Differentiable, GPU-accelerated RCWA

Frank van der Ceelen

Delft University of Technology

TU Delft

Differentiable, GPU-accelerated RCWA

by

Frank van der Ceelen

Student Name	Student Number
Frank	4587464

Project Supervisors:	W. M. J. Coene
	Y. Shao
Project Duration:	March, 2023 - November, 2023
Faculty:	Applied Sciences and Technology, Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Summary

This project sought to implement an automatic differentiation scheme around the Rigorous Coupled-Wave Analysis algorithm to create a tool for parameter retrieval of a device's structure for 3D-profilometry applications.

Rigorous Coupled-Wave Analysis (RCWA) is a semi-analytical Maxwell solver, used to numerically evaluate the properties of three-dimensional structures which are periodic in two dimensions. The three-dimensional Maxwell equations are numerically evaluated in two dimensions using the Plane Wave Expansion Method. This results in a matrix differential equation along the remaining dimension. This differential equation is then solved to transmission and reflection values of incident light using the Transfer Matrix Method.

Automatic Differentiation is a method of evaluating the derivative/gradient/Jacobian of a function. By evaluating the gradient of a function, it is possible to perform gradient-descent to find local extrema of a function and the inputs which achieve that extrema. By finding the minimum of a loss function, gradient descent becomes an optimization tool. We sought to apply this to an RCWA algorithm where the loss function is minimized with respect to parameters which define the structure of the device. These optimal parameters then define the structure of a numerical model which most closely replicates measured data, and thereby become ”best guesses” of the device attempted to model.

Automatic differentiation was achieved by implementing a conventional RCWA algorithm in TensorFlow. TensorFlow is a software package which has built-in functionality for Automatic Differentiation and gradient descent. For this scheme, we have demonstrated the following:

- The ability to replicate the provided validation dataset down to near-rounding error accuracy.
- When the parameters are perturbed, the model is able to perform backpropagation and perform parameter retrieval to regain the correct value for every parameter of the device.

During the process of implementation, we have invented an alternate formulation for the S-parameter matrix, whose terms serve as analogues to those found in the conventional Scattering parameter matrix method. These are analogous to the **W**, **V**, and **X** matrices. We have demonstrated that using these analogues in lieu of the known values inside the S-parameter matrix algorithm will numerically yield the same values as those found by eigendecomposition.

We have demonstrated that these analogues can be calculated using iterative methods. These methods involve the use of matrix multiplications, scalar multiplications, matrix additions, and matrix solves. We have demonstrated that this algorithm can be up to 20 times faster on the HPC hardware provided by TU Delft.

The following areas for further research are suggested;

- Validation of the algorithm for a wider range of cases;
- Exploration of adjustments to the overall RCWA algorithm for improved performance;
- Alternate gradient-descent methods which could provide accelerated convergence;
- Attempting the use of gradient-descent methods for other Maxwell solvers.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
RCWA	Rigorous coupled-wave Analysis
FFF	Fast Fourier Factorization
PWEM	Plane-wave Expansion Method
TMM	Transfer matrix method
S-matrix	Scattering parameter matrix
T-matrix	Transfer parameter matrix
ETM	Enhanced Transfer parameter matrix
CPU	Central Processing Unit
GPU	Graphical Processing Unit
TPU	Tensor Processing Unit
expm	Matrix exponential
sqrtm	Matrix square root
sinm	Matrix sine
cosm	Matrix cosine
logm	Matrix logarithm

