsymbol{D}_x^y \\ -\boldsymbol{D}_x^z & \boldsymbol{D}_y^z & -\boldsymbol{\Sigma}^z \end{bmatrix} \boldsymbol{y} - \begin{bmatrix} \boldsymbol{m}_x \\ \boldsymbol{m}_y \\ \boldsymbol{j}_z \end{bmatrix} \right). \tag{2.4}$$

If we assume that $\sigma_m = \sigma = 0$ we are left with the matrix:

$$\boldsymbol{A} = \boldsymbol{D}\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{M}^x & & \\ & \boldsymbol{M}^y & \\ & & \boldsymbol{E}^z \end{bmatrix}^{-1} \begin{bmatrix} & & -\boldsymbol{D}_y^x \\ & & \boldsymbol{D}_x^y \\ -\boldsymbol{D}_x^z & \boldsymbol{D}_y^z & \end{bmatrix}, \tag{2.5}$$

where $\boldsymbol{D}$ is a diagonal matrix with positive entries and $\tilde{\boldsymbol{A}}$ is skew-symmetric since $\boldsymbol{D}_x^{zT} = -\boldsymbol{D}_y^x$ and $\boldsymbol{D}_y^{zT} = -\boldsymbol{D}_x^y$. The matrix $\boldsymbol{D}^{-1}\tilde{\boldsymbol{A}}$ is similar to $\boldsymbol{D}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}}$ which is skew-symmetric since $\tilde{\boldsymbol{A}}$ is skew-symmetric and $\boldsymbol{D}^{-\frac{1}{2}}$ is diagonal and hence symmetric. Therefore all eigenvalues of $\boldsymbol{D}^{-1}\tilde{\boldsymbol{A}}$ are purely imaginary since it is similar to a skew-symmetric matrix.
If we introduce positive values of σ and σ_m we introduce negative eigenvalues to the system.
If we assume that all our sources can be divided in a constant spatial part and all depend on time in the same way we can write equation (2.4) as:

$$\boldsymbol{y}' = -\boldsymbol{A}\boldsymbol{y} + \alpha(t)\boldsymbol{g}_s.$$

Matrix $\boldsymbol{A}$ is an $3n_xn_y+n_x+n_y \times 3n_xn_y+n_x+n_y$ matrix. But it has only a few entries for every row, hence it is very sparse. In figure 2.1 you can see an example of a matrix like (2.5), this case has $n_x = n_y = 100$ and has a fill in of 0.0088%.

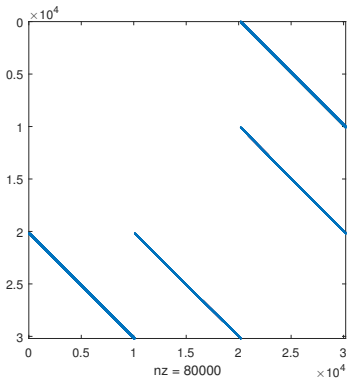

Figure 2.1: Example of a sparse matrix like (2.4) with $n_x = n_y = 100$

2.2 PML boundaries

In the discretization we used Perfect Electrically Conducting (PEC) boundaries, which perfectly reflect all incoming waves. This can be useful if one wants to simulate a closed cavity but we want to simulate an infinite domain, therefore we want to implement absorbing boundaries such that the fields disappear when reaching the boundaries. We can do this by setting σ and σ_m higher than zero in a layer at the boundary, but this causes reflection where the material parameters change and hence alter our field within the boundaries. That is why we use a more sophisticated approach that is called Perfectly Matched Layer (PML) boundaries. We use the approach as described by Johnson [8].
The idea of PML is based on the fact that we perform a variable transformation in both x and y coordinates as:

$$\tilde{x} = x + if(x), \quad \tilde{y} = y + ig(y),$$

where $f(x)$ and $g(y)$ are zero in the normal domain but grow in the boundary regions.
This idea is based on the fact that under certain conditions [8] we can write the solution $\boldsymbol{y}(\boldsymbol{r}, t)$ of a wave equation as the superposition of plane waves as:

$$\boldsymbol{y}(\boldsymbol{r}, t) = \sum_{\boldsymbol{k},\omega} \boldsymbol{Y}_{\boldsymbol{k},\omega} e^{i(\boldsymbol{k}\boldsymbol{r} - \omega t)}$$

which can be decomposed into solutions of the form:

$$\boldsymbol{Y}(y, z) e^{i(kx - \omega t)}.$$

If we introduce our coordinate transformation here this transforms into:

$$\boldsymbol{Y}(y, z) e^{i(kx - \omega t)} e^{-kf(x)}.$$

Hence our solution decays exponentially at the boundaries as $f(x)$ grows. Which means that the boundaries act as an absorbing material but the solutions in the domain where $f(x) = 0$ are not altered.
The derivative of our coordinate transformation is:

$$\frac{d\tilde{x}}{dx} = 1 + if'(x) = 1 + i\frac{\sigma_x(x)}{\omega}.$$

It turns out that it is easier to write the derivative of f as $\frac{\sigma_x}{\omega}$. Now in our new coordinates the derivative $\frac{\partial}{\partial \tilde{x}}$ become:

$$\frac{1}{1 + i\frac{\sigma_x(x)}{\omega}} \frac{\partial}{\partial x},$$

the same happens for y. Now let us rewrite our equations.
We use the dimensionless Maxwell equations in TM^z mode without source currents and damping.

$$\begin{aligned}
\frac{\partial H_x}{\partial t} &= -\frac{1}{\mu_r}\frac{\partial E_z}{\partial y},\\
\frac{\partial H_y}{\partial t} &= \frac{1}{\mu_r}\frac{\partial E_z}{\partial x},\\
\frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon_r}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right).
\end{aligned}$$

For the complex coordinate transformation the following happens to our derivatives:

$$\begin{aligned}
\frac{\partial}{\partial \tilde{x}} &\implies \left(\frac{1}{1+i\frac{\sigma_x}{\omega}}\right)\frac{\partial}{\partial x} = \frac{1}{s_x}\frac{\partial}{\partial x},\\
\frac{\partial}{\partial \tilde{y}} &\implies \left(\frac{1}{1+i\frac{\sigma_y}{\omega}}\right)\frac{\partial}{\partial y} = \frac{1}{s_y}\frac{\partial}{\partial y}.
\end{aligned}$$

We assume the solution depends on time as $e^{-i\omega t}$ hence multiplying with $-i\omega$ corresponds to differentiation with respect to t in the frequency domain. We will now transform the equations one by one, we start with the equation for H_x:

$$\begin{aligned}
-i\omega H_x &= -\frac{1}{\mu_r}\frac{1}{s_y}\frac{\partial E_z}{\partial y},\\
-s_y i\omega H_x &= -\frac{1}{\mu_r}\frac{\partial E_z}{\partial y},\\
-i\omega H_x + \sigma_y H_x &= -\frac{1}{\mu_r}\frac{\partial E_z}{\partial y},\\
\frac{\partial H_x}{\partial t} &= -\frac{1}{\mu_r}\frac{\partial E_z}{\partial y} - \sigma_y H_x.
\end{aligned}$$

For H_y this derivation is similar:

$$\begin{aligned}
-i\omega H_y &= \frac{1}{\mu_r}\frac{1}{s_x}\frac{\partial E_z}{\partial x},\\
-s_x i\omega H_y &= \frac{1}{\mu_r}\frac{\partial E_z}{\partial x},\\
-i\omega H_y + \sigma_x H_y &= \frac{1}{\mu_r}\frac{\partial E_z}{\partial x},\\
\frac{\partial H_y}{\partial t} &= \frac{1}{\mu_r}\frac{\partial E_z}{\partial x} - \sigma_x H_y.
\end{aligned}$$

These two derivations were quite simple, however for E_z it is a bit more complicated because here we are going to need some auxiliary differential equations:

$$\begin{aligned}
-i\omega E_z &= \frac{1}{\epsilon_r}\left(\frac{1}{s_x}\frac{\partial H_y}{\partial x} - \frac{1}{s_y}\frac{\partial H_x}{\partial y}\right),\\
-s_x s_y i\omega E_z &= \frac{1}{\epsilon_r}\left(s_y\frac{\partial H_y}{\partial x} - s_x\frac{\partial H_x}{\partial y}\right),\\
\left(-i\omega + \sigma_x + \sigma_y + i\frac{\sigma_x\sigma_y}{w}\right)E_z &= \frac{1}{\epsilon_r}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) + \frac{1}{\epsilon_r}\left(i\frac{\sigma_y}{\omega}\frac{\partial H_y}{\partial x} - i\frac{\sigma_x}{\omega}\frac{\partial H_x}{\partial y}\right),\\
\frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon_r}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) - (\sigma_x + \sigma_y)E_z + B_x + B_y + B_z,
\end{aligned}$$

with:

$$B_x = -\frac{1}{\epsilon_r} i \frac{\sigma_x}{\omega} \frac{\partial H_x}{\partial y},$$
$$B_y = \frac{1}{\epsilon_r} i \frac{\sigma_y}{\omega} \frac{\partial H_y}{\partial x},$$
$$B_z = -i \frac{\sigma_x \sigma_y}{\omega} E_z.$$

We can transform these to the time domain to arrive at:

$$\frac{\partial B_x}{\partial t} = -\frac{\sigma_x}{\epsilon_r} \frac{\partial H_x}{\partial y},$$
$$\frac{\partial B_y}{\partial t} = \frac{\sigma_y}{\epsilon_r} \frac{\partial H_y}{\partial x},$$
$$\frac{\partial B_z}{\partial t} = -\sigma_x \sigma_y E_z,$$

which are our auxiliary differential equations. So if we use the PML boundary layers we get three extra differential equations, however these are only defined on regions where $\sigma_x > 0, \sigma_y > 0$ or if both are positive. Hence they are only a small part of our total discretized system.

Now we rewrite our ODE (2.4) by adding the auxiliary equations for the PML layer. Let $\widehat{\boldsymbol{E}^{z,-1}\boldsymbol{\Sigma}_x^z}$ be the rows of the matrix $\boldsymbol{E}^{z,-1}\boldsymbol{\Sigma}_x^z$ corresponding to values where $\boldsymbol{\sigma}_x^z$ is nonzero. Let the same hold for $\boldsymbol{E}^{z,\widehat{-1}\boldsymbol{\Sigma}_y^z}\boldsymbol{D}_x^z$ for the rows where $\boldsymbol{\sigma}_y^z$ is nonzero and $\widehat{\boldsymbol{\Sigma}_x^z\boldsymbol{\sigma}_y^z}$ for the rows where both σ_x^z and σ_y^z are nonzero. Furthermore let $\hat{\boldsymbol{I}}_{n_x(n_y+1)}$ be the $n_x(n_y+1) \times n_x(n_y+1)$ identity matrix but with only the columns that correspond with nonzero entries of $\boldsymbol{\sigma}_x^z$. The same holds for $\hat{\boldsymbol{I}}_{(n_x+1)n_y}$ and $\hat{\boldsymbol{I}}_{n_x n_y}$ and the columns that correspond to entries where respectively $\boldsymbol{\sigma}_y^z$ and both $\boldsymbol{\sigma}_x^z$ and $\boldsymbol{\sigma}_y^z$ are nonzero. Using these row or column reduced matrices we write our total system with PML layers as:

$$\boldsymbol{y}' = \begin{bmatrix} \boldsymbol{h}_x' \\ \boldsymbol{h}_y' \\ \boldsymbol{e}_z' \\ \boldsymbol{b}_x' \\ \boldsymbol{b}_y' \\ \boldsymbol{b}_z' \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\Sigma}_y^x & & -\boldsymbol{M}^{x,-1}\boldsymbol{D}_y^x & & & \\ & -\boldsymbol{\Sigma}_x^y & \boldsymbol{M}^{y,-1}\boldsymbol{D}_x^y & & & \\ -\boldsymbol{E}^{z,-1}\boldsymbol{D}_y^x & \boldsymbol{E}^{z,-1}\boldsymbol{D}_x^z & -(\boldsymbol{\Sigma}_x^z + \boldsymbol{\Sigma}_y^z) & \hat{\boldsymbol{I}}_{n_x(n_y+1)} & \hat{\boldsymbol{I}}_{(n_x+1)n_y} & \hat{\boldsymbol{I}}_{n_x n_y} \\ -\boldsymbol{E}^{z,\widehat{-1}\boldsymbol{\Sigma}_x^z}\boldsymbol{D}_y^z & & & & & \\ & \boldsymbol{E}^{z,\widehat{-1}\boldsymbol{\Sigma}_y^z}\boldsymbol{D}_x^z & & & & \\ & & -\widehat{\boldsymbol{\Sigma}_x^z\boldsymbol{\sigma}_y^z} & & & \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_x \\ \boldsymbol{h}_y \\ \boldsymbol{e}_z \\ \boldsymbol{b}_x \\ \boldsymbol{b}_y \\ \boldsymbol{b}_z \end{bmatrix} - \begin{bmatrix} \boldsymbol{m}_x \\ \boldsymbol{m}_y \\ \boldsymbol{j}_z \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}.$$

This derivation is for a PML layer on both sides of the domain. It is also possible to implement PML boundaries on only one side of the domain, if one wants to simulate a waveguide for example. In this case we set $\sigma_y = 0$ and the last two auxiliary differential equations disappear.

