MASTER

Preconditioning dense complex linear systems from a VIM discretization

Belachew, Melisew T.

Award date:
2009

Preconditioning Dense Complex Linear Systems from a VIM Discretization

A master's thesis submitted to the Department of Mathematics for the partial fulfillment of the requirements for the degree of MSc in Industrial and Applied Mathematics specializing in Computational Science and Engineering.

Melisew Tefera Belachew

Supervisor : Dr. Joseph M.L. Maubach

August 2009

Eindhoven, the Netherlands.

Table of Contents

List of Figures

List of Tables

Acknowledgement

It would be meaningless to start acknowledging people before I thank the almighty God for making everything possible.

My next heartfelt thanks go to my supervisor Dr. Joseph M. L. Maubach. If his continuous support, encouragement, valuable advices and constructive comments were not there for me this thesis would not have this shape at all. Jos deserves a big thank you for making matters easier.

I also wish to thank the experts who provided support; Dr. M.E. Hochstenbach (**TU/e**), and the **ASML** crew Dr. Wim Coene, Dr. Irwan Setija, Ir. Mark van Kraaij, and Dr. Martijn van Beurden (**TU/e - ASML**). The biweekly meetings and discussions with the **ASML** group played a great role in broadening the thesis and incorporating new ideas. My special thanks go to Ir. Mark van Kraaij for his quick responses to my questions and for providing help whenever needed.

I would like to pass my notable gratitude to my mom, my sister and my brothers for their continuous motivation and encouragement. My dear friends also deserve a thank you for being there when I needed them.

It is worth to mention my appreciation to all the staff members at the department of Mathematics at **TU/e** for their quality education and kind services.

I would like to extend my gratitude to the international office (**STU**) for making my stay enjoyable and also for contributing to my study in one way or another.

Finally yet importantly, I am grateful to the company **ASML** for the financial support throughout the entire study and for giving me the opportunity to do my master's thesis on one of their prestigious research projects.

The company, ASML

ASML is the world's leading provider of lithography systems for the semiconductor industry, manufacturing complex machines that are critical to the production of integrated circuits (IC) or microchips. Headquartered in Veldhoven, the Netherlands, **ASML** designs, develops, integrates, markets and services these advanced systems, which continue to help their customers - the major chip makers - reduce the size and increase the functionality of microchips, and consumer electronic equipment.

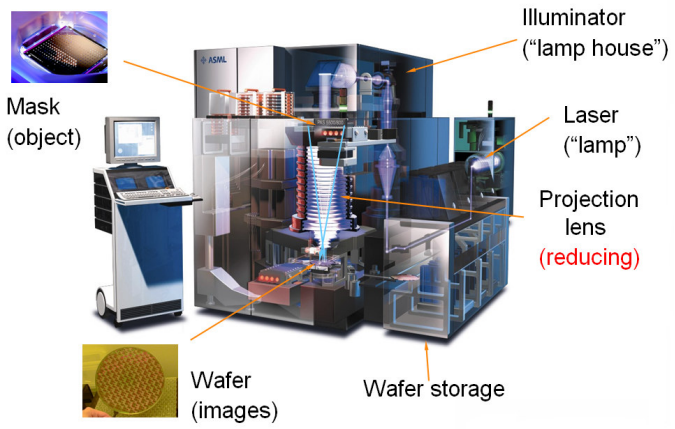
ASML's TWINSCAN

Figure 1: **ASML**'s TWINSCAN

The TWINSCAN NXT:1950i Step-and-Scan system is a high productivity, dual stage immersion lithography tool designed for volume production 300-mm wafers at the 32-nm node and beyond.

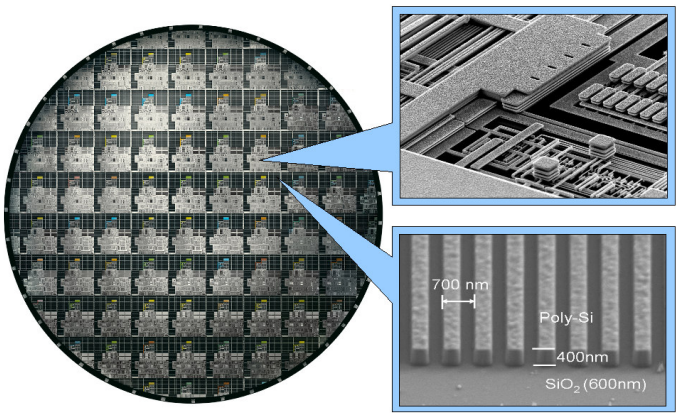
Advanced in-situ metrology per wafer together with a comprehensive set of options to seamlessly input off-line metrology data to the scanner enable maximum overlay, focus and CDU performances on product wafers.

Microchips are built using several layers of connected wires which have to fit precisely on top of each other. To achieve this, one must very accurately determine the position and dimension of the wires before one builds a new layer on top. If two pieces of a wire connect perfectly (over a 100% of their intended contact area) the overlay is called perfect. This thesis is concerned with part of the mathematics related to the determination of the position and dimension of a wire piece.



(a)

Figure 2: ASML's wafer stepper



(a)

Figure 3: Wafer, chip and grating respectively

Chapter 1

Introduction

Gratings (i.e., optical device consisting of a surface with many parallel grooves on it; disperses a beam of light into its wavelengths to produce its spectrum) are widely used in lithography for various metrology tasks such as alignment, overlay metrology and CD metrology. In general lithographic methods produce in general symmetric profiles with sinusoidal, rectangular or trapezoidal grooves. However, for very high and very low groove frequencies the real profiles have much more complicated forms. Moreover, in the far-infrared of wavelength metal can be assumed with infinite conductivity, but in the visible region and for shorter wavelengths the finite conductivity complicates the grating response and requires different theoretical methods. An example of a non-trivial grating can been seen in Figure (1.1) below. Diffraction gratings are categorized according to several criteria: geometry, material, efficiency behavior, manufacturing method or usage. The most well known gratings are reflection and transmission gratings:

- *Transmission gratings* can have various groove shapes: triangular, rectangular or sinusoidal. The material is transparent and the back face needs an anti-reflection coating to prevent light losses due to reflection or to prevent multiple scattering effects inside the substrate. While most transmission gratings have a use limited to the visible spectrum, it is possible to extend their performance into the ultraviolet and infrared with choice of appropriate materials. These gratings are frequently used as beam splitters.

- *Reflection gratings* are usually surface relief gratings, covered with some highly reflecting material. Depending on the spectral region, reflectivity can vary significantly, so that the choice of material is critical for grating performance. In particular, gratings that are reflective in one spectral region, can become transmittive in other regions. Its advantage over a transmission grating is that it produces a spectrum extending from ultraviolet to infrared, since the light does not pass through the grating material. Because of the wide spectral band coverage and minimal order overlap, most of the reflection gratings are used in first order.

Angular-resolved optical scatterometry is a new promising technology for metrology in near-future lithography. In so-called critical-dimension (CD) metrology, the characteristics of a pattern written in the resist layer are determined, yielding a parametrized shape of the resist lines, with parameters as width, height and slope of these lines. Typically, test-patterns with 1D periodic gratings are used. In an iterative reconstruction loop, the parameterized shape is retrieved via a least-squares fitting procedure, hereby

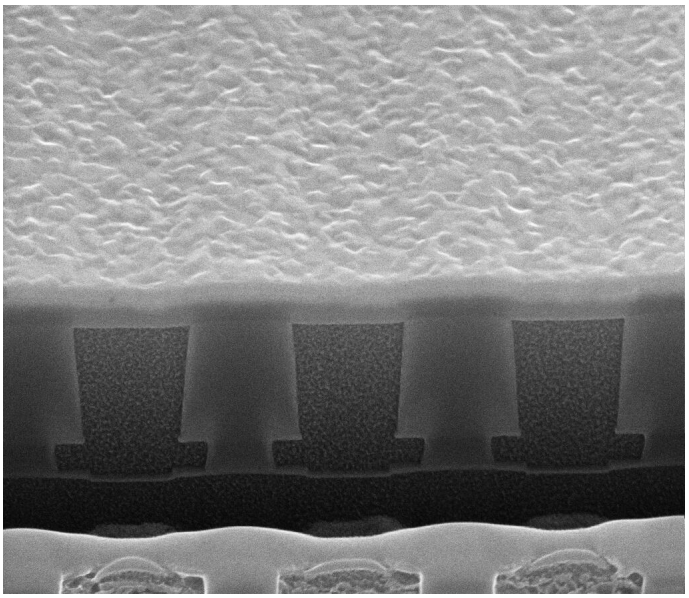


Figure 1.1: Cross-section of a complex layered grating.

fitting the experimental diffraction image to a simulated diffraction image that has been computed for a given parameter set. For 1D periodic gratings, scatterometry is a well developed technique reaching the high throughput demands of less than a second per target. In addition to the regular 1D gratings, current CD metrology needs also functionality for characterization of ellipsoidal contact holes, and brickwall structures of rectangular features in DRAM, arranged on a periodic 2D lattice. Unfortunately, the number of harmonics which parameterize the electric filed increases from P for 1D to P^2 for 2D. The "widely" used Maxwell solver RCWA has a computational complexity that scales asymptotically with the third power of P (and P^2). This leads to unacceptably large computational times that are incompatible with the required high throughput.

To avoid the large execution time of RCWA for 2D problems, different discretization of the Maxwell's equations have been examined; the finite element method (FEM) by Joseph Maubach, Pisarenco and Rook, and the volume integral method (VIM) by M. V. Beurden. Both FEM and VIM lead to linear systems of equations such that the matrix vector multiplication is optimally fast ($\mathcal{O}(N)$ or $\mathcal{O}(N\log N)$ time, N the number of degrees of freedom) and both methods use iterative solvers to determine the solution of the linear systems.

The parameterized linear systems of equations are the result of a discretization of the 3D Maxwell equations applied for the solution of an optical diffraction problem for a (double) periodic grating. The particular discretization of interest determines how the linear systems depend on the parameters (geometry, material parameters, wave length, and angles of incidence).

This thesis focuses on the VIM method. The VIM discretization leads to a dense complex linear system for the electric field for a 2D-periodic grating which is solved in an iterative way. This method is a factor of 20 faster for a typical 2D-periodic application on resist than the RCWA method.

This thesis is aimed at designing an optimized preconditioner for speeding-up the iterative solution of the linear system obtained from VIM discretization. Ideally, the spectrum of the total system of the preconditioner and the VIM coefficient matrix is concentrated around one and should be independent of the physical parameters, such that the number of iterations is stable under parameter changes. Further, the construction of the preconditioner should have low computational complexity.

The actual purpose of solving the linear systems is to reconstruct the specific parameter values which led to a measured result. Whenever the nonlinear reconstruction algorithm converges, several linear systems for almost identical parameter values have to be solved.

For given parameters $\boldsymbol{c}$, the VIM discretization leads to a linear system $\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c})$, where $\boldsymbol{A}(\boldsymbol{c}) = \boldsymbol{C}(\boldsymbol{c}) - \boldsymbol{G}(\boldsymbol{c}) \cdot \boldsymbol{M}(\boldsymbol{c}),\ \boldsymbol{C}(\boldsymbol{c}), \boldsymbol{G}(\boldsymbol{c}), \boldsymbol{M}(\boldsymbol{c}) \in \mathbb{C}^{N\times N}$. Matrix $\boldsymbol{A}$ is full (almost all the entries are non-zero), but a matrix vector multiplication $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ can be calculated in almost linear time (linear proportional to $N\log N$).

The two main objectives of this thesis are:

1. To search for an iterative solution method and a preconditioner which minimizes the number of iterations and the overall computational time, preferably for a wide range of parameters $\boldsymbol{c}$.

2. To determine (if possible) a better suitable initial guess than the zero vector for the case that $\boldsymbol{A}(\boldsymbol{c}+\Delta\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}+\Delta\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c}+\Delta\boldsymbol{c})$ has to be solved after $\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c})$ has been solved;

This thesis describes several possible preconditioning techniques, links the construction of an estimate for $\boldsymbol{x}(\boldsymbol{c} + \Delta c)$ to derivatives and presents a reconstruction algorithm with an implementation.

The remainder of this thesis is organized as follows. Chapter (2) is devoted to the differential formulation of the Maxwell's equations for the optical diffraction problem. Though the actual discretization method in Chapter (4) uses the equivalent integral equation formulation, we present the differential formulation because it is more accessible to understand, it serves its purpose to introduce all relevant physical parameters and it is the basis for at least two competing discretization methods examined by the Scientific computing group inside **CASA**: the rigorously coupled wave analysis (RCWA) and the finite element method (FEM) . Sliced grating structure and Fourier representation of the electric field are discussed in Chapter (3). The discretization method VIM uses layer-wise decomposition of the domain and representation of the electric field. A brief description of the VIM discretization, structure and properties of the coefficient matrix are presented in Chapter (4). The applicability of the iterative solvers and related preconditioners for the solution of the linear systems depends on the properties of the coefficient matrix of the discretized system and those of the preconditioner. Chapter (5) and (6) present the range of applicable preconditioners. Because the coefficient matrix under consideration does not satisfy any desirable properties required for the construction of fast preconditioners such as multi-grid ones we take our resort to preconditioners which do not require such properties. Chapter (5) introduces iterative methods and preconditioning techniques. The first part of Chapter (6) is devoted to sparse approximate inverses based on Frobenius norm minimization method while the rest of the chapter discusses preconditioners based on incomplete LU factorizations combined with dropping strategy. Numerical results

and their interpretations are given in detail in Chapter (7). Chapter (8) contains a brief summary of the conclusions obtained from the numerical experiments, and also some recommendations are proposed. Chapter (9) presents the future work and further research areas. In Appendix (A) the description of the non-linear reconstruction problem is given. As an example, subsection (A.3.2) presents few simple boundary value problems. The determination of a good initial guess for $\boldsymbol{x}(\boldsymbol{c} + \Delta\boldsymbol{c})$ is given in Appendix (B).

Chapter 2

The Physical Model

In section (2.1) a set of equations is given, including Maxwell's equations, the continuity equation and the constitutive relations. Under certain circumstances the Helmholtz equation can be derived for the electric and magnetic fields. In section (2.2) a one-dimensional and two-dimensional periodic grating is defined. The one-dimensional periodic grating gives rise to two diffraction cases that are explained separately: planar diffraction and conical diffraction. The grating equations and grating boundary conditions are derived in section (2.3) and (2.4) respectively.

2.1 Maxwell's equations

In a medium with magnetic displacement $\boldsymbol{B}$, electric displacement $\boldsymbol{D}$, current density $\boldsymbol{J}$ and charge density ρ, Maxwell's equations for the electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{H}$ can be formulated as:

$$\nabla \times \boldsymbol{E} = -\frac{\partial}{\partial t}\boldsymbol{B}, \qquad \nabla \times \boldsymbol{H} = \boldsymbol{J} + \frac{\partial}{\partial t}\boldsymbol{D}, \qquad \nabla \cdot \boldsymbol{D} = \rho, \qquad \nabla \cdot \boldsymbol{B} = 0. \tag{2.1}$$

The continuity equation is given by

$$\nabla \cdot \boldsymbol{J} = -\frac{\partial}{\partial t}\rho. \tag{2.2}$$

The additional constraints needed to solve (2.1) and (2.2) are the constitutive relations. These relations characterize materials on a macroscopic level and are described in terms of three scalar quantities, the permittivity ε, the permeability μ and the conductivity σ. Assuming that no primary or external sources are present and that the medium is linear, isotropic (i.e., the physical properties of the medium at each point are independent of the direction) and time-invariant, then:

$$\boldsymbol{D} = \varepsilon \boldsymbol{E}, \qquad \boldsymbol{B} = \mu \boldsymbol{H}, \qquad \boldsymbol{J} = \sigma \boldsymbol{E}. \tag{2.3}$$

Note that the three scalar quantities are real if the medium is dispersion-free (i.e., if the response of the medium is instantaneous). If on the other hand the medium is dispersive,

the scalar quantities are complex. Looking at the electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{H}$, Maxwell's equations become:

$$\nabla \times \boldsymbol{E} = -\mu \frac{\partial}{\partial t} \boldsymbol{H}, \tag{2.4}$$

$$\nabla \times \boldsymbol{H} = \varepsilon \frac{\partial}{\partial t} \boldsymbol{E} + \sigma \boldsymbol{E}. \tag{2.5}$$

From now on only time-harmonic field quantities are considered and in that case the complex field notation $\boldsymbol{E}(\boldsymbol{x})e^{j\omega t}$ and $\boldsymbol{H}(\boldsymbol{x})e^{j\omega t}$ can be used for the electric and magnetic field. Combining the permittivity ε with the conductivity σ gives a new complex permittivity $\tilde{\varepsilon} = \varepsilon - j\sigma/\omega$ so that Maxwell's equations become:

$$\nabla \times \boldsymbol{E} = -j\omega\mu \boldsymbol{H}, \tag{2.6}$$

$$\nabla \times \boldsymbol{H} = j\omega\tilde{\varepsilon} \boldsymbol{E}. \tag{2.7}$$

From Maxwell's equations (2.6) and (2.7) a relation can be derived for the electric field. The curl of (2.6), combined with (2.7) and the identities

$$\nabla \times \nabla \times \boldsymbol{E} = \nabla(\nabla \cdot \boldsymbol{E}) - \nabla^2 \boldsymbol{E}$$

and

$$\nabla \times (\mu \boldsymbol{H}) = \mu \nabla \times \boldsymbol{H} + \nabla \mu \times \boldsymbol{H}$$

result in:

$$\nabla(\nabla \cdot \boldsymbol{E}) - \nabla^2 \boldsymbol{E} = \omega^2 \mu \tilde{\varepsilon} \boldsymbol{E} - j\omega \nabla \mu \times \boldsymbol{H}. \tag{2.8}$$

For a homogeneous medium the complex permittivity $\tilde{\varepsilon}$ and permeability μ are both constant and $\nabla\mu = 0$. The divergence of (2.7) and the identity $\nabla \cdot (\nabla \times \boldsymbol{H}) = 0$ show the electric field is divergence free. In this case the electric field $\boldsymbol{E}$ based on (2.8) satisfies a Helmholtz equation:

$$\nabla^2 \boldsymbol{E} + \omega^2 \mu \tilde{\varepsilon} \boldsymbol{E} = \boldsymbol{0}. \tag{2.9}$$

In this situation, the magnetic field $\boldsymbol{H}$ also satisfies a Helmholtz equation:

$$\nabla^2 \boldsymbol{H} + \omega^2 \mu \tilde{\varepsilon} \boldsymbol{H} = \boldsymbol{0}. \tag{2.10}$$

2.2 Grating definition

2.2.1 One-dimensional periodic gratings

In order to derive the boundary conditions, Maxwell's equations need to be solved in a domain. Therefore a one-dimensional periodic grating in $\mathbb{R}^3$ is considered in Figure (2.1). In the x-direction the periodic grating continues infinitely and has period Λ. The y-direction is parallel to the grating grooves which are again infinitely long. Above and below the grating there are two infinite halfspaces in the z-direction. So the grating profile

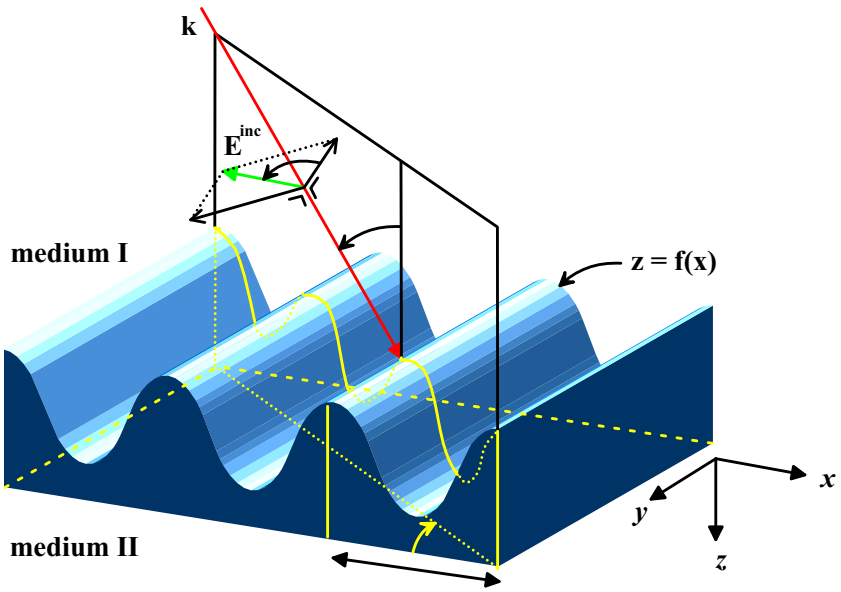


Figure 2.1: One-dimensional periodic grating in $\mathbb{R}^3$.

in $\mathbb{R}^3$ can be described by a function $z = f(x)$ such that $f(x + n\Lambda) = f(x)$ for $x \in \mathbb{R}$ and $n \in \mathbb{Z}$. If the grating profile cannot be described by a function, a parametrization can be used. Since the periodic grating is infinitely long in the x-direction, the domain of interest can be restricted to only one period with length Λ. Furthermore, the grating is invariant in the y-direction. Above the grating lies the upper halfspace for $z < 0$ and below the grating lies the lower halfspace for $z > D$. Under these circumstances the original 3-dimensional grating can be reduced to a 2-dimensional subdomain of the xz-plane where the grating profile is described by $z = f(x)$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$. In Figure (2.1)

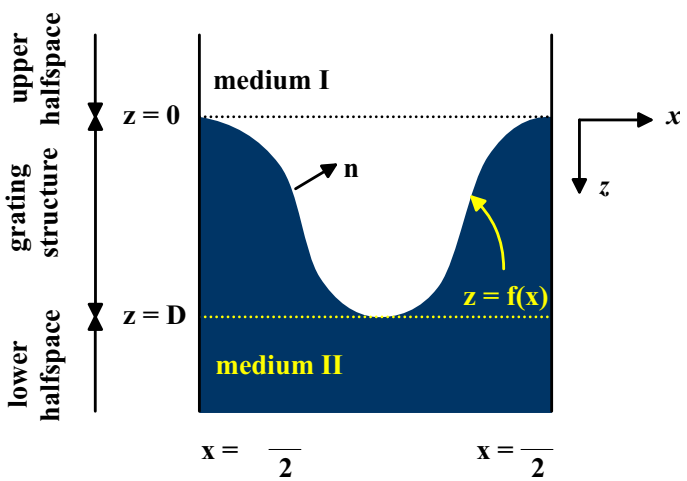


Figure 2.2: Reduced one-dimensional grating in $[-\frac{\Lambda}{2}, \frac{\Lambda}{2}] \times (-\infty, \infty)$.

the incident plane wave is also shown which is assumed to be linearly polarized. The plane wave is incident on the grating at a polar angle θ and azimuthal angle ϕ. Furthermore, ψ is the angle between the incident electric field $\boldsymbol{E}^{\text{inc}}$ and the plane of incidence. In the case of planar diffraction ($\phi = 0$) the incident polarization can be decomposed into a

transverse electric (TE) polarization part ($\psi = \pi/2$, electric field is oscillating normal to the plane of incidence) and a transverse magnetic (TM) polarization part ($\psi = 0$, electric field is oscillating in the plane of incidence) which are handled independently. Here all the transmitted and reflected orders lie in the same plane (the plane of incidence, the xz-plane). For the general three-dimensional problem or conical diffraction ($\phi \neq 0$) two uncoupled equations for the x-component of the electric and magnetic field are solved. Here the transmitted and reflected orders lie on the surface of a cone. Note that in the case of planar diffraction all electromagnetic field components are independent of the y-coordinate while in a conical mount these field components still depend on the y-coordinate.

Two-dimensional periodic gratings

In order to derive the boundary conditions, a domain is defined where Maxwell's equations are solved. Therefore a two-dimensional periodic grating in $\mathbb{R}^3$ is considered in Figure (2.3). In both the x- and y- directions the periodic grating continues infinitely and has period Λ_x and Λ_y respectively. Above and below the grating there are two infinite halfspaces in the z-direction. So, the grating profile in $\mathbb{R}^3$ can be described by a function $z = f(x, y)$ such that $f(x + n\Lambda_x, y + m\Lambda_y) = f(x, y)$ for $x, y \in \mathbb{R}$ and $n, m \in \mathbb{Z}$. If the grating profile cannot be described by a function, a parametrization can be used.

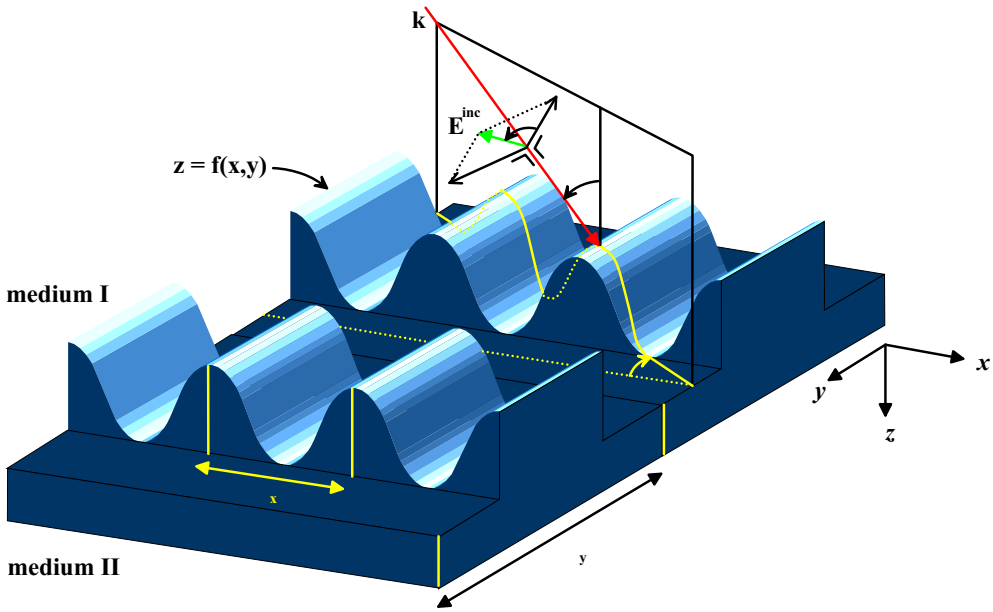


Figure 2.3: Two-dimensional periodic grating in $\mathbb{R}^3$.