Symbols

Symbol	Definition
ϵ_0	The permittivity of vacuum.
μ_0	The permeability of vacuum.
ϵ	The relative permittivity of a material.
μ	The relative permeability of a material.
$\vec{E}$	Electric field vector in real space.
$\vec{D}$	Electric displacement vector in real space.
$\vec{B}$	Magnetic field vector in real space.
$\vec{H}$	Magnetic field density in real space.
$\tilde{\vec{H}}$	Normalized magnetic field density.
k_{inc}	the angular wavenumber of the incident plane wave.
$k_{x,\text{inc}}$	the angular spatial frequency of the incident plane wave along the x-direction.
$k_{y,\text{inc}}$	the angular spatial frequency of the incident plane wave along the y-direction.
θ	The polar angle of the incident light.
ϕ	The azimuthal angle of the incident light.
$\mathbf{s_x}$	A one-dimensional array of values, containing the Fourier coefficients E_x.
$\mathbf{s_y}$	A one-dimensional array of values, containing the Fourier coefficients E_y.
$\mathbf{u_x}$	A one-dimensional array of values, containing the Fourier coefficients $\tilde{H}_x$.
$\mathbf{u_y}$	A one-dimensional array of values, containing the Fourier coefficients $\tilde{H}_y$.

Symbol	Definition
m	The index for the (pseudo-) Fourier sum in the x-direction.
M	The maximum and minimum index for the (pseudo-) Fourier sum in the x-direction; harmonics beyond this range are truncated.
n	The index for the (pseudo-) Fourier sum in the y-direction.
N	The maximum and minimum index for the (pseudo-) Fourier sum in the y-direction; harmonics beyond this range are truncated.
j	The one-dimensional index for every possible permutation of the ordered pair (m, n).
J	The maximum index of j.
$\mathbf{K_x}$	A two-dimensional matrix which represents taking the partial derivative of a pseudoperiodic function with respect to x.
$\mathbf{K_y}$	A two-dimensional matrix which represents taking the partial derivative of a pseudoperiodic function with respect to y.
$[\![\epsilon]\!]$	A two-dimensional matrix which represents a convolution of a pseudo periodic function with the periodic permittivity ϵ.
$[[\frac{1}{\epsilon}]_x^{-1}]_y$	The permittivity Fast Fourier Factorization operator which acts as a deconvolution in x and a convolution in y.
$[[\frac{1}{\epsilon}]_y^{-1}]_x$	The permittivity Fast Fourier Factorization operator which acts as a deconvolution in y and a convolution in x.
$[\![\mu]\!]$	A two-dimensional matrix which represents a convolution of a pseudo periodic function with the periodic permeability μ.
$[[\frac{1}{\mu}]_x^{-1}]_y$	The permittivity Fast Fourier Factorization operator which acts as a deconvolution in x and a convolution in y.
$[[\frac{1}{\mu}]_y^{-1}]_x$	The permittivity Fast Fourier Factorization operator which acts as a deconvolution in y and a convolution in x.
L_x	Periodicity, or pitch, of the unit cell in the x-direction.
L_y	Periodicity, or pitch, of the unit cell in the y-direction.
x_w	Width of a pixel in the x-direction.
y_w	Width of a pixel in the y-direction.
x_c	Center of a pixel in the x-direction.
y_c	Center of a pixel in the y-direction.
p	The index for the pixel in the x-direction.
P	The total number of pixels in the x-direction.
q	The index for the pixel in the y-direction.
Q	The total number of pixels in the y-direction.
$\mathbf{P}$	The matrix which relates the partial derivative in the z-direction of the transverse electric field vectors, to the transverse magnetic field vectors.
$\mathbf{Q}$	The matrix which relates the partial derivative in the z-direction of the transverse magnetic field vectors, to the transverse electric field vectors.
$\mathbf{\Omega^2}$	The matrix given by the matrix multiplication **PQ**.
$\mathbf{\Omega}$	The matrix square root of $\mathbf{\Omega^2}$.
$\mathbf{W}$	The matrix which contains the set of eigenvectors of both $\mathbf{\Omega^2}$ and $\mathbf{\Omega}$. It represents the eigen-modes of the electric field as it propagates through the material.
$\boldsymbol{\lambda^2}$	The set of eigenvalues of $\mathbf{\Omega^2}$, expressed as a diagonal matrix.
$\boldsymbol{\lambda}$	The set of eigenvalues of $\mathbf{\Omega}$, expressed as a diagonal matrix.
$\mathbf{V}$	A matrix which represents the eigen-modes of the magnetic field as it propagates through the material.
$\mathbf{X}$	The propagation matrix of the eigenmodes.
$\mathbf{S}$	The scattering parameter matrix.
$\mathbf{T}$	The scattering transfer parameter matrix.