2.3 Homogenization of ϵ_r

In a scattering medium we have a lot of isles with a different value of ϵ_r. On the border of such isles we have a discontinuity in ϵ_r. To correctly represent such isles on our discrete grid it is useful to use a homogenization method. In this section we shortly discuss how we do this.

Our algorithm starts with a grid of very high resolution for ϵ_r such that the isles are represented accurately (J.J.W. van der Vegt, personal communication, march 2017). After that we perform the following iteration on ϵ:

$$\begin{aligned} \epsilon_{i,j}^{n+1} &= \tfrac{1}{2}\epsilon_{i,j}^n + \tfrac{1}{8}\left(\epsilon_{i+1,j}^n + \epsilon_{i-1,j}^n + \epsilon_{i,j+1}^n + \epsilon_{i,j-1}^n\right), \\ &= \epsilon_{i,j}^n + \frac{(\Delta x)^2}{8}\frac{\epsilon_{i+1,j}^n - 2\epsilon_{i,j}^n + \epsilon_{i-1,j}^n}{(\Delta x)^2} + \frac{(\Delta y)^2}{8}\frac{\epsilon_{i,j+1}^n - 2\epsilon_{i,j}^n + \epsilon_{i,j-1}^n}{(\Delta y)^2}, \\ &\approx \epsilon_{i,j}^n + \Delta t\, \Delta\epsilon, \end{aligned}$$

where in the last step we assumed $\Delta x = \Delta y$ and defined $\Delta t = \frac{(\Delta x)^2}{8}$. Hence our homogenization scheme finds an approximation of ϵ for some t when ϵ satisfies the equation $\frac{\partial \epsilon}{\partial t} = \Delta \epsilon$, which is just the heat equation.
After sufficient iterations (i.e. when ϵ doesn't change much anymore) we interpolate these values of ϵ on a coarser grid and hopefully have a better representation of our isles.
This method is different from the approach used in MEEP [15].

Chapter 3

ITR scheme

We will develop several Krylov methods to perform the time integration in the next chapter. But we also need another reference time-integration method to measure the performance of our Krylov methods. For this we use the implicit trapezoidal rule scheme (ITR).
For the discretization Matrix $\boldsymbol{A}$ with $\sigma_m = \sigma = 0$ we proved that all eigenvalues are purely imaginary. After introducing the damping of the PML layer some eigenvalues of $\boldsymbol{A}$ have a positive real part. But we still have some eigenvalues very close to the imaginary axis [2].
As explained by Hundsdorfer et al. [6] the stability function of forward euler is $R(z) = 1 + z$ with $z = \Delta t \gamma_i$ for some eigenvalue of $\boldsymbol{A}$ γ_i. The scheme is stable as long as $|R(z)| \leq 1$ for all eigenvalues γ_i, the stability region of $R(z)$ is a circle with radius 1 around the point $(-1, 0)$. Since we have some eigenvalues that are very close to purely imaginary eigenvalues we have to choose a very small time step in order to have a stable integration scheme using forward Euler.
When we choose an implicit scheme like the trapezoidal rule which can be represented by the Butcher array:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

or differently as:

$$\begin{aligned} \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{\Delta t}{2}\boldsymbol{F}(\boldsymbol{y}_n, t_n) + \frac{\Delta t}{2}\boldsymbol{F}(\boldsymbol{y}_{n+1}, t_{n+1}) \\ &= \boldsymbol{y}_n + \frac{\Delta t}{2}(-\boldsymbol{A}\boldsymbol{y}_n + \alpha(t_n)\boldsymbol{g}_s) + \frac{\Delta t}{2}(-\boldsymbol{A}\boldsymbol{y}_{n+1} + \alpha(t_{n+1})\boldsymbol{g}_s), \\ \boldsymbol{y}_{n+1} &= \left(\boldsymbol{I} + \frac{\Delta}{2}\boldsymbol{A}\right)^{-1}\left(\left(\boldsymbol{I} - \frac{\Delta}{2}\boldsymbol{A}\right)\boldsymbol{y}_n + \frac{\Delta}{2}\left(\alpha(t_n) + \alpha(t_{n+1})\right)\boldsymbol{g}_s\right) \end{aligned}$$

we have stability function $R(z) = \frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}$. In this case $|R(z)| < 0$ as long as $Re(z) < 0$. Since the real part of all eigenvalues of $-\boldsymbol{A}$ is nonpostive we have an unconditionally stable integration scheme.
Also ITR seems like a good integration scheme for Maxwell's equations since Verwer [22] shows that the scheme is conservative for the Maxwell equations in the absence of sources. Furthermore in the case of two dimensions the action of the inverse of $(\boldsymbol{I} - \frac{\Delta t}{2}\boldsymbol{A})$ can easily be calculated using an LU decomposition [22], which can be computed efficiently for the 2D case.

Chapter 4

Krylov time integration for multiple inputs

In this chapter we will look at Krylov methods for time integration of the following problem:

$$\begin{aligned} \boldsymbol{y}' &= -\boldsymbol{A}\boldsymbol{y} + \alpha(t)\boldsymbol{g}_s, \\ \boldsymbol{y}(0) &= 0, \end{aligned} \tag{4.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{g}_s \in \mathbb{R}^n$, $\boldsymbol{y} : \mathbb{R} \to \mathbb{R}^n$ and $\alpha : \mathbb{R} \to \mathbb{R}$. We assume that this is a stable system, hence all eigenvalues of $\boldsymbol{A}$ are nonnegative.
We want to be able to solve this system efficiently for multiple input functions $\alpha(t)$, therefore we look at different ways to exploit Krylov methods to do so.
The first method we discuss is Krylov without stepping in time but only with restarting in dimension. The advantage of this method is that we can search for solutions of all input functions in the same Krylov subspace.
The next method we discuss uses stepping in time. The problem here is that we get a different initial value for every input function which causes a lot of additional computational work. To counter this we try to make a low rank approximation of all different initial vectors and use a block method to solve the system, this method is similar to the method described in [1]. If it is possible to make a low rank approximation with low dimension for all time steps this method could be very efficient.
The last method we discuss uses the solution of the problem in the frequency domain to transform the problem to a homogeneous problem which can possibly be more efficient to solve for Krylov methods. Also the shift-invariance of the Krylov subspace can be exploited here to efficiently find frequency domain solutions.
We discuss these methods one by one in the following sections.

4.1 Introduction to Krylov methods and restarting in dimension

In this section we give a general introduction to Krylov methods. Then we discuss a Krylov method that uses restarting in dimension for solving problems like (4.1). In this section we only discuss the Krylov subspace with a block-size of one, a more general description using a Krylov block method can be found in [1].
The analytical solution of the problem (4.1) is:

$$\boldsymbol{y}(t) = \int_0^t e^{\boldsymbol{A}(s-t)}\boldsymbol{g}_s\alpha(t)\,\mathrm{d}s.$$

We try to find an approximation to this solution in the Krylov subspace:

$$\mathrm{K}_m(\boldsymbol{A}, \boldsymbol{g}_s) = \{\boldsymbol{g}_s, \boldsymbol{A}\boldsymbol{g}_s, \boldsymbol{A}^2\boldsymbol{g}_s, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{g}_s\}.$$

Let $\boldsymbol{y}_i$ be eigenvectors of $\boldsymbol{A}$ with corresponding eigenvalues λ_i, we can write $\boldsymbol{g}_s$ as $\sum_i^n a_i \boldsymbol{y}_i$. The Krylov subspace consists of the vectors: $\sum_i^n a_i \lambda_i^k \boldsymbol{y}_i$ for $k = 0, 1, \ldots, m-1$. This means that eigenvectors with higher eigenvalues are more dominant in the Krylov subspace. This is not desirable since our eigenvalues appear in the matrix exponential as $e^{-\lambda t}$, hence the smaller eigenvalues are more important here. Therefore we often use another approach which is called the Shift and Invert (SAI) method. In the SAI method we try to find a solution in the shifted and then inverted version of $\boldsymbol{A}$, hence in $\mathrm{K}_m\left((\boldsymbol{I}+\gamma\boldsymbol{A})^{-1}, \boldsymbol{g}_s\right)$ for some value of γ. The eigenvalues of $(\boldsymbol{I}+\gamma\boldsymbol{A})$ are then $\frac{1}{1+\gamma\lambda}$, where now the smallest eigenvalues of $\boldsymbol{A}$ are the largest. The convergence of the algorithm depends largely on the right choice of γ, Botchev proposes a useful method for finding γ in [2].
The first step we take to find a solution in a Krylov subspace is to find an orthonormal basis of the subspace. The Arnoldi method is used to construct this basis, and gives the following equation:

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_m \boldsymbol{H}_{m,m} + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T,$$

where $\boldsymbol{V}_m$ is an orthonormal basis of $\mathrm{K}_m(\boldsymbol{A}, \boldsymbol{g}_s)$, $\boldsymbol{v}_{m+1}$ is the last column of $\boldsymbol{V}_{m+1}$, and $\boldsymbol{H}_{m,m}$ is upper Hessenberg. A matrix that is upper Hessenberg is an upper triangular matrix which also has nonzero entries on the diagonal beneath the main diagonal.
We can do the same for the SAI method to arrive at:

$$(\boldsymbol{I}+\gamma\boldsymbol{A})^{-1}\boldsymbol{V}_m = \boldsymbol{V}_m \tilde{\boldsymbol{H}}_{m,m} + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T,$$

Which can be rewritten to:

$$\begin{aligned}
\boldsymbol{A}\boldsymbol{V}_m &= \boldsymbol{V}_m\left(\frac{\tilde{\boldsymbol{H}}_{m,m}^{-1}-\boldsymbol{I}}{\gamma}\right) - \frac{h_{m+1,m}}{\gamma}(\boldsymbol{I}+\gamma\boldsymbol{A})\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T\tilde{\boldsymbol{H}}_{m,m}^{-1} \\
\boldsymbol{A}\boldsymbol{V}_m &= \boldsymbol{V}_m\boldsymbol{H}_{m,m} - \frac{h_{m+1,m}}{\gamma}(\boldsymbol{I}+\gamma\boldsymbol{A})\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T\tilde{\boldsymbol{H}}_{m,m}^{-1}
\end{aligned}$$

Now first we define the residual $\boldsymbol{r}_i(t)$ and error $\boldsymbol{e}_i(t)$ for an approximation $\boldsymbol{y}_i$ as follows:

$$\begin{aligned}
\boldsymbol{r}_i &:= -\boldsymbol{y}_i' - \boldsymbol{A}\boldsymbol{y}_i + \alpha(t)\boldsymbol{g}_s, \\
\boldsymbol{e}_i &:= \boldsymbol{y} - \boldsymbol{y}_i.
\end{aligned}$$

The residual here is the error of the approximation in the equation it has to satisfy. The error is the difference between our approximation and the unknown exact solution.
We can find that the residual and the error are related in the following way:

$$\begin{aligned}
\boldsymbol{e}_i' &= -\boldsymbol{A}\boldsymbol{e}_i + \boldsymbol{r}_i, \\
\boldsymbol{e}_i(0) &= \boldsymbol{0}.
\end{aligned} \tag{4.2}$$