Since the periodic grating is infinitely long in both the x- and y-direction, the domain of interest can be restricted to only one period in each direction. Above the grating lies the upper halfspace for $z < 0$ and below the grating lies the lower halfspace for $z > D$. This means that the original 3-dimensional grating can be reduced to a 3-dimensional subdomain of $\mathbb{R}^3$ where the grating profile is described by $z = f(x, y)$ for $-\frac{\Lambda_x}{2} \leq x \leq \frac{\Lambda_x}{2}$ and $-\frac{\Lambda_y}{2} \leq y \leq \frac{\Lambda_y}{2}$.

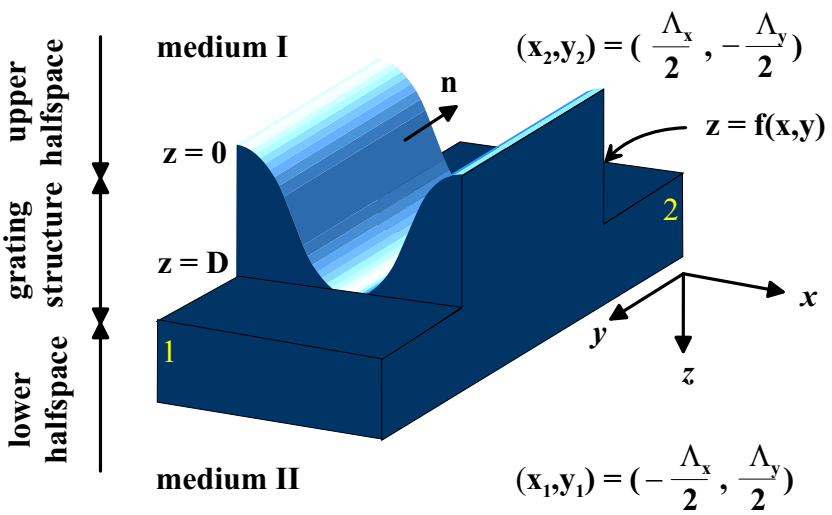


Figure 2.4: Reduced two-dimensional grating in $[-\frac{\Lambda_x}{2}, \frac{\Lambda_x}{2}] \times [-\frac{\Lambda_y}{2}, \frac{\Lambda_y}{2}] \times (-\infty, \infty)$.

In Figure (2.3) the incident plane wave is also shown which is assumed to be linearly polarized. The plane wave is incident on the grating at a polar angle θ and azimuthal angle ϕ. Furthermore, ψ is the angle between the incident electric field vector and the plane of incidence. For two-dimensional periodic gratings the perpendicular and parallel components of the electric and magnetic fields are always coupled and must be obtained simultaneously. Note that for two-dimensional periodic gratings all the electromagnetic field components depend on all the three coordinates.

2.3 Grating equations for 1D planar diffraction

The previous section explained how the grating structure could be reduced to only one period. This means that for one-dimensional periodic gratings in Maxwell's equations (2.6) and (2.7) the permeability is given by $\mu = \mu(x, z)$ and the complex permittivity by $\tilde{\varepsilon} = \tilde{\varepsilon}(x, z)$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z \in \mathbb{R}$. Similarly, for two-dimensional periodic gratings the permeability is given by $\mu = \mu(x, y, z)$ and the complex permittivity by $\tilde{\varepsilon} = \tilde{\varepsilon}(x, y, z)$ for $-\frac{\Lambda_x}{2} \leq x \leq \frac{\Lambda_x}{2}$, $-\frac{\Lambda_y}{2} \leq y \leq \frac{\Lambda_y}{2}$ and $z \in \mathbb{R}$.

There are three different cases of interest, Transverse Electric (TE) Polarization, Transverse Magnetic (TM) Polarization and Conical diffraction. For the sake of brevity, only the first case is described below (2.3.1).

2.3.1 Transverse electric (TE) polarization

For TE polarization the electric field has only a y component that depends on the x and z coordinates, so $\boldsymbol{E} = E_y(x, z)\boldsymbol{e}_y$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z \in \mathbb{R}$. Substituting this

in Maxwell's equations (2.6) and (2.7) results in the following set of equations for the electric and magnetic field:

$$\frac{\partial}{\partial z}E_y(x,z) = j\omega\mu(x,z)H_x(x,z), \tag{2.11a}$$

$$\frac{\partial}{\partial x}E_y(x,z) = -j\omega\mu(x,z)H_z(x,z), \tag{2.11b}$$

$$\frac{\partial}{\partial z}H_x(x,z) = j\omega\tilde{\varepsilon}(x,z)E_y(x,z) + \frac{\partial}{\partial x}H_z(x,z). \tag{2.11c}$$

The equations above can be rewritten into one second order differential equation for the electric field component $E_y(x,z)$. Substituting (2.11a) and (2.11b) into (2.11c) gives:

$$\frac{\partial}{\partial z}\left(\frac{1}{\mu(x,z)}\frac{\partial}{\partial z}E_y(x,z)\right) = -\omega^2\tilde{\varepsilon}(x,z)E_y(x,z) - \frac{\partial}{\partial x}\left(\frac{1}{\mu(x,z)}\frac{\partial}{\partial x}E_y(x,z)\right) \tag{2.12}$$

2.4 Grating boundary conditions for 1D periodic gratings

For one-dimensional periodic gratings the permittivity $\varepsilon = \varepsilon(x,z)$ can be split up into two parts so that in medium I the permittivity is given by $\varepsilon_I = \varepsilon_I(x,z)$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z < f(x)$ and in medium II by $\varepsilon_{II} = \varepsilon_{II}(x,z)$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z > f(x)$. A similar expression holds for the permeability, conductivity, electric field and magnetic field. Along the grating surface $z = f(x)$ where the two different media I and II intersect, the so called boundary jump conditions can be derived from equations (2.1). The normal unit vector $\boldsymbol{n}$ is pointing from medium II towards medium I. If the grating surface current density is denoted by $\boldsymbol{J}_s$ and the grating surface charge density by ρ_s, then the following relations hold for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z = f(x)$:

$$\boldsymbol{n} \times (\boldsymbol{E}_I - \boldsymbol{E}_{II}) = \boldsymbol{0}, \tag{2.13}$$

$$\boldsymbol{n} \times (\boldsymbol{H}_I - \boldsymbol{H}_{II}) = \boldsymbol{J}_s, \tag{2.14}$$

$$\boldsymbol{n} \cdot (\boldsymbol{J}_I - \boldsymbol{J}_{II}) = -j\omega\rho_s, \tag{2.15}$$

$$\boldsymbol{n} \cdot (\boldsymbol{D}_I - \boldsymbol{D}_{II}) = \rho_s, \tag{2.16}$$

$$\boldsymbol{n} \cdot (\boldsymbol{B}_I - \boldsymbol{B}_{II}) = 0. \tag{2.17}$$

For materials with finite conductivity, no grating surface current density and grating surface charge density exists, so $\boldsymbol{J}_s = \boldsymbol{0}$ and $\rho_s = 0$. Furthermore, for time-harmonic field quantities the boundary conditions (2.16) and (2.17) do not provide any new information. Since only the electric and magnetic fields are of interest and not the charge density which follows from the continuity equation, only the first two boundary jump conditions remain. So for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z = f(x)$:

$$\boldsymbol{n} \times (\boldsymbol{E}_I - \boldsymbol{E}_{II}) = \boldsymbol{0}, \qquad \boldsymbol{n} \times (\boldsymbol{H}_I - \boldsymbol{H}_{II}) = \boldsymbol{0}. \tag{2.18}$$

Furthermore, the so-called pseudo-periodic boundary condition is applied to the electric and magnetic fields which introduces a phase-correction originating from the incident field. The wave number is defined by $k_0 = 2\pi/\lambda_0$ with λ_0 the wavelength of the incident light. Together with $n_I = \sqrt{\mu_I \tilde{\varepsilon}_I}/\sqrt{\mu_0 \varepsilon_0}$, the relative refraction index of medium I, the pseudo-periodic boundary condition becomes for $z \in \mathbb{R}$:

$$\boldsymbol{E}(\Lambda/2, z) = e^{-jk_0 n_I \Lambda \sin\theta \cos\phi} \boldsymbol{E}(-\Lambda/2, z), \tag{2.19}$$

$$\boldsymbol{H}(\Lambda/2, z) = e^{-jk_0 n_I \Lambda \sin\theta \cos\phi} \boldsymbol{H}(-\Lambda/2, z). \tag{2.20}$$

Finally, in the upper halfspace the diffracted electric field (this is the difference between the total electric field $\boldsymbol{E}_I$ and the incident electric field $\boldsymbol{E}_I^{\text{inc}}$) has to satisfy the radiation condition for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z < 0$. This means that the diffracted field can be written as a superposition of outgoing plane waves and is bounded. In the lower halfspace the diffracted electric field (this is just the total electric field $\boldsymbol{E}_{II}$) also has to satisfy the radiation condition for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z > D$.

For two-dimensional periodic gratings similar boundary conditions apply.

Chapter 3

Reduced Grating Structure and Fourier Representation

In the first three sections the medium is sliced into different layers where per layer all material properties are regarded as "constant". The last part of this section discusses Fourier representation of the electric field.

3.1 Reduced grating structure

3.1.1 One-dimensional periodic gratings

The first model reduction simplifies the two media I and II in Figure (2.2), they are assumed to be homogeneous. In the entire domain the permeability is given by $\mu(x,z) = \mu_0$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z \in \mathbb{R}$. However, the complex permittivity $\tilde{\varepsilon}(x,z)$ is still a function of both the x and z coordinates and is given by $\tilde{\varepsilon}_I(x,z) = \tilde{\epsilon}_I$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z < f(x)$ and by $\tilde{\varepsilon}_{II}(x,z) = \tilde{\epsilon}_{II}$ for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z > f(x)$. The second model reduction deals with simplifying the grating structure. In general the exact form is unknown and therefore the grating is approximated with a finite number of K layers having variable layer-thicknesses d_i for $i = 2, \ldots, K+1$. Within each layer the complex permittivity depends only on the x-coordinate and is independent of the z-coordinate. The upper and lower halfspace in Figure (2.2) are denoted by the (semi-infinite) layers 1 and $K+2$ in Figure (3.1). Within these layers the permeability and complex permittivity are both constant. The normal unit vector $\boldsymbol{n}$ on each layer is pointing in the negative z-direction. From now on the subscript i is used in the complex permittivity and electric field to indicate the layer where these variables are valid ($i = 1$ indicates the semi-infinite layer 1 which is the upper halfspace, $i = K+2$ indicates the semi-infinite layer $K+2$ which is the lower halfspace and $i = 2, \ldots, K+1$ indicates one of the layers that approximate the grating). For example, in layer 2 the y-component of the electric field is given by $E_{2,y}(x,z)$ and is valid only for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $D_1 \leq z \leq D_2$. In the same way the complex permittivity in layer 2 is given by $\tilde{\varepsilon}_2(x)$ which takes on the value $\tilde{\epsilon}_I$ or $\tilde{\epsilon}_{II}$.

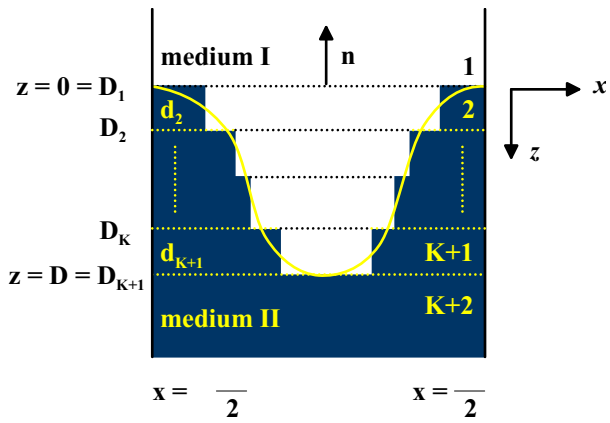


Figure 3.1: Layered one-dimensional grating in $[-\frac{\Lambda}{2}, \frac{\Lambda}{2}] \times (-\infty, \infty)$.

3.1.2 Two-dimensional periodic gratings

The first model reduction simplifies the two media I and II in Figure (2.4), they are assumed to be homogeneous. In the entire domain the permeability is given by $\mu(x, y, z) = \mu_0$ for $-\frac{\Lambda_x}{2} \leq x \leq \frac{\Lambda_x}{2}$, $-\frac{\Lambda_y}{2} \leq y \leq \frac{\Lambda_y}{2}$ and $z \in \mathbb{R}$. However, the complex permittivity $\tilde{\varepsilon}(x, y, z)$ is still a function of the x , y and z coordinates and is given by $\tilde{\varepsilon}_I(x, y, z) = \tilde{\epsilon}_I$ for $-\frac{\Lambda_x}{2} \leq x \leq \frac{\Lambda_x}{2}$, $-\frac{\Lambda_y}{2} \leq y \leq \frac{\Lambda_y}{2}$ and $z < f(x, y)$ and by $\tilde{\varepsilon}_{II}(x, y, z) = \tilde{\epsilon}_{II}$ for $-\frac{\Lambda_x}{2} \leq x \leq \frac{\Lambda_x}{2}$, $-\frac{\Lambda_y}{2} \leq y \leq \frac{\Lambda_y}{2}$ and $z > f(x, y)$. The second model reduction deals with simplifying the grating structure. In general the exact form is unknown and therefore the grating is approximated with a finite number of K layers having variable layer-thicknesses d_i for $i = 2, \ldots, K + 1$. Within each layer the complex permittivity depends only on the x and y coordinates and is independent of the z coordinate. The upper and lower halfspace in Figure (2.4) are denoted by the (semi-infinite) layers 1 and $K + 2$ in Figure (3.2). Within these layers the permeability and complex permittivity are both constant. Note that the pictures in Figure (3.2) are cross-sections of the two-dimensional layered grating: the picture on the left is a cross-section at $y = 0$ and the picture on the right is a cross-section at $x = \frac{\Lambda_x}{2}$. The normal unit vector $\boldsymbol{n}$ on each layer is pointing in the negative z-direction.

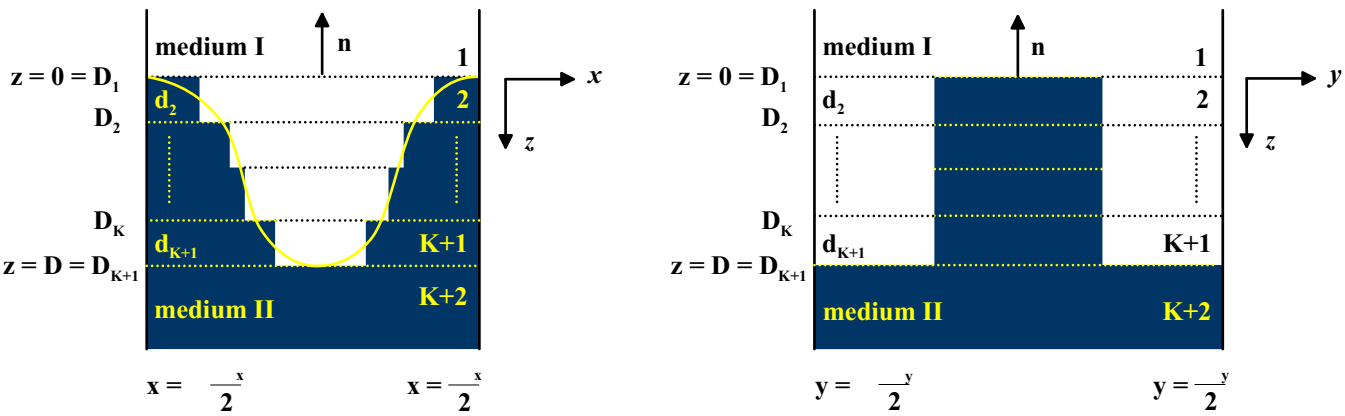


Figure 3.2: Layered two-dimensional grating in $[-\frac{\Lambda_x}{2}, \frac{\Lambda_x}{2}] \times [-\frac{\Lambda_y}{2}, \frac{\Lambda_y}{2}] \times (-\infty, \infty)$

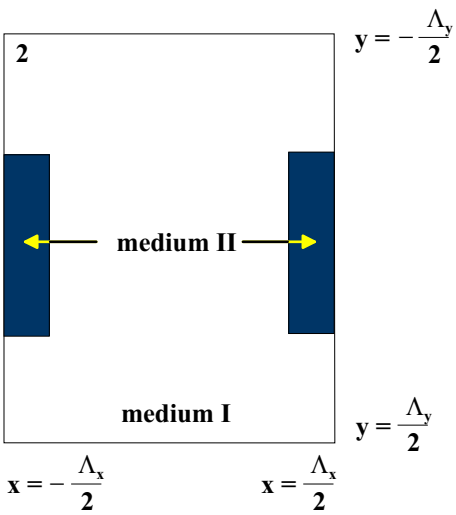


Figure 3.3: Top view of layered two-dimensional grating in $[-\frac{\Lambda_x}{2}, \frac{\Lambda_x}{2}] \times [-\frac{\Lambda_y}{2}, \frac{\Lambda_y}{2}]$.

From now on the subscript i is used in the complex permittivity and electric field to indicate the layer where these variables are valid ($i = 1$ indicates the semi-infinite layer 1 which is the upper halfspace, $i = K + 2$ indicates the semi-infinite layer $K + 2$ which is the lower halfspace and $i = 2, \ldots, K + 1$ indicates one of the layers that approximate the grating). For example, in layer 2 the y-component of the electric field is given by $E_{2,y}(x, y, z)$ and is valid only for $-\frac{\Lambda_x}{2} \leq x \leq \frac{\Lambda_x}{2}$, $-\frac{\Lambda_y}{2} \leq y \leq \frac{\Lambda_y}{2}$ and $D_1 \leq z \leq D_2$. In the same way the complex permittivity in layer 2 is given by $\tilde{\varepsilon}_2(x, y)$ which takes on the value $\tilde{\epsilon}_I$ or $\tilde{\epsilon}_{II}$. In the picture below a top view of layer 2 is depicted of thickness d_2.

3.2 Reduced grating equations for 1D planar diffraction

3.2.1 TE polarization

Applying both model reductions to equation (2.12) and using $k_0^2 = \omega^2 \mu_0 \varepsilon_0$ gives the following set of equations for $i = 1, \ldots, K + 2$:

$$\frac{\partial^2}{\partial z^2} E_{i,y}(x, z) = -k_0^2 \frac{\tilde{\varepsilon}_i(x)}{\varepsilon_0} E_{i,y}(x, z) - \frac{\partial^2}{\partial x^2} E_{i,y}(x, z). \tag{3.1}$$

Note that for $i = 1$ and $i = K+2$ equation (3.1) is a Helmholtz equation since $\tilde{\varepsilon}_1(x) = \tilde{\epsilon}_I$ and $\tilde{\varepsilon}_{K+2}(x) = \tilde{\epsilon}_{II}$. This was already derived in equation (2.9) if the entire domain is homogeneous (which is true for the semi-infinite homogeneous layers 1 and $K + 2$).

3.3 Reduced grating boundary conditions for 1D planar diffraction

3.3.1 TE polarization

The boundary jump conditions (2.18) are now applied to all $K+1$ interfaces. Equation (2.11a) is used to eliminate the x-component of the magnetic field. For $i = 1, \ldots, K+1$ and $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z = D_i$ the boundary jump conditions become:

$$E_{i,y}(x,z) = E_{i+1,y}(x,z), \tag{3.2a}$$

$$\frac{\partial}{\partial z} E_{i,y}(x,z) = \frac{\partial}{\partial z} E_{i+1,y}(x,z). \tag{3.2b}$$

The pseudo-periodic boundary condition (2.19) is applied to all $K+2$ layers. So for $i = 1, \ldots, K+2$ and $z \in \mathbb{R}$:

$$E_{i,y}(\Lambda/2, z) = e^{-jk_0 n_I \Lambda \sin\theta} E_{i,y}(-\Lambda/2, z). \tag{3.3}$$

In layer 1 the diffracted electric field still has to satisfy the radiation condition. This means that the y-component of the electric field can be written as a superposition of outgoing plane waves and is bounded for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$ and $z < 0$. A similar condition holds for the diffracted electric field in layer $K+2$ where $z > D$.

3.4 Fourier representation of the electric field

In this section Fourier expansions are used for the electric field as well as for the complex relative permittivity function.

3.4.1 1D planar diffraction : TE polarization

In the case of TE polarization the normalized incident electric field is perpendicular to the plane of incidence. So the electric field has only a y-component and equation (3.1) together with the boundary conditions (3.2), (3.3) and the radiation condition describe the entire diffraction problem. The magnetic field components can be derived from equations (2.11a) and (2.11b) once the electric field is known. The normalized incident electric field is given by:

$$E_y^{\text{inc}}(x,z) = e^{-jk_0 n_1 (x\sin\theta + z\cos\theta)}. \tag{3.4}$$

The electric field in the two semi-infinite homogeneous layers 1 and $K+2$ is given by the Rayleigh expansions and the Floquet condition

$$E_{1,y}(x,z) = \sum_n R_n e^{-j(k_{xn}x - k_{1,zn}z)} + E_y^{\text{inc}}(x,z), \tag{3.5}$$

$$E_{K+2,y}(x,z) = \sum_n T_n e^{-j(k_{xn}x + k_{K+2,zn}(z-D))}, \tag{3.6}$$

$$k_{xn} = k_0(n_1 \sin\theta - n(\lambda_0/\Lambda)), \tag{3.7}$$

$$k_{1,zn} = (k_0^2 n_1^2 - k_{xn}^2)^{1/2}, \tag{3.8}$$

$$k_{K+2,zn} = (k_0^2 n_{K+2}^2 - k_{xn}^2)^{1/2}. \tag{3.9}$$

In the expansions the unknown constant field amplitudes R_n and T_n stand for the reflected and transmitted fields respectively.

The electric field in the K layers is expanded in a Fourier series with unknown field components $S_{i,n}(z)$ that depend on the z-coordinate for $i = 2, \ldots, K+1$. In the same way as above a Fourier expansion is used for the complex relative permittivity $\tilde{\varepsilon}_i^r(x) = \tilde{\varepsilon}_i(x)/\varepsilon_0$ for $i = 2, \ldots, K+1$:

$$E_{i,y}(x,z) = \sum_n S_{i,n}(z) e^{-jk_{xn}x}, \tag{3.10}$$

$$\tilde{\varepsilon}_i^r(x) = \sum_g \tilde{\epsilon}_{i,g} e^{j\frac{2\pi}{\Lambda}gx}, \qquad \tilde{\epsilon}_{i,g} = \frac{1}{\Lambda}\int_{-\frac{\Lambda}{2}}^{\frac{\Lambda}{2}} \tilde{\varepsilon}_i^r(x) e^{-j\frac{2\pi}{\Lambda}gx} dx. \tag{3.11}$$

Substituting the expansions above into equation (3.1) and using the relation $k_{xn} - \frac{2\pi}{\Lambda}g = k_{x(n+g)}$ gives for $i = 2, \ldots, K+1$:

$$\begin{aligned}
\sum_n \frac{\mathrm{d}^2}{\mathrm{d}z^2} S_{i,n}(z) e^{-jk_{xn}x} &= -k_0^2 \sum_g \sum_n \tilde{\epsilon}_{i,g} S_{i,n}(z) e^{-jk_{x(n+g)}x} + \sum_n k_{xn}^2 S_{i,n}(z) e^{-jk_{xn}x} \\
&= -k_0^2 \sum_p \sum_n \tilde{\epsilon}_{i,p-n} S_{i,n}(z) e^{-jk_{xp}x} + \sum_n k_{xn}^2 S_{i,n}(z) e^{-jk_{xn}x} \\
&= -k_0^2 \sum_n \sum_p \tilde{\epsilon}_{i,n-p} S_{i,p}(z) e^{-jk_{xn}x} + \sum_n k_{xn}^2 S_{i,n}(z) e^{-jk_{xn}x}
\end{aligned} \tag{3.12}$$

Since equation (3.12) holds for $-\frac{\Lambda}{2} \leq x \leq \frac{\Lambda}{2}$, this means that after projection on the basis functions for $i = 2, \ldots, K+1$ and $n \in \mathbb{Z}$:

$$\frac{\mathrm{d}^2}{\mathrm{d}z^2} S_{i,n}(z) = -k_0^2 \sum_p \tilde{\epsilon}_{i,n-p} S_{i,p}(z) + k_{xn}^2 S_{i,n}(z). \tag{3.13}$$

Summarizing in each layer $i = 1, ..., K + 2$, the electric field is approximated with the use of a Fourier basis which in 1D case is:

$$\begin{aligned} E_{i,y}(x,z) &= \sum_{n} c_{i,n}(z) e^{-jk_{xn}x} \\ &= \sum_{n=-\infty}^{\infty} c_{i,n}(z) e^{-jk_{xn}x} \\ &\doteq \sum_{n_x=-M_x}^{M_x} c_{i,n_{n_x}}(z) e^{-jk_{xn}x}, \qquad i = 1, ..., K+2. \end{aligned} \tag{3.14}$$

In the 2D case each of the three electric field components are approximated with a Fourier expression

$$E_{i,y}(x,y,z) = \sum_{n_x=-M_x}^{M_x} \sum_{n_y=-M_y}^{M_y} c_{i,n}(z) e^{-jk_{xn}x} e^{-jk_{yn}y}, \quad i = 1, ..., K+2. \tag{3.15}$$

Chapter 4

The VIM Discretization

4.1 Introduction to the volume integral method (VIM)

Before going into detail about the specific diffraction problem first the volume integral method for a general electro-magnetic setting is given. Like with RCWA the starting point are the time harmonic Maxwell equations in partial differential equation form (see equations (2.6), (2.7))

$$\nabla \times \boldsymbol{H} = j\omega\varepsilon_0\varepsilon_r\boldsymbol{E}, \tag{4.1a}$$

$$\nabla \times \boldsymbol{E} = -j\omega\mu_0\boldsymbol{H}, \tag{4.1b}$$

where ε_0 and μ_0 respectively are the permittivity and permeability of vacuum. Here the standard constitutive equations were used for the electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{H}$ assuming non-magnetic but otherwise general lossy or lossless media with relative permittivity ε_r. Naturally the fields and relative permittivity depend on all three spatial coordinates $\boldsymbol{r} = (x, y, z)$ which are left out for notational convenience. Eliminating the magnetic field from these equations and introducing the wave number of vacuum $k_0 = \omega\sqrt{\varepsilon_0\mu_0}$ results in

$$\nabla \times \nabla \times \boldsymbol{E} = k_0^2\varepsilon_r\boldsymbol{E}. \tag{4.2}$$