1 Introduction

A multi-variable function is a function which depends on multiple variables. As an example, take the waves in a pool; the water level depends on

- Where you are in the pool; this is the x-direction and y-direction, or longitude and latitude.
- When you are; call this t (for time).

A partial derivative of a function is how this function changes with only one of these variables, while the others are constant. If you stand still and observe how the height of where you stand changes with time, that is the partial derivative of height with respect to time. If you move along the x-direction while staying at the same y-direction and freezing time, that is the partial derivative of height with respect to x-direction, etc. A partial differential equation is an equation which describes how all the partial derivatives of a function are defined. However, this information is not enough to find the solution. We also need to know the boundary conditions; where does the pool end? Are the boundaries of the pool;

- Just regular walls, which let no water in or out?
- Are they infinity pool walls, which fix the water height at one value?
- Are the walls portals, where you can swim through the wall to come out through the other wall?

This last one sounds ridiculous, but it allows one to model an infinitely large, repeating pool with just a small section accurately. For example, an olympic swimming pool, which consists of 10 swimming lanes, can be modelled somewhat accurately by just looking at a single lane.

If we know the boundary conditions for all boundaries, we can then calculate the function itself. While it is possible to find the solution using pen-and-paper for the simplest problems, many are just too complex for this to be feasible. Instead, the problem is usually approximated using linear algebra, the solution to the linear algebra problem is calculated by a computer, and then the linear algebra solution is converted back to values we can readily interpret.

1.1. Maxwell equations

The Maxwell equations are a series of partial differential equations which describe the behavior of the electromagnetic fields. They are:

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \tag{1.1}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.2}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.3}$$

$$\nabla \times \mathbf{B} = \mu_0(\mathbf{J} + \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}) \tag{1.4}$$

In case of vacuum, we actually know that there are no point charges, and thus no currents. In such a case, the equations simplify to the following:

$$\nabla \cdot \mathbf{E} = 0 \tag{1.5}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.6}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.7}$$

$$\nabla \times \mathbf{B} = \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} \tag{1.8}$$

This actually leads to the following:

$$\nabla \times \nabla \times \mathbf{E} = \nabla \times (-\frac{\partial \mathbf{B}}{\partial t}) = -\frac{\partial (\nabla \times \mathbf{B})}{\partial t} = -\epsilon_0 \mu_0 \frac{\partial^2 \mathbf{E}}{\partial t^2} \tag{1.9}$$

$$\nabla \times \nabla \times \mathbf{E} = \nabla(\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E} = -\nabla^2 \mathbf{E} \tag{1.10}$$

which finally leads to:

$$\nabla^2 \mathbf{E} = \epsilon_0 \mu_0 \frac{\partial^2 \mathbf{E}}{\partial t^2} \tag{1.11}$$

In short, in a vacuum, the vectorial Maxwell equations become analogous to regular scalar wave equations. The same goes for stretches of homogeneous media. Because of this, the explanation of solution to the problem can be done by means of analogy.

For our case, we have a problem; we do not know all the boundary conditions. We have a total of four variables; x,y,z, and time. For three of the four variables (x, y, and time), we actually know the boundary conditions; they are periodic (this is the wall-portal boundary). However, we do not know the boundary conditions for the z-direction; in fact, this is the unknown to be solved for.

Rigorous Coupled-Wave Analysis (RCWA) deals with this in a peculiar way; it solves the Maxwell equations for all dimensions, except the z-dimension. The solution to this new problem is not the electromagnetic fields directly, but instead a description of how the electromagnetic fields change with respect to z. This then becomes an “ordinary” differential equation since it only depends on one dimension, which is then solved with pen-and-paper more easily. The catch is that, while these values only depend on z, the values themselves are vectors and matrices, which make working with them somewhat difficult. This is known as the field of matrix differential equations.