Our initial guess is $\boldsymbol{y}_0 = \boldsymbol{0}$, which results in $r_0(t) = \alpha(t)\|\boldsymbol{g}_s\|, \boldsymbol{g}_0 = \frac{\boldsymbol{g}_s}{\|\boldsymbol{g}_s\|}$ and $\boldsymbol{e}_0 = \boldsymbol{y}(t)$. We will always write our residual $\boldsymbol{r}_i$ in the form $r_i(t)\boldsymbol{g}_i$ with $\|\boldsymbol{g}_i\| = 1$ as we see later on.
We try to find an approximation $\boldsymbol{e}_i^m = \boldsymbol{V}_m\boldsymbol{u}^m$ of our error $\boldsymbol{e}_i$ in $\mathrm{K}_m(\boldsymbol{A}, \boldsymbol{g}_i)$ or $\mathrm{K}_m\left((\boldsymbol{I}+\gamma\boldsymbol{A})^{-1}, \boldsymbol{g}_i\right)$. We do this by substituting $\boldsymbol{e}_i^m$ in equation (4.2) and then finding the projection with respect to $\boldsymbol{V}_m$. So it is like using the finite element method with base functions $\boldsymbol{v}_j$. Solving this for the regular Krylov method results in:

$$\begin{aligned}
\boldsymbol{V}_m^T\boldsymbol{e}_i^{m'} &= \boldsymbol{V}_m^T\left(-\boldsymbol{A}\boldsymbol{e}_i^m + r_i(t)\boldsymbol{g}_i\right), \\
\boldsymbol{u}_m' &= -\boldsymbol{H}_{m,m}\boldsymbol{u}_m + r_i(t)\boldsymbol{e}_1,
\end{aligned}$$

as projected differential equation for $\boldsymbol{u}^m$. We can find the residual of this solution:

$$\begin{aligned}
\boldsymbol{r}_{i+1} &= -h_{m+1,m}\left(\boldsymbol{e}_m^T\boldsymbol{u}^m\right)\boldsymbol{v}_{m+1}, \\
r_{i+1}(t) &= -h_{m+1,m}\left(\boldsymbol{e}_m^T\boldsymbol{u}_m\right), \\
\boldsymbol{g}_{i+1} &= \boldsymbol{v}_{m+1}.
\end{aligned}$$

Now we have a new approximation $\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \boldsymbol{e}_i^m$ which is hopefully be better than the previous one. In this case we call m the amount of iterations per restart, and we call i the amount of restarts.
For the SAI method we use the same projected ODE, but now the residual is a bit different:

$$\begin{aligned} \boldsymbol{r}_{i+1} &= \frac{h_{m+1,m}}{\gamma} \left(\boldsymbol{e}_m^T \tilde{\boldsymbol{H}}_{m,m}^{-1} \boldsymbol{u}^m \right) (\boldsymbol{I} + \gamma \boldsymbol{A})\, \boldsymbol{v}_{m+1}, \\ r_{i+1}(t) &= \frac{h_{m+1,m}}{\gamma} \left(\boldsymbol{e}_m^T \tilde{\boldsymbol{H}}_{m,m}^{-1} \boldsymbol{u}_m \right) \| (\boldsymbol{I} + \gamma \boldsymbol{A})\, \boldsymbol{v}_{m+1} \|, \\ \boldsymbol{g}_{i+1} &= \frac{(\boldsymbol{I} + \gamma \boldsymbol{A})\, \boldsymbol{v}_{m+1}}{\| (\boldsymbol{I} + \gamma \boldsymbol{A})\, \boldsymbol{v}_{m+1} \|}. \end{aligned} \tag{4.3}$$

We assumed that all eigenvalues of $\boldsymbol{A}$ are nonnegative, hence looking at (4.2) we see that the error is small as long as we keep the residual small. We can continue expanding the Krylov basis until some stopping criterion on the residual is satisfied, and then we have a solution of the desired accuracy. In this case we can for example define a norm for the residual:

$$\sqrt{\frac{\int_0^T \|\boldsymbol{r}^m\|^2 \,\mathrm{d}t}{\int_0^T \alpha^2(t) \,\mathrm{d}t}},$$

for final time T, if this norm is smaller than a certain tolerance we can stop the simulation. Then we know that the relative error is of about the same order as the tolerance.
Expanding the Krylov basis becomes more expensive every iteration, since for iteration m we need to orthogonalize a vector with respect to m vectors. Therefore it is often useful to perform restarts in dimension.
If we perform a restart for a different function $\alpha(t)$ the part of the residual that changes is only the scalar function $r_i(t)$, the vector $\boldsymbol{g}_i$ stays exactly the same. Therefore restarting in dimension seems like a good method to simulate the problem for many different functions $\alpha(\omega, t)$ at the same time. We only have to remember the residual functions $r_i(\omega, t)$ for all values of ω.
Apart from restarting in dimension we can also try restarting in time. The problem here is that after the restart we do not have initial value $\mathbf{0}$ anymore, hence we have to expand a Krylov basis with more than one vector at the same time. Also for multiple input functions this new initial value depends on the function $\alpha(\omega, t)$. In the next chapter we propose a method to deal with this issue.
Expanding the Krylov basis is often the most expensive part of this algorithm. Krylov methods often take large time steps as compared to Runge-Kutta methods, but if we want to perform only one big time step so that we can restart in dimension but don't have to restart in time, this time step is also very big for Krylov methods. This means that the time we have to invest in the ODE solver becomes larger which can be a disadvantage.

4.2 Krylov method with restarting in time

Now we try to simulate the following problem:

$$\begin{aligned} \boldsymbol{y}_\omega' &= -\boldsymbol{A}\boldsymbol{y}_\omega + \alpha(\omega, t)\boldsymbol{g}_s, \\ \boldsymbol{y}_\omega(0) &= \boldsymbol{y}_{\omega,0}, \end{aligned}$$

for $\omega \in \Omega$ for some finite set of values Ω. If the functions $\alpha(\omega, t)$ are very alike we can also suppose that the initial values $\boldsymbol{y}_{\omega,0}$ do not differ much. We can rewrite this problem by introducing the variable $\tilde{\boldsymbol{y}_\omega} = \boldsymbol{y}_\omega - \boldsymbol{y}_{\omega,0}$, with this variable we get the equivalent system:

$$\begin{aligned} \tilde{\boldsymbol{y}_\omega}' &= -\boldsymbol{A}\tilde{\boldsymbol{y}}_\omega - \boldsymbol{A}\boldsymbol{y}_{\omega,0} + \alpha(\omega, t)\boldsymbol{g}_s, \\ \tilde{\boldsymbol{y}}_\omega(0) &= \mathbf{0}. \end{aligned}$$

The next step is to make a singular value decomposition (SVD) of the following matrix:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{g}_s & -\boldsymbol{A}\boldsymbol{y}_{\omega_1,0} & -\boldsymbol{A}\boldsymbol{y}_{\omega_2,0} & \dots & -\boldsymbol{A}\boldsymbol{y}_{\omega_p,0} \end{bmatrix}$$

The SVD of an $n \times p$ matrix $\boldsymbol{G}$ consists of an $n \times n$ unitary matrix $\boldsymbol{U}$, an $n \times p$ diagonal matrix $\boldsymbol{\Sigma}$ with nonnegative entries and an $p \times p$ unitary matrix $\boldsymbol{V}$ such that:

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*.$$

The diagonal of $\boldsymbol{\Sigma}$ is ordered such that the largest values are in the upper left corner. We can truncate this decomposition by taking only the first j columns of $\boldsymbol{U}$, a diagonal matrix with the first j diagonal values of $\boldsymbol{\Sigma}$, and also the first j columns of $\boldsymbol{V}$. Now we get:

$$\boldsymbol{G} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^*,$$

where $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\Sigma}}$ and $\tilde{\boldsymbol{V}}^*$ are the truncated versions of their originals. The error of this approximation in the Frobenius norm is:

$$\|\boldsymbol{G} - \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^*\| = \sum_{i=j+1}^{\min(m,p)} \sigma_i^2,$$

where σ_i are the diagonal entries of $\boldsymbol{\Sigma}$. So if we choose a proper value of j to truncate our SVD we can make a good approximation. Let $\boldsymbol{Q} = \tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^*$, then we can rewrite our problem as follows:

$$\begin{aligned} \tilde{\boldsymbol{y}_{\omega_i}}' &= -\boldsymbol{A}\tilde{\boldsymbol{y}}_{\omega_i} - \boldsymbol{A}\boldsymbol{y}_{\omega_i,0} + \alpha_{\omega_i}(t)\boldsymbol{g}_s, \\ &\approx -\boldsymbol{A}\tilde{\boldsymbol{y}}_{\omega_i} + \tilde{\boldsymbol{U}}\boldsymbol{Q}\boldsymbol{e}_{i+1} + \alpha(\omega_i, t)\tilde{\boldsymbol{U}}\boldsymbol{Q}\boldsymbol{e}_1, \\ &= -\boldsymbol{A}\tilde{\boldsymbol{y}}_{\omega_i} + \tilde{\boldsymbol{U}}\boldsymbol{\beta}_{\boldsymbol{\omega_i}}(t), \end{aligned}$$

which can be solved using a krylov block method [1]. If we can make an efficient SVD truncation with $j << |\Omega|$, this can probably be a useful method to simulate a lot of different input functions at the same time.

4.3 Different approach with asymptotical time periodic solution

In this chapter we assume that our input function is in the form of a harmonic function. In this case we can find a frequency domain solution of our differential equation which we can find in a faster way than solving a projected ODE system for a long time period. Hence we want to find a asymptotic time periodic solution of the following differential equation:

$$\begin{aligned} \boldsymbol{y}' &= -\boldsymbol{A}\boldsymbol{y} + \boldsymbol{g}_s e^{2\pi i\omega t}, \\ \boldsymbol{y}(0) &= \boldsymbol{0}. \end{aligned}$$

Let us suppose that $\boldsymbol{A}$ is diagonalizable, hence $\boldsymbol{A} = \boldsymbol{T}\boldsymbol{D}\boldsymbol{T}^{-1}$ with $\boldsymbol{D}$ a diagonal matrix with on its diagonal the eigenvalues of $\boldsymbol{A}$. If $\boldsymbol{A}$ is not diagonalizable we can perturb it slightly to get a diagonalizable matrix since the set of diagonalizable matrices is dense in the set of matrices. Furthermore, let us

assume that $2\pi i\omega \notin \lambda(\boldsymbol{A})$, then we can find an analytical solution of this problem as follows:

$$\begin{aligned}
\boldsymbol{y}(t) &= \int_0^t e^{\boldsymbol{A}(s-t)}\boldsymbol{g}_s e^{-2\pi i\omega s}\,\mathrm{d}s,\\
&= e^{-\boldsymbol{A}t}\int_0^t e^{\boldsymbol{A}s}\boldsymbol{g}_s e^{2\pi i\omega s}\,\mathrm{d}s,\\
&= e^{-\boldsymbol{A}t}\boldsymbol{T}\int_0^t e^{\boldsymbol{D}s}e^{2\pi i\omega s}\,\mathrm{d}s\boldsymbol{T}^{-1}\boldsymbol{g}_s,\\
&= e^{-\boldsymbol{A}t}\boldsymbol{T}\int_0^t e^{(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})s}\,\mathrm{d}s\boldsymbol{T}^{-1}\boldsymbol{g}_s,\\
&= e^{-\boldsymbol{A}t}\boldsymbol{T}\left|e^{(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})s}(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})^{-1}\right|_{s=0}^{t}\boldsymbol{T}^{-1}\boldsymbol{g}_s,\\
&= \boldsymbol{T}e^{-\boldsymbol{D}t}\boldsymbol{T}^{-1}\boldsymbol{T}e^{\boldsymbol{D}t}(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})^{-1}\boldsymbol{T}^{-1}\boldsymbol{g}_s e^{2\pi i\omega t} - e^{-\boldsymbol{A}t}\boldsymbol{T}(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})^{-1}\boldsymbol{T}^{-1}\boldsymbol{g}_s,\\
&= \boldsymbol{T}(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})^{-1}\boldsymbol{T}^{-1}\boldsymbol{g}_s e^{2\pi i\omega t} - e^{-\boldsymbol{A}t}\boldsymbol{T}(\boldsymbol{D}+2\pi i\omega\boldsymbol{I})^{-1}\boldsymbol{T}^{-1}\boldsymbol{g}_s,\\
&= (2\pi i\omega\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{g}_s e^{2\pi i\omega t} - e^{-\boldsymbol{A}t}(2\pi i\omega\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{g}_s,\\
&= \boldsymbol{y}_{st}e^{2\pi i\omega t} - e^{-\boldsymbol{A}t}\boldsymbol{y}_{st}.
\end{aligned}$$