From this point onwards VIM typically differs from RCWA and starts by rewriting (4.2) into

$$\nabla \times \nabla \times \boldsymbol{E} - k_0^2\varepsilon_r^b\boldsymbol{E} = k_0^2\big(\varepsilon_r - \varepsilon_r^b\big)\boldsymbol{E}. \tag{4.3}$$

Here essentially the relative permittivity is split up into two parts $\varepsilon_r = \varepsilon_r^b + \big(\varepsilon_r - \varepsilon_r^b\big)$ where ε_r^b is called the background relative permittivity and the term within brackets is sometimes referred to as the permittivity contrast term. The trick is to choose the background relative permittivity in such a way that the homogeneous part of (4.3), putting the right-hand side to zero, can be solved easily while trying to make the permittivity contrast

term equal to zero in most of the physical domain. The next step is to use the linearity of Maxwell's equations and to split up the electric field into two parts

$$\boldsymbol{E} = \boldsymbol{E}^{\text{inc}} + \boldsymbol{E}^{\text{scat}}, \tag{4.4}$$

such that

$$\nabla \times \nabla \times \boldsymbol{E}^{\text{inc}} - k_0^2 \varepsilon_r^b \boldsymbol{E}^{\text{inc}} = \boldsymbol{0}, \tag{4.5a}$$

$$\nabla \times \nabla \times \boldsymbol{E}^{\text{scat}} - k_0^2 \varepsilon_r^b \boldsymbol{E}^{\text{scat}} = k_0^2 \big(\varepsilon_r - \varepsilon_r^b\big) \boldsymbol{E}. \tag{4.5b}$$

Here $\boldsymbol{E}^{\text{inc}}$ is called the incident electric field and is the solution of Maxwell's equations in absence of the contrast term and only with the background permittivity present. Moreover $\boldsymbol{E}^{\text{scat}}$ is then the scattered electric field and is the solution of Maxwell's equations where the contrast term is treated as a source term. Note that the scattered electric field has to satisfy radiation conditions. As already mentioned the solution to (4.5a) should be easy to find and is usually known in closed form. The solution to (4.5b) is found with the help of the so-called Green's tensor. The Green's tensor $\overline{\overline{\boldsymbol{G}}}(\boldsymbol{r}, \boldsymbol{r}')$ is the solution of the same equation as for the scattered field, except that the right-hand side is replaced with a mathematical dipole, thus

$$\nabla \times \nabla \times \overline{\overline{\boldsymbol{G}}} - k_0^2 \varepsilon_r^b \overline{\overline{\boldsymbol{G}}} = \delta \overline{\overline{\boldsymbol{I}}}, \tag{4.6}$$

where $\delta = \delta(\boldsymbol{r} - \boldsymbol{r}')$ the Dirac delta function and $\overline{\overline{\boldsymbol{I}}}$ simply the unit tensor. As was the case for the scattered electric field also the Green's tensor has to satisfy the radiation condition. Having found the Green's tensor the scattered electric field is then given by

$$\boldsymbol{E}^{\text{scat}} = \int_V \overline{\overline{\boldsymbol{G}}} \cdot k_0^2 \big(\varepsilon_r - \varepsilon_r^b\big) \boldsymbol{E} \, \mathrm{d}\boldsymbol{r}'. \tag{4.7}$$

To arrive at this equation first multiply (4.6) by the right-hand side of equation (4.5b) but evaluated at $\boldsymbol{r}'$ instead of $\boldsymbol{r}$. Then perform an integration in the variable $\boldsymbol{r}'$ over $\mathbb{R}^3$. Since the contrast term is almost zero everywhere this boils down to an integration over the subdomain $V \subset \mathbb{R}^3$ (i.e. the domain or volume where the contrast is non-zero). Finally because the Maxwell operator is linear and acts on the position variable $\boldsymbol{r}$ it can be taken out of the integral to arrive at (4.7). Using the decomposition in (4.4) one can now derive an integral equation for the unknown total electric field

$$\boldsymbol{E} = \boldsymbol{E}^{\text{inc}} + \int_V \overline{\overline{\boldsymbol{G}}} \cdot k_0^2 \big(\varepsilon_r - \varepsilon_r^b\big) \boldsymbol{E} \, \mathrm{d}\boldsymbol{r}'. \tag{4.8}$$

The fact that (4.8) is an integral equation and the integral is taken over a certain volume V explains the name volume integral method. At this point we have arrived at equation (1) of the Chang article [29].

The next step is to discretize (4.8) for the specific diffraction problem at hand. The assumption of an infinitely double-periodic structure suggest the use of a (quasi-periodic) Fourier series (i.e. Bloch-Floquet modes) in the xy-plane. Equation (4.6) then simply reduces to finding the Green's function along the vertical z-direction for all Fourier modes separately. This can still be done analytically for a homogeneous layer stack. Semi-discretizing (4.8) with Fourier series then actually results in a line-integral along the

z-direction. The Fourier coefficients of the electric field which are still a function of z can be discretized in many ways. One possible way is described at the beginning of section 3 in [29] where a (uniform) mesh is chosen along the z-direction in combination with piecewise constant or piecewise linear basis functions. The integrals that remain can then still be evaluated analytically. A linear system can be derived in several ways. In the Chang article [29] one typically uses test functions or moment matching techniques to arrive at a linear system. With a bit of imagination it can be seen that the linear system belonging to equation (4.8) must have the following form

$$\big(\boldsymbol{C} - \boldsymbol{G}\boldsymbol{M}\big)\boldsymbol{x} = \boldsymbol{b}, \tag{4.9}$$

where $\boldsymbol{C}$ is a matrix coming from discretizing the left-hand side of (4.8) (following [29] and using piecewise linear expansion and test functions ([29], p641)), $\boldsymbol{G}$ is the matrix containing the terms related to the Green's function in the integral and $\boldsymbol{M}$ is the matrix related to the contrast term in the integral. Moreover the vector $\boldsymbol{b}$ contains the expansion coefficients of the incident electric field coming from discretizing the first term in the right-hand side of (4.8). Finally the vector $\boldsymbol{x}$ simply contains the expansion coefficients of the total electric field.

4.2 VIM parameters

4.2.1 VIM parameters

The matrices $\boldsymbol{C}$, $\boldsymbol{G}$ and $\boldsymbol{M}$ depend on the discretization (choices) parameters as well as on geometry and material parameters, i.e. on all parameters discussed in Chapter (2) and section (3.4) (in a non-linear manner):

1. Discretization parameters:
   - $-M_x$ and $+M_x$: lower and upper index of modes along first reciprocal vector
   - $-M_y$ and $+M_y$: lower and upper index of modes along second reciprocal vector
   - N_z: number of layers along grating
2. Incident field $\boldsymbol{E}^{\text{inc}}$ parameters :
   - θ: angle of incidence (angle with z axis) in degrees
   - ϕ: angle of incidence (angle with x axis) in degrees
   - $\boldsymbol{E}_\theta$: incident field parallel to the plane of incidence (classical TM or p polarization)
   - $\boldsymbol{E}_\phi$: incident field perpendicular to the plane of incidence (classical TE or s polarization)
   - f: frequency of operation
3. Material parameters (ϵ, μ, in equation (2.3)):
   - ϵ_∞: relative permittivity of background

- ϵ_1: relative permittivity of layer (possibly complex): BARC
- $\epsilon_{-\infty}$: relative permittivity of lower halfspace (possibly complex): Silicon
- ϵ_2: relative permittivity of grating (possibly complex): Resist

4. Geometry parameters:

   - **dv1**: x and y component of 1st lattice vector

     This vector specifies the first part of the unit-cell. In the test cases it is always chosen along the x-direction. So, $\mathbf{dv1} = (a, 0)$ meaning that the period/pitch of the grating along this direction is equal to $\text{norm}(\mathbf{dv1}) = a$.
   - **dv2**: x and y component of 2nd lattice vector

     Similarly this vector specifies the second part of the unit-cell. In our examples it is always chosen along the y - direction. So, $\mathbf{dv2} = (0, b)$ meaning that the period/pitch of the grating along this direction is equal to $\text{norm}(\mathbf{dv2}) = b$.
   - thickness: thickness of grating in z-direction

     This is simply the total height of the grating.
   - dslab: thickness of the layer

     This is simply the height/thickness of the homogeneous layer below the grating.
   - bottomcoordx/y(1): x- and y- coordinates of vertex at the bottom, relative with respect to dv1 and dv2 (lower left corner)
   - bottomcoordx/y(2): x- and y- coordinates of vertex at the bottom, relative with respect to dv1 and dv2 (lower left corner)

     The last two coordinates are used to specify the grating itself, assuming a square block of resist one simply needs to specify two opposite corner points of this square in the xy - plane. From these corner points the width of the (resist or silicon) block in the x - and y - direction can be found.

5. Accuracy parameters:

   - MFFT: fast Fourier transform (fft) size in transverse plane greater or equal to $\max(2M_x + 1, 2M_y + 1)$, product of small primes
   - NFFT: fft size in z direction greater or equal to $2N_z - 1$, product of small primes.

4.2.2 Degrees of freedom

The degrees of freedom $\boldsymbol{x} \in \mathbb{C}^N$ are the discretization degrees of freedom in the order (i, m_1, m_2, n_z) where the first index is the fast running index and the last one is the slow running index:

- $i = 1, 2, 3$ the three components E_1, E_2, E_3 of the electric field $\boldsymbol{E}$

- $m_1 = -M_x, ..., +M_x$ where $M_x \in \mathbb{N}$, Fourier discretization in the x direction
- $m_2 = -M_y, ..., +M_y$ where $M_y \in \mathbb{N}$, Fourier discretization in the y direction
- $n_z = 0, ..., N_z$ spatial discretization/sampling in the z direction.

This implies that the total amount of degrees of freedom is

$$N = 3(2M_x + 1)(2M_y + 1)(N_z + 1).$$

In general, the larger M_x, M_y and N_z are the more accurate the electric field $\boldsymbol{E}$ is approximated.

4.2.3 Parameter dependencies

1. $\boldsymbol{C}$ and $\boldsymbol{M}$ depend on $M_x, N_y, \epsilon_\infty, \epsilon_2$, bottomcoordx/y(1), and bottomcoordx/y(2).
2. $\boldsymbol{G}$ depends on $M_x, N_y, N_z, \theta, \phi, f, \epsilon_1, \epsilon_{-\infty}, \epsilon_\infty$ dv1, dv2, dslab, and thickness.

4.3 The test cases, geometry

There are three test cases related to the different geometries. Test case 2 (T2) and 3 (T3) are related to the geometry (material properties) in Figure (4.2) and test case 1 (T1) is related to the geometry (material properties) in Figure (4.1). Section (4.4.3) explains the choice for these specific test cases.

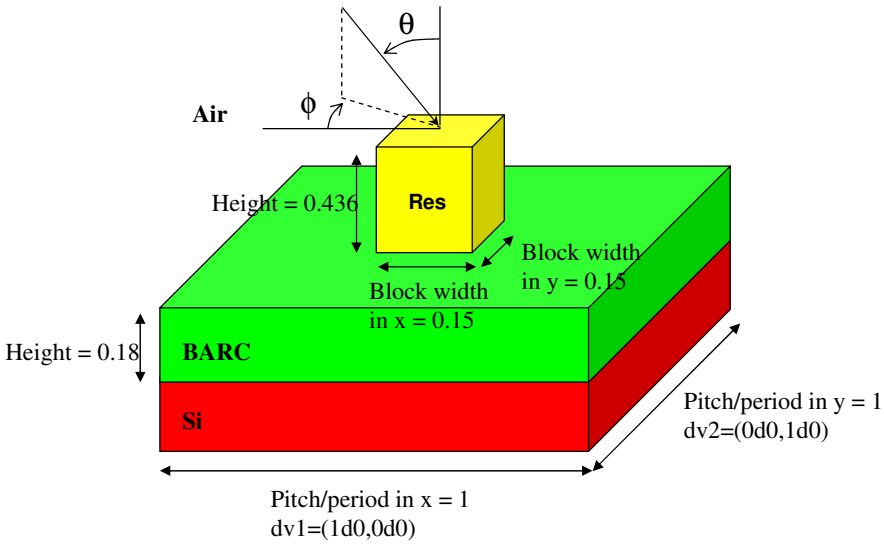
Case	M_x	M_y	N_z	MFFT	NFFT	Bytes	Figure	Difficulty
T1	3	3	8	16	16	25.2 MB	(4.1)	Easy
T2	5	5	8	32	16	146 MB	(4.2)	Medium
T3	7	7	24	64	64	>2GB	(4.2)	Hard

Table 4.1: Parameter values.

Making the size of the grating larger and (or) increasing the permittivity directly influence the number of matrix vector multiplications (they can increase quite dramatically). These are mainly the parameters **ASML** changed to distinguish between the easy and the medium test cases; in addition to using different kinds of materials for some of the layers. For the medium and hard cases a Silicon block floats on the air, where as for the easy case we have a block of Resist (Res) lying on top of a layer of bottom anti reflection coating (BARC) which sits on top of a Silicon substrate.

unit-cell easy case

Angle of incidence (theta,phi) = (8.13010235deg, 45.0000000deg)
Incident polarization (Etheta,Ephi) = ((0+0*i),(1+0*i)) ~ TE (Transverse Electric) polarization
Incident frequency (freq) = 299792458 (~ incident wavelength = 1)

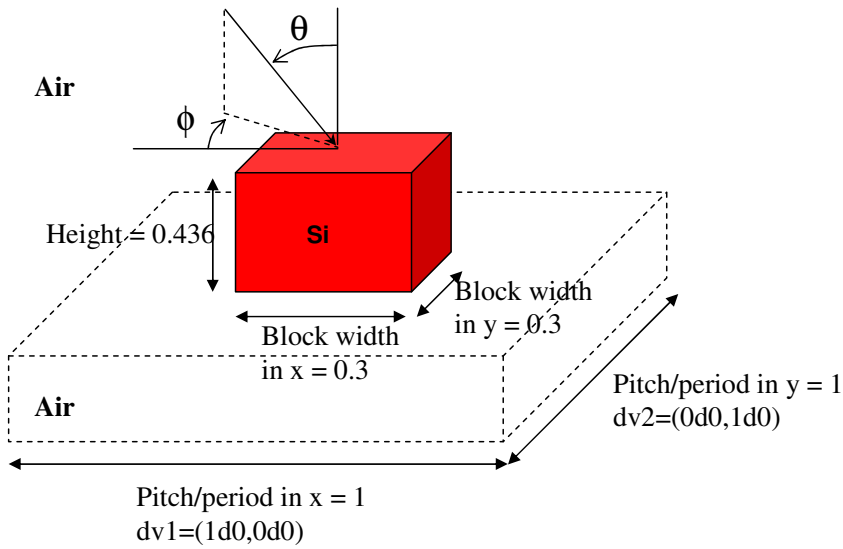


Air, relative permittivity = 1
Res (Resist), relative permittivity = 2.32410025
BARC (Bottom Anti Reflection Coating), relative permittivity = 2.46985106091296
Si (Silicon), relative permittivity = 18.37919466326700-i*0.40307129895600

Figure 4.1: Geometry T1

unit-cell medium/hard case

Angle of incidence (theta,phi)	= (8.13010235deg, 45.0000000deg)
Incident polarization (Etheta,Ephi)	= ((0+0*i),(1+0*i)) ~ TE (Transverse Electric) polarization
Incident frequency (freq)	= 299792458 (~ incident wavelength = 1)



Air,	relative permittivity = 1
Si (Silicon),	relative permittivity = 18.37919466326700-i*0.40307129895600

Figure 4.2: Geometry T2/T3

4.4 Properties of A, C, G and M

In order to determine the best (fastest) method to solve $\boldsymbol{Ax} = \boldsymbol{b}$, we first examine the structure of the non-zero entries of $\boldsymbol{A}$ and its spectrum. Its spectral properties help to determine the best iterative solution method.

4.4.1 Matrix vector multiplication

The non-zero sparsity pattern of the matrix (parts which constitute) $\boldsymbol{A}$ is a direct consequence of the VIM discretization and the choice of basis. The matrix $\boldsymbol{A}$ (canonical basis), a combination of the sparse matrices $\boldsymbol{C}$, $\boldsymbol{M}$ and $\boldsymbol{G}$, is dense. But because $\boldsymbol{x} \mapsto \boldsymbol{Cx}$, $\boldsymbol{x} \mapsto \boldsymbol{Gx}$ and $\boldsymbol{x} \mapsto \boldsymbol{Mx}$ can be calculated in linear time in N, the amount of degrees of freedom, so can a matrix-vector product $\boldsymbol{x} \mapsto \boldsymbol{Ax}$. The VIM discretization is such that the matrix vector product can be calculated even faster if one uses a Fast Fourier Transformation, i.e., if one uses a different basis for the representation of the linear operator.

4.4.2 Sparsity pattern

The VIM discretization of (an integral-equation reformulation for (layer-wise) Maxwell's equations described in Chapter (2)) of the linear equation (2.9) with boundary conditions (2.18) leads to a system $\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c})$, where $\boldsymbol{A}(\boldsymbol{c}) \in \mathbb{C}^{N\times N}, \boldsymbol{b}(\boldsymbol{c}) \in \mathbb{C}^N$, and the unknown vector $\boldsymbol{x}(\boldsymbol{c}) \in \mathbb{C}^N$ which all depend on the parameters $\boldsymbol{c}$ described in section (4.2).

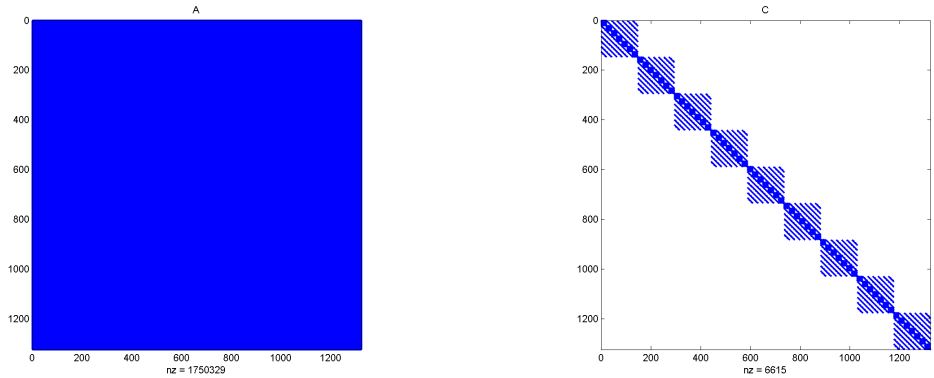
The coefficient matrix $\boldsymbol{A}$ is given by the formula $\boldsymbol{A}(\boldsymbol{c}) = \boldsymbol{C} - \boldsymbol{G} \cdot \boldsymbol{M}$ and it is a dense non-Hermitian (see (C) complex matrix. The matrices $\boldsymbol{C}$, $\boldsymbol{M}$ and $\boldsymbol{G}$ are sparse and have block structures as described below (see Figure (4.3) and (4.4)). The unknowns in the system $\boldsymbol{Ax} = \boldsymbol{b}$ are derived from a 4D array which is converted into one long vector (hence the block structures below).

The block structures

1. $\boldsymbol{C} = \boldsymbol{B}_x + \boldsymbol{B}_y + \boldsymbol{I}_z$ where $\boldsymbol{B}_x$ is diagonal-block-toeplitz-block-diagonal-block-diagonal, $\boldsymbol{B}_y$ is diagonal-block-diagonal-block-toeplitz-block-diagonal, and $\boldsymbol{I}_z$ is block diagonal. See Figure (4.3) (b).

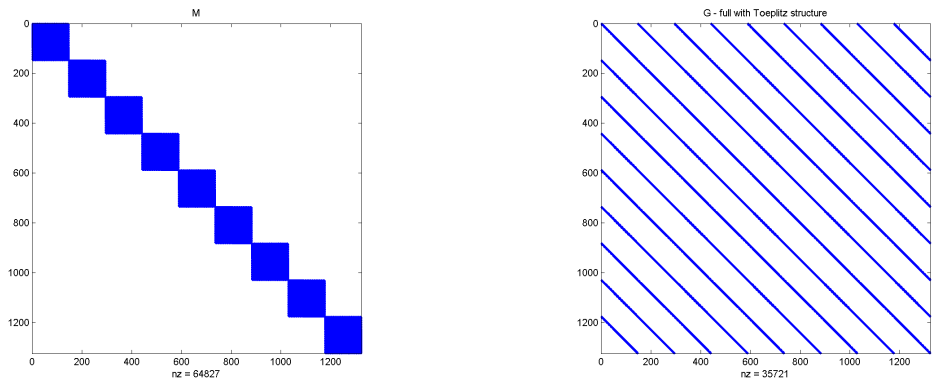
2. $\boldsymbol{M}$ is diagonal-block-toeplitz-block-toeplitz-block-diagonal. See Figure (4.4) (a).

3. $\boldsymbol{G}$ is a sparse toeplitz matrix. See Figure (4.4) (b).



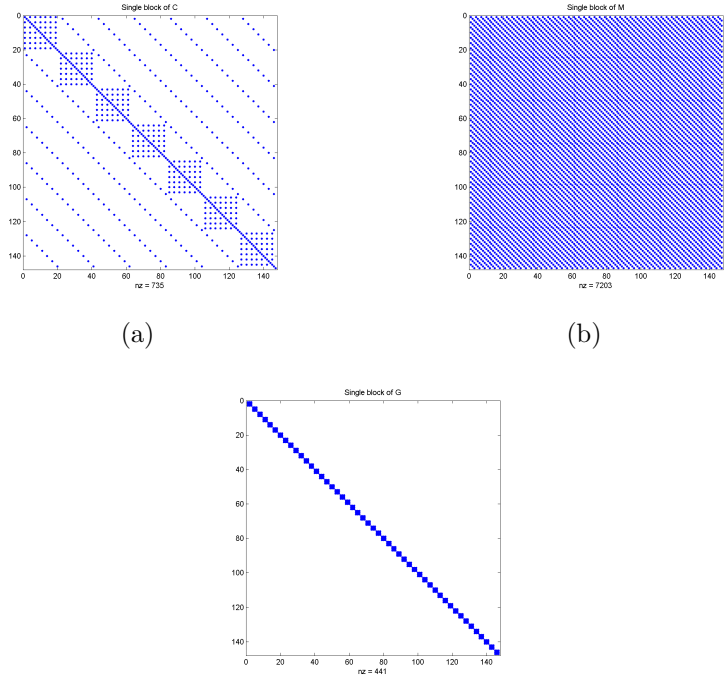
(a) (b)

Figure 4.3: T1 (a) The coefficient matrix $\boldsymbol{A}$ and (b) the matrix $\boldsymbol{C}$



(a) (b)

Figure 4.4: T1 (a) the matrix $\boldsymbol{M}$ and (b) the matrix $\boldsymbol{G}$.



(a) (b)

(c)

Figure 4.5: Block of (a) $\boldsymbol{C}$ (b) $\boldsymbol{M}$ and (c) $\boldsymbol{G}$.

Figure (4.5) shows the structure of an individual (non-zero) block of $\boldsymbol{C}, \boldsymbol{M}$ and $\boldsymbol{G}$.

(a). Each block of $\boldsymbol{C}$ has a Toeplitz structure containing $2M_y + 1$ Toeplitz blocks at the diagonal and 6 upper and lower 'diagonals'.

(b). Each block of $\boldsymbol{M}$ consists of $2M_y + 1$ Toeplitz blocks in which each of them again consists of blocks with Toeplitz structures.

(c). Each block in $\boldsymbol{G}$ is a $3(2M_x+1)(2M_y+1)$ block diagonal matrix of non-overlapping 3×3 blocks.

Theorem 4.4.1. *There exists a permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{PGP}^{-1}$ is a block diagonal matrix.*

Proof. Inspection of the non-zero pattern yields that (`MatLab` notation)

$$\begin{aligned}\boldsymbol{P} = \Big[&[1,2,3],[1,2,3]+3\Delta,\ldots,[1,2,3]+3\Delta N_z;\\ &[4,5,6],[4,5,6]+3\Delta,\ldots,[4,5,6]+3\Delta N_z;\\ &\ldots\\ &3\Delta-[2,1,0],6\Delta-[2,1,0],\ldots,3\Delta(N_z+1)-[2,1,0]\Big]\end{aligned}$$

where $\Delta = (2M_x+1)(2M_y+1)$. □

4.4.3 Analysis of the properties of $\boldsymbol{A}$

All of our analysis are "numerical" (i.e. experimental) since the formulae for (the entries) the matrix blocks $\boldsymbol{C}$, $\boldsymbol{G}$ and $\boldsymbol{M}$ are not provided as an information (NoA). We can resort to a numerical analysis since **ASML** provides a `MatLab` code which calculates

$$\boldsymbol{c} \rightarrow \boldsymbol{C}(\boldsymbol{c}),\ \boldsymbol{c} \rightarrow \boldsymbol{G}(\boldsymbol{c}),\ \boldsymbol{c} \rightarrow \boldsymbol{M}(\boldsymbol{c})$$

which reads the parameter values $\boldsymbol{c}$ from files.

The first challenge is to find test cases of physical relevance which fit into memory of the largest (RAM sense) available departmental computer (16GB RAM) which can run a version of `MatLab` (64 bits) that can deal with big (multi GB) matrices $\boldsymbol{C}$, $\boldsymbol{G}$, $\boldsymbol{M}$ and $\boldsymbol{A}$. This challenge turns out to be non-trivial. Even small numbers of harmonics ($M = M_x = M_y$) and number of layers (N_z) lead to a matrix $\boldsymbol{A}$ of several megabytes large (see Table (4.1): T1, T2) and a physically interesting case is easily several gigabytes large (Table (4.1):T3). Because for operations (such as the determination of singular values) `MatLab` uses a factor of the GB size of $\boldsymbol{A}$, T3 is the "largest" test case for which a numerical analysis is possible.

The reason to use the same geometry (Figure (4.2)) for a range of tests (T1, T2) is motivated by the wish to get a bit of insight into the asymptotical behavior of the properties of the matrix for the case $M_x,\ M_y,\ N_z \rightarrow \infty$.

The reason to have test T1 (Figure (4.1)) next to T2, T3 is that the material for T1 (Resist, BARC) is "easier" than that of T2, T3; which is Silicon. "Easier" is related to an observation made by **ASML**: using specific method (NoA) the amount of iterations increases if

1. for constant material properties $M_x = M_y,\ N_z \rightarrow \infty$ - a more accurate solution.
2. for $M_x = M_y,\ N_z$ constant material properties $\varepsilon_i \rightarrow c$ (less contrast).