For such an equation, there are some electromagnetic fields where the change with respect to z is the same shape as the electromagnetic fields themselves. These are, in effect, waves going through the medium. These waves come in couples; one going forward, from small to large z, and another going backward, from large to small z. As they move, they can either bob up and down like regular waves, or they can exponentially decay, which are known as evanescent waves. These are the coupled waves in Rigorous Coupled-Wave Analysis.

A numerical method for a partial differential equation is a method which, evaluates a solution for a partial differential equation. These methods typically numerically approximate the terms inside the equation, and arrange these terms into a matrix. The solution to the matrix problem is then found, and the solution of the matrix problem serves as a surrogate for the solution of the partial differential equation.

These methods are not perfect, as they suffer from two sources of error;

- The first is that the numerical approximation is not perfect, so the matrix form does not perfectly replicate the real partial differential equation. This error is typically referred to as ”truncation error”.
- In the process of solving the problem, rounding errors are introduced, and propagate through the system as the problem is solved.

1.2. The tube & the two-port network

A lot of optical systems we have are constructed somewhat similar in shape to a tube. You have light going in or out at one end, you have light going in or out at another end, how these two relate to each other is determined by what is inside the tube. The tube is a finite slice in the remaining two directions. A good example of this is the telescope or the microscope. However, this can also be used to describe fiber optics for internet transmission, or glasses, or meta-lenses, or diffractive gratings.

If you have a relation between the fields at one end and the fields at another end, you can describe the fields at end 2 to be the fields at end 1 which are transferred. Mathematically, the operation which transfers the fields at end 1 to end 2, when expressed as a matrix, are known as a transfer matrix. The method which calculates the fields at end 2 as a function of the field at end 1, is known as the Transfer Matrix Method.

We know that the electromagnetic fields propagate as waves. In relatively simple situations, such as in a vacuum with no charges or currents, the Maxwell equations (the equations which describe how the electromagnetic fields behave) simplify to a set of wave equations of the Helmholtz type. The solutions to these equations are known as waves.

There are multiple ways to describe such waves. One way to describe them is as spherical waves. Just like how dropping a rock in a pond produces a circular wave on the surface of the water, electromagnetic waves can be seen as spheres rippling through the 3d surface that we refer to as space.

Another way of modelling waves is as plane waves. These are waves with no curvature, that stretch infinitely far. While this does not really happen in the real world, instead starting as something more closely related to the spheres previously described, we can describe these spherical waves as a superposition of plane waves. The process of describing waves as a superposition of these simpler plane waves is known as plane wave expansion. It is arguably a misnomer, since in practice, this is oftentimes no different from a Fourier decomposition, where the Fourier basis functions are simply plane waves. Most importantly, the way plane waves propagate is quite well-known.

With this information, we can do the following:

- We take the fields at one end of the tube, and decompose the fields into plane wave using plane wave expansion.
- We can calculate how these new waves propagate from one end to another.
- We then transform these new plane waves back to the fields, which then gives us a relation between the fields.

Doing such steps for a tube filled with vacuum leads us to an equation such as the Rayleigh-Sommerfeld propagation equation:

$$U(x,y,z) = \frac{z}{i\lambda}\int\int U(x',y',0)\frac{e^{ik\sqrt{(x-x')^2+(y-y')^2+z^2}}}{(x-x')^2+(y-y')^2+z^2}dx'dy' \tag{1.12}$$

While there does not seem to be a Fourier transform in the present form, this is because of the convolution theorem; a multiplication in Fourier space becomes a convolution in real space, and vice versa. This two-dimensional integral is effectively a convolution, which can be derived using the plane-wave expansion method, although this is not the only way of deriving this equation.