Here we define $\boldsymbol{y}_{st}^{\omega} := (2\pi i\omega\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{g}_s$ the steady state solution. We can see that our solution consists of two parts, the first part is $\hat{\boldsymbol{y}} := \boldsymbol{y}_{st}e^{2\pi i\omega t}$ which is the time periodic part of our solution with the same frequency as our source. The second part is $\tilde{\boldsymbol{y}} = e^{-\boldsymbol{A}t}\boldsymbol{y}_{st}$ which is the solution of the following initial value problem:

$$\begin{aligned}
\tilde{\boldsymbol{y}}' &= -\boldsymbol{A}\tilde{\boldsymbol{y}},\\
\tilde{\boldsymbol{y}}(0) &= \boldsymbol{y}_{st}.
\end{aligned} \tag{4.4}$$

This goes to zero under the assumption that all eigenvalues of $\boldsymbol{A}$ have a positive real part. Hence under this assumption the solution converges to the harmonic steady state solution $\boldsymbol{y}_{st}^{\omega}$.
Under the assumption that $\boldsymbol{y}(t)$ converges to $\hat{\boldsymbol{y}}$ quickly enough it can be advantageous to search directly for the solution $\boldsymbol{y}_{st}$ instead of solving (4.4) for a large time interval.
It is possible that the homogeneous system (4.4) is easier to solve for Krylov methods than an inhomogeneous problem like (4.1). The only problem here is that we have a different input function for every frequency hence we cannot exploit the advantages of Krylov methods. An advantage that we can maybe exploit is that solutions in the neighborhood of one frequency lie close to each other. Let $\boldsymbol{y}_{st} = (2\pi i\omega_1\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{g}_s$ and $\tilde{\boldsymbol{y}}_{st} = (2\pi i\omega_2\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{g}_s$ with $|\omega_1-\omega_2|$ small. If $\|\boldsymbol{y}_{st}-\tilde{\boldsymbol{y}}_{st}\| \ll \|\boldsymbol{y}_{st}\|$, we can solve:

$$\begin{aligned}
\bar{\boldsymbol{y}}' &= -\boldsymbol{A}\bar{\boldsymbol{y}},\\
\bar{\boldsymbol{y}}(0) &= \tilde{\boldsymbol{y}}_{st} - \tilde{\boldsymbol{y}}_{st},
\end{aligned}$$

with lower tolerance and use the solution we have already calculated for $\boldsymbol{y}$ to get a solution for $\tilde{\boldsymbol{y}}$. We cannot exploit this fact using ITR because we cannot control the error here.

4.3.1 Finding the frequency domain solution

We take a look at some different methods of finding $\boldsymbol{y}_{st}$ for multiple values of ω. Let us first define:

$$\boldsymbol{y}_{st}^{\omega} = (2\pi i\omega\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{g}_s.$$

One way to find multiple $\boldsymbol{y}_{st}^{\omega}$ would be to solve this system for all required values of ω but since an LU decomposition of $(2\pi i\omega\boldsymbol{I}+\boldsymbol{A})$ is created every time the system is solved this seems inefficient. It may

be better to use this LU decomposition as a preconditioner for solving the system for other values of ω. Because if ω_1 and ω_2 lie close to each other we hope that we can assume that $(2\pi i\omega_1 \boldsymbol{I} + \boldsymbol{A})^{-1}(2\pi i\omega_2 \boldsymbol{I} + \boldsymbol{A}) \approx \boldsymbol{I}$. If this assumption is true and holds for a wide range of ω we can probably solve this system very efficiently for a lot of different ω. So we should research for what values of ω_1 and ω_2 the frequency domain solutions $\boldsymbol{y}_{st}^{\omega_1}$ and $\boldsymbol{y}_{st}^{\omega_2}$ lie close to each other.
Another way to efficiently find multiple solutions would be to search in a Krylov subspace for all solutions as we have done for restarting in dimension. This method is very similar to GMRES, and uses the fact that the Krylov subspace is shift invariant. Let us start by defining a residual and error again.

$$\begin{aligned}\boldsymbol{e}_i^\omega &= \boldsymbol{y}_s^\omega - \boldsymbol{y}_{s,i}^\omega,\\ \boldsymbol{r}_i^\omega &= \boldsymbol{g}_s - (2\pi i\omega \boldsymbol{I} + \boldsymbol{A})\boldsymbol{y}_{s,i}^\omega = r_i \boldsymbol{g}_i,\end{aligned}$$

where $\boldsymbol{y}_{s,i}^\omega$ is an approximation of $\boldsymbol{y}_s^\omega$, and $\|\boldsymbol{g}_i\| = 1$.
Now we can find the following relation between the error and the residual:

$$\begin{aligned}(2\pi i\omega \boldsymbol{I} + \boldsymbol{A})\boldsymbol{e}_i^\omega &= (2\pi i\omega \boldsymbol{I} + \boldsymbol{A})\boldsymbol{y}_s^\omega - (2\pi i\omega \boldsymbol{I} + \boldsymbol{A})\boldsymbol{y}_{s,i}^\omega\\ &= \boldsymbol{g}_s - (2\pi i\omega \boldsymbol{I} + \boldsymbol{A})\boldsymbol{y}_{s,i}^\omega\\ &= \boldsymbol{r}_i^\omega\end{aligned}$$

Now define $\boldsymbol{y}_{s,0}^\omega = \boldsymbol{0}$ such that $\boldsymbol{e}_0^\omega = \boldsymbol{y}_s^\omega$ and $\boldsymbol{r}_0^\omega = \boldsymbol{g}_s = r_0^\omega \boldsymbol{g}_0$. Now we use the Krylov subspace $K_m\left((\boldsymbol{I} - \gamma \boldsymbol{A})^{-1}, \boldsymbol{g}_i\right)$ to find a solution. It is important to notice that this space is the same for all ω because of the shift-invariance of Krylov spaces. We try to find an approximation of the form $\boldsymbol{e}_{i,m}^\omega = \boldsymbol{V}_m \boldsymbol{u}_m^\omega$ which results in the following projected problem:

$$(2\pi i\omega \boldsymbol{I} + \boldsymbol{H}_{m,m})\boldsymbol{u}_m^\omega = \boldsymbol{e}_1.$$

This problem is easy to solve since m is relatively small. The new residual is:

$$\begin{aligned}\boldsymbol{r}_{i+1}^\omega &= \boldsymbol{g}_i - (2\pi i\omega \boldsymbol{I} + \boldsymbol{A})\, \boldsymbol{V}_m \boldsymbol{u}_m^\omega \\ &= \boldsymbol{g}_i - \boldsymbol{V}_m\, (2\pi i\omega \boldsymbol{I} + \boldsymbol{H}_{m,m})\, \boldsymbol{u}_m^\omega + \frac{h_{m+1,m}}{\gamma}\, (\boldsymbol{I} + \gamma \boldsymbol{A})\, \boldsymbol{v}_{m+1} \boldsymbol{e}_m^T \tilde{\boldsymbol{H}}_{m,m} \boldsymbol{u}_m^\omega \\ &= \frac{h_{m+1,m}}{\gamma} \left(\boldsymbol{e}_m^T \tilde{\boldsymbol{H}}_{m,m} \boldsymbol{u}_m^\omega\right) \|(\boldsymbol{I} + \gamma \boldsymbol{A})\boldsymbol{v}_{m+1}\| \frac{(I + \gamma \boldsymbol{A})\boldsymbol{v}_{m+1}}{\|(I + \gamma \boldsymbol{A})\boldsymbol{v}_{m+1}\|} \\ &= r_{i+1}^\omega \boldsymbol{g}_{i+1}.\end{aligned} \tag{4.5}$$

We see that in our new residual the vector is independent of ω so we can restart in a very cheap way using this algorithm for a lot of different values of ω.

4.4 Discussion

We have looked at some different methods to simulate problem (4.1) for multiple input functions. The first method we looked at was to take a very big time step and restart only in dimension. The advantage here is that we can search for our solution in the same Krylov subspace for all input functions. A disadvantage can be that the cost of the ODE solver for the projected problem is more dominant for such a large time interval. Also, we do not know if it will work to take such a large time step since Krylov methods are usually used on much smaller time intervals.
The second method we discussed uses Krylov with restarting in time. It calculates an SVD truncation of all initial value vectors every step. If it is possible to reduce the total rank of the matrix with initial values significantly using this SVD truncation the total costs of the algorithm might also reduce significantly. The method continues using a block method to solve the truncated system for all frequencies.
The final method we have discussed assumes that we have a harmonic input. It transforms the system

into a homogeneous system by first solving a linear system. If all solutions in the neighborhood of some frequency are similar, this method exploits this fact by reducing the tolerance. Furthermore, we suppose that Krylov methods work better for homogeneous systems because they have been proven to work properly for these systems.

Chapter 5

Results

In this chapter we discuss all Krylov methods introduced in chapter 4 one by one. We compare their results with the ITR algorithm as a reference. Before we discuss these results we are going to introduce two test cases we want to test our methods on, and verify the implementation of our algorithm by comparing our results with results of meep.

5.1 Test cases

We are going to introduce two test cases here. The first case is a simple case that allows us to easily verify our implementation and to perform simple tests on. The second case involves a scattering medium and is more important for this research.

5.1.1 Test case 1

The first test case consists of open space with an obstacle in the middle. There is a source on one side of the obstacle and we are interested in what happens to the field on the other side of the obstacle.
The domain has a size of 12×12 unit length, and has PML boundaries with width one on all sides. The value of σ_x and σ_y go quadratically from 0 to σ_{max} in the PML area. There is a cylinder in the middle of the domain consisting of material with a higher dielectric constant. This cylinder has a radius of $1,5$ unit length and has dielectric constant $\epsilon_r = 2.25$, outside we just have $\epsilon_r = 1$. The source is of the form $\alpha(t)g_s(x)$, where $\boldsymbol{g}_s$ is an evenly distributed line on the left side of the domain, which goes linearly to zero in the PML area. The source is $\sin 2\pi\omega t$, we are interested in values of ω in the interval $[0.85, 1.15]$.

5.1.2 Test case 2

This test case is a waveguide with a scattering medium in it. The domain is $[-3\,,33] \times [0\,,10]$ with PEC ($E_z = 0$) boundaries on the upper and lower boundary. Furthermore we have PML boundaries for $x < -2$ and $x > 32$. The whole domain has dielectric value $\epsilon_r = 2.25$, but in the domain $[0\,,30] \times [0\,,10]$ there are 750 cylinders with dielectric value $\epsilon_r = 1$. These cylinders are randomly placed in this domain under the following conditions: the centers of the cylinders are placed on a minimum distance of 0.2 from the wall, the centers of the cylinders are placed at a minimum distance of 0.25 from each other, the radius of the cylinders is 0.1. Hence the domain looks like the domain in figure 5.1.2.

For the proper implementation of ϵ_r we have used the homogenization as described in section 2.3. First we implemented ϵ_r on a grid with resolution 256 after that we performed 200 iterations because after this many iterations the value of ϵ does not seem to change much anymore (figure 5.1.2). The resulting homogenized values of ϵ_r can be seen in figure 5.1.2.