4.4.4 Hermitian

Definition 1: A complex matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ is said to be *Hermitian (self-adjoint)* if $\boldsymbol{A}^H = \boldsymbol{A}$. If $\boldsymbol{A}$ is *Hermitian* then its diagonal entries are real valued.

By using the `MatLab` command `isequal(A,A')`, one can check that all the coefficient matrices in the three test cases are not *Hermitian*.

4.4.5 Diagonal dominance

Definition 2: A square matrix $\boldsymbol{A} = [a_{ij}]_{i,j=1}^n$ is said to be *(weakly) diagonally dominant* if

$$|a_{ii}| \geq \sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}| \qquad \forall\ i, j. \tag{4.10}$$

By using the `MatLab` script in Appendix (C) it is checked that the coefficient matrix $\boldsymbol{A}$ is not diagonally dominant for all the three test cases, see Table (4.2).

Table (4.2) shows the number of rows, i which satisfy

$$|a_{ii}| < \sum_{i \neq j} |a_{ij}|. \tag{4.11}$$

Case	Number of rows satisfying (4.11)
T1	629
T2	3267
T3	16875

Table 4.2: The number of rows i which satisfy (4.11).

4.4.6 Spectrum, positive definiteness

Every iterative method requires the coefficient matrix to have certain properties. The convergence of most iterative methods is highly dependent on the spectral properties of the coefficient matrix. The singular values and the distribution of the eigenvalues play important roles for speeding up Krylov space type methods. For the distribution of the eigenvalues of the coefficient matrices of the three test cases, see Figure (4.6). This thesis is devoted to finding preconditioners $\boldsymbol{M}$ that can make the spectrum of the preconditioned matrix $\boldsymbol{M}^{-1}\boldsymbol{A}$ accumulate around one.

Definition 3: A matrix $\boldsymbol{A} \in \mathbb{C}^n$ is called positive definite if

$$\mathrm{Re}((\boldsymbol{A}\boldsymbol{x}, \boldsymbol{x})_2) > 0 \qquad \forall \boldsymbol{x} \in \mathbb{C}^n, \boldsymbol{x} \neq \boldsymbol{0}.$$

One can show that $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is called positive definite if and only if its *Hermitian* part

$$\frac{\boldsymbol{A} + \boldsymbol{A}^{\mathrm{H}}}{2}$$

is positive definite.

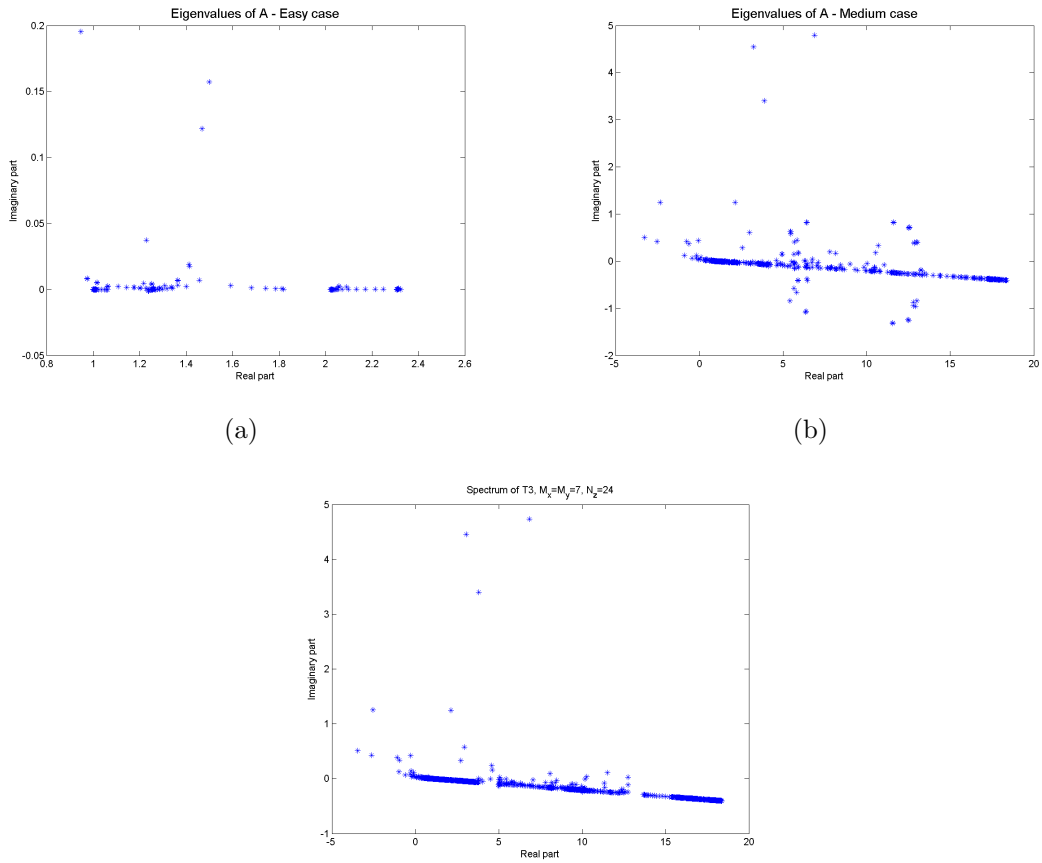
Definition 4: A matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is said to be *normal* if and only if $\boldsymbol{A}\boldsymbol{A}^{\mathrm{H}} = \boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}$.

By definition the *Hermitian* (symmetric) part of a matrix $\boldsymbol{A}$ is *normal* whence its numerical range

$$\left\{\mathrm{Re}((\boldsymbol{A}\boldsymbol{x}, \boldsymbol{x})_2) : \boldsymbol{x} \in \mathbb{C}^n, ||\boldsymbol{x}||_2 = 1\right\}$$

is the convex hull of its (real-valued) eigenvalues which implies that a complex valued matrix is positive definite if and only if the eigenvalues of its *Hermitian* part are all positive.

By implementing the above definitions and the corresponding `MatLab` codes, we found out that the coefficient matrix for the easy case is positive definite but not that of the medium and hard cases. Moreover, it is checked that all the matrices are not *normal*.



(a) (b)

(c)

Figure 4.6: Spectra of the test cases (a) T1 (b) T2 and (c) T3

4.4.7 Spectral radius and singular values

Let $\sigma_{\min}(\boldsymbol{A})$ and $\sigma_{\max}(\boldsymbol{A})$ respectively be the minimum and maximum singular values of $\boldsymbol{A}$ over $\sigma(\boldsymbol{A})$. Then, since

$$\kappa_2(\boldsymbol{A}) = ||\boldsymbol{A}||_2||\boldsymbol{A}^{-1}||_2 = \frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A})},$$

the extreme singular values (eigenvalues) (see Table (4.3)) strongly influence the amount of iterations.

In the case when $\boldsymbol{A}$ is a *normal* matrix the condition number becomes

$$\kappa_2(\boldsymbol{A}) = \frac{\lambda_{\max}(\boldsymbol{A})}{\lambda_{\min}(\boldsymbol{A})},$$

where $\lambda_{\max}(\boldsymbol{A})$ and $\lambda_{\min}(\boldsymbol{A})$ respectively are the maximum and minimum eigenvalues (in modulus) of $\boldsymbol{A}$.

Case	$\lambda_{\min}(\boldsymbol{A})$	$\lambda_{\max}(\boldsymbol{A})$	$\sigma_{\min}(\boldsymbol{A})$	$\sigma_{\max}(\boldsymbol{A})$	$\kappa_2(\boldsymbol{A})$
T1	0.9469+0.1953i	2.3241+0.0000i	0.7083	2.4928	3.5193
T2	-0.0362+0.0453i	18.3792-0.4031i	0.0286	25.5369	927.7855

Table 4.3: Extreme singular and eigenvalues

Additionally, in all the three cases one can compute (approximate) $\kappa_1(\boldsymbol{A}), \kappa_2(\boldsymbol{A})$, and $\kappa_\infty(\boldsymbol{A})$ with the `MatLab` command `cond` (`condest`) (see Table (4.4)):

Case	$\kappa_1(\boldsymbol{A})$	$\kappa_2(\boldsymbol{A})$	$\kappa_\infty(\boldsymbol{A})$
T1	11.9860	3.5193	36.6195
T2	6.9332e+003	927.7855	4.7050e+004
T3	7.0953e+004	484.3420	4.1939e+004

Table 4.4: Approximation of the condition numbers

Finally, to get a more precise insight of the spectrum Table (4.5) lists

$$\{\lambda \in \sigma(\boldsymbol{A}) : |\lambda| < 1\},$$

i.e. all eigenvalues inside the unit disc.

Because $\boldsymbol{A}$ is a full (almost) matrix, we will try to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with an iterative method in Chapter (5). A direct solver can be used only for $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ with N small, but even in that case calculating a full Gaussian elimination costs $\mathcal{O}(N^3)$ FLOPS, i.e, it is (too) expensive. Note that $\boldsymbol{A}$ depends on parameters, i.e, $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{c})$. Unfortunately, for some limit parameter values (such as when distance between two adjacent markers goes down to zero), $\kappa_2(\boldsymbol{A}) \mapsto \inf$. For such cases, we need to precondition the iterative solution method.

-0.7734 + 0.4128 i	0.4223 + 0.0326 i
-0.8922 + 0.1238 i	0.4883 + 0.0388 i
-0.6412 + 0.3709 i	0.5902 + 0.0104 i
-0.4397 + 0.0613 i	0.5918 + 0.0111 i
-0.2173 + 0.0673 i	0.6172 + 0.0108 i
-0.1923 + 0.1266 i	0.6216 + 0.0170 i
-0.0754 + 0.4361 i	0.6337 + 0.0162 i
-0.0438 + 0.0390 i	0.6923 + 0.0173 i
-0.0362 + 0.0452 i	0.7130 + 0.0088 i
0.0424 + 0.0444 i	0.7144 + 0.0079 i
0.0974 + 0.0868 i	0.7172 + 0.0214 i
0.2626 + 0.0280 i	0.7298 + 0.0292 i
0.3215 + 0.0203 i	0.7269 + 0.0092 i
0.3402 + 0.0268 i	0.7334 + 0.0143 i
0.3566 + 0.0204 i	0.7358 + 0.0071 i
0.3821 + 0.0466 i	0.7550 + 0.0068 i
0.3852 + 0.0250 i	0.7572 + 0.0073 i
0.4106 + 0.0163 i	0.7855 + 0.0065 i
0.4189 + 0.0284 i	0.8124 + 0.0047 i
0.8186 + 0.0042 i	0.8605 + 0.0032 i
0.8723 + 0.0032 i	0.8313 + 0.0040 i
0.8921 + 0.0053 i	0.8313 + 0.0041 i
0.8930 + 0.0049 i	0.8357 + 0.0212 i
0.8936 + 0.0051 i	0.8397 + 0.0193 i
0.8935 + 0.0022 i	0.8540 + 0.0048 i
0.8937 + 0.0023 i	0.8541 + 0.0041 i
0.8940 + 0.0038 i	0.8597 + 0.0039 i

Table 4.5: $\{\lambda \in \sigma(\boldsymbol{A}) : |\lambda| < 1\}$, test case T2.

Chapter 5

Iterative Methods and Preconditioning Techniques

As mentioned in Chapter (4), we need to determine an (or the best) iterative method to solve $\boldsymbol{Ax} = \boldsymbol{b}$, possibly preconditioned.

First we give an overview of the iterative methods examined and next we show their performance on our test cases. We also attempt to investigate which eigenvalues of the spectrum cause the large amount of iterations. Thereafter, we discuss preconditioning and preconditioners.

5.1 Iterative methods

The term *iterative method* refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system at each step. The most effective iterative solvers currently available are multi-gird methods and the class of the Krylov subspace methods [5]. We focus solely on Krylov subspace methods because our discrete system is in integral form and to the best of our knowledge there exist no multi-grid methods for this discretization method.

5.1.1 Stationary iterative methods

Iterative methods that can be expressed in the form

$$\boldsymbol{x}^k = \boldsymbol{B}\boldsymbol{x}^{k-1} + \boldsymbol{c}$$

are called stationary iterative methods. The most common stationary methods are: *Jacobi, Gauss-Seidel, successive over relaxation, and symmetric successive over relaxation*, [5]. These methods converge only for matrices which satisfy certain properties. For instance, point Jacobi gives a better convergence rate for systems in which the coefficient matrix $\boldsymbol{A}$ is strictly or irreducibly diagonally dominant.

5.1.2 Non-stationary iterative methods

The main difference between the stationary and non-stationary methods lies in the fact that in the latter case the computations involve information that changes at each iteration. Typically, constants are computed by taking inner products of residuals or other vectors arising from the iterative method [5]. The iteration matrices associated with the stationary iterative methods are often used as preconditioners of the non-stationary ones. Examples of non-stationary methods include CG, BiCGSTAB, QMR, GMRES and the likes.

(a) GMRES and restarted GMRES (GMRES(m)) GMRES is designed to solve positive definite nonsymmetric linear systems. It has the property of minimizing at every step the norm of the residual vector over a Krylov subspace. The algorithm is derived from the Arnoldi process for constructing an l_2-orthoginal basis of Krylov subspaces. It can be considered as a generalization of the MINRES algorithm [22] and is theoretically equivalent to the Generalized Conjugate Residual (GCR) method [23], [24] and to ORTHODIR [25]; but it presents several advantages over GCR and ORTHODIR. [17]

The most popular form of GMRES is based on the modified Gram-Schmidt procedure, and uses restarts to control storage requirements. If no restarts are used, it will converge in no more than n steps, where n is the order of the matrix. Of course this is of no practical value when n is large; moreover the storage and computational requirements in the absence of restarts are prohibitive. The crucial element for the successful application of GMRES(m) is the choice of m. [5]

(b) Bi-CGSTAB(l) For a number of linear systems of equations arising from realistic problems, using Bi-CGSTAB algorithm [26] to solve these equations is very attractive. Unfortunately a large class of equations, where, for instance, Bi-CG [27] performs well, the convergence of Bi-CGSTAB stagnates. This was observed specifically in case of discretized advection dominated PDE's ([4]). The stagnation is due to the fact that for this type of equations the matrix has almost pure imaginary eigen values. In our case, the coefficient matrices associated with all the test cases seem to have such kind of eigenvalues. Even if Bi-CGSTAB2 [28] attempts to avoid this stagnation it also has its own shortcomings. For that matter, we prefer to use the Bi-CGSTAB(l) algorithm developed by Gerard L.G. Sleijpen and Diederik R. Fokkema [4]. This algorithm generalizes the Bi-CGSTAB algorithm further to handle also problems with negative (real part) eigenvalues and overcome some shortcomings of Bi-CGSTAB2. In some sense, it combines GMRES(m) [17] and Bi-CG and profits from both and seems to converge faster than any of those. [4]

Methods such as GMRES and QMR converge for all real/complex valued matrices because they build a Krylov space (the computational cost grows with each iteration) over which they perform a minimization. However, in general it is not clear for which matrices the restarted versions will converge.

5.2 The amount of iterations

In Table (5.1) we list some iterative methods together with the number of iterations these methods take to find a solution for the linear system in the medium case. Since the medium and hard cases lead to an indefinite matrix $\boldsymbol{A}$, the conditions on $\boldsymbol{A}$ of a few iterative methods are not met and as a result these methods fail to converge. In all cases the maximum number of iteration was set to 1000 and the stopping criteria was $||\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}||_2 < 10^{-7}||\boldsymbol{b}||_2$.

Iterative method	Number of iterations	computational time
CGS	NO conv.	-
QMR	NO conv.	-
TFQMR	NO conv.	-
GMRES	NO conv.	-
GMRESR	115	23.08s
GMRES(m=100)	99	1.915e+3s
BiCGSTAB($l = 2$)	104	2.91s
BiCGSTAB($l = 4$)	40	1.83s

Table 5.1: Iteration of iterative methods: test case T2

Furthermore, the amount of iterations for BiCGSTAB(l) is observed to increase considerably from the easy case to the hard case.

Case	Number of iterations
T1	3
T2	104
T3	173

Table 5.2: BiCGSTAB($l = 2$) iterations

5.3 Spectrum shifting

Since so few methods converge for the test case under consideration, we next examine the spectrum of $\boldsymbol{A}$ i.e., to figure out what might be the cause of this.

Let $0 < \alpha \in \mathbb{R}$. Assume that $\sigma(\boldsymbol{A} + \alpha\boldsymbol{I})$ has more favorable properties than $\sigma(\boldsymbol{A})$ whence one wants to solve

$$(\boldsymbol{A} + \alpha\boldsymbol{I})\boldsymbol{y} = \boldsymbol{b}$$

rather than $\boldsymbol{Ax} = \boldsymbol{b}$.

To this end assume that $\boldsymbol{ARy} = \boldsymbol{b}$. Then

$$\boldsymbol{AR} = \boldsymbol{A} + \alpha \boldsymbol{I} \;\Leftrightarrow\; \boldsymbol{R} = \boldsymbol{I} + \alpha \boldsymbol{A}^{-1}$$

is the required preconditioner. Alternatively one can write

$$\boldsymbol{R} \doteq \boldsymbol{I} + \alpha(\boldsymbol{LU})^{-1}$$

where $\boldsymbol{L}$ and $\boldsymbol{U}$ are (approximate) factors of $\boldsymbol{A}$ such that $\boldsymbol{A} \doteq \boldsymbol{LU}$.

After the calculation of $\boldsymbol{y}$, one can compute

$$\boldsymbol{x} = \boldsymbol{Ry} \doteq [\boldsymbol{I} + \alpha(\boldsymbol{LU})^{-1}]\boldsymbol{y} = \boldsymbol{y} + \alpha(\boldsymbol{LU})^{-1}\boldsymbol{y}.$$

In general one shifts the spectrum such that

a. The small eigenvalues (in modulus) become bigger.

b. The amount of eigenvalues with negative real parts decreases.

In other words, depending on the iterative method, the amount of iterations not only (strongly) depends on $\kappa_2(\boldsymbol{A})$ but it may also depend on other spectral properties such as

1. Small eigenvalues (in modulus) $\lambda_{\min}(\boldsymbol{A})$ of $\sigma(\boldsymbol{A})$ and
2. The amount of eigenvalues in $\{\lambda \in \sigma(\boldsymbol{A}) : \mathrm{Re}(\boldsymbol{A}) < 0\}$.

For the proper choice or construction of iterative methods and possible preconditioners we here examine these properties.

In order to investigate whether the eigenvalues with negative real part cause the large number of iterations in case of T2, we solved

$$(\boldsymbol{A} + 8\boldsymbol{I})\boldsymbol{x} = \boldsymbol{b}, \tag{5.1}$$

and observed that the amount of iterations goes down from 104 to 5, see Figure (5.1).

Figure (5.1) shows that the spectrum $\sigma(\boldsymbol{A} + 8\boldsymbol{I})$ is situated in the right hand side of the complex plane.

Of course the solution $\boldsymbol{x}$ of the shifted system no longer solves $\boldsymbol{Ax} = \boldsymbol{b}$. Moreover, the large shift also shifts $\lambda_{\min}$ away from zero.

We also solved the preconditioned system

$$(\boldsymbol{A} + 8\boldsymbol{I})\boldsymbol{x} = \boldsymbol{Mb},$$

where $\boldsymbol{MA} = \boldsymbol{A} + 8\boldsymbol{I}$ and $\boldsymbol{M} = \boldsymbol{I} + 8\boldsymbol{A}^{-1}$. This new system is equivalent to the original system, but unfortunately the iterative method took 90 iterations to converge.

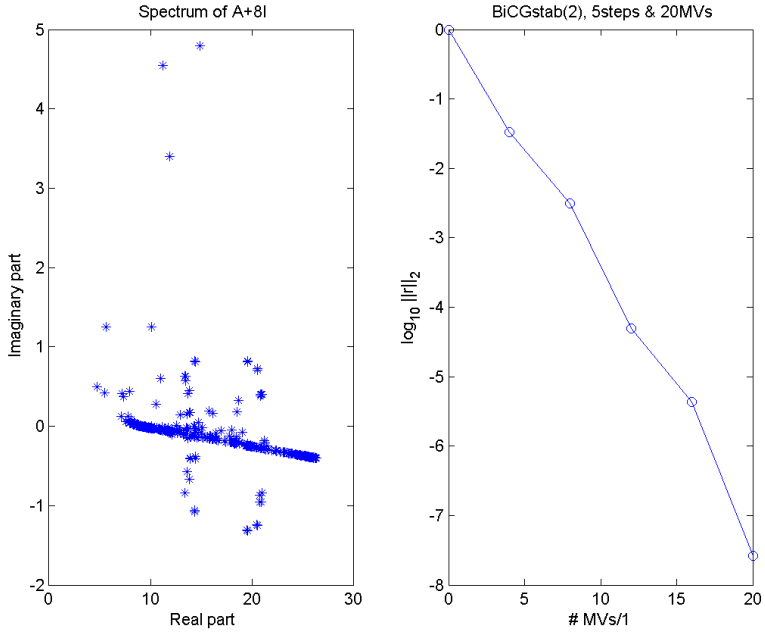


Figure 5.1: Spectrum of $\boldsymbol{A} + 8\boldsymbol{I}$: test case T2.

Furthermore, to see whether the small eigenvalues $\lambda_{\min}$ cause the problem (related to an eigenvector which represents a localized wave (localized in space))between markers), in the sense that hopefully

$$\kappa_2(\boldsymbol{A}_k) \doteq \frac{|\lambda_{\min}(\boldsymbol{A})|}{|\lambda_{\min}(\boldsymbol{A})|}$$

increases, we solved

$$\underbrace{(\boldsymbol{A} - (1 - 1/2^k)\lambda_{\min}(\boldsymbol{A})\boldsymbol{I})}_{=\boldsymbol{A}_{1,k}}\boldsymbol{x} = \boldsymbol{b}, \qquad k = 0, 1, 2, 3 \tag{5.2}$$

(see Table (5.3)) and

$$\underbrace{(\boldsymbol{A} - (1 - 1/2^k)\mathrm{Re}(\lambda_{\min}(\boldsymbol{A}))\boldsymbol{I})}_{=\boldsymbol{A}_{2,k}}\boldsymbol{x} = \boldsymbol{b}, \qquad k = 0, 1, 2, 3 \tag{5.3}$$

(see Table (5.3)). Table (5.3) shows that as the value of k increases the number of

k	iter. (5.2)	iter. (5.3)	cond($\boldsymbol{A}_{1,k}$)	cond($\boldsymbol{A}_{2,k}$)	$\lambda_{\min}(\boldsymbol{A}_{1,k})$	$\lambda_{\min}(\boldsymbol{A}_{2,k})$
1	115	120	1.78e+003	1.15e+003	-0.018+0.23i	-0.025 + 0.039i
5	170	100	1.96e+003	1.33e+003	-0.001+001i	-0.008 + 0.039i
10	303	113	6.29e+005	1.34e+003	-3.53e-5+4.42e-5i	-0.007 + 0.039i
20	515	111	6.45e+008	1.34e+003	-3.45e-8+4.32e-8i	-0.007 + 0.039i

Table 5.3: Iterations for the shifted systems: test case T2

iterations required to solve (5.2) increases linearly while the iteration of (5.3) seems to decrease slowly (but with no clear pattern). In this table we can see that the real part of $\lambda_{\min}$ alone does not affect the linear system. So, by looking at the 2-nd column we can conclude that the small eigenvalues are the ones to be blamed for the large number of iterations.

Table (5.3) indicates that the small eigenvalues may be the ones causing problem. Because zero is not an accumulation point of the eigenvalues (M.V. Beurden) and because there are iterative solver methods which can handle a few (up to 3) small eigenvalues efficiently we next examine what happens if we "leave out" some of the small singular values (eigenvalues). The expectation is that "leaving out" small singular values (eigenvalues) will not improve the amount of iterations much because they are more or less linearly distributed inside $(0, 1]$.

5.4 The small singular values

In this section we estimate the amount of impact of specific eigenvalue(s) (or singular values) on the amount of iterations. In many cases the smallest singular values(s) can have a large impact. To establish the size of the impact, one might need to solve a similar system of equations, but one without the $k \geq 1$ smallest singular values (eigenvalues). Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ be non-singular, to be solved is $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{b} \in \mathbb{C}^n$.

1: Solve $\boldsymbol{A}^*\boldsymbol{x}^* = \boldsymbol{b}^*$ with $\boldsymbol{A}^* \in \mathbb{C}^{n\times n}$.