In spirit, RCWA attempts to perform the same trick. However, the structure we are attempting to model has a series of complicating factors;

- The tube is no longer a tube, but a periodic structure with periodic boundary constraints; instead of hitting a wall, it is as if you loop back on the other side.
- The inside of the ”tube” is no longer homogeneous, which causes the tube to not let all light through, but also absorb and reflect some of the light.
- Furthermore, the fact that the inside of the tube is no longer homogeneous makes the propagation of the waves more complex than the Rayleigh-Sommerfeld propagator.

- The tube is actually a series of different tubes, stitched together end-to-end, so now there is actually light travelling forward and backward at the same time through basically all the tubes.
- The waves propagating now also include evanescent waves, which shrink going forward and grow going backward, and risk making further calculations useless unless dealt with properly.

1.2.1. Two-port network

The interesting part of the tube is that the ”entrance” and ”exit” are arbitraty; Light can enter from end 1 and exit end 2, with some reflected light going out end 1, or it can enter from end 2 and exit end 1, with some reflected light going out end 2. We can model this as a two-port network; Two inputs on either side, two outputs at either side.

Figure 1.1: Illustration of a tube as a two-port network.

It is important to note that, in this case, it is not the fields themselves going in or out of the two-port network, but field excitations which we know as light. The electromagnetic fields can have parts of light going forward and going backward, so the idea of all the fields together moving forward does not make sense physically. While there is a relation in the electromagnetic fields, this is not one end being a function of the other; by contrast, the outgoing light on either side is a function of the incoming light on either side. Figuring out whether light is going one way or another is not a trivial task; usually in the propagator it describes both how forward-moving and backward-moving light changes. However, there are some ways to disentangle the forward-moving and backward-moving light.

From such tubes, we can conceptually construct a new one by ”concatenation”, which is basically adding two two-port networks end-to-end, who then together function as a new singular two-port network.

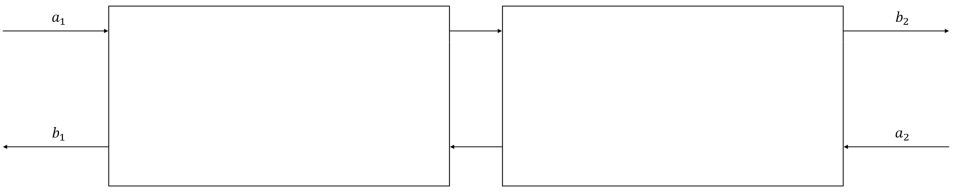


Figure 1.2: Illustration of two two-port networks concatenated.

With this, we can effectively construct an optical system from a bunch of ”elementary” two port networks by stringing them all together in proper order.

Because of this, the game plan is as follows:

- We derive a numerical model for the propagation of the plane wave expansion with what is known as the plane wave expansion method, or Fourier decomposition method;
- We then separate out the forward-moving light from the backward-moving light inside each tube;

- Using that, we construct a bunch of two-port networks which describe the incoming-outgoing light relations for each slice;
- Then, we finally construct the overall system by stringing together the tubes using concatenation.

The first step in RCWA falls under the Plane Wave Expansion Method, while steps 2 to 4 fall under the Transfer Matrix Method.

1.3. Problem inversion through automatic differentiation

Ultimately, the goal of this Master Thesis project was to produce an RCWA algorithm which was differentiable. Plenty of implementations of the forward model already exist, so while a forward implementation is difficult, it would not yield new insights or progress unless the implementation was novel. If the method is differentiable, however, it would allow for problem inversion, and thus be worth exploring. In the end, the resulting algorithm is both novel and differentiable.

One currently popular method of problem inversion is the gradient descent method. In essence, if one is able to calculate the gradient of the output with respect to all its inputs, updating the inputs dependent on the gradients allows one to iteratively approach a local optimum of the function. If this function in question is a loss function (a function which quantifies how closely two datasets correspond with one another), this allows us to minimize the loss function by gradient descent, and thereby to produce a model with strong predictive power.