The source is a line source at $x = -2$, which has the same value everywhere except for the ends, it

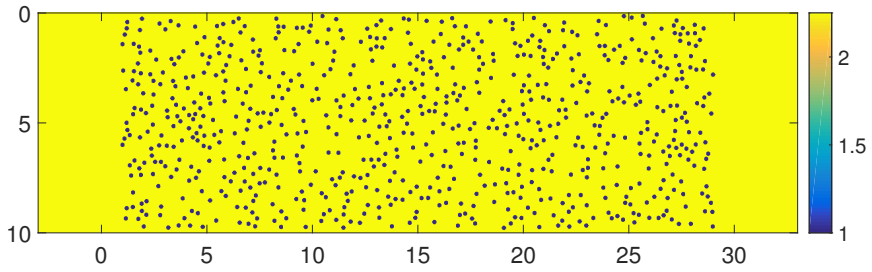

Figure 5.1: Example of a scattering domain in a waveguide

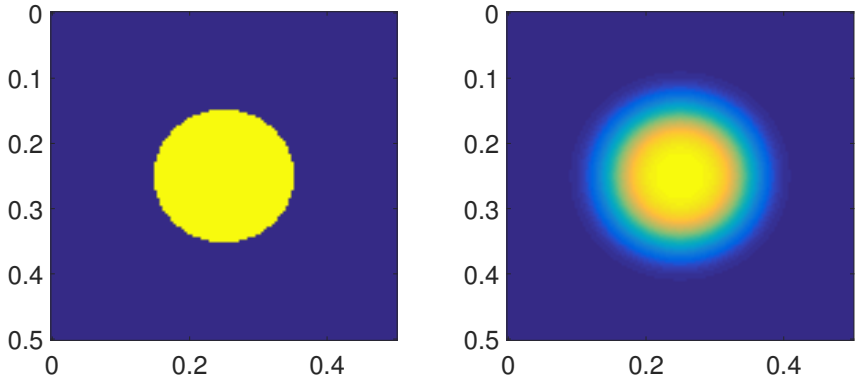

Figure 5.2: Example of an isle (left) and the same isle after homogenization (right)

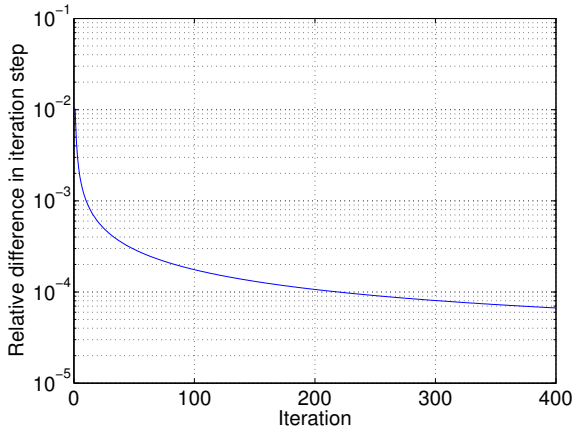

Figure 5.3: Relative difference between $\boldsymbol{\epsilon}^{i+1}$ and $\boldsymbol{\epsilon}^{i}$. We stop at $i = 200$.

Table 5.1: The error between the same simulation for ITR and meep for different resolutions

resolution	relative error
16	7.96 $\times 10^{-2}$
32	4.72 $\times 10^{-2}$
64	2.26 $\times 10^{-2}$

Table 5.2: Performance of ITR on testcase 1 with resolution 64. Here the time step is $\Delta x 2^{-i}$.

i	Relative error	CPU
0	3.13×10^{-2}	6.97×10^{3}
1	7.46×10^{-3}	1.44×10^{4}
2	1.50×10^{-3}	2.90×10^{4}

goes linearly to zero for $y < 1$ and $y > 9$ to satisfy the PEC boundaries. The temporal part of the source is $\alpha(t)$. Mostly we have $\alpha(t) = \sin 2\pi\omega t$ for $\omega \in [0.85, 1.15]$.

5.2 Code verification

To validate our implementation of the Maxwell equations we compare our results with the results of the same case in meep. We know that the results in meep are correct, so if our results converge to the results in meep we know our code is correct. We consider test case 1 with $\omega = 1$.
The output of meep is the E_z-field on the original domain without the PML layers. The outputs in meep are given at $(i + 0.5, j + 0.5)\Delta$, where $\Delta = \frac{1}{\text{resolution}}$. In my implementation the E_z field is given at $(i, j)\Delta$ hence we need a quadrature to get the approximation at the same points.
We perform the time integration of this case using the ITR scheme as described in chapter 3.
Since we are using two different ways of simulating the same problem we can not expect the results to be exactly the same. But they should be within a certain error margin. If we make the resolution of our grid higher the results should be more accurate and hence the results of our two methods should converge to the same values. So if we have this convergence we can conclude that our implementation is also correct.
In table 5.1 we show the error between meep and our method for different resolutions.
In this table we see the desired convergence. Hence we can conclude that our implementation is correct.

5.3 Reference Solutions

For both cases we calculate some reference solutions using ITR. The basic time step we use for ITR is $\Delta t = \Delta x$. We do multiple runs with time step $\Delta t = \frac{\Delta x}{2^i}$, and estimated the relative error of our solution as:

$$\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_{i_{\text{max}}}\|}{\|\boldsymbol{y}_{i_{\text{max}}}\|}$$

where $\boldsymbol{y}_i$ is the solution corresponding to time step Δt and i_{max} is the maximum value of i that we use. For both test cases we perform the simulations with resolution 64, $\sigma_{\text{max}} = 20$ and $\omega = 1$. For test case 1 $i_{max} = 3$ and for test case 2 $i_{\text{max}} = 5$. The results for test case 1 are in table 5.2 and for test case 2 in table 5.3.

Table 5.3: Performance of ITR on testcase 2 with resolution 64. Here the time step is $\Delta x 2^{-i}$.

i	Relative error	CPU
0	4.12×10^{-1}	1.84×10^{4}
1	1.07×10^{-1}	3.74×10^{4}
3	6.33×10^{-3}	1.32×10^{5}
4	1.27×10^{-3}	2.34×10^{5}

Table 5.4: Results for Krylov with restarting in time. Here f_ω is the frequency width, f_c is the center frequency and n_f is the amount of different frequencies. The total time needed for the simulation is CPU and block size is the required block size for the SVD truncation at the last time step.

f_w	f_c	n_f	CPU	block size
0.3	1	100	1798	21
0.6	1	100	2130	34
0.3	1	300	3587	21
$-$	1	1	25.3	2

5.4 Krylov with restarting in time

In this section we discuss the results of Krylov with restarting in time as described in section 4.2. We have done some test runs using this method with $\alpha_\omega(t) = \sin 2\pi\omega t$ for n_f frequencies ω distributed evenly in the interval $[f_c - \frac{f_w}{2}, f_c + \frac{f_w}{2}]$. We did this simulation on test case 1 for 20 time units, the results are in table 5.4. These results are produced with regular Krylov and time step 0.1. For the real problem we have to simulate for a much longer time than 20 dimensionless time step to let the signal pass fully through the domain. Since the dimension that is needed to make a proper SVD truncation is growing steadily we can expect it to become bigger and bigger.
The parameter block size is the size of the block that remained after the SVD truncation on the last time step. During the simulation the block size grows to this number. We see that a larger value of f_w results in a higher block size. But changing the density of frequencies n_f does not really alter the block size. So it is more efficient to simulate a lot of different frequencies that lie close to each other. In the case of $f_w = 0.3$ we see that using this method is faster than doing n_f single runs with all frequencies. But the advantage is really small and will probably disappear if we simulate for a longer time. So we conclude that this method is not an efficient method for simulating the same problem for a lot of different frequencies. The large block sizes needed for a good approximation of all signals make the simulation too slow.

5.5 Krylov with restarting in dimension

In this section we take a look at some results for Krylov with restarting in dimension. The method is described in section 4.1. We look at results for both test cases and are mainly interested in a time step of 500.

Table 5.5: The residual norm of Krylov after 100 Krylov subspace iterations for test case 1 for different values of the final time T, γ and σ_{max}.

For $T = 50$

σ_{max}, $\frac{\gamma}{T}$	0.1	0.05	0.025	0.01	0.005	0.0025	0.001
20	7.42×10^{-1}	1.35×10^{-1}	2.38×10^{-2}	1.21×10^{-3}	2.70×10^{-3}	5.87×10^{-2}	9.30×10^{-1}
40	1.59×10^{0}	2.17×10^{-1}	2.38×10^{-2}	3.63×10^{-3}	2.56×10^{-3}	7.10×10^{-2}	1.64×10^{0}
80	1.81×10^{2}	4.14×10^{-1}	1.14×10^{-1}	4.36×10^{-1}	4.23×10^{-3}	1.10×10^{-1}	2.69×10^{0}
160	2.13×10^{0}	6.01×10^{-1}	4.85×10^{-2}	1.44×10^{-2}	1.46×10^{-2}	1.04×10^{-2}	6.28×10^{0}

For $T = 200$

σ_{max}, $\frac{\gamma}{T}$	0.05	0.025	0.01	0.005	0.0025	0.001	0.0005
20	2.16×10^{7}	7.41×10^{-1}	5.62×10^{-2}	7.84×10^{-3}	5.86×10^{-4}	6.65×10^{-3}	1.09×10^{-1}
40	3.76×10^{7}	1.57×10^{0}	8.85×10^{-2}	8.94×10^{-3}	8.47×10^{-4}	6.82×10^{-3}	1.20×10^{-1}
80	5.79×10^{7}	3.62×10^{11}	1.86×10^{1}	1.34×10^{-2}	6.91×10^{2}	8.19×10^{-3}	1.49×10^{-1}
160	7.43×10^{7}	2.13×10^{0}	2.41×10^{-1}	1.99×10^{-2}	1.55×10^{-3}	8.98×10^{4}	2.30×10^{-1}

For $T = 500$

σ_{max}, $\frac{\gamma}{T}$	0.01	0.005	0.0025	0.001	0.0005	0.00025	0.0001
20	7.41×10^{-1}	1.35×10^{-1}	2.42×10^{-2}	5.89×10^{-4}	2.20×10^{-3}	5.66×10^{-2}	9.27×10^{-1}
40	1.57×10^{0}	2.17×10^{0}	2.50×10^{-2}	4.47×10^{-4}	2.12×10^{-3}	6.91×10^{-2}	1.64×10^{0}
80	3.83×10^{30}	4.13×10^{-1}	4.88×10^{-2}	3.61×10^{9}	3.15×10^{-3}	1.07×10^{-1}	2.69×10^{0}
160	2.16×10^{0}	6.01×10^{-1}	4.32×10^{-2}	1.53×10^{-3}	5.79×10^{-3}	1.02×10^{-1}	6.27×10^{0}

5.5.1 Test case 1

To get a feeling for this method, at first we only look at this case. This is mainly because this test case allows us to play a bit more, so we can get a feeling for the problem. In this case we have a cylinder with radius 1.5 instead of cylinders with radius 0.1 in the second case which means that we can use a lower resolution and still have a good representation of the cylinder on the domain. We use SAI instead of regular Krylov, from [2] we know that a proper strategy for choosing γ is doing a fixed amount of iterations for different values of γ and then choosing the value of γ corresponding with the smallest residual. Apart from choosing a proper value of γ we are also interested in the influence of σ_{max} on our simulation results.
The following results are made with $\omega = .5$ and resolution 8. We have done 100 iterations for 1 restart. In table 5.5 the results for different values of the final time T, different values of σ_{max} and different values of $\frac{\gamma}{T}$ are listed.
The residual norm is calculated here as:

$$\frac{\int_{T/2}^{T} |r(t)|^2 \, \mathrm{d}t}{\int_{T/2}^{T} |\alpha(t)|^2 \, \mathrm{d}t}.$$

In all cases we see the best results for $\sigma_{\text{max}} = 20$, but they do not differ much with $\sigma_{\text{max}} = 40$. Setting σ_{max} higher than that has a huge impact on the accuracy in some cases. For $T = 50$ the best value of $\frac{\gamma}{T} = 0.01$ seems to be the best options, for $T = 200$ this seems to be 0.0025 and for $T = 500$ we see that 0.001 is the best value. The interesting thing here is that in all cases we have an optimal value of $\gamma = 0.5$ seems to be the optimal value. This is different from small time steps where a linear relation between γ and T seems optimal. However these results suggest that a constant value of γ is optimal for large time steps. For the following simulations we use $\gamma = 0.5$ and σ_{max}.