Let $\boldsymbol{A}$ have a singular value decomposition $\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^{\mathrm{H}}$, with $\Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_n)$ and $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices; i.e., $\boldsymbol{U}\boldsymbol{U}^{\mathrm{H}} = \boldsymbol{I}$ and $\boldsymbol{V}\boldsymbol{V}^{\mathrm{H}} = \boldsymbol{I}$. Assume that the singular values are sorted as: $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. Let $k \in \mathbb{N}$, for all $\boldsymbol{x} \in \mathbb{R}^n$ define the splitting

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{V}^{\mathrm{H}}\boldsymbol{x} = \sum_i \underbrace{(\boldsymbol{v}_i, \boldsymbol{x})_2}_{(\boldsymbol{V}^{\mathrm{H}}\boldsymbol{x})_i} \boldsymbol{v}_i = \underbrace{\sum_{i=1}^{n-k} (\boldsymbol{v}_i, \boldsymbol{x})_2 \boldsymbol{v}_i}_{\boldsymbol{x}^+} + \underbrace{\sum_{i=n-k+1}^{n} (\boldsymbol{v}_i, \boldsymbol{x})_2 \boldsymbol{v}_i}_{\boldsymbol{x}^-}.$$

Likewise, for matrix $\boldsymbol{W} \in \mathbb{C}^{n\times n}$, let $\boldsymbol{W}^+ \in \mathbb{C}^{n\times n-k}$ denote the matrix which contains its first $n-k$ columns and let $\boldsymbol{W}^- \in \mathbb{C}^{n\times k}$ denote the matrix which consists of its last k columns:

$$\boldsymbol{W} = [\boldsymbol{W}^+ | \boldsymbol{W}^-].$$

Let $\boldsymbol{D}$ be a non-singular diagonal matrix given by

$$\boldsymbol{D} = \mathrm{diag}(1, ..., 1, \sigma^{\mathrm{o}}_{n-k+1,n-k+1}, ..., \sigma^{\mathrm{o}}_{n,n}) = \begin{bmatrix} \boldsymbol{I}_k & \\ & \Sigma^{\mathrm{o}} \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Then

$$\begin{aligned}
\boldsymbol{D}\,\boldsymbol{V}^{\mathrm{H}}\boldsymbol{x}^{+} &= \boldsymbol{D}\,\boldsymbol{V}^{\mathrm{H}}\sum_{i=1}^{n-k}(\boldsymbol{v}_i,\boldsymbol{x})_2\boldsymbol{v}_i = \boldsymbol{D}\sum_{i=1}^{n-k}(\boldsymbol{v}_i,\boldsymbol{x})_2\,\boldsymbol{V}^{\mathrm{H}}\boldsymbol{v}_i \\
&= \sum_{i=1}^{n-k}(\boldsymbol{v}_i,\boldsymbol{x})_2\boldsymbol{D}\,\boldsymbol{e}_i = \sum_{i=1}^{n-k}(\boldsymbol{v}_i,\boldsymbol{x})_2\boldsymbol{e}_i \\
&= \boldsymbol{V}^{\mathrm{H}}\boldsymbol{x}^{+}.
\end{aligned}$$

In addition

$$\begin{aligned}
\boldsymbol{A}\boldsymbol{x} &= \boldsymbol{b} \;\Leftrightarrow \\
\boldsymbol{A}\boldsymbol{x}^{+} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{-} \;\Leftrightarrow \\
\boldsymbol{U}\Sigma\,\boldsymbol{V}^{\mathrm{H}}\boldsymbol{x}^{+} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{-} \;\Leftrightarrow \\
\boldsymbol{U}\Sigma\boldsymbol{D}\,\boldsymbol{V}^{\mathrm{H}}\boldsymbol{x}^{+} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{-} \;\Leftrightarrow \\
\boldsymbol{U}\begin{bmatrix} \Sigma^{+} & \\ & \Sigma^{-}\Sigma^{\mathrm{o}} \end{bmatrix}\boldsymbol{V}^{\mathrm{H}}\boldsymbol{x}^{+} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{-} \;\Leftrightarrow \\
\underbrace{\boldsymbol{U}\begin{bmatrix} \Sigma^{+} & \\ & \Sigma^{*} \end{bmatrix}\boldsymbol{V}^{\mathrm{H}}}_{\boldsymbol{A}^{*}}\underbrace{\boldsymbol{x}^{+}}_{\boldsymbol{x}^{*}} &= \underbrace{\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{-}}_{\boldsymbol{b}^{*}}. \\
\boldsymbol{A}^{*}\boldsymbol{x}^{*} &= \boldsymbol{b}^{*}.
\end{aligned} \tag{5.4}$$

For a *normal* matrix $\boldsymbol{A}$ and decomposition $\boldsymbol{A} = \boldsymbol{V}\Sigma\,\boldsymbol{V}^{\mathrm{H}}$ one also obtains the same result.

Now a reasonable choice for Σ^0 is one such that $\Sigma^* = \mathrm{diag}(\sigma^*, ..., \sigma^*)$ where σ^* is one of the first k singular values (already in Σ^+). For this choice $\boldsymbol{A}^*$'s singular values are all of the singular values of $\boldsymbol{A}$ except the k smallest singular values. Similarly, all of $\boldsymbol{A}^*$'s right singular vectors are equal to right singular vectors of $\boldsymbol{A}$. If the singular values have an accumulation point (for a sequence of matrices $\{\boldsymbol{A}_n\}_n$) one might choose σ^* to be an accumulation point.

For the matrices under consideration, it turns out that 1 is an accumulation point of the singular values, whence we choose $\sigma^* = 1$. Numerical experiments for the medium case have been done to solve (5.4), and 397 small singular values have been replaced by 1. Moreover, these experiments show that BiCGSTAB($l = 2$) took 43 steps to converge at the cost of some computational time and the new condition number is 26.5369. Recall that the original system for the medium case is solved after 104 iterations (steps) and the condition number is 972.7855.

2: Solve $\boldsymbol{A}^*\boldsymbol{x}^* = \boldsymbol{b}^*$ with $\boldsymbol{A}^* \in \mathbb{C}^{n-k\times n-k}$.

Note that the solution $\boldsymbol{x}^+$ of $\boldsymbol{A}\boldsymbol{x}^+ = \boldsymbol{b}^*$ is a linear combination of the columns of $\boldsymbol{V}^+$. Therefore, define V^+ to be the linear span of the columns of $\boldsymbol{V}^+$ and observe:

$$\boldsymbol{0} = \min_{\boldsymbol{x}\in\mathbb{C}^n}\{||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^*||_2\} = \min_{\boldsymbol{x}\in V^+}\{||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^*||_2\}.$$

Due to this, the minimizer $\boldsymbol{x}^+ \in V^+$ is a linear combination

$$\boldsymbol{x}^+ = \boldsymbol{V}^+ \boldsymbol{c}$$

where $\boldsymbol{c} \in \mathbb{C}^{n-k}$ satisfies

$$\underbrace{(\boldsymbol{V}^+)^{\mathrm{H}} \boldsymbol{A}\, \boldsymbol{V}^+}_{\boldsymbol{A}^0} \boldsymbol{c} = \underbrace{(\boldsymbol{V}^+)^{\mathrm{H}} \boldsymbol{b}^*}_{\boldsymbol{b}^0},$$

where $\boldsymbol{b}^* = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^-$. This shows that to compute $\boldsymbol{x}^+$ one can use an iterative method to solve

$$\boldsymbol{A}^0 \boldsymbol{c} = \boldsymbol{b}^0. \tag{5.5}$$

Numerical experiments for the medium case show that in solving (5.5) BiCGSTAB($l = 2$) took 29 steps to converge at the cost of some computational time, which is a little more than a quarter of the total amount of iterations required for the original system. Unfortunately, the condition number remained the same for this particular case.

The first numerical test shows that many small singular values (eigenvalues) have to be eliminated to get the iteration count down by a factor of 2.41.

5.5 Preconditioning

The aim of *preconditioning* is to obtain the solution $\boldsymbol{x}$ to the linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with less arithmetical operations, i.e., faster and/or cheaper.

The word *preconditioning* contains the word *condition* which refers to the condition number

$$\kappa(\boldsymbol{A}) = ||\boldsymbol{A}||||\boldsymbol{A}^{-1}||$$

of a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$. The condition number of a matrix is strongly related to

1. the amount of iterations (of iterative methods) for the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.
2. the accuracy of the solution $\boldsymbol{x}$ (for all finite fixed-precision solution methods).

Generally for iterative solution methods, a larger condition number stands for more iterations and a less accurate (final iterand) solution $\boldsymbol{x} = \boldsymbol{x}^k$ which satisfies the relative stoping criterion $||\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}||/||\boldsymbol{b}|| < \epsilon \in \mathbb{R}$. For direct solution methods, the accuracy is influenced in a different manner, not due to the stopping criterion but due to the accumulation of round-off when fixed precision arithmetic is used[1]

[1] which is the case when the standard IEEE double precision or single precision fixed-precision cpu-arithmetic is used. The use of flexible-precision arithmetic (interval-arithmetic) is possible but will lead to longer-precision numbers which makes scalar addition and multiplication more expensive.

Summarizing the main effect of the condition number: A larger condition number requires more floating point operations (FLOPS) to attain the solution $\boldsymbol{x}$ with the same amount of correct decimal places of precision.

With this effect of the condition number in mind, a linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{b},\boldsymbol{x} \in \mathbb{R}^n$$

is called a *preconditioned system* if instead one solves a different system with the same solution, for instance

$$\boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}\underbrace{(\boldsymbol{M}_{\mathrm{R}}^{-1}\boldsymbol{x})}_{y} = \boldsymbol{M}_{\mathrm{L}}\boldsymbol{b}, \qquad \boldsymbol{M}_{\mathrm{L}}, \boldsymbol{M}_{\mathrm{R}} \in \mathbb{C}^{n\times n}.$$

Together the linear operators (matrices) $\boldsymbol{M}_{\mathrm{L}}, \boldsymbol{M}_{\mathrm{R}}$ are called *preconditioners*, the idea is that the coefficient matrix

$$\boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}$$

has a (much) smaller condition number:

$$\kappa(\boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}) \ll \kappa(\boldsymbol{A})$$

such that the solution $\boldsymbol{x}$ can be calculated with less FLOPS (i.e., cheaper/faster).

Unfortunately, section (5.4) showed that this classical definition preconditioning (via the condition number) is not too useful, since in section (5.4) the condition number decreases to a factor of 34.96 for the numerical test using (5.4) whereas the amount of iterations decreases only to a factor of 2.41. On the other hand for the test which uses (5.5) the condition number remained the same but the amount of iterations decreases to a factor of 3.58.

5.6 Iterative methods and amount of FLOPS

Let us determine the amount of FLOPS required for the iterative solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

Assume one uses an iterative method (fixed point type) to determine the solution $\boldsymbol{x}$:

$$\boldsymbol{x}^{(0)} \in \mathbb{C}^n, \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{G}(\boldsymbol{x}^{(k)}), \qquad k = 1, 2, ...,$$

with the stopping criterion $k > k_{\max}$ and/or

$$||\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}||_2 < \epsilon,$$

where $k_{\max} \in \mathbb{N}$ and $0 < \epsilon < 1$ are predetermined numbers.

A typical example of a simple iterative method is Richardson's method

$$\boldsymbol{x}^0 \in \mathbb{C}^{n\times n}, \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \tau(\boldsymbol{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}) = \boldsymbol{G}(\boldsymbol{x}^{(k)}), \qquad k = 1, 2, ...$$

which is a fixed point method for

$$\boldsymbol{G}(\boldsymbol{x}) = (\boldsymbol{I} - \tau\boldsymbol{A})\boldsymbol{x}^{(k)} + \tau\boldsymbol{b},$$

where τ must be determined in advance - for the sake of convenience assume that $\tau = 1$. The related stopping criterion

$$||\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}||_2 = ||\boldsymbol{x}^{(k)} - \tau(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}) - \boldsymbol{x}^{(k)}||_2 = ||\boldsymbol{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}||_2 < \epsilon,$$

i.e., one stops if the k-th *residual* $||\boldsymbol{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}||_2$ is smaller than ϵ.

With this iterative method in mind, the total amount of FLOPS depends on:

1. The amount of FLOPS to construct $\boldsymbol{A}$, $\mathsf{construct}\ (\boldsymbol{A})$ - once
2. The amount of FLOPS per iteration (i.e., the amount of FLOPS for $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ and vector operations);
3. The amount of iterations until $||\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}||_2 < \epsilon$.

For a preconditioned method, the extra costs are

1. The amount of FLOPS to construct $\boldsymbol{M}_{\mathrm{L}}, \boldsymbol{M}_{\mathrm{R}}$, $\mathsf{construct}\ (\boldsymbol{M}_{\mathrm{L}}) + \mathsf{construct}\ (\boldsymbol{M}_{\mathrm{R}})$ - once;
2. The amount of FLOPS per iteration (i.e., the amount of FLOPS for $\boldsymbol{x} \mapsto \boldsymbol{M}_{\mathrm{L}}\boldsymbol{x}$ and $\boldsymbol{x} \mapsto \boldsymbol{M}_{\mathrm{R}}\boldsymbol{x}$);
3. The amount of iterations until $||\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}||_2 < \epsilon$.

5.7 The calculation of $\mathbf{x} \mapsto \mathbf{M_L A M_R x}$

In the preconditioned case one has to calculate $\boldsymbol{x} \mapsto \boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}\boldsymbol{x}$. Because matrix-vector and vector-vector operations on complex valued matrices and vectors are associative, implementations

$$((\boldsymbol{M}_{\mathrm{L}}(\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}))\boldsymbol{x}) \tag{5.6}$$

and

$$(\boldsymbol{M}_{\mathrm{L}}(\boldsymbol{A}(\boldsymbol{M}_{\mathrm{R}}\boldsymbol{x}))) \tag{5.7}$$

lead to identical results with exact arithmetic. On a computer these implementations differ considerably:

- Implementation (5.6) first calculates the entire product matrix $\boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}$ which must fit into computer RAM memory before the multiplication $\boldsymbol{x} \mapsto \boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}\boldsymbol{x}$ can take place. Even if $\boldsymbol{A},\ \boldsymbol{M}_{\mathrm{L}},\ \boldsymbol{M}_{\mathrm{R}} \in \mathbb{C}^{n\times n}$ are sparse the product $\boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}$ can be a full matrix. i.e., $\boldsymbol{x} \mapsto \boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}\boldsymbol{x}$ can cost $\mathcal{O}(n^3)$ FLOPS and take a lot of CPU-time.

- Implementation (5.7) calculates three matrix - vector products which will cost $\mathcal{O}(n)$ FLOPS if $\boldsymbol{A}, \boldsymbol{M}_{\mathrm{L}}$ and $\boldsymbol{M}_{\mathrm{R}}$ are sparse.

Even if both implementations can be realized (enough RAM and CPU-time) both will lead to a different answer $\boldsymbol{M}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}\boldsymbol{x}$ since the standard (employed) IEEE computer arithmetic is finite precision. The finite precision scalar multiplication and additions are both associative but still lead to a different answer because in both cases the aforementioned operations are performed with different numbers and in a different order, which leads (in general) to different round-off.

5.8 A preconditioner overview

Most existing preconditioners can be categorized as *implicit* or *explicit*. A preconditioner is implicit if its application, within each step of the chosen iterative method, requires the solution of a linear system. An example of an *implicit* preconditioner $\boldsymbol{M}$ is $\boldsymbol{M} = (\bar{\boldsymbol{L}}\bar{\boldsymbol{U}})^{-1}$ where $\bar{\boldsymbol{L}}$ and $\bar{\boldsymbol{U}}$ are triangular matrices obtained by an incomplete LU factorization of $\boldsymbol{A}$. The application of $\boldsymbol{x} \mapsto \boldsymbol{M}\boldsymbol{x}$ requires solving triangular systems. For an explicit preconditioner, $\boldsymbol{x} \mapsto \boldsymbol{M}\boldsymbol{x}$ can be calculated without solving systems (all entries of $\boldsymbol{M}$ are explicitly available). Algorithms with explicit preconditioners are easier to parallelize than algorithms with implicit preconditioners. Furthermore, the construction of some types of approximate inverse preconditioners can be performed in parallel. For these reasons, sparse approximate preconditioners are becoming popular as an alternative to more traditional implicit techniques.

5.8.1 Implementation of preconditioners

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be non-singular and assume that a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, also non-singular, is close to $\boldsymbol{A}$ ($\|\boldsymbol{A} - \boldsymbol{M}\|$ is small), then the transformed system

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b},$$

has the same solution as the original system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, but the spectral properties of its coefficient matrix $\boldsymbol{M}^{-1}\boldsymbol{A}$ could be more favorable. From a practical point of view, the only requirement for $\boldsymbol{M}$ is that it is inexpensive to solve linear systems $\boldsymbol{M}\boldsymbol{u} = \boldsymbol{v}$. This is because the preconditioned algorithms will all require a linear system solution with the matrix $\boldsymbol{M}$ at each step.

Standard preconditioning of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with the use of $\boldsymbol{M}$ is :

1. Left preconditioning: $\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$.

2. Right preconditioning: $\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{y} = \boldsymbol{b}$, with $\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{y}$.

3. Two-sided (split) preconditioning: $\boldsymbol{M}$ is factored as $\boldsymbol{M} = \boldsymbol{M}_{\mathrm{L}}\boldsymbol{M}_{\mathrm{R}}$. Then

$$\boldsymbol{M}_{\mathrm{L}}^{-1}\boldsymbol{A}\boldsymbol{M}_{\mathrm{R}}^{-1}\boldsymbol{y} = \boldsymbol{M}_{\mathrm{L}}^{-1}\boldsymbol{b},$$

   This form of preconditioning may be used for preconditioners that come in factored form. It can be seen as a compromise between left- and right- preconditioning.[2]

5.8.2 Preconditioners from stationary iterative methods

Consider a matrix $\boldsymbol{A}$ given by

$$\boldsymbol{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix},$$

Then we can write $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{E} - \boldsymbol{F}$, where

$$\boldsymbol{D} = \begin{pmatrix} a_{1,1} & & \\ & \ddots & \\ & & a_{n,n} \end{pmatrix}, \qquad \boldsymbol{E} = -\begin{pmatrix} 0 & 0 & \dots & 0 \\ a_{2,1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & 0 \end{pmatrix},$$

$$\boldsymbol{F} = -\begin{pmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ 0 & 0 & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

with $-\boldsymbol{E}$ and $-\boldsymbol{F}$ respectively being the strictly lower and strictly upper triangular parts of $\boldsymbol{A}$

1. Point Jacobi : $\boldsymbol{M}_{JA} = \boldsymbol{D}$. This preconditioner is implicit if one stores $\boldsymbol{D}$ and explicit if one stores $\boldsymbol{D}^{-1}$.

2. Block Jacobi

   a. Non-overlapping block Jacobi (see Figure (5.2) (a)).
   Block versions of the Jacobi preconditioner can be derived by a partitioning of the variables. If the index set $S = \{1, 2, ..., n\}$ is partitioned as $S = \bigcup_i S_i$ with the sets S_i mutually disjoint, then

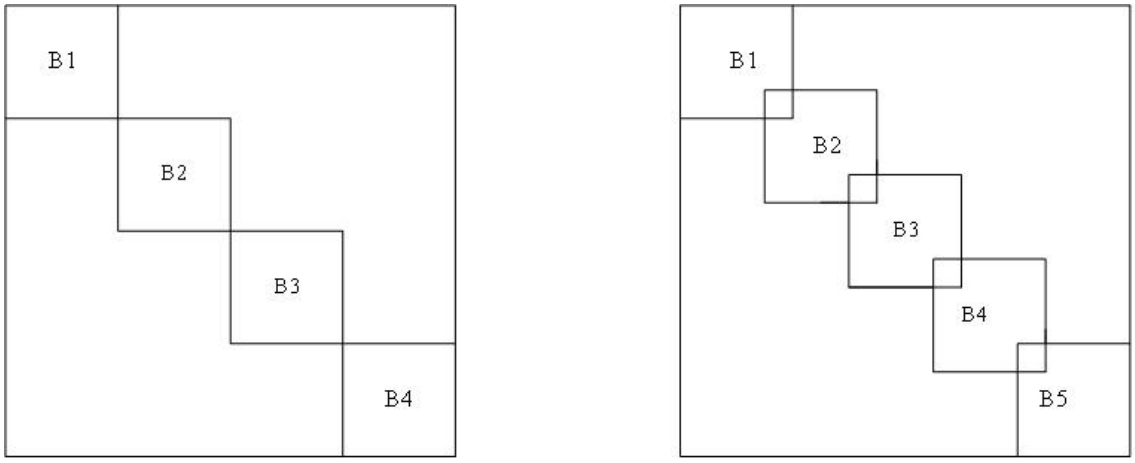
$$m_{ij} = \begin{cases} a_{ij} & \text{if i and j are in the same index set} \\ 0 & \text{otherwise} \end{cases}$$

   the preconditioner, $\boldsymbol{M} = [m_{ij}]$, is now a non-overlapping block-diagonal matrix.

b. Overlapping block Jacobi (see Figure (5.2) (b)) - in this case the overlapping index sets are defined by

$$S_i = \{j \mid l_i \leq j \leq r_i\}$$

where $l_1 = 1,\ r_p = n,\ l_{i+1} < r_i\ 1 \leq i \leq p-1$, and p is the number of blocks [1]. Jacobi preconditioners need very little storage, even in the block case, and they are easy to implement.



(a) Non-overlapping (b) Overlapping

Figure 5.2: Block Jacobi preconditioners.

3. Gauss - Seidel : $\boldsymbol{M}_{\text{GS}} = \boldsymbol{D} - \boldsymbol{E}$.

4. Successive over relaxation - is derived from Gauss-Seidel by introducing an extrapolation (relaxation) parameter ω:

$$\boldsymbol{M}_{\text{SOR}} = \frac{1}{\omega}(\boldsymbol{D} - \omega\boldsymbol{E}).$$

The constant coefficient ω, the relaxation parameter uniformly scales the equations of the preconditioned system.

5. Symmetric SOR (SSOR) - performs better than the SOR preconditioner when applied to non-stationary iterative methods.

$$\boldsymbol{M}_{\text{SSOR}} = \frac{1}{\omega(2-\omega)}(\boldsymbol{D} - \omega\boldsymbol{E})\boldsymbol{D}^{-1}(\boldsymbol{D} - \omega\boldsymbol{F}),$$

where $-\boldsymbol{F}$ is the strictly upper triangular part of $\boldsymbol{A}$.

6. ROWNORM scaling : $\boldsymbol{M} = \text{diag}(1/r_i(\boldsymbol{A}))$, where $r_i(\boldsymbol{A})$ stands for the norm of the i-th row of $\boldsymbol{A}$. This form of scaling is used to ensure that all the rows of the new coefficient matrix have the same norm.

Block versions of GS, SOR and SSOR are obtained from a block partitioning of $\boldsymbol{A}$. Let $\boldsymbol{A}$ be partitioned in the form

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \dots & \boldsymbol{A}_{1,n} \\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} & \dots & \boldsymbol{A}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{n,1} & \boldsymbol{A}_{n,2} & \dots & \boldsymbol{A}_{n,n} \end{pmatrix},$$

where each diagonal block $\boldsymbol{A}_{i,i}$ is square and non-singular. From this partitioning, one can define the block matrices

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{A}_{1,1} & & \\ & \ddots & \\ & & \boldsymbol{A}_{n,n} \end{pmatrix}, \qquad \boldsymbol{E} = -\begin{pmatrix} \boldsymbol{O} & \boldsymbol{O} & \dots & \boldsymbol{O} \\ \boldsymbol{A}_{2,1} & \boldsymbol{O} & \dots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{n,1} & \boldsymbol{A}_{n,2} & \dots & \boldsymbol{O} \end{pmatrix},$$

$$\boldsymbol{F} = -\begin{pmatrix} \boldsymbol{O} & \boldsymbol{A}_{1,2} & \dots & \boldsymbol{A}_{1,n} \\ \boldsymbol{O} & \boldsymbol{O} & \dots & \boldsymbol{A}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \dots & \boldsymbol{O} \end{pmatrix}$$

where the matrices $-\boldsymbol{E}$ and $-\boldsymbol{F}$ respectively are the strictly lower and strictly upper triangular parts of $\boldsymbol{A}$. Therefore, one can define the block versions of GS, SOR, and SSOR in the same way as above.

For our matrix $\boldsymbol{A}$, several block preconditioners are "natural": 3×3 blocks (the three components of the electromagnetic field), $N_z \times N_z$ blocks, etc.

Chapter 6

Sparse Approximate Preconditioners

6.1 Sparse approximate inverses (SPAI) using Frobenius

The purpose of this kind of preconditioning is to construct a sparse explicit preconditioner $\boldsymbol{M}$, such that $\boldsymbol{AM}$ is close to the identity matrix $\boldsymbol{I}$ as much as possible or mathematically one can write $\boldsymbol{AM} \doteq \boldsymbol{I}$.

6.1.1 Frobenius norm minimization methods

Often Frobenius norm minimization is used to compute explicit preconditioners. Here we simply need to find the sparse approximate inverse as the matrix $\boldsymbol{M}$ which minimizes $\|\boldsymbol{I}\text{-}\boldsymbol{AM}\|_F$ for right preconditioning, subject to some sparsity constraints. The main reason behind choosing the Frobenius norm minimization method is because it allows the decoupling of the constrained minimization problem into n independent linear least-squares problems, where n stands for the order of the matrix $\boldsymbol{A}$ ([3], [9]). The n problems are due to the fact that we need to give one least square problem for each column of $\boldsymbol{M}$ (when preconditioning from the right) or row of $\boldsymbol{M}$ (when preconditioning from the left). The minimization problem can then be decoupled into the sums of the squares of the 2-norms of the individual columns of the residual matrix $\boldsymbol{I}\text{-}\boldsymbol{AM}$ as

$$\min_{M} \|\boldsymbol{I}\text{-}\boldsymbol{AM}\|_F^2 = \sum_{j=1}^{n} \min_{m_j} \|\boldsymbol{e}_j - \boldsymbol{Am}_j\|_2^2, \qquad j = 1, 2, ..., n \tag{6.1}$$

where $\boldsymbol{e}_j$ and $\boldsymbol{m}_j$ are the jth column vectors of the matrices $\boldsymbol{I}$ and $\boldsymbol{M}$, respectively. The number of unknowns for each problem being equal to the number of nonzeros allowed in each column (or row) of $\boldsymbol{M}$. The resulting least squares problems can be solved independently of each other; because the minimization problem (6.1) is equivalent to minimizing the individual functions

$$\min_{m_j} \|\boldsymbol{e}_j - \boldsymbol{Am}_j\|_2, \qquad j = 1, 2, ..., n. \tag{6.2}$$

In Frobenius norm minimization a good selection of the nonzero pattern of $\boldsymbol{M}$ plays a great role in solving the least squares problems. One should keep in mind that $\boldsymbol{M}$

has to be reasonably sparse while containing the "large" entries of the inverse, which are expected to contribute the most to the quality of the preconditioner [21]. The simplest criterion to select a nonzero pattern is to choose a threshold parameter $\epsilon \in (0, 1)$ and to include the position (i, j) in the nonzero pattern of $\boldsymbol{M}$ if

$$|a_{i,j}| > \epsilon \cdot \max_{\hat{i}} |a_{\hat{i}j}|. \tag{6.3}$$

Defining a sparsity pattern a priori does not guarantee an actual reduction in the cost for the construction of this type of preconditioners. Especially when compared to standard implicit techniques like ILU, preconditioners based on Frobenius norm minimization always take more time [3].

6.1.2 Sparsity pattern and construction of a preconditioner

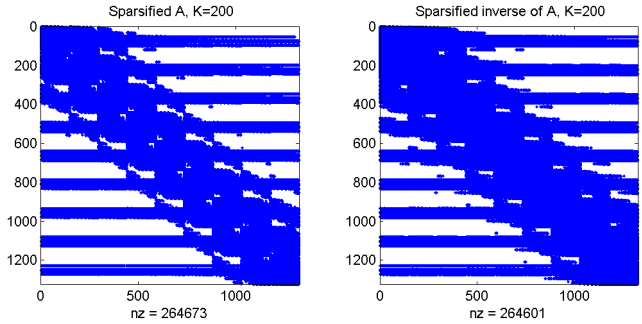
Let us define the nonzero pattern of $\boldsymbol{M}$ as a set $\mathcal{B} \subseteq \{(i, j) : 1 \leq i, j \leq n\}$ such that $m_{ij} = 0$ if $(i, j) \notin \mathcal{B}$. Fix j and consider the set $J = \{i : (i, j) \in \mathcal{B}\}$, which specifies the nonzero pattern of $\boldsymbol{m}_j$. Let the nonzero entries of $\boldsymbol{m}_j$ be denoted by $\hat{\boldsymbol{m}}_j = \boldsymbol{m}_j(J)$. It is easy to see that the columns of $\boldsymbol{A}$ that enter in the definition of $\hat{\boldsymbol{m}}_j$ are those whose index is in J. Let $\boldsymbol{A}(:, J)$ be the sub-matrix of $\boldsymbol{A}$ formed with such columns. Depending on the sparsity of $\boldsymbol{A}$ the sub matrix $\boldsymbol{A}(:, J)$ is expected to have some zero rows; for that matter, let T be the set of indices of nonzero rows of $\boldsymbol{A}(:, J)$. Then after removing the zero rows, we can restrict our attention to the matrix $\hat{\boldsymbol{A}} = \boldsymbol{A}(T, J)$, to the unknown vector $\hat{\boldsymbol{m}}_j$ and to the right-hand side $\hat{\boldsymbol{e}}_j = \boldsymbol{e}_j(T)$. Therefore, the individual minimization problem (6.2) is reduced to the following least squares problem [3]

$$\min_{\hat{\boldsymbol{m}}_j} \|\hat{\boldsymbol{e}}_j - \hat{\boldsymbol{A}}\hat{\boldsymbol{m}}_j\|_2, \qquad j = 1, 2, ..., n. \tag{6.4}$$

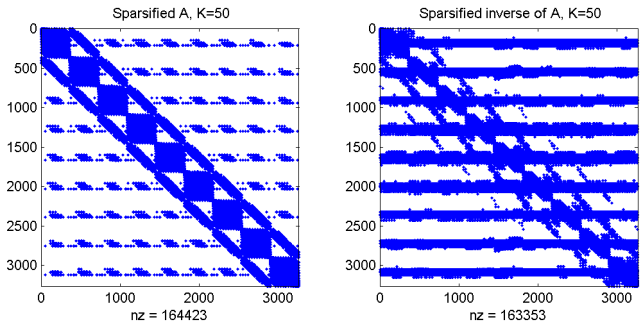
There are a variety of methods available to solve (6.4). One approach is to use a dense QR - factorization of $\boldsymbol{A}(T, J)$ (recommended for large matrices). It is obvious that the complexity of the least squares solve will be reduced if $\boldsymbol{A}$ is replaced by a sparse approximation of $\boldsymbol{A}$. [9]

Algebraic strategies for sparsity pattern

In selecting a suitable sparsity pattern for $\boldsymbol{M}$, the sparsity pattern of $\boldsymbol{A}$ is often suggested, because it is not possible to determine the structure of $\boldsymbol{A}^{-1}$ (without computing it) [9] or because $\boldsymbol{A}^{-1}$ is a full matrix. By numerical experiments we determined the location of the large entries of $\boldsymbol{A}$ and $\boldsymbol{A}^{-1}$. From the experiments we concluded that for the easy test case the large entries of $\boldsymbol{A}$ tend to be located in the positions corresponding to the large entries of $\boldsymbol{A}^{-1}$, see Figure (6.1) (a). But for the medium test case there is less similarity, see Figure (6.1) (b).



(a)



(b)

Figure 6.1: Sparsified $\boldsymbol{A}$, $\boldsymbol{A}^{-1}$ for the test cases (a) T1 (b) T2

Below we give the descriptions of some techniques that help us prescribe $\boldsymbol{B}$, the sparsity pattern of $\boldsymbol{M}$.

1. For a fixed positive integer k, with $k \ll n$, find the k largest entries (in modulus) in each column of $\boldsymbol{A}$. Then $\boldsymbol{B}$ is the set of positions (i, j) of these entries.

2. $\boldsymbol{B}$ is the set of positions (i, j) such that a_{ij} satisfies equation (6.3).

3. The combination of 1 and 2;

Compared to other kinds of heuristics, the ones given above have obvious advantages from the point of view of parallel implementation. The first heuristic has the additional

advantage that the amount of nonzeros in the approximate inverse is controlled by the user. A drawback common to all the heuristics is the need to find good values of the parameters involved.

Sparsifying techniques

(a) a column-wise thresholding (dropping) strategy.

$$c_{ij} = \begin{cases} a_{ij} & \text{if } |a_{ij}| > \epsilon \cdot \max_{\hat{i}} |a_{\hat{i}j}| \qquad 1 \leq \hat{i}, j \leq n \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

(b) pick the k largest (in modulus) elements from each column of A.

$$c_{ij} = \begin{cases} a_{ij} & \text{if } |a_{ij}| >= Ak \qquad 1 \leq i, j \leq n \\ 0 & \text{otherwise,} \end{cases} \tag{6.6}$$

where Ak is the k-th largest element in column j.

(c) combination of the above two.

$$c_{ij} = \begin{cases} a_{ij} & \text{if } |a_{ij}| > \epsilon \cdot \max_{\hat{i}} |a_{\hat{i}j}|, \quad i \in J \quad \text{and } 1 \leq \hat{i}, j \leq n \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

where J indexes the k largest elements (in modulus) in column j of A. Thus in each column we only take the k largest elements, provided they are above the threshold given in (6.3).

Construction

We construct the preconditioner $\boldsymbol{M}$ as follows:

1. Compute $\boldsymbol{G} \approx \boldsymbol{A}$ using one of the above techniques.
2. Compute $\boldsymbol{C} \approx \boldsymbol{A}$ using the same heuristics as above either using the same parameters ϵ, k or different ones.
3. Compute sparse approximate inverse $\boldsymbol{M}$ of $\boldsymbol{G}$ using (6.4) where the sparsity pattern of $\boldsymbol{M}$ is chosen to be the same as that of $\boldsymbol{C}$.

6.2 Sparse approximate preconditioners using ILUT(τ)

To construct a sparse approximate preconditioner we use the sparsifying techniques (6.5) - (6.7).

6.2.1 Construction and application

- sparsify $\boldsymbol{A}$ with some threshold parameter ϵ, and denote the new matrix by $\boldsymbol{C}$.
- factorize $\boldsymbol{C}$ using an incomplete LU factorization with a drop tolerance τ
- the product of the inverses of $\boldsymbol{L}$ and $\boldsymbol{U}$ can now be used as a left (right) preconditioner.
- one can also use central preconditioning as $\boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{U}^{-1}$.

Chapter 7

Numerical Results

The BiCGSTAB(l) iterative method is used to test different kinds of preconditioners for the system $\boldsymbol{Ax} = \boldsymbol{b}$, where the coefficient matrix $\boldsymbol{A}$ and the right-hand-side vector, $\boldsymbol{b}$, are built by the VIM algorithm (note that VIM also uses BiCGSTAB(l)). Some of the parameters have been set as follows; the suitable initial guess was assumed to be the null (zero) vector, the stopping criteria (tolerance), $\epsilon = 1e - 7$ is taken and the maximum number of iterations is set to 1000 and also $l = 2$ has been used. In all cases left preconditioning is chosen unless indicated otherwise.

The VIM algorithm requires 14 and 400 matrix-vector multiplications, respectively, for the easy and medium test cases; therefore decisions on choosing the best preconditioner are made based on these two results and the ones that are presented in the following tables.

In the following tables aware of the following notations

1. iter. - number of iterations
2. relres - relative residual which is defined by $||\boldsymbol{b} - \boldsymbol{Ax}||/||\boldsymbol{b}||$
3. t_{solve} - time needed to solve the system
4. $t_{\text{const.}}$ - time needed to construct the corresponding preconditioner
5. t_{inv} - time needed to invert the preconditioner
6. t_{invsp} - time needed to invert the sparsified coefficient matrix
7. t_{LU} - time needed to compute an incomplete LU factorization of the sparsified matrix
8. $\boldsymbol{M}_{\text{D}}$ - density of the preconditioner $\boldsymbol{M}$ where density is defined by the number of non-zero entries divided by N^2 (N order of the matrix)
9. sp. $\boldsymbol{M}_{\text{D}}^{-1}$ - density of the sparsified inverse of $\boldsymbol{M}$
10. sp. $\boldsymbol{A}_{\text{D}}$ - density of the sparsified coefficient matrix $\boldsymbol{A}$
11. $\boldsymbol{L}_{\text{D}}$ - density of the matrix $\boldsymbol{L}$
12. $\boldsymbol{U}_{\text{D}}$ - density of the matrix $\boldsymbol{U}$

7.1 Numerical results for Jacobi, GS, SOR, and SSOR

The main objective of this report, as explained before, is to find a preconditioner which can drastically minimize the number of iterations and the amount of matrix-vector multiplications (MVs).

In the following two tables you find expressions like OV block Jacobi(n, m) which means the preconditioner is an overlapping block Jacobi where each block is of order n, and m is the order of the matrix where the Jacobi blocks intersect. In addition, NOV block Jacobi(n) stands for a non-overlapping block diagonal matrix of order n.

As one can see from Table (7.1), almost all of the preconditioners ended up with three iteration and 12 MVs, which is a little bit better than the VIM algorithm which took 14 MVs. But, for OV block Jacobi (400,50) and SSOR preconditioners the iterative method took 2 iterations and 8 MVs which is good compared to the others. The former takes less construction time but the latter guarantees a very small relative residual.

For the medium test case the VIM algorithm needs 400 MVs to converge. By looking at the results given in Table (7.2) the non-overlapping block Jacobi with block size 2000 seems to perform better than the rest of them, since it took only 24 steps and 96 MVs, this makes it a good preconditioner so far.

7.1.1 Easy case, test case T1

Preconditioner	iter.	MVs	relres	t_{solve}	$t_{\text{const.}} + t_{\text{inv.}}$
None	3	12	2.3104e-008	0.016s	-
Point Jacobi	3	12	1.0015e-009	0.015s	0.06s
Overlapped(OV) block Jacobi(400,50)	2	8	2.3298e-008	0.156s	1.17s
Non-OV (NOV) block Jacobi(400)	3	12	1.6903e-010	0.234s	1.16s
Gauss - Seidel(GS)	3	12	1.0080e-009	0.219s	0.60s
SOR, $\omega = 0.97$	3	12	1.0081e-009	0.235s	0.64s
SSOR, $\omega = 1$	2	8	2.0367e-012	0.141s	4.07s
Rownorm scaling	4	16	5.348e-008	0.156s	0.07s

Table 7.1: Test case T1

7.1.2 Medium case, test case T2

Preconditioner	iter.	MVs	relres	t_{solve}	$t_{\text{const.}} + t_{\text{inv.}}$
None	104	416	8.5172e-008	2.91s	-
Point Jacobi	-	-	-	-	0.34s
NOV block Jacobi(700)	55	220	3.7093e-008	22.4s	12.75s
NOV block Jacobi(2000)	24	96	2.2380e-008	9.59s	12.93s
OV block Jacobi(2000,50)	40	160	6.4274e-008	16.3s	12.33s
OV block Jacobi(700,10)	67	268	9.6653e-008	27.1s	12.94s
Gauss Seidel(GS)	120	480	2.9936e-008	48.6s	5.37s
SOR, $\omega = 0.97$	124	496	3.4109e-008	50.4s	5.39s
SSOR, $\omega = 1$	57	228	6.0315e-008	23.2s	34.29s
Rownorm scaling	138	552	8.6739e-008	29.6s	0.39s

Table 7.2: Test case T2

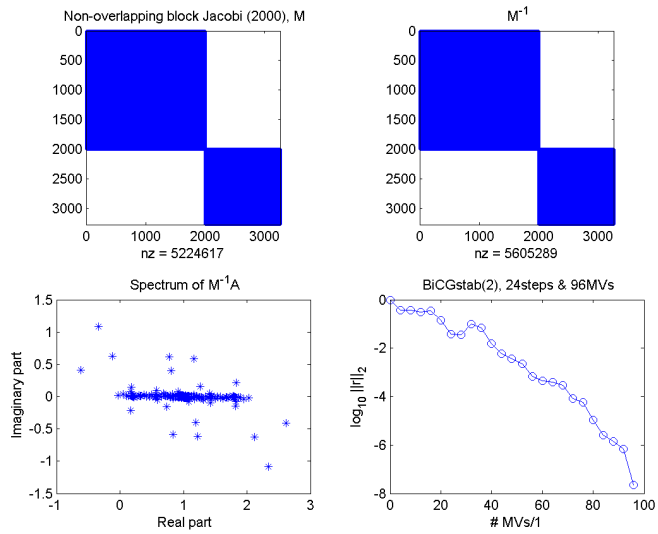


Figure 7.1: Non-overlapping block Jacobi (2000).

In Tables (7.1) and (7.2) the numerical results corresponding to preconditioners obtained from stationary iterative methods are presented. We want to give emphasis to the block Jacobi preconditioners, from Table (7.2) we can see that, given the same size, the non-overlapping blocks perform better than the overlapping ones. In spite of their popularity for sparse systems, preconditioners like point Jacobi and SSOR seems to take more iterations than the block Jacobi preconditioners.

We can also see in Table (7.2) that some preconditioners made the condition of the system much worse which is implied by the number of iterations the iterative method needed to converge. It is also presented that the iterative method failed to converge in the case of point Jacobi.

7.2 Numerical results for SPAI Frobenius

In the following tables you find the expression sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$, it means the density of the sparsified coefficient matrix $\boldsymbol{A}$ and that of the preconditioner $\boldsymbol{M}$ are equal.

Tables (7.3) - (7.8) show the amount of iterations for Frobenius based preconditioners using BiCGSTAB($l = 2$) and the sparsification techniques (6.5) - (6.7).

7.2.1 Easy case, test case T1

ϵ	sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\text{const.}}$
7e-2	0.23 %	2	8	9.1831e-008	0.156s	0.98s
4e-4	34.10 %	1	4	8.2478e-008	0.062s	849.85s

Table 7.3: Frobenius SPAI, technique (6.5).

k	sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\text{const.}}$
4	0.30 %	2	8	6.2687e-008	0.172s	1.33s
500	37.80 %	1	4	7.7450e-008	0.078s	836.24s

Table 7.4: Frobenius SPAI, technique (6.6).

k, ϵ	sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\text{const.}}$
4, 2e-2	0.27 %	2	8	8.4408e-008	0.156s	1.49s
500, 3e-4	33.10 %	1	4	9.2428e-008	0.078s	755.38s

Table 7.5: Frobenius SPAI, technique (6.5).

7.2.2 Medium case, test case T2

ϵ	sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
5e-2	1.44%	67	268	7.8692e-008	27.2s	66.22s
1e-2	6.15%	55	220	6.7801e+008	22.1s	2.03e+003s

Table 7.6: Frobenius SPAI, technique (6.5).

k	sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
100	3.07%	69	276	7.2182e-008	28.1s	112.23s
300	9.20%	57	228	6.6517e-008	23.2s	1.91e+003s

Table 7.7: Frobenius SPAI, technique (6.6).

k, ϵ	sp. $\boldsymbol{A}_{\mathrm{D}} = \boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
100, 1e-3	3.07%	69	276	7.2182e-008	28.1s	114.46s
200, 1e-2	3.10%	56	224	5.3995e-008	22.8s	201.09s

Table 7.8: Frobenius SPAI, technique (6.7).

7.2.3 Interpretation of the numerical results

From Table (7.3) one can see that the quality of the preconditioner increases as the value of the threshold parameter ϵ decreases, where the quality of the preconditioner is measured by the number of iterations and the time the iterative method took to find one accurate solution.

By looking at Table (7.4) one can observe that the quality of the preconditioner increases with the value of k, the number of largest entries from each column one wants to include in the sparsity pattern.

Table (7.5) is where we used the combination of the two previous strategies. It is observed that for large values of k and relatively small values of ϵ it is possible to obtain quality preconditioners. Moreover, by fixing k one can play with the values of ϵ (and vise versa) until a desired quality is achieved.

7.3 Numerical results for ILUT(τ) preconditioners

Tables (7.9) - (7.15) show the amount of iterations for ILUT(τ) based preconditioners using BiCGSTAB($l = 2$) and the sparsification techniques (6.5) - (6.7).

7.3.1 Easy case, test case T1

ϵ, τ	sp. $\boldsymbol{A}_{\mathrm{D}}$	$\boldsymbol{L}_{\mathrm{D}}$	$\boldsymbol{U}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	t_{LU}
7e-2, 7e-2	0.23	0.15 %	0.15 %	2	8	6.3178e-008	0.015s	0.19s
8e-4, 2e-4	22.52 %	13.25%	14.66%	1	4	9.3505e-008	0.031s	2.77s

Table 7.9: ILUT(τ), technique (6.6).

k, τ	sp. $\boldsymbol{A}_{\mathrm{D}}$	$\boldsymbol{L}_{\mathrm{D}}$	$\boldsymbol{U}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	t_{LU}
10, 1e-2	0.77 %	0.34%	0.33%	2	8	2.3291e-008	0s	0.31s
400, 1e-4	30.24 %	18.12%	20.85%	1	4	8.6090e-008	0.031s	3.86s

Table 7.10: ILUT(τ), technique (6.6).

k, ϵ, τ	sp. $\boldsymbol{A}_{\mathrm{D}}$	$\boldsymbol{L}_{\mathrm{D}}$	$\boldsymbol{U}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	t_{LU}
10, 7e-2, 1e-2	0.23 %	0.34%	0.33 %	2	8	6.3178e-008	0.015s	0.50s
400, 3e-4, 1e-4	28.28 %	18.12%	20.85%	1	4	9.0312e-008	0.047s	4.02s

Table 7.11: ILUT(τ), technique (6.7).

7.3.2 Medium case, test case T2

ϵ, τ	sp. $\boldsymbol{A}_{\mathrm{D}}$	$\boldsymbol{L}_{\mathrm{D}}$	$\boldsymbol{U}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	t_{LU}
3e-3, 1e-3	13.08%	10.34%	13.52%	10	40	3.0612e-009	4.2s	27.84s
9e-4, 6e-4	24.07%	14.38%	18.85%	5	20	2.1076e-010	3.59s	44.16s
6e-5, 1e-5	52.11%	35.03%	44.98%	1	4	8.2678e-012	2.8s	118.79s

Table 7.12: ILUT(τ), technique (6.6).

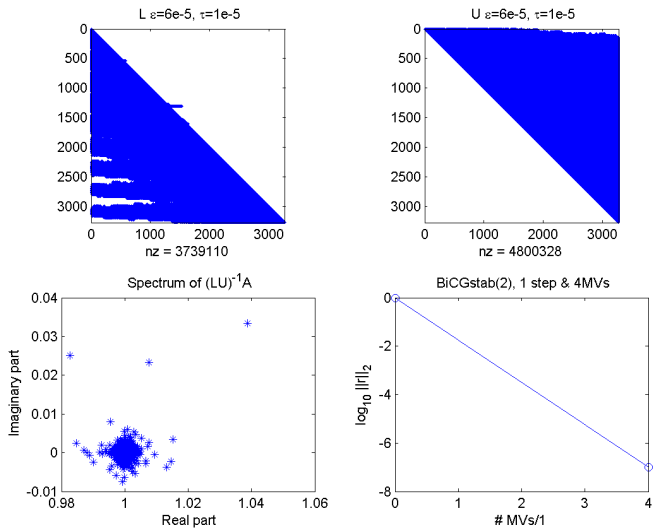


Figure 7.2: $\boldsymbol{L}$, $\boldsymbol{U}$ with $\epsilon = 6e-5$, $\tau = 1e-5$, spectrum of of $(\boldsymbol{LU})^{-1}A$ and MVs.

k, τ	sp. $\boldsymbol{A}_{\mathrm{D}}$	$\boldsymbol{L}_{\mathrm{D}}$	$\boldsymbol{U}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	t_{LU}
600, 1e-3	18.38%	11.01%	14.51%	10	40	3.1872e-009	4.64s	35.32s
900, 4e-4	27.56%	17.03%	22.63%	5	20	6.4734e-010	3.84s	55.48s
2100, 1e-5	64.29%	35.47%	45.28%	1	4	1.2751e-008	2.88s	118.74s

Table 7.13: ILUT(τ), technique (6.6).

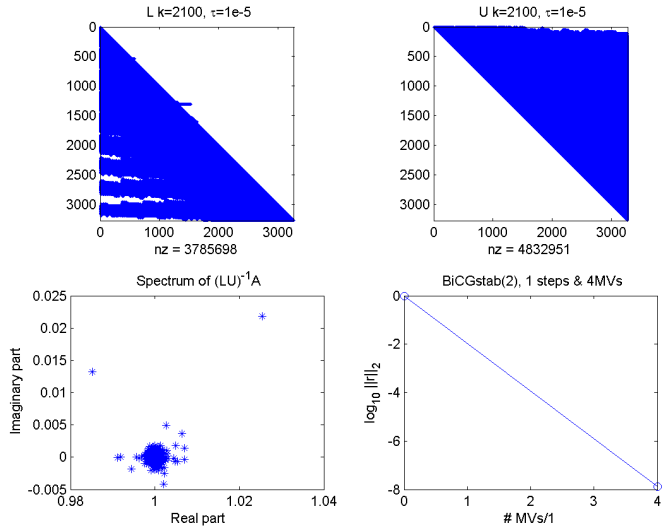


Figure 7.3: $\boldsymbol{L}$, $\boldsymbol{U}$ with $=2100, \tau = 1e-5$, spectrum of $(\boldsymbol{LU})^{-1}A$ and MVs.

k, ϵ, τ	sp. $\boldsymbol{A}_{\mathrm{D}}$	$\boldsymbol{L}_{\mathrm{D}}$	$\boldsymbol{U}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	t_{LU}
600, 3e-3, 1e-3	9.84%	10.00%	13.15%	12	48	6.4909e-008	5.25s	29.07s
900, 9e-4, 6e-4	19.20%	14.21%	18.66%	5	20	7.3717e-008	3.56s	45.92s
2100, 3e-5, 1e-5	55.73%	35.17%	45.12%	1	4	7.1476e-008	2.89s	121.35s

Table 7.14: ILUT(τ), technique (6.7).

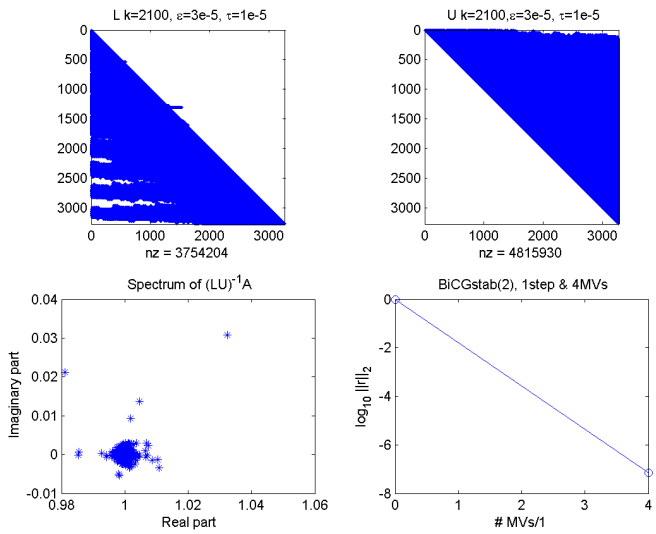


Figure 7.4: $\boldsymbol{L}, \boldsymbol{U}$ with $k = 2100, \epsilon = 3e-5, \tau = 1e-5$, spectrum of $(\boldsymbol{LU})^{-1}A$ and MVs.

7.3.3 Hard case, test case T3

When the sparsified matrix $\boldsymbol{C} = (c_{ij})$ satisfies equation (6.5).

ϵ, τ	sp. A_{D}	iter.	MVs	relres	t_{solve}	t_{LU}
2e-3, 1e-3	5.82%	26	104	9.6896e-008	26.1s	788.14s
9e-4, 9e-4	9.81%	17	68	3.0132e-008	18s	1.0651+003s
4e-4, 4e-4	15.12%	8	32	2.0790e-009	16.3s	2.1647+003s
1e-4, 1e-4	26.46%	3	12	1.8282e-010	13s	5.3801+003s

Table 7.15: ILUT(τ), technique (6.5).

7.3.4 Interpretation of the numerical results

To construct preconditioners based on incomplete LU factorization with dropping tolerance (τ), one has to keep in mind the magnitude of the elements to be dropped. In addition to the parameters k and ϵ, the quality of these kind of preconditioners depend on the value of τ also, see (7.3). It is also observed that for fixed values of k and ϵ the smaller the τ the better the preconditioner is.

7.4 Sparsity patterns as preconditioners using single and double threshold sparsifying strategies

In this section we use the sparsified coefficient matrices as preconditioners (see (7.4.1), (7.4.2)). These types of preconditioners are applied to matrices where it is possible to find the inverse of the sparsified matrix by using the backslash operator in `MatLab`.

Tables (7.16) - (7.27) show the amount of iterations for preconditioners based on single and double threshold sparsifying strategies using BiCGSTAB($l = 2$) and the sparsification techniques (6.5) - (6.7).

7.4.1 Single threshold strategy (SPST) - numerical results

Construction

- sparsify $\boldsymbol{A}$ and denote it by $\boldsymbol{S}$
- compute the inverse $\boldsymbol{S}^{-1}$ by using the backslash operator in `MatLab`
- use the computed inverse as a left (right) preconditioner

Single threshold easy case, T1

ϵ	$\boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
7e-2	0.23%	2	8	6.3178e-008	0.156s	0.19s
6e-4	27.19%	1	4	2.8270e-008	0.078s	0.17s

Table 7.16: Single threshold, technique (6.5).

k	$\boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
3	0.23%	2	8	9.6373e-008	0.141s	0.25s
300	19.89%	1	4	8.7516e-008	0.078s	0.24s

Table 7.17: Single threshold, technique (6.6).

k, ϵ	$\boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
4, 7e-2	0.21%	2	8	5.0148e-008	0.14s	0.47s
300, 6e-4	22.68%	1	4	9.4022e-008	0.078s	0.51s

Table 7.18: Single threshold, technique (6.7).

Single threshold medium case, T2

ϵ	$\boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
4e-3	11.13%	9	36	7.9148e-009	3.72s	1.00s
9e-4	24.07%	4	16	5.4302e-010	1.7s	0.93s
4e-5	55.74%	1	4	1.4172e-008	0.438s	0.90s

Table 7.19: Single threshold, technique (6.5).

k	$\boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
500	15.32%	11	44	4.4040e-008	4.5s	1.62s
700	21.44%	5	20	2.8150e-008	2.06s	1.56s
2050	62.76%	1	4	4.1089e-008	0.453s	1.53s

Table 7.20: Single threshold, technique (6.6).

k, ϵ	$\boldsymbol{M}_{\mathrm{D}}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}}$
500, 1e-3	13.68%	12	48	2.8596e-008	4.88s	2.87s
700, 9e-4	17.15%	6	24	1.2663e-008	2.45s	2.95s
2200, 4e-5	54.30%	1	4	6.2283e-008	0.438s	2.85s

Table 7.21: Single threshold, technique (6.7).