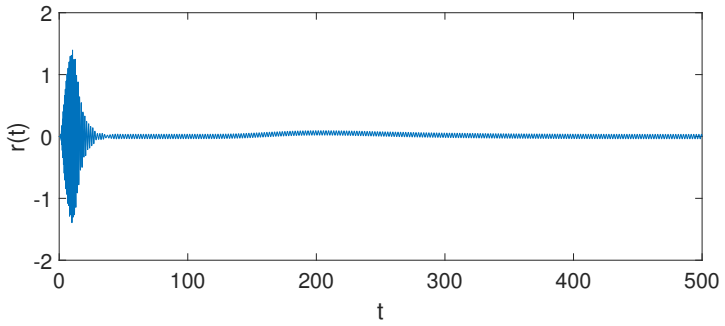

Figure 5.4: Residual after 1 restart with 100 iterations and $\gamma = 0.1$

It seems strange that the residual for $T = 50$ is often larger than the residual for $T = 200$ and $T = 500$, this can be explained by looking at the function of the residual $r_i(t)$. In figure 5.5.1 we see an example of this function after one restart with 100 iterations and $\gamma = 0.5$ and $\sigma_{\text{max}}20$. We observe that the first part of the residual is very large as compared to the tail which is already very small. This is also the reason why we only look at the second halve of the time step for the residual norm. If the tail of the residual function is small enough we can approximate the residual as:

$$r_i(t) \approx \begin{cases} r_i(t) & 0 \leq t \leq T_1, \\ 0 & T_1 < t \leq T. \end{cases}$$

If we want to solve the error for these equations we can first solve the part till T_1 to find a value of $\boldsymbol{e}_i(T_1)$, after this we are left with the following system:

$$\begin{aligned} \boldsymbol{e}_i' &= -\boldsymbol{A}\boldsymbol{e}_i, \\ \boldsymbol{e}_i(T_1) &= \boldsymbol{e}_i(T_1), \end{aligned}$$

for $t \in [T_1, T]$. Under the assumptions that this interval is large and that we have positive eigenvalues we assume that the contribution of this solution is really small. Hence we assume that it is sufficient if the tail of our residual is small.

Now that we have found efficient parameters we continue by looking for optimal values to restart the algorithm. The first thing we do is find how many iterations we need to have convergence in one restart. After that we perform restarts at a fixed amount of iterations and see if the algorithm converges and how many restarts are needed. In the following tables we have used $\omega = 1$ and $\frac{\gamma}{T} = 0.00075$ which is a bit less than the chosen value, but this seems to work better with a higher frequency. The residual norm is only evaluated at the end of a restart because the ODE solver takes a lot of time for a time step of 500, also when we do not have to solve the system very accurately only to evaluate the residual norm. The simulation stops when the residual norm is below a certain tolerance which is 10^{-4} in this case. We have chosen 10^{-4} because we want the temporal error to be of the same order as the spatial error which is of order $\mathrm{O}(\Delta x^2)$ which is close to 10^{-4} in the case of a resolution of 64. The results of this test are stated in table 5.6. There are some important things we can notice here. The first is that we seem to have convergence independent of the resolution of our grid. For resolution 16, 32 and 64 we need almost the same amount of restarts for convergence. This means that this method is more efficient for finer grids. The second observation is that we do not have convergence independent of the size of our restart. For a lower value of n_{restart} we seem to need a higher amount of total iterations. As can be seen in [2] we can expect convergence in the same amount of iterations for smaller values of n_{restart} in some cases. Unfortunately this does not seem to be true in our case.

For resolution 64 and restarting every 200 Krylov iterations we have an error of 1.00×10^{-2}, which is more accurate and slower than ITR with the maximum time step. But it is faster and less accurate compared to ITR with time step $\frac{\Delta x}{2}$, hence the performance is similar to ITR.

Table 5.6: Performance of Krylov with restarting for different resolutions and different restart sizes. Here n_{restart} is the amount of iterations in one restart, CPU_{tot} is the CPU time needed for the whole simulation and CPU_{ODE} is the CPU time needed only for solving the projected ODEs.

resolution	n_{restart}	n_{restarts}	residual norm	CPU_{tot}	CPU_{ODE}
16	200	3	1.35×10^{-6}	91.8	65.5
16	150	3	8.95×10^{-5}	63.7	47.5
16	100	7	1.62×10^{-5}	125.0	106.5
16	50	19	7.63×10^{-6}	407.0	391.2
32	200	3	5.42×10^{-6}	317.1	189.1
32	150	3	9.96×10^{-5}	224.2	143.4
32	100	7	3.68×10^{-5}	326.4	232.7
32	50	17	3.43×10^{-5}	569.4	492.8
64	200	3	4.60×10^{-5}	926.0	334.9
64	150	4	1.70×10^{-5}	782.2	318.9
64	100	7	8.57×10^{-5}	853.5	445.8
64	50	17	2.88×10^{-5}	1266.7	866.2

It is usually beneficial in Krylov methods to have a small value of n_{restart} because the orthonormalization in the Arnoldi algorithm needs more time every iteration, also for larger values of n_{restart} more memory is required. But in our case it does not seem beneficial to perform many restarts because every restart an ODE system has to be solved which takes a big share of the total work for very large time steps. We would expect that this would not be of influence the total time invested in solving the projected systems because they are smaller every restart, but contrary to our expectations this seems to be the case. So for large time steps it seems beneficial to perform less restarts which requires larger Krylov dimensions per restarts and therefore requires more memory.
At $t = 500$ we expect the solution to be very similar to the steady state solution as derived in section 5.6, therefore we suspect that the value of the residual in the GMRES method for the steady state solution (4.5) is very similar to the amplitude of the residual in (4.3). In figure 5.5.1 we see the residual norm calculated in two different ways. As we observe they are almost exactly the same. Hence (4.5) seems to be a quick way to approximate the residual between restarts.

5.5.2 The eigenvalues of $\tilde{\boldsymbol{H}}$

Let λ_i be an eigenvalue of $\boldsymbol{A}$, then $\frac{1}{1+\gamma\lambda_i}$ is an eigenvalue of $(\boldsymbol{I} + \gamma\boldsymbol{A})^{-1}$. Hence we expect the eigenvalues of $\tilde{\boldsymbol{H}}$ to be in the same range as the eigenvalues $\frac{1}{1+\gamma\lambda_i}$. If we then transform $\tilde{\boldsymbol{H}}$ to $\boldsymbol{H}$ as $\boldsymbol{H} = \frac{\tilde{\boldsymbol{H}}^{-1} - \boldsymbol{I}}{\gamma}$ we have the eigenvalues λ_i again.
Sometimes we see this is not the case and we have negative eigenvalues in $\boldsymbol{H}$.
Let $\tilde{\lambda}_i$ be an eigenvalue of $\tilde{\boldsymbol{H}}$ we expect $0 < \Re\tilde{\lambda}_i < 1$ since $\lambda - i > 0$ and $\tilde{\lambda}_i$ is an approximation of some

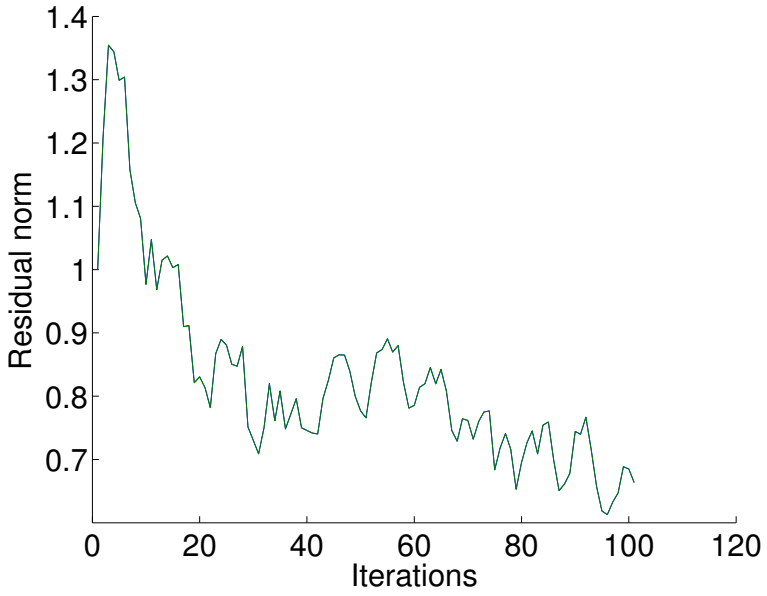

Figure 5.5: Residual calculated using the steady-state GMRES method (green) and by solving the projected ODE (blue), these lines can hardly be distinguished. Simulation is done on test case 1 with $\gamma = 0.5$ and resolution 16.

eigenvalue $\frac{1}{1+\gamma\lambda_i}$. But after transforming we get:

$$\Re\left(\frac{\frac{1}{\tilde{\lambda}_i} - 1}{\gamma}\right) < 0$$

$$\frac{\Re\left(\tilde{\lambda}_i\right)}{\Re\left(\tilde{\lambda}_i\right)^2 + \Im\left(\tilde{\lambda}_i\right)^2} < 1$$

$$\Re\left(\tilde{\lambda}_i\right) < \Re\left(\tilde{\lambda}_i\right)^2 + \Im\left(\tilde{\lambda}_i\right)^2$$

$$|\Im\left(\tilde{\lambda}_i\right)| > \sqrt{\Re\left(\tilde{\lambda}_i\right) - \Re\left(\tilde{\lambda}_i\right)^2}$$

Hence if the imaginary part of $\tilde{\lambda}_i$ is too large in comparison with the real part we get a wrong eigenvalue in $\boldsymbol{H}$ which means $\tilde{\gamma}_i$ is already wrong. This happens often when γ is too large. It also happens that $\tilde{\lambda}_i \approx 0$, which means that $\tilde{\boldsymbol{H}}$ is almost singular which can give errors when transforming back to $\boldsymbol{H}$ and results in very large eigenvalues in $\boldsymbol{H}$. These large eigenvalue are no problem when solving the projected ODE because they have almost no influence on the solution, but they make the system stiff so it is important to use a stiff ODE solver.

5.5.3 Convergence for different ω

So far all our simulations have been focused on a single frequency $\omega = 1$. To find out if the same parameter γ is also efficient for other values of ω we do a quick test with the method based on GMRES to find the frequency domain solution as described in section . We do this test for the frequencies $\omega_j = .85 + 0.03j$ for $j = 0, 1, \ldots, 10$. The simulation stops as the highest residual norm is lower than 10^{-4} and we choose $\gamma = 0.5$ and restart every 100 iterations. In figure 5.6 we can see the results of

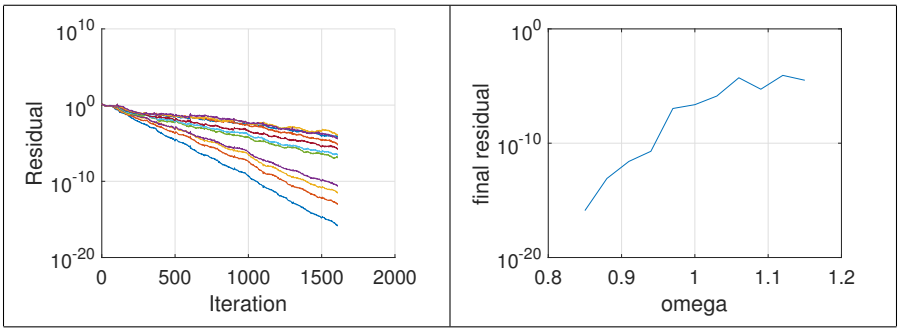

Figure 5.6: *In the image on the left we see the convergence of the residual for all frequencies, here the highest values belong to the highest frequencies. In the image on the right we see the lowest of the residual for every frequency after 1607 iterations.*

Table 5.7: The residual norm for different values of γ for test case 2 with resolution 16 and $\omega = 1$. The value of the residual norm is calculated after 400 iterations. A * means that some negative eigenvalues were found in H, which were replaced by 0. The ** means that too many wrong eigenvalues were replaced which altered the solution too much.