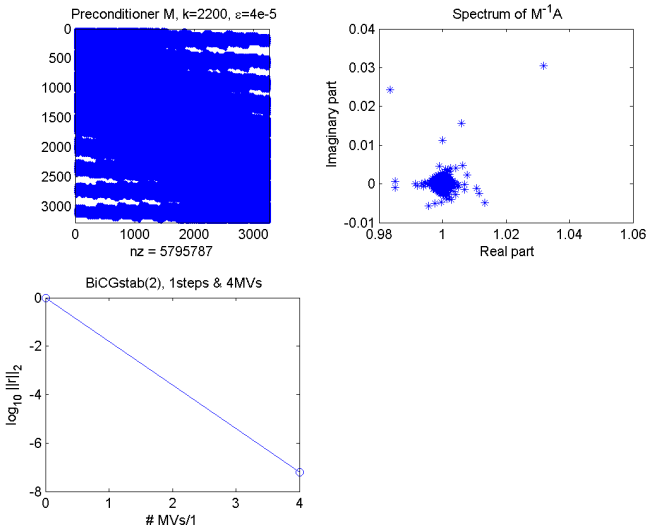


Figure 7.5: Single threshold T2 - Sparsified A, spectrum of $\boldsymbol{M}^{-1}A$ and MVs respectively.

7.4.2 Double threshold sparsifying strategy (SPDT) - numerical results

- sparsify $\boldsymbol{A}$ and denote it by $\boldsymbol{S}$.
- compute the inverse $\boldsymbol{S}^{-1}$, just using the backslash operator in `MatLab`.
- sparsify $\boldsymbol{S}^{-1}$.
- use the resulting matrix as a left (right) preconditioner.

Double threshold easy case, T1

ϵ_1, ϵ_2	$\boldsymbol{M}_{\mathrm{D}}$	sp. $\boldsymbol{M}_{\mathrm{D}}^{-1}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}} + t_{\mathrm{invsp.}}$
7e-2, 7e-2	0.23%	0.23%	2	8	6.2598e-008	0.188s	0.17+1.23s
6e-4, 6e-4	27.19%	25.07%	1	4	7.8555e-008	0.078s	0.17+1.47s

Table 7.22: Double threshold, technique (6.5).

k, ϵ	$\boldsymbol{M}_{\mathrm{D}}$	sp. $\boldsymbol{M}_{\mathrm{D}}^{-1}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}} + t_{\mathrm{invsp.}}$
3, 6e-3	0.23%	0.30%	2	8	9.7944e-008	0.172s	0.29+1.21s
300, 8e-5	22.68%	46.37%	1	4	9.8380e-008	0.078s	0.26+1.29s

Table 7.23: Double threshold, technique (6.6).

$k, \epsilon_1, \epsilon_2$	$\boldsymbol{M}_{\mathrm{D}}$	sp. $\boldsymbol{M}_{\mathrm{D}}^{-1}$	iter	MVs	relres	t_{solve}	$t_{\mathrm{const.}} + t_{\mathrm{invsp.}}$
4,7e-2, 7e-2	0.21%	0.21%	2	8	7.3244e-008	0.219s	0.48+1.94s
300, 6e-4, 5e-5	19.89%	53.43%	1	4	9.7529e-008	0.078s	0.48+1.32s

Table 7.24: Double threshold, technique (6.7).

Double threshold medium case, T2

ϵ_1, ϵ_2	$\boldsymbol{M}_{\mathrm{D}}$	sp. $\boldsymbol{M}_{\mathrm{D}}^{-1}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}} + t_{\mathrm{invsp.}}$
4e-3, 5e-3	11.13%	57.71%	9	36	6.9358e-008	3.7s	1.02+13.77s
9e-4, 3e-3	24.07%	78.99%	4	16	5.0341e-008	1.66s	0.95+13.54s
4e-5, 7e-4	55.74%	92.75%	1	4	5.5341e-008	0.437s	0.91+13.56s

Table 7.25: Double threshold, technique (6.5).

k, ϵ	$\boldsymbol{M}_{\mathrm{D}}$	sp. $\boldsymbol{M}_{\mathrm{D}}^{-1}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}} + t_{\mathrm{invsp.}}$
500, 4e-3	15.32%	80.47%	11	44	8.6488e-008	4.56s	1.68+13.42s
700, 4e-5	21.44%	99.33%	5	20	8.7884e-008	2.09s	1.69+13.43s
2050, 6e-4	62.76%	93.69%	1	4	7.9012e-008	0.422s	1.69+13.43s

Table 7.26: Double threshold, technique (6.7).

$k, \epsilon_1, \epsilon_2$	$\boldsymbol{M}_{\mathrm{D}}$	sp. $\boldsymbol{M}_{\mathrm{D}}^{-1}$	iter.	MVs	relres	t_{solve}	$t_{\mathrm{const.}} + t_{\mathrm{invsp.}}$
500, 1e-3, 6e-3	13.68%	73.18%	12	48	5.1825e-008	4.94s	2.89+13.64s
700, 9e-4, 4e-5	17.15%	99.31%	6	24	9.4534e-008	2.5s	2.89+13.51s
2200, 4e-5, 4e-5	54.30%	99.22%	1	4	6.2354e-008	0.438s	2.97+13.63s

Table 7.27: Double threshold, technique (6.7).

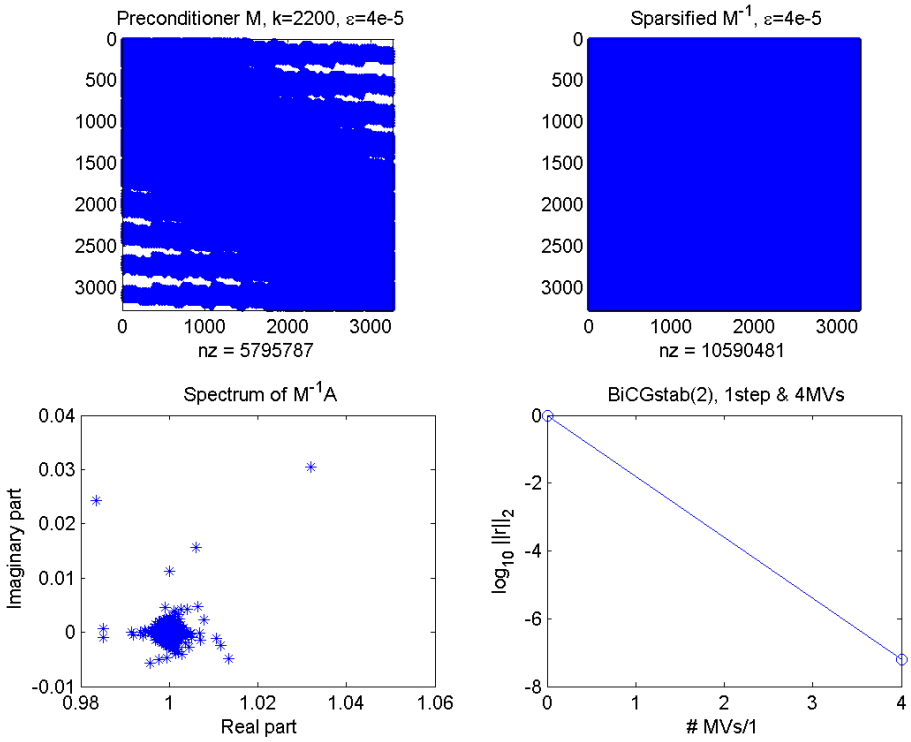


Figure 7.6: Double threshold T2 - Preconditioner $\boldsymbol{M}$, spectrum of $\boldsymbol{M}^{-1}A$ and MVs respectively.

7.4.3 Interpretation of the numerical results

As already been stated, these types of preconditioners are only applicable when it is possible to use the backslash operator in `MatLab` to find the inverse of a sparsified matrix. The quality of these kind preconditioners is affected in the same way as that of the other type of preconditioners discussed above. That is, the values of the sparsifying parameters k and ϵ plays the same roles in this case as well. These types of preconditioners have an impressive advantage in the sense that only very small time is spent for their construction when compared to the others. One of the main disadvantages is when the sparsified matrices are still big and dense which means they need large storage capacity and the other disadvantage is these techniques can not be used to invert matrices of large size.

7.5 Reuse of preconditioners for different parameters, test case T2

Because in the reconstruction loop one has to solve $\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c})$ for different (almost identical) values of $\boldsymbol{c}$ it is important to know whether a preconditioner for value $\boldsymbol{c}_1$ is also a good preconditioner for value $\boldsymbol{c} \in (\boldsymbol{c}_1 - \Delta\boldsymbol{c}, \boldsymbol{c}_1 + \Delta\boldsymbol{c})$.

By changing the angles of incidence of the plane wave with the x and z - axes, ϕ and θ respectively, we observed a significant variation in the number of matrix - vector products (MVs). This is shown in Table (7.28) in the 4-th and 5-th columns. Column 4 shows the amount of MVs obtained with the `FORTRAN`-VIM implementation and column 5 shows the amount of MVs obtained with the equivalent `MatLab` implementation. By original system we mean the system with angles of incidence $\phi = 45$ and $\theta = 8.13010235$. In Table (7.28) PBiCGSTAB stands for the preconditioned BiCGSTAB. We have constructed an ILUT(τ) preconditioner (see (6.2)) by using a column wise threshold strategy for the parameters $\epsilon = 3e - 3$ and $\tau = 1e - 3$.

When we take larger angles for θ and ϕ the new system seems to be better than the original one. To the contrary, the preconditioned system for these new angles seems to take a little more MVs than the preconditioned original system. But still the results (6-th column) imply that the preconditioner built from the original coefficient matrix is also good for other angles of incidence.

System	$\Delta\phi$	$\Delta\theta$	VIM	BiCGSTAB($l = 2$)	PBiCGSTAB($l = 2$)
Original	0	0	400 MVs	416 MVs	40 MVs
New system 1	0	0.2	432 MVs	392 MVs	40 MVs
New system 2	0	10	316 MVs	360 MVs	48 MVs
New system 3	0	20	351 MVs	308 MVs	52 MVs
New system 4	10	0	392 MVs	396 MVs	36 MVs
New system 5	20	0	380 MVs	360 MVs	40 MVs
New system 6	10	10	368 MVs	376 MVs	44 MVs
New system 7	20	10	372 MVs	376 MVs	48 MVs
New system 8	10	20	395 MVs	384 MVs	56 MVs
New system 9	20	20	400 MVs	384 MVs	60 MVs

Table 7.28: MVs for systems of different angles of incidence ϕ, θ : test case T2.

Chapter 8

Conclusion and Recommendation

8.1 Conclusion

By shifting the spectrum to the right hand side of the complex plane we observed a significant decrease in the number of iterations. Moreover by moving the small eigenvalues towards zero we came to realize that the system become much worse conditioned. This implies that the small eigenvalues, not just their negative real parts, have a huge impact in affecting the system.

By the experiments in section (5.4) we tried to see what happens when the singular values less than one are replaced by 1. In one of these experiments we found a number of iterations of 43 and in the other 29. From these two experiments we came to realize that the small singular values are not the only ones ruining the system. So, instead one can conclude that the distribution of the singular values (eigenvalues) between the two extremes plays decisive roles. For instance, if we have a system with a large condition number but the eigenvalues are, let us say, accumulated in two regions; experts claim that the system can be solved in just two steps. On the other hand if another system has a relatively small condition number but the eigenvalues are distributed everywhere, it might take quite a while before such a system converges.

By varying the angles of incidence of the plane wave, we observed a significant variation in the number of matrix - vector products (MVs). It is also observed that by using a fixed preconditioner for all the systems, the changes in the angles of incidence have a little effect on the preconditioned system when compared to the un-preconditioned ones, see Table (7.28).

The preconditioner for a fixed value of ϕ and θ is also a good preconditioner for systems obtained by varying these two angles like $\phi + \Delta\phi$ and $\theta + \Delta\theta$ even for relatively large $\Delta\phi$ and $\Delta\theta$, see (7.5).

Sections (7.2) - (7.4) show that sensible sparsifications depend on the test case, i.e., there is no unique "optimal" value for k and ϵ.

By looking at the amount of time needed to construct the preconditioners one can observe that the high the quality of the preconditioner the more time it needs for construction. What can also be observed from the tables is the density of the preconditioners has a direct relationship with the quality of the preconditioner, the denser the preconditioner the better the quality. Moreover as can easily be concluded, the time needed to compute the accurate solution keeps decreasing when the quality of the preconditioner increases.

We conclude this section by jotting down more reasons why one needs to sparsify dense coefficient matrices for constructing preconditioners.

- For storage purposes.
- Easier to compute inverses.
- For Frobenius minimization methods, the complexity of the least square problems gets reduced when the matrix is sparse.

8.2 Recommendation

1. Advantage of SPDT over SPST - the inverse of the preconditioner is less dense in the former case due to the fact that the inverse is also sparsified using a second dropping tolerance.
2. For matrices where the backslash operator is possible to find inverses SPDT (SPST) is recommended.
3. If the above is not possible, we recommend ILUT(τ) based preconditioners.
4. In the case when the above two methods fail Frobenius based preconditioners come to the rescue, even if it always takes too much time to construct such kind of preconditioners . These preconditioners are recommended provided that one uses multi-core (multi-processor) computers. The columns of the Frobenius preconditioners better be computed in parallel using these kind of computers, since their computation needs too much time in ordinary computers.
5. In the case when all the above three are impossible we would like to recommend the block Jacobi preconditioners.
6. Among the sparsification strategies, we would like to recommend the column wise threshold strategy due to the fact that it enables a quality preconditioner with smaller density when compared to the others. But we would also like to bring to the readers' attention that the "k - largest" strategy has the advantage of enabling researchers to decide the density of their preconditioners prior to computing them.
7. Preconditioners can also be reused for different angles of incedence.

Chapter 9

Future Work

1. Relevant information for the construction of the preconditioner can be extracted from the meshes of the underlying physical problem. In particular, two types of information are directly available

   - The connectivity graph, describing the topological neighborhood amongst the edges
   - The coordinates of the nodes in the mesh, describing geometric neighborhoods amongst the edges

   We believe it is a good practice to explore these two strategies and construct preconditioners out of them.

2. Use some information from the properties of the grids and use something like multigrid preconditioners.

3. Use multi-core (multi-processor) computers to construct Frobenius based preconditioners.

Appendix A

The non-linear reconstruction problem

The motivation for the diffraction problem is that of reconstruction: A beam of (laser) light is targeted at a grating and the reflected image is shot with a camera. The aim of this exercise is to exactly determine the grating's dimensions and material properties.

The beam of laser light is focused at the grating through a lens which causes the light to come from all directions θ (see Figure (2.3)). This implies that the reflected image is a superposition (intensities) of the reflected electromagnetic field over all angles θ. In practice each angle of incidence contributes to 2 or more "pixels" in the resulting image, an amount which depends on the incoming wave's amount of diffraction angles.

In order to create a (computer simulated) reflected image for a specific set of geometry and material parameters one therefore must solve the Maxwell equations for (a discrete subset) all θ and accumulate the computed "pixels" into an image. After this is done the calculated image is compared to the measured (shot) one and geometry and material parameters are adapted to find a proper fit.

This section first provides an impression of the mathematical nature of the sketch procedure. Thereafter, several standard algorithms to obtain a best fit are provided and explained. To conclude with, as part of this thesis a `MatLab` code has been written which implements these best fit determination algorithms **not for the Maxwell equations** but for scalar pde's – which depend in a non-linear manner on parameters. The extensive amount of implicit differentiation and determination of higher order derivatives (not vectors but tensors as iterands) makes the `MatLab` code far from trivial.

For a structure (grating) with geometry and material parameterized by real valued parameters $\boldsymbol{g} \in \mathbb{R}^d$ and geometry parameters $\boldsymbol{w} \in \mathbb{R}^{n-d-1}$, the mapping

$$\boldsymbol{U} : \mathbb{R}^{n-1} \;\rightarrow\; \mathbb{R}^N \tag{A.1}$$

produces a (discrete) image. The parameters $\boldsymbol{w}$ (in the sequel) are subject to minimization, but not the parameters $\boldsymbol{g}$. Below an element $\boldsymbol{u} \in \mathbb{R}^N$ is regarded as $\boldsymbol{u} \in \mathbb{R}^{m\times m}$ ($m\times m = N$) the discrete version of an image in $L^2(\Omega)$ ($\Omega = (0,1)^2$), i.e, a matrix of pixels (see Figure (A.1)). The problem of interest is that when an image $\boldsymbol{M} \in \mathbb{R}^N$ (a measured image) is given and when one has to reconstruct the geometry parameters $\boldsymbol{w} \in \mathbb{R}^{n-d-1}$

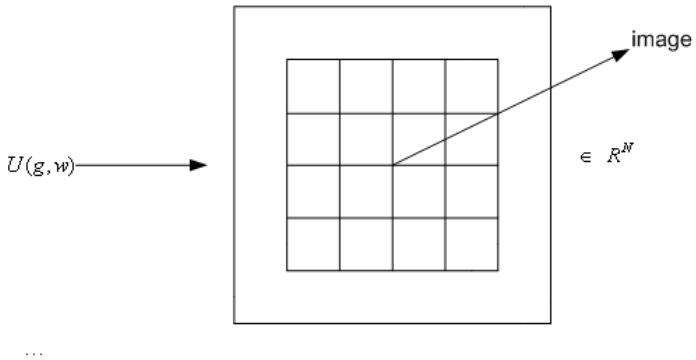


Figure A.1: Image $\in \mathbb{R}^N$.

which led to the image. In such cases one needs to determine $\boldsymbol{w}$ that minimizes

$$\min_{\boldsymbol{w}} \frac{1}{2} \| \boldsymbol{U}(\boldsymbol{g}, \boldsymbol{w}) - \boldsymbol{M} \|^2. \tag{A.2}$$

Now, we define $\boldsymbol{U}$ for the specific case of interest. Let $\theta \in \mathbb{R}$, $(\boldsymbol{g}, \boldsymbol{w}) \in \mathbb{R}^d \times \mathbb{R}^{n-d-1}$ be geometry parameters and assume that

$$\boldsymbol{b} : \mathbb{R}^{n-1} \times \mathbb{R} \to \mathbb{R}^N, \qquad \boldsymbol{A} : \mathbb{R}^{n-1} \times \mathbb{R} \to \mathbb{R}^{N \times N}.$$

Let

$$\boldsymbol{u} : \mathbb{R}^{n-1} \times \mathbb{R} \ \to \ \mathbb{R}^N$$

solve

$$\boldsymbol{A}(\boldsymbol{g}, \boldsymbol{w}, \theta) \boldsymbol{u}(\boldsymbol{g}, \boldsymbol{w}, \theta) = \boldsymbol{b}(\boldsymbol{g}, \boldsymbol{w}, \theta). \tag{A.3}$$

Let $\theta_i \in \mathbb{R}$ and define $\boldsymbol{U} : \mathbb{R}^{n-1} \ \to \ \mathbb{R}^N$ by

$$\boldsymbol{U}(\boldsymbol{g}, \boldsymbol{w}) = \sum_i \boldsymbol{u}(\boldsymbol{g}, \boldsymbol{w}, \theta_i) = \sum_i \boldsymbol{u}_i. \tag{A.4}$$

For the specific problem of interest, $\boldsymbol{U}$ is non-linear in the argument $\boldsymbol{w}$, and $\boldsymbol{u}$ solves the linear system (A.3).

Figure (A.2) shows an example: In that figure $N = 25$, thus

$$\boldsymbol{U} : \mathbb{R}^{n-1} \ \to \ \mathbb{R}^{25}.$$

Each $\boldsymbol{u}_i = \boldsymbol{u}(\boldsymbol{g}, \boldsymbol{w}, \theta_i)$ is a 5×5 image. Therefore the entire picture has only three non-zero pixels such that

$$\boldsymbol{U}(\boldsymbol{g}, \boldsymbol{w}) = \sum_{i=1}^{3} \boldsymbol{u}_i = \sum_{i=1}^{3} \boldsymbol{u}(\boldsymbol{g}, \boldsymbol{w}, \theta_i) \in \mathbb{R}^{25} \ (\equiv \mathbb{R}^{5 \times 5}).$$

For our application the linear system

$$\boldsymbol{A}(\boldsymbol{g}, \boldsymbol{w}, \theta) \boldsymbol{u}(\boldsymbol{g}, \boldsymbol{w}, \theta) = \boldsymbol{b}(\boldsymbol{g}, \boldsymbol{w}, \theta)$$

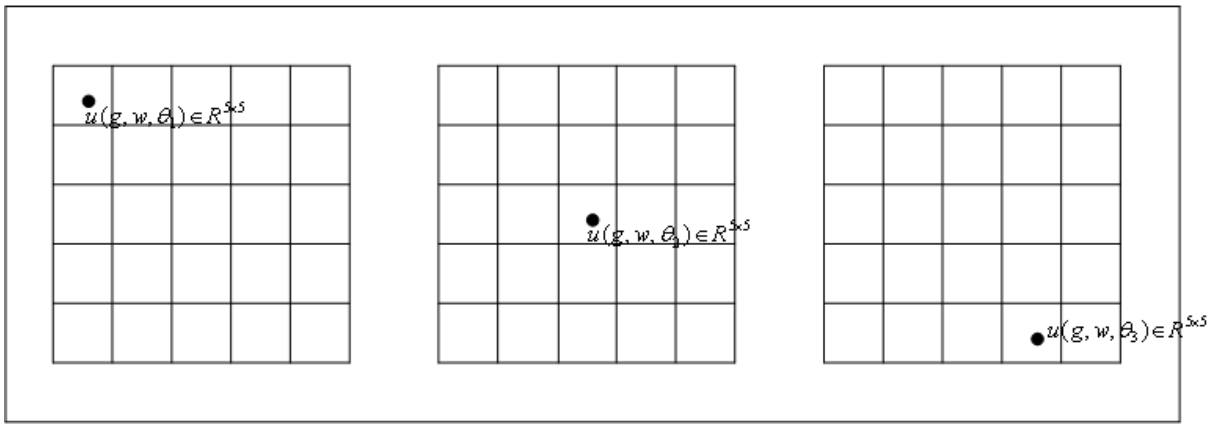


Figure A.2: Special case: Three 5×5 images with one non-zero pixel $\left(\boldsymbol{u}(\boldsymbol{g}, \boldsymbol{w}, \theta_i)\right)_{i=1}^{3}$.

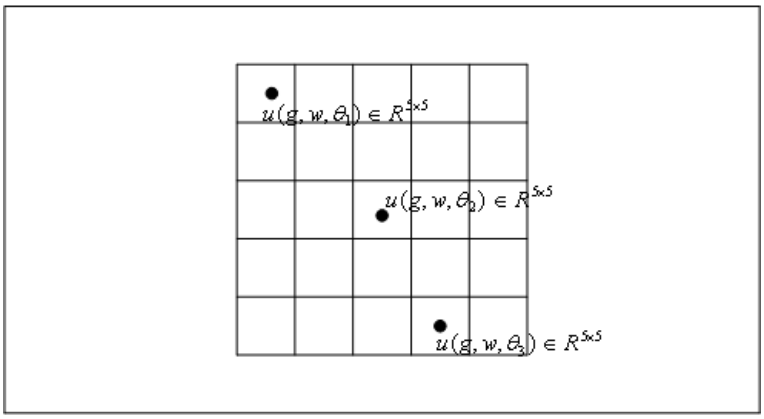


Figure A.3: Special case: The sum of the images in Figure (A.2) as seen by the camera.

is the VIM discretized system described in Chapter (4) based on the layer-wise discretization depicted in Figure (3.2) in Chapter (3).

In order to abbreviate notations let

$$\mathbb{R}_c = \mathbb{R}^d \times \mathbb{R}^{n-d-1} \times \mathbb{R},$$

then one can write (A.3) as

$$\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c}).$$

This shows that to calculate one image one has to solve several systems (see (A.4))

$$\boldsymbol{A}(\boldsymbol{c}_p)\boldsymbol{x}(\boldsymbol{c}_p) = \boldsymbol{b}(\boldsymbol{c}_p), \qquad \forall p \in P$$

where $P \subset \mathbb{N}$.

A.1 The reconstruction algorithm

First we introduce the reconstruction algorithms. The next section applies them to one specific image reconstruction case.

A.1.1 Newton's method

First take a suitable initial guess $\boldsymbol{u}^{(0)}$, then the remaining iterates are calculated according to

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} - \underbrace{[\partial \boldsymbol{F}(\boldsymbol{u}^{(k)})]^{-1}\boldsymbol{F}(\boldsymbol{u}^{(k)})}_{\boldsymbol{d}} \quad \forall k = 0, 1, 2, \ldots. \tag{A.5}$$

Each step of this method needs to calculate the direction $\boldsymbol{d} \in \mathbb{R}^N$ such that

$$\partial \boldsymbol{F}(\boldsymbol{u}^{(k)})\boldsymbol{d} = \boldsymbol{F}(\boldsymbol{u}^{(k)}). \tag{A.6}$$

A.1.2 The generalized Gauss - Newton method

Let $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^N$. Let $\boldsymbol{f} : \mathbb{R}^m \mapsto \mathbb{R}^n$ and let $g : \mathbb{R}^N \mapsto \mathbb{R}$:

$$g(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}||_2^2. \tag{A.7}$$

Then g is differentiable and a critical point $\boldsymbol{x}$ of g satisfies

$$\nabla g(\boldsymbol{x}) = (\partial \boldsymbol{f}(\boldsymbol{x}))^T(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}) = \boldsymbol{0}. \tag{A.8}$$

This shows that a critical point is either a solution of

$$\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y} = \boldsymbol{0} \tag{A.9}$$

or that a real critical point for which

$$\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y} \in \ker(\partial \boldsymbol{f}(\boldsymbol{x}))^T. \tag{A.10}$$

If $m = n$ and $\partial \boldsymbol{f}(\boldsymbol{x})$ is of maximal rank (for instance if it is non-singular) then only solutions of (A.9) exist, not other critical points.

To find critical points we have to find the roots of

$$\boldsymbol{h}(\boldsymbol{x}) = \nabla g(\boldsymbol{x}) = (\partial \boldsymbol{f}(\boldsymbol{x}))^T(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}).$$

Then one step of the Newton method for this critical point search (also called *modified Newton* for the search of a "root" of $\boldsymbol{x} \mapsto \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}$):

$$\partial \boldsymbol{h}(\boldsymbol{x}^k)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) = -\boldsymbol{h}(\boldsymbol{x}^k) \tag{A.11}$$

Note that if $\boldsymbol{f}$ is affine then

$$\partial \boldsymbol{h} = (\partial^2 \boldsymbol{f})\boldsymbol{f} + \partial \boldsymbol{f}^T \partial \boldsymbol{f} = \partial \boldsymbol{f}^T \partial \boldsymbol{f}.$$

Also $\partial \boldsymbol{f}(\boldsymbol{x})^T \partial \boldsymbol{f}(\boldsymbol{x})$ is non-singular if $\partial \boldsymbol{f}(\boldsymbol{x})$ is of maximal rank.

A *generalized steepest descent method* for problem (A.8) would be: Determine $\mathbb{R} \ni \lambda_k > 0$ and set

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \lambda_k \nabla g(\boldsymbol{x}^k) = \boldsymbol{x}^k - \lambda_k (\partial \boldsymbol{f}(\boldsymbol{x}^k))^T (\boldsymbol{f}(\boldsymbol{x}^k) - \boldsymbol{y}).$$

The *generalized Newton method* for problem $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}$ is the Newton method applied to (A.8), which is

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \left(\left(\partial^2 \boldsymbol{f}(\boldsymbol{x}^k)\right) \boldsymbol{f}(\boldsymbol{x}^k) + \partial \boldsymbol{f}(\boldsymbol{x}^k) \partial \boldsymbol{f}(\boldsymbol{x}^k)\right)^{-1} \partial \boldsymbol{f}(\boldsymbol{x}^k)^T \left(\boldsymbol{f}(\boldsymbol{x}^k) - \boldsymbol{y}\right). \tag{A.12}$$

If one substitutes $\lambda_k \boldsymbol{I}_n$ for the second derivative term $\left(\partial^2 \boldsymbol{f}(\boldsymbol{x}^k)\right) \boldsymbol{f}(\boldsymbol{x}^k)$ one obtains the *Levenberg method*:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \left(\partial \boldsymbol{f}(\boldsymbol{x}^k)^T \partial \boldsymbol{f}(\boldsymbol{x}^k) + \lambda_k \boldsymbol{I}_n\right)^{-1} \partial \boldsymbol{f}(\boldsymbol{x}^k)^T \left(\boldsymbol{f}(\boldsymbol{x}^k) - \boldsymbol{y}\right) \tag{A.13}$$

and if one substitutes $\lambda_k \mathrm{diag}(\partial \boldsymbol{f}(\boldsymbol{x}^k)^T \partial \boldsymbol{f}(\boldsymbol{x}^k))$ instead one obtains the *Levenberg-Marquardt method*

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \left(\partial \boldsymbol{f}(\boldsymbol{x}^k)^T \partial \boldsymbol{f}(\boldsymbol{x}^k) + \lambda_k \mathrm{diag}(\partial \boldsymbol{f}(\boldsymbol{x}^k)^T \partial \boldsymbol{f}(\boldsymbol{x}^k))\right)^{-1} \partial \boldsymbol{f}(\boldsymbol{x}^k)^T \left(\boldsymbol{f}(\boldsymbol{x}^k) - \boldsymbol{y}\right). \tag{A.14}$$

A.2 The specific non-linear case

Let $\boldsymbol{U}$ be defined as in (A.4) and let $\boldsymbol{M}$ be a measured image. For the sake of argument assume that $\boldsymbol{c} \in \mathbb{R}^{n-1} = \mathbb{R}^3$. We start off by the minimization problem

$$\min_{\boldsymbol{c} \in \mathbb{R}^3} \frac{1}{2} \| \underbrace{\boldsymbol{U}(\boldsymbol{c}) - \boldsymbol{M}}_{\in \mathbb{R}^N} \|^2. \tag{A.15}$$

A minimizer $\hat{\boldsymbol{c}}$ of (A.15) satisfies

$$\underbrace{\left(\nabla \boldsymbol{U}(\boldsymbol{c})\right)^T}_{\in \mathbb{R}^{3 \times N}} \left(\boldsymbol{U}(\boldsymbol{c}) - \boldsymbol{M}\right) = \boldsymbol{0} \tag{A.16}$$

where (see (A.4))

$$\boldsymbol{U}(\boldsymbol{c}) = \sum_i \boldsymbol{u}(\boldsymbol{c}, \theta_i), \quad \nabla \boldsymbol{U}(\boldsymbol{c}) = \sum_i \nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i),$$

with $\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)$ to be determined later by (A.20).

To solve (A.16) with Newton (see (A.5)) one needs the derivative

$$\nabla H(\boldsymbol{c}) = \underbrace{\left(\nabla(\nabla \boldsymbol{U}(\boldsymbol{c}))\right)^T}_{\in \mathbb{R}^{3 \times 3 \times N}} \left(\boldsymbol{U}(\boldsymbol{c}) - \boldsymbol{M}\right) + \left(\nabla \boldsymbol{U}(\boldsymbol{c})\right)^T \left(\underbrace{\nabla \boldsymbol{U}(\boldsymbol{c})}_{\in \mathbb{R}^{N \times 3}}\right) \tag{A.17}$$

of

$$H(\boldsymbol{c}) = \Big(\nabla \boldsymbol{U}(\boldsymbol{c})\Big)^T \Big(\boldsymbol{U}(\boldsymbol{c}) - \boldsymbol{M}\Big).$$

Note that

$$\nabla\Big(\nabla \boldsymbol{U}(\boldsymbol{c})\Big) = \sum_i \nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)),$$

with $\nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i))$ to be determined later from (A.22).

Observe that in the Maxwell's applications $\boldsymbol{c} \notin \mathbb{R}^{n-1}$. In that application $\boldsymbol{c}$ is an element of a convex subset (closed and bounded) of $\mathbb{R}^{n-1}$

Let $\boldsymbol{c} = (c_1, c_2, c_3) \in \mathbb{R}^3$ and assume that $\boldsymbol{u}(\boldsymbol{c}, \theta_i)$ solves

$$\boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)) = \boldsymbol{0}. \tag{A.18}$$

Under some assumptions, by the implicit function theorem one can show that there is a unique (differentiable) $\boldsymbol{u}(\boldsymbol{c}, \theta_i)$ satisfying (A.18).

In order to determine $\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)$ one implicitly differentiates (take the total derivative of) (A.18) which leads to

$$\mathrm{d}_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)) \overset{\text{Chain rule}}{=} \underbrace{\nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))}_{\in \mathbb{R}^{N\times 3}} + \underbrace{\nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))}_{\in \mathbb{R}^{N\times N}} \underbrace{\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)}_{\in \mathbb{R}^{N\times 3}} = \underbrace{\boldsymbol{0}}_{\in \mathbb{R}^{N\times 3}}, \tag{A.19}$$

which implies

$$\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i) = -[\nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))]^{-1} \nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)). \tag{A.20}$$

In order to determine $\nabla_{\boldsymbol{c}} \boldsymbol{u}$ one can apply Newton's method and solve

$$H_{\boldsymbol{c}}(\boldsymbol{v}) = \boldsymbol{0}.$$

where $H_{\boldsymbol{c}} : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times 3}$ is defined by

$$H_{\boldsymbol{c}}(\boldsymbol{v}) = \nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)) + \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))\boldsymbol{v}.$$

To this end one needs the derivative

$$\nabla_{\boldsymbol{v}} H_{\boldsymbol{c}}(\boldsymbol{v}) = \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)). \tag{A.21}$$

And by taking the total derivative of (A.19) one gets

$$\begin{aligned}\nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))) + \nabla_{\boldsymbol{u}}(\nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)))\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i) + \nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)\\ + \nabla_{\boldsymbol{u}}(\nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)))\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i) + \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))\nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)) = \boldsymbol{0}.\end{aligned} \tag{A.22}$$

In order to determine the double gradient $\nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{c}} \boldsymbol{u})$ one can apply Newton's method and solve

$$H_{\boldsymbol{c}}(\boldsymbol{v}) = \boldsymbol{0}.$$

where

$$\begin{aligned}H_{\boldsymbol{c}}(\boldsymbol{v}) = \nabla_{\boldsymbol{c}}(\nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))) + \nabla_{\boldsymbol{u}}(\nabla_{\boldsymbol{c}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)))\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)\\ + \nabla_{\boldsymbol{u}}(\nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)))\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i) + \nabla_{\boldsymbol{u}}(\nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c})))\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)\nabla_{\boldsymbol{c}} \boldsymbol{u}(\boldsymbol{c}, \theta_i)\\ + \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i))\boldsymbol{v}.\end{aligned} \tag{A.23}$$

To this end one needs the gradient

$$\nabla_{\boldsymbol{v}} H_{\boldsymbol{c}}(\boldsymbol{v}) = \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c}, \theta_i)). \tag{A.24}$$

A.3 The non-linear numerical test cases

A.3.1 Mathematical problem formulation

Let $\Omega = (0,1)$, let $n \in \mathbb{N}$, $m = n+1$, $h = 1/m$ and $x_i = i \cdot h$, $i = 0, \ldots, n+1$. Let $\Omega_h = \{x_i\}_{i=1}^n$ and $\partial\Omega_h = \{x_0, x_{n+1}\}$. Let $\boldsymbol{c} \in \boldsymbol{c}_{\boldsymbol{c}}$ be a parameter in the parameter-set $\mathbb{R}_{\boldsymbol{c}}$. Let $a, f, g \colon \mathbb{R}_{\boldsymbol{c}} \times \Omega \mapsto \mathbb{R}$ be real valued smooth functions defined on the domain Ω, which in addition depend on a parameter/parameters $\boldsymbol{c}$. Let

$$L = -a(\boldsymbol{c})\partial^2, \qquad L_h = -a(\boldsymbol{c})D^+D^- =: -a(\boldsymbol{c})\partial_h^2$$

be two linear operators (for the sake of demonstration) in u. The boundary value problems (BVPs) are: Let $\boldsymbol{c} \in \mathbb{R}^3$. Find $u_{\text{bvp}} \in \mathcal{C}^2(\bar{\Omega})$ such that

$$L(\boldsymbol{c})(u) = f(\boldsymbol{c}) \quad \text{in } \Omega, \qquad u = g(\boldsymbol{c}) \quad \text{in } \partial\Omega. \tag{A.25}$$

Application of a finite difference method (FDM) leads to: Let $\boldsymbol{c} \in \mathbb{R}^3$. Find $\boldsymbol{u}_{\text{fdm}} \in \mathbb{R}^N$ such that

$$L_h(\boldsymbol{c})(u) = f(\boldsymbol{c}) \quad \text{in } \Omega_h, \qquad u = g(\boldsymbol{c}) \quad \text{in } \partial\Omega_h. \tag{A.26}$$

The FDM discretization depends on parameter h which can be considered as one of the parameters $\boldsymbol{w}$ related to (A.26). Let the linear system related to (A.26) be written as

$$\boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}_{\text{fdm}}(\boldsymbol{c}) = \boldsymbol{f}(\boldsymbol{c})$$

A.3.2 The specific non-linear case

The specific non-linear case of interest is of the form

$$\boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c})) = \boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}(\boldsymbol{c}) - \boldsymbol{b}(\boldsymbol{c}) = \boldsymbol{0}, \tag{A.27}$$

where $\boldsymbol{F} : \mathbb{R}^3 \times \mathbb{R}^N \to \mathbb{R}^N$.

Example 1. Let $\boldsymbol{c} = (c_1, c_2, c_3)$. The function $u_{\text{bvp}}(\boldsymbol{c}, x) = c_1 + c_2 x + c_3 x^2$ solves

$$\boldsymbol{F}(\boldsymbol{u}(\boldsymbol{c}), \boldsymbol{c}) = \boldsymbol{0} \Leftrightarrow \boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}(\boldsymbol{c}) = \boldsymbol{f}(\boldsymbol{c}) \Leftrightarrow \begin{cases} -u'' = -2c_3 & \text{in } (0,1) \\ u \ = c_1 & \text{at } x = 0 \\ u' = c_2 + 2c_3 & \text{at } x = 1. \end{cases} \tag{A.28}$$

where $\boldsymbol{L} = -a(\boldsymbol{c})\partial_h^2$. Here $\boldsymbol{F}$ is linear in both $\boldsymbol{u}$ and $\boldsymbol{c}$

Example 2. Let $\boldsymbol{c} = (c_1, c_2, c_3)$. The function $u_{\text{bvp}}(\boldsymbol{c}, x) = c_1(x - c_2)(x - c_3)$ solves

$$\begin{cases} -u'' = -2c_1 & \text{in } (0,1) \\ u \ = c_1 c_2 c_3 & \text{at } x = 0 \\ u' = c_1(2 - c_2 - c_3) & \text{at } x = 1. \end{cases} \tag{A.29}$$

Here $\boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c})) = \boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}(\boldsymbol{c}) - \boldsymbol{f}(\boldsymbol{c})$ where $\boldsymbol{L}$ is linear in $\boldsymbol{u}$ and $\boldsymbol{c}$ but $\boldsymbol{f}$ is non-linear in $\boldsymbol{c}$.

Example 3. Let $\boldsymbol{c} = (c_1, c_2, c_3)$. The function $u_{\text{bvp}}(\boldsymbol{c}, x) = c_1(x - c_2)(x - c_3)$ solves

$$\begin{cases} -c_1 u'' = -2c_1c_1 & \text{in } (0,1) \\ u = c_1c_2c_3 & \text{at } x = 0 \\ u' = c_1(2 - c_2 - c_3) & \text{at } x = 1. \end{cases} \tag{A.30}$$

Here $\boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c})) = \boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}(\boldsymbol{c}) - \boldsymbol{f}(\boldsymbol{c})$ where $\boldsymbol{L}$ is linear in $\boldsymbol{u}$ and $\boldsymbol{c}$ but $\boldsymbol{f}$ is non-linear in $\boldsymbol{c}$.

Example 4. Let $\boldsymbol{c} = (c_1, c_2, c_3)$. The function $u_{\text{bvp}}(\boldsymbol{c}, x) = c_1(x - c_2)(x - c_3)$ solves

$$\begin{cases} -(1/c_1)u'' = -2 & \text{in } (0,1) \\ u = c_1c_2c_3 & \text{at } x = 0 \\ u' = c_1(2 - c_2 - c_3) & \text{at } x = 1. \end{cases} \tag{A.31}$$

Here $\boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c})) = \boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}(\boldsymbol{c}) - \boldsymbol{f}(\boldsymbol{c})$ where $\boldsymbol{L}$ is linear in $\boldsymbol{u}$, but non-linear in $\boldsymbol{c}$ and $\boldsymbol{f}$ is non-linear in $\boldsymbol{c}$.

Example 5. Let $\boldsymbol{c} = (c_1, c_2, c_3)$. The function $u_{\text{bvp}}(\boldsymbol{c}, x) = c_1 + c_2x + c_3x^2$ solves

$$\begin{cases} -u'' - ku = -2c_3 - k(c_1 + c_2x + c_3x^2) & \text{in } (0,1) \\ u = c_1 & \text{at } x = 0 \\ u' = c_2 + 2c_3 & \text{at } x = 1. \end{cases} \tag{A.32}$$

Here $\boldsymbol{F}(\boldsymbol{c}, \boldsymbol{u}(\boldsymbol{c})) = \boldsymbol{L}(\boldsymbol{c})\boldsymbol{u}(\boldsymbol{c}) - \boldsymbol{f}(\boldsymbol{c})$ where $\boldsymbol{L}$ is linear in $\boldsymbol{u}$, but non-linear in $\boldsymbol{c}$ and $\boldsymbol{f}$ is non-linear in $\boldsymbol{c}$.

Appendix B

The determination of an initial guess

Let $\boldsymbol{c} \in \mathbb{R}^n$ and $\boldsymbol{A}(\boldsymbol{c}) \in \mathbb{R}^{N\times N}$ be non-singular, $\boldsymbol{b}(\boldsymbol{c}) \in \mathbb{R}^N$ and assume that $\boldsymbol{x}(\boldsymbol{c}) \in \mathbb{R}^N$ is defined by

$$\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c}) \;\Rightarrow\; \boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{A}^{-1}(\boldsymbol{c})\boldsymbol{b}(\boldsymbol{c}). \tag{B.1}$$

Now a good initial guess for $\boldsymbol{x}(\boldsymbol{c}+\Delta\boldsymbol{c})$ would be $\boldsymbol{x}(\boldsymbol{c}+\Delta\boldsymbol{c})$ itself or (since unknown) an approximation based on $\boldsymbol{x}(\boldsymbol{c})$:

$$\boldsymbol{x}(\boldsymbol{c}+\Delta\boldsymbol{c}) = \boldsymbol{x}(\boldsymbol{c}) + \nabla\boldsymbol{x}(\boldsymbol{c})\Delta\boldsymbol{c} + \mathcal{O}(|\Delta\boldsymbol{c}|^2).$$

The following **Lemma** calculates $\nabla\boldsymbol{x}(\boldsymbol{c})$.

Lemma B.0.1.

$$\nabla\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{A}^{-1}(\boldsymbol{c})\Big(\nabla\boldsymbol{b}(\boldsymbol{c}) - (\nabla\boldsymbol{A}(\boldsymbol{c}))^T\boldsymbol{A}^{-1}(\boldsymbol{c})\boldsymbol{b}(\boldsymbol{c})\Big). \tag{B.2}$$

Proof. For the scalar case, i.e., when $c \in \mathbb{R}$

$$\boldsymbol{x}(c) = \boldsymbol{A}^{-1}(c)\boldsymbol{b}(c) \;\Rightarrow\; (\boldsymbol{x}'(c))^T = \boldsymbol{A}^{-1}(c)(\boldsymbol{b}'(c))^T + [\boldsymbol{A}^{-1}]'(c)\boldsymbol{b}(c). \tag{B.3}$$

Let $\boldsymbol{A} : \mathbb{R}^n \;\rightarrow\; \mathbb{R}^{N\times N}$ be non-singular. Then

$$\boldsymbol{A}(c)\boldsymbol{A}^{-1}(c) = \boldsymbol{I}_N \;\Rightarrow\; \boldsymbol{A}'(c)\boldsymbol{A}^{-1}(c) + \boldsymbol{A}(c)[\boldsymbol{A}^{-1}]'(c) = 0. \tag{B.4}$$

Thus,

$$[\boldsymbol{A}^{-1}]'(c) = -\boldsymbol{A}^{-1}(c)\boldsymbol{A}'(c)\boldsymbol{A}^{-1}(c). \tag{B.5}$$

Therefore, substituting (B.5) into (B.3) gives the desired result.

□

Under the assumption that one can calculate $\nabla\boldsymbol{x}(\boldsymbol{c})$ with Lemma (B.0.1) (or by other means) one can construct at least two equally good initial guesses; $\boldsymbol{x}(\boldsymbol{c}) + \Delta\boldsymbol{c}$ itself and

$$\boldsymbol{x}(\boldsymbol{c}+\Delta\boldsymbol{c}) \doteq \boldsymbol{x}(\boldsymbol{c}) + \nabla\boldsymbol{x}(\boldsymbol{c})\Delta\boldsymbol{c}$$

since $\boldsymbol{A}(\boldsymbol{c})\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{c})$. Another option is to use $\boldsymbol{x}(\boldsymbol{c})$ as an initial guess for the determination of $\boldsymbol{x}(\boldsymbol{c}+\Delta\boldsymbol{c})$.

In fact note that (B.2) gives

$$(\boldsymbol{x}')^T = \boldsymbol{A}^{-1}((\boldsymbol{b}')^T - \boldsymbol{A}'\boldsymbol{A}^{-1}\boldsymbol{b}) \;\Leftrightarrow\; \boldsymbol{A}(\boldsymbol{x}')^T = (\boldsymbol{b}')^T - \boldsymbol{A}'\boldsymbol{x},$$

which can also be obtained directly by implicit differentiation

$$\begin{aligned}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} &\Rightarrow \boldsymbol{A}'\boldsymbol{x} + \boldsymbol{A}(\boldsymbol{x}')^T = (\boldsymbol{b}')^T \\ &\Leftrightarrow \boldsymbol{A}(\boldsymbol{x}')^T = (\boldsymbol{b}')^T - \boldsymbol{A}'\boldsymbol{x}.\end{aligned}$$

Appendix C

Script for checking diagonal dominance

```
function isdiagonallydominantf(A)
%-------------------------------------------------------------------------
% checks whether a square matrix A is (weakly) diagonally dominant or not.
%
% Definition: A square matrix is said to be
% (weakly) diagonally dominant if for all rows the absolute value of the
% diagonal element in a row is  greater than or equal to the sum of the
% absolute values of the rest of the elements in that row.
%-------------------------------------------------------------------------
%
% Example 1. NOT diagonally dominant
% A=[-2  -6+3i  0  1;5  11  -8  -9; 0  20  22 -4;3  2+3i  1  -7];
%
%-------------------------------------------------------------------------
% Example 3. Diagonally dominant
% A=[10,2,2,2,2,2;2,10,2,2,2,2;2,2,10,2,2,2;2,2,2,10,2,2;2,2,2,2,10,2;2,2,2,2,2,10];
%
%-------------------------------------------------------------------------
%
% Example 4:
%
  A=buildAxb;
%
%-------------------------------------------------------------------------
rowcol=size(A);
%size of the matrix A (how many rows/columns ).
n=rowcol(1);
% count = the number of rows which satisfy the inequality.
count=0;
for i=1:1:n
    sumrow=0;

    for j=1:1:n
        if i~=j
            sumrow=sumrow+abs(A(i,j));
        end
    end
    if abs(A(i,i))-sumrow >=0
        count=count+1;
    end

end
if count==n
    disp('YES')
else
    disp('NO')
end
count2=n-count    % counts the number of rows where abs(A(i,i))-sumrow < 0 count=629
```

Bibliography

[1] Yousef Saad, *Iterative methods for sparse linear systems.* PWS publishing (2000).

[2] Henk A, van der Vorst, *Iterative Krylov methods for large linear systems.* Cambridge University press. Monographs in applied and computational mathematics (2003).

[3] G. Allèon, M. Benzi, and L. Giraud, *Sparse approximate inverse preconditioning for dense linear systems arising in computational elecromagnetics.* Numerical Algorithms, 16(1):1-15, 1997.

[4] Gerard L.G. Sleijpen and Diederik R. Fokkema, *Bi-CGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum.* Electronic transactions on numerical analysis(2003); volume 1, 11-32.

[5] Richard Barett -et al, *Templates for the solution of linear systems: Building blocks for iterative methods.* SIAM books (2006).

[6] Are Magnus Bruaset, *A survey of preconditioned iterative methods.* Longman scientific and technical (1992).

[7] M. Benzi and D. Bertaccini, *Block preconditioning for real-valued iterative algorithms for complex linear systems.* IMA Journal of Numerical Analysis (2008) 28, 598-618.

[8] R. J. Arms, L. D. Gates, and B. Zondek, *A method of block iterations.* SIAM Journal of Applied MAthematics (1956)4, 220-229.

[9] M. Ganesh and S.C. Hawkins, *Sparse approximate inverse preconditioners for electromagnetic surface scaterring simulations.* ANZIAM J. 49 (EMAC2007) pp.C155-C169, 2007.

[10] M. Benzi, J.C. Haws and M. Tùma, *Preconditioning highly indefinite and nonsymmetric matrices.* SIAM Journal of Scientific Computing (2000) 22, 1333-1353.

[11] Anne Greenbaum, *Iterative methods for solving linear systems.* SIAM(1997) Frontiers in applied mathematics.

[12] David M. Young, *Iterative solution of large linear systems.* Academic Press (1971) Computer science and applied mathematics.

[13] Owe Axelsson, *Iterative solution methods.* Cambridge University press (1994).

[14] Richard S. Varga, *Matrix iterative analysis.* Springer series in computational mathematics (2000).

[15] Marcus J. Grote and Thomas Huckle, *Parallel preconditioning with sparse approximate inverses.* Siam Journal of Scientific computing (1997) .

[16] Jeonghwa Lee, Jun Zhang, and Cai-Cheng Lu, *Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics.* Technical report No. 363-02, Department of Computer Science, University of Kentucky, Lexington.

[17] Yousef Saad and Martin H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.* SIAM Journal of Scientific and Statistical Computing 1986.

[18] Yin Wang, Jeonghwa Lee, and Jun Zhang, *A short survey on preconditioning techniques for large-scale dense complex linear systems in electromagnetics.* International Journal of Computer Mathematics Vol. 84, No. 8, August 2007.

[19] M. Ganesh and S. C. Hawkins, *Sparse preconditioners for dense complex linear systems arising in some radar cross section computations.* ANZIAM J. 48 (CTAC2006) pp.C233-C248, 2007.

[20] Jeonghwa Lee, Jun Zhang and Cai-Cheng Lu, *Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems.* Journal of Computational Physics (2003).

[21] B. Carpentieri, I.S. Duff, and L. Giraud, *Sparse pattern selection for robust Frobenius norm minimization preconditioners in electromagnetism.* CERFACS Technical Report TR/PA/00/05.

[22] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations.* SIAM J. Numerical Analysis 12, 617-629 (1975).

[23] H. C. Elman, *Iterative methods for large sparse nonsymmetric systems of linear equations, Ph.D. thesis.* Computer Science Dept., Yale Univ., New Haven, CT, 1982.

[24] P. K. W. Vinsome, *ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations, in Proc.* Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149-159.

[25] K. C. Jea and D. M. Young, *Generalized conjugate gradient acceleration ofnonsymmetrizable iterative methods.* Linear Algebra and Applications, 34 (1980), pp. 159-194.

[26] H.A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems.* SIAM J. Sci. Statist. Comput., 13 (1992) pp. 631-644.

[27] R. Fletcher, *Conjugate gradient methods for indefinite systems.* V - 506 Lecture notes Math. pp 73-89. Springer newyork 1976.

[28] M.H. Gutknecht, *Variants of BiCGStab for matrices with complex spectrum.* IPS Research Report No. 91-14, 1991.

[29] Yia-Chung Chang et al, *Efficient finite-element, Green's function approach critical-dimension metrology of three-dimensional garating on multilayer films.* J.Opt. Soc. Am. A/Vol. 23, No. 3/March 2006.

[30] Li Wung and Jun Zhang, *A new stabilization strategy for incomplete LU preconditioning of indefinite matrices.* Applied Mathematics and Coputation 144(2003) 75-87.

[31] T.A. Manteuffel, *An incomplete factorization technique for positive definite linear systems.* Math. Comput. 34 (1980) 473-497.