$\frac{\gamma}{T}$	residual norm
$0.1*$	1.63×10^{0}
0.05	1.60×10^{0}
$0.025*$	1.47×10^{0}
0.01	1.54×10^{0}
0.005	1.36×10^{0}
0.0025	1.39×10^{0}
$0.001*$	1.10×10^{0}
$0.0005*$	9.15×10^{-1}
$0.00025**$	7.60×10^{-5}

this simulation. We see that for higher frequencies we need many more iterations than for the lower frequencies. This suggests that it is beneficial to optimize the problem for the highest frequencies that are used. In the same simulation with $\gamma = 0.25$ and $\gamma = 0.1$ we needed respectively 917 and 836 compared to the 1607 needed for $\gamma = 0.5$. For $\gamma = 1$ we had no convergence within 2000 iterations. In all cases the highest frequencies had the worst performance.
It is important to not here that we have rewritten the formulas for $(2\pi\omega i\boldsymbol{I} + \boldsymbol{H})^{-1}\boldsymbol{e}_1$ and the residual in such a way that the inverse of $\tilde{\boldsymbol{H}}$ did not have to be calculated. Furthermore this method does not suffer from positive eigenvalues in $\boldsymbol{H}$. Using these small values of γ might give issues when solving a system of ODEs because we have seen this in other simulations.

5.5.4 Test case 2

For the second test case we will first search for an optimal value of γ, we do this on the grid with resolution 16 and $\omega = 1$. For σ_{max} we choose 20 because this seemed like a proper value in the previous section. The results are listed in table 5.7. These results do not assist us in choosing a proper value of γ, this can be a consequence of the scattering medium or of the bigger domain. To find out which of these two is the case we perform the same simulation on the same domain but now with only one cylinder in the middle like in case 1, the results are in table 5.8. This time we encountered some

Table 5.8: The residual norm for different values of γ for the same domain as test case 2 but with only one cylinder instead of the scattering medium. The resolution is 16 and $\omega = 1$. The value of the residual norm is calculated after 400 iterations. A * means that some negative eigenvalues were found in H, which were replaced by 0. The ** means that too many wrong eigenvalues were replaced which altered the solution too much

$\frac{\gamma}{T}$	residual norm
$0.1*$	1.43×10^{0}
$0.05*$	1.19×10^{0}
$0.025*$	1.35×10^{0}
$0.01*$	1.36×10^{0}
$0.005*$	1.13×10^{0}
0.0025	1.16×10^{0}
$0.001*$	6.19×10^{-1}
$0.0005*$	2.49×10^{-1}
$0.00025**$	2.61×10^{-2}

Table 5.9: Results for test case 2 with resolution 16. The residual is calculated every n_{restart} iterations and the algorithm stops after the residual is lower than 10^{-4}, n_{restarts} is the amount of restart till the algorithm stops.

resolution	n_{restart}	n_{restarts}	residual norm	relative error	$\text{CPU}_{\text{total}}$	CPU_{ODE}
16	400	14	4.88×10^{-5}	$-$	2334.5	1252.0
64	400	12	1.53×10^{-5}	5.76×10^{-3}	3.09×10^{4}	6.21×10^{3}

negative eigenvalues for every $\frac{\gamma}{T}$, but again for $\frac{\gamma}{T} = 0.00025$ we had many more. Now in both cases the convergence is very minimal but for $\frac{\gamma}{T} = 0.0005$ we have seen the best results. This value is also very close to the optimum values found for test case 1, so we continue using this value.
We do not want to use a value of n_{restart} bigger than 400 because this requires too much memory, especially for the case with resolution 64 which we are interested in. In this case our Matrix $\boldsymbol{A}$ has dimension 4498307×4498307, a 4498307×400 matrix to store $\boldsymbol{V}_m$ already needs about 14.4 Gb of memory hence we are reaching limits of our computer here. In 5.9 we see the results for resolution 16 and 64. Another simulation was done with 200 iterations per restart, but without convergence. A minimal amount of about 400 per restart does not seem very beneficial for solving large systems at high resolution because of memory requirements.
The work we have to do double if we want to solve for another frequency is the CPU_{ODE}, this is only 6.21×10^{3} in the case of resolution 64 which is about 3 times as fast as ITR with the biggest time step but then with much higher accuracy.

5.6 Method with time periodic asymptotic solution

Finally we take a look at some results of our last method with the time periodic solution described in section . All results in this sections are made for test case 2 with $\sigma_{\text{max}} = 20$. First we are interested in how far away two solutions with two different frequencies lie from each other. If they lie close to each other we can use this to find results based on the difference of two solutions which can be faster. It

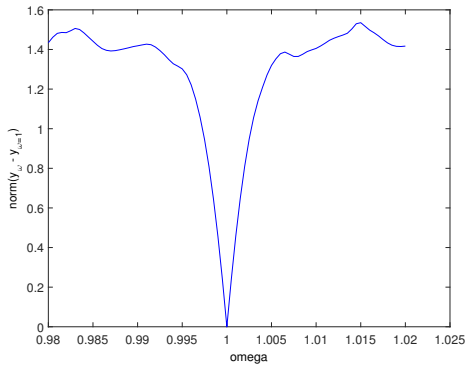

Figure 5.7: Difference of frequency domain solutions calculated for several values of ω compared to the solution for $\omega = 1$.

Table 5.10: Results of GMRES for solving (5.1) for $\omega = 1 + \mathrm{d}\omega$ with the LU decomposition of $(2\pi i\boldsymbol{I} + \boldsymbol{A})$ as a preconditioner. The case $\mathrm{d}\omega = 0$ gives the results for directly solving the system.

$\mathrm{d}\omega$	method	CPU
0	Linear Solver	49.5
0.0001	GMRES	9.6
0.001	GMRES	35.5
0.002	GMRES	66.9

also means we can probably use GMRES to find multiple frequency domain solutions based on one LU decomposition. For figure 5.6 we calculated solutions $\boldsymbol{y}_\omega$ as:

$$\boldsymbol{y}_{st}^{\omega} = (2\pi i\omega\boldsymbol{I} + \boldsymbol{A})^{-1}\boldsymbol{g}_s, \tag{5.1}$$

and calculated the relative error between $\boldsymbol{y}_{st}^{\omega}$ and $\boldsymbol{y}_{st}^{1}$ for $\omega \in [0.85, 1.15]$. These results are made with resolution 16. We observe that these solutions are similar for a very small interval around $\omega = 1$. So we can use this advantage only for a very small neighborhood around ω.
Let us have a look at the performance of GMRES if we use the LU decomposition of $(2\pi i\omega\boldsymbol{I} + \boldsymbol{A})$ as a preconditioner. We set $\omega = 1$ and try to find $\boldsymbol{y}_s^{\omega+\mathrm{d}\omega}$ for resolution 64. In table 5.10 we see that only for very small values of ω we can find solutions quickly using GMRES. For $\mathrm{d}\omega = 0.002$ GMRES is already slower than the backslash operator.
After we have found a frequency domain solution we can find the exact solution of the system:

$$\begin{aligned} \boldsymbol{y}' &= -\boldsymbol{A}\boldsymbol{y} + \sin(2\pi\omega t)\boldsymbol{g}_s, \\ \boldsymbol{y}(0) &= \boldsymbol{0}, \end{aligned} \tag{5.2}$$

as:

$$\boldsymbol{y}(t) = \Im\left(\boldsymbol{y}_s^{\omega} e^{2\pi i\omega t}\right) - \tilde{\boldsymbol{y}}(t) = \hat{\boldsymbol{y}}(t) - \tilde{\boldsymbol{y}}(t),$$

where $\tilde{\boldsymbol{y}}(t) = e^{-\boldsymbol{A}t}\Im(\boldsymbol{y}_{s,\omega})$ is the solution of:

$$\begin{aligned} \tilde{\boldsymbol{y}}' &= -\boldsymbol{A}\tilde{\boldsymbol{y}} + \sin(2\pi\omega t)\boldsymbol{g}_s, \\ \tilde{\boldsymbol{y}}(0) &= \Im(\boldsymbol{y}_s^{\omega}). \end{aligned} \tag{5.3}$$

We can solve the homogeneous system with restarting in time, before we do this we find some efficient parameters for restarting. We have found these to be $t_{\text{restart}} = 1$, $\gamma = 0.01$ and $n_{\text{restart}} = 25$. With these

Table 5.11: The performance of different solvers for solving problem (5.2), or (5.3), we see that Krylov solvers are more efficient at high accuracy.

Method	Relative error	CPU
Krylov homogenous	6.39×10^{-3}	2.53×10^{4}
Krylov inhomogenous	7.08×10^{-3}	4.01×10^{4}
ITR $(\frac{\Delta x}{8})$	6.33×10^{-3}	1.32×10^{5}
ITR (Δx)	4.12×10^{-1}	1.84×10^{4}
Time periodic asymptotic solution	1.60×10^{-2}	4.95×10^{1}
Krylov homogeneous, $\mathrm{d}\omega = 0.001$	$-$	9.75×10^{3}

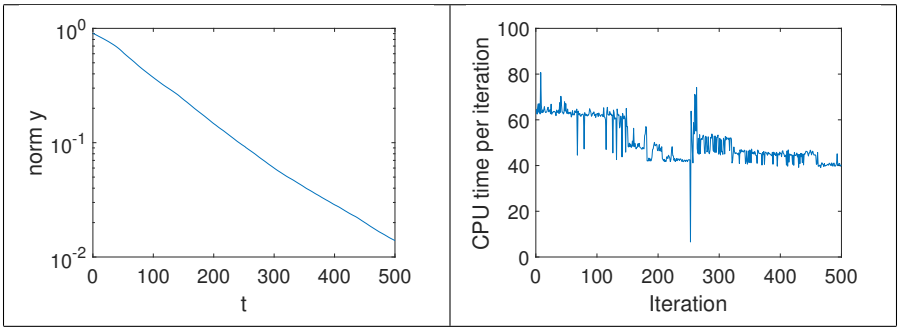

Figure 5.8: *In the left figure we see the norm of $\tilde{\boldsymbol{y}}$ over time by solving equation* (5.3)*. In the figure on the right we see the resulting reduction in CPU time as a consequence of relaxing the tolerance.*

parameters we get the results in table 5.11. Here Krylov homogeneous is just the method as described above where we solve only (5.3). For Krylov inhomogeneous we solve the system (5.2) using a Krylov method. And Krylov difference finds the solution for $\omega = 1.001$ based on the difference of $\boldsymbol{y}_s^1$ and $\boldsymbol{y}_s^{1.001}$ and the solution for $\omega = 1$. Because this difference is small we relax the tolerance a bit and therefore this method is faster. We have used an efficient method here to solve the inhomogeneous problem that uses the fact that $\alpha(t)$ is periodic. It uses restarting in time on intervals $[T_i, T_{i+1}]$ and splits problem (5.2) into two subproblems:

$$\begin{cases} \hat{\boldsymbol{y}}_i' &= -\boldsymbol{A}\hat{\boldsymbol{y}}_i, \\ \hat{\boldsymbol{y}}_i(T_i) &= \boldsymbol{y}(T_i). \end{cases} \qquad \begin{cases} \tilde{\boldsymbol{y}}_i' &= -\boldsymbol{A}\tilde{\boldsymbol{y}}_i + \sin(2\pi\omega t), \\ \tilde{\boldsymbol{y}}_i(T_i) &= \boldsymbol{0}. \end{cases}$$

After this is solved we can calculate $\boldsymbol{y}(T_{i+1}) = \hat{\boldsymbol{y}}_i(T_{i+1}) + \tilde{\boldsymbol{y}}_i(T_{i+1})$. The next step is that we choose $T_i = \frac{i}{\omega}$ such that every time step is exactly one period. In this case the solution of the second problem is always the same $(\tilde{\boldsymbol{y}}_i(T_{i+1}) = \tilde{\boldsymbol{y}}_j(T_{j+1}))$ and we only have to solve the homogeneous part of the problem for every time step.
It seems strange that both methods that solve a homogeneous system with the same parameters perform different. But this is due to the fact that $\tilde{\boldsymbol{y}}$ in (5.3) converges to zero due to the positive eigenvalues of $\boldsymbol{A}$, hence we can relax the tolerance during the simulation. In figure 5.8 we see the reduction in the CPU time for every restart as a result of relaxing the tolerance. We also see how $\tilde{\boldsymbol{y}}$ converges to zero. In (5.11) we observe that Krylov methods perform almost as fast as ITR with $\Delta t = \Delta x$ but with much higher accuracy. Another important thing to notice is that at $T = 500$, $\hat{\boldsymbol{y}}(t)$ is already so close to $\boldsymbol{y}(t)$ that it is more accurate than ITR with $\Delta t = \Delta x$ and also a lot faster, about a factor 400. It is also much faster than Krylov where we have to spend a lot of time for a small increase in accuracy.

Chapter 6

Conclusions

The goal of this research is to investigate if we can efficiently use Krylov methods to simulate scattering media for monochromatic sources to calculate a multi-frequency optical response. This amounts to solving ODE systems of the form

$$\begin{aligned} \boldsymbol{y}' &= -\boldsymbol{A} + \alpha(t)\boldsymbol{g}_s, \\ \boldsymbol{y}(0) &= \boldsymbol{0}, \end{aligned}$$

for multiple functions $\alpha(t)$ efficiently. Here $\boldsymbol{A}$ is a discretization matrix for Maxwell's equations that models a scattering medium and $\boldsymbol{g}_s$ is fixed and is the spatial part of the source. Usually $\alpha(t)$ is an harmonic function or a combination of harmonic functions.
Existing methods make use of the FTDT method to simulate sources with a wide range of frequencies and make a Fourier transform of the response of this source after it interacted with the scattering medium [16]. An disadvantage here is that the accuracy depends on the width of the frequency range, for a higher accuracy multiple runs with a small frequency spectrum and different center frequencies have to be done.
In Krylov exponential time integration the system is projected on a smaller ODE, this projection is done independent of the function $\alpha(t)$. Therefore the idea arised that Krylov methods could be exploited to perform multiple runs for different functions $\alpha(t)$ at relatively low costs.
Three methods are developed to simulate this problem using Krylov methods. We discuss them one by one.

6.1 Krylov subspaces with restarting in dimension

The first one is described in 4.1, it makes one big time step using restarting in dimension which means we can search for all solutions in the same Krylov subspace. The advantage of this method is that all different $\alpha(t)$ can be simulated in the same problem without addition work required for expanding the Krylov bases. The disadvantages are that in the case of a very larger time step a big part of the total CPU time is invested in solving the projected ODE system. While restarting in dimension usually reduces the total amount of time needed to simulate the problem using Krylov methods in this case the total time needed becomes longer with more restarts. Also there seems to be a limit on how small the maximal amount of iterations per restart can be in order to have convergence. This means that in the case of large systems with high resolution a lot of memory is needed for the efficient use of Krylov with restarting in dimension. In the simulation of a scattering medium with the required resolution of 64 we needed a matrix of 14.4 Gb Matrix to store our Krylov basis. The simulation would possibly have been faster using a bigger basis. We observed for another case that in the case of such a big time step we still have convergence independent of the resolution as we would expect from cases with smaller timesteps [2]. But we did not have convergence independent of the maximum amount of restarts. For

bigger restarts we needed a smaller amount of total iterations and a smaller time invested in solving the projected ODE.

6.2 Krylov block subspaces with restarting in time

When we restart our Krylov method in time we get a new initial value problem with an initial value that is not zero. To solve this using a Krylov method we have to use a block method which builds up a Krylov subspace with $\boldsymbol{g}_s$ and the new initial value. When we have multiple inputs $\alpha(t)$ we also have multiple different initial values. In this method we tried to make an SVD truncation of all these initial values. If it was possible to make a very low rank SVD truncation for all t, this would be an efficient method. Sadly this is not the case, in section 5.4 we see that the block size increases fast and because of the extra work required for the block method this also becomes inefficient very fast. Also we see in figure 5.6 that solutions for different ω differ very much when they are close to their time periodic solution. This suggests that it is not possible to make a low rank truncation for large t.

6.3 Krylov subspace with time periodic asymptotic method

For this method we observe that in the case that our input $\alpha(t) = \sin(2\pi\omega t)$ we can write the outcome as the sum of a periodic solution and a solution that converges to zero (the asymptotic part). To divide the solution into these subsolutions we need to solve a linear system, after that the total solution is found by solving a homogeneous system of ODEs.
We observe that if the solutions for different ω are close we can exploit this by solving homogeneous systems for the difference between two solutions with lowered tolerance which is faster. We observe in figure 5.6 that the difference in ω must be small for two solutions to lie close. But if we need two solutions for a small difference in ω we show that we can speed up the algorithm by more than a factor two, and we can then use the same method to speed up the the algorithm for the solutions next to these. In this way we can reduce the total simulation time. Also in the case of one run the homogeneous solver is much faster than the solver for the inhomogeneous system. It performs almost as fast as ITR with time step $\Delta t = \Delta x$ but with higher accuracy. For the same accuracy ITR needs almost 5 times as long. With only the periodic part of the solution a higher accuracy was reached than using ITR with the biggest time step, but the linear solver is about 400 times as fast. Krylov decreased the error of the periodic solution by about 4 by solving the homogeneous system, this is only a small profit compared to the amount of work needed. The part that solving the homogeneous system adds in accuracy becomes smaller with the final time. This is because the homogeneous system converges to zero.
The biggest disadvantage of this method is that a single run has to be performed for every frequency. These runs can be sped up if the solutions of two frequencies lie close.

6.4 General conclusions and recommendations

One disadvantage of Krylov methods in general is the amount of tuning needed before they work properly. It takes effort to find optimal values for γ in SAI, the amount of iterations per restart and the time step.
Of all suggested methods both the method with restarting in dimension 5.5 and the method that uses the time periodic asymptotic solution 5.6 show some promise. They are both slower than ITR with the biggest time step for the first value of ω but are more accurate. When a solution for a second value of ω needs to be calculated both methods are faster than ITR with the biggest time step. The most important results are summarized in table 6.1.
For the method with restarting in dimension to work efficiently even more tuning is needed. Currently

Table 6.1: Most important results of the Krylov subspace method with restarting in dimension and the time periodic asymptotic method.

Method	Error	CPU time	Gain factor	Notes
ITR $\left(\frac{\Delta x}{8}\right)$	6.33×10^{-3}	1.32×10^{5}	1	Reference method
Krylov homog.	6.39×10^{-3}	2.53×10^{4}	5.22	
Krylov homog. next ω	$-$	9.75×10^{3}	13.52	Efficient as long as the next ω is close
Krylov rest. dim.	5.76×10^{-3}	3.09×10^{4}	4.27	
Krylov rest. dim. next ω	$-$	6.21×10^{3}	21.26	As long as the next ω is smaller

the ODE solver takes too much time. After restarting the residual looks periodic, possibly Fourier analysis of the residual and analytical solutions could help to make the solver more efficient. Also variable values of γ for restart could be tried ([5], [4], [3]). And the residual functions should be studies in more detail.
A solver similar to GMRES which makes use of SAI is used to solve multiple frequency domain solutions at once. This solver can be used to find an optimal value for γ for Krylov solver for exponential time integration when it uses a large time step. In figure 5.5.1 we see that the convergence is similar for both methods. The results in 5.6 suggest that it is best to optimize the Krylov subspace methods for the highest frequency. It seems that for higher frequencies lower values of γ are more efficient and more iterations are needed.
In the method with the time periodic asymptotic solution depending on the required accuracy it can be that the homogeneous problem does not have to be solved. If the asymptotic part of the solution converges quickly to the periodic part, solving only the periodic part is sufficient. The problem here is that we do not know how quickly the asymptotic time periodic solution converges to the time periodic solution, so it is useful to find a quick way to estimate this difference. Possibly it is enough to solve the same system on a lower resolution to get a proper estimate of the convergence but this should be investigated.
A big advantage of Krylov over ITR is that we can control the error by controlling the residual. We can use this to our advantage if our solution converges to zero because in this case we can relax the tolerance a bit throughout the simulation so that it becomes faster.
ITR is still a bit faster if accuracy is not required but since the the spacial resolution is 64 and the spatial error is about $\Delta x^2 \approx 2 \times 10^{-4}$ the accuracy added by Krylov does not seem unnecessary.

Bibliography

[1] M.A Botchev. A block Krylov subspace time-exact solution method for linear ordinary differential equation systems. *Numerical linear algebra with applications*, 20(4):557–574, 2013.

[2] M.A. Botchev. Krylov subspace exponential time domain solution of Maxwells equations in photonic crystal modeling. *Journal of computational and applied mathematics*, 293:20–34, 2016.

[3] V. Druskin, C. Lieberman, and M. Zaslavsky. On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. *SIAM Journal on Scientific Computing*, 32(5):2485–2496, 2010.

[4] V. Druskin and V. Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems & Control Letters*, 60(8):546–560, 2011.

[5] S. Güttel and L. Knizhnerman. A black-box rational Arnoldi variant for Cauchy-Stieltjes matrix functions. *BIT Numerical Mathematics*, 53(3):595–616, 2013.

[6] W. Hundsdorfer and J.G Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33. Springer Science & Business Media, 2013.

[7] J.D. Joannopoulos, S.G. Johnson, J.N. Winn, and R.D. Meade. *Photonic crystals: molding the flow of light*. Princeton university press, 2011.

[8] S.G. Johnson. Notes on perfectly matched layers (PMLs). *Lecture notes, Massachusetts Institute of Technology, Massachusetts*, 29, 2008.

[9] B. Judkewitz, R. Horstmeyer, I.M. Vellekoop, I.N. Papadopoulos, and C. Yang. Translation correlations in anisotropically scattering media. *Nature physics*, 11(8):684–689, 2015.

[10] E. Larose, L. Margerin, A. Derode, B. van Tiggelen, M. Campillo, N. Shapiro, A. Paul, L. Stehly, and M. Tanter. Correlation of random wavefields: An interdisciplinary review. *Geophysics*, 71(4):SI11–SI21, 2006.

[11] Y. Liu, P. Lai, C. Ma, X. Xu, A.A. Grabar, and L.V. Wang. Optical focusing deep inside dynamic scattering media with near-infrared time-reversed ultrasonically encoded (true) light. *Nature communications*, 6, 2015.

[12] Z. Merali. Super vision: using techniques adapted from astronomy, physicists are finding ways to see through opaque materials such as living tissue. *Nature*, 518(7538):158–161, 2015.

[13] A.P. Mosk, A. Lagendijk, G. Lerosey, and M. Fink. Controlling waves in space and time for imaging and focusing in complex media. *Nature photonics*, 6(5):283–292, 2012.

[14] M. Mounaix, D. Andreoli, H. Defienne, G. Volpe, O. Katz, S. Grésillon, and S. Gigan. Spatiotemporal coherent control of light through a multiple scattering medium with the multispectral transmission matrix. *Physical review letters*, 116(25):253901, 2016.

[15] A.F. Oskooi, C. Kottke, and S.G. Johnson. Accurate finite-difference time-domain simulation of anisotropic media by subpixel smoothing. *Optics letters*, 34(18):2778–2780, 2009.

[16] A.F. Oskooi, D. Roundy, M. Ibanescu, P. Bermel, J.D. Joannopoulos, and S.G. Johnson. Meep: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications*, 181(3):687–702, 2010.

[17] B. Redding, S.F. Liew, R. Sarma, and H. Cao. Compact spectrometer based on a disordered photonic chip. *Nature Photonics*, 7(9):746–751, 2013.

[18] A. Taflove and S.C. Hagness. *Computational electrodynamics*. Artech house, 2005.

[19] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM Journal on Scientific Computing*, 27(4):1438–1457, 2006.

[20] M.C.W. van van Rossum and T.M. Nieuwenhuizen. Multiple scattering of classical waves: microscopy, mesoscopy, and diffusion. *Reviews of Modern Physics*, 71(1):313, 1999.

[21] I.M. Vellekoop and A.P. Mosk. Universal optimal transmission of light through disordered materials. *Physical review letters*, 101(12):120601, 2008.

[22] J.G. Verwer and M.A. Botchev. Unconditionally stable integration of Maxwell's equations. *Linear Algebra and its Applications*, 431(3-4):300–317, 2009.

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