These methods have sprung into the limelight recently, where they are known as AI. In those cases, the function is an Artificial Neural Network, which is a numerical model which seeks to replicate the structure of the human brain, which thereby hopes to inherit the same emergent property of pattern recognition as humans have. This extremely non-linear system is then trained with the use of gradient-descent methods.

With the increasing success of AI over the last decade, a whole ecosystem has emerged to support it. This includes hardware, in the form of highly-parallelized computer architecture to evaluate the inherently parallel structure of neural networks, but also software, which seeks to make the implementation of Neural Networks and the optimization regime to train these neural networks faster and easier.

The goal of this Master Thesis project is to implement an RCWA algorithm in Tensorflow - one of the more popular software packages for Artificial Neural Networks - so that we can make use of its optimization regime. We use this regime to find the ”optimal” internal structure of the device, so that it replicates the measured diffraction pattern as closely as possible. By doing so, we would have developed software which is able to demonstrate parameter retrieval, and find an as-close-as-possible solution to the inverse problem.

1.4. Structure of this Thesis

The second and third chapters are dedicated to the Plane Wave Expansion Method and Transfer Matrix Method, respectively. Together, they contain all the equations used to implement the overall RCWA algorithm.

The fourth chapter contains an explanation of Tensorflow, what it attempts to do vis-a-vis machine learning, how we have adapted these processes for RCWA, and the limitations that come with this application.

The fifth chapter contains the information on how we modelled the structure itself, how these model structures are then calculated into operators used by the Plane Wave Expansion Method, and how the choices of the modelling were influenced by both the needs coming from RCWA as well as the limitations applied by the backpropagation algorithm.

The sixth chapter contains an explanation of the improved formulation, which we found to be up to twenty times faster on GPU architecture than the previous formulation.

The seventh chapter contains another alternate formulation which is not fully complete in implementation.

The eighth chapter contains the results of our algorithm for certain structure cases. This includes validation of the RCWA algorithm, as well as a demonstration of its ability to perform parameter retrieval. This chapter also contains data on the computational performance of the improvements as outlined in the sixth chapter.

The ninth chapter contains discussion about the results, as well as some previously-unmentioned algorithms that are worth mentioning but are otherwise not integrated into the wider RCWA algorithm.

The final chapter contains the conclusion and recommendation for future research.

2

Plane Wave Expansion Method

2.1. Maxwell equations

In case of no free charges, no free currents, and all fields being time-harmonic, the Maxwell equations simplify to the following;

$$\begin{aligned}
\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} &= k_0 \mu_r \tilde{H}_x \\
\frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} &= k_0 \mu_r \tilde{H}_y \\
\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} &= k_0 \mu_r \tilde{H}_z \\
\frac{\partial \tilde{H}_z}{\partial y} - \frac{\partial \tilde{H}_y}{\partial z} &= k_0 \epsilon_r E_x \\
\frac{\partial \tilde{H}_x}{\partial z} - \frac{\partial \tilde{H}_z}{\partial x} &= k_0 \epsilon_r E_y \\
\frac{\partial \tilde{H}_y}{\partial x} - \frac{\partial \tilde{H}_x}{\partial y} &= k_0 \epsilon_r E_z
\end{aligned} \tag{2.1}$$

The H-field present in these equations is actually a slight normalization of the actual H-field, so as to simplify the equations. It is therefore denoted with a tilde.

$$\vec{\tilde{H}} = -i\sqrt{\frac{\mu_0}{\epsilon_0}}\vec{H} \tag{2.2}$$

2.2. Floquet theorem

Floquet theory is mathematical theory, which applies to differential equations of the following kind;

$$\frac{dx}{dt} = A(t)x \tag{2.3}$$

where $A$is periodic with respect to variable t, with periodicity L. Solutions to such equations come in the following form;

$$\psi(t) = e^{ikt}p(t) \tag{2.4}$$

where $p(t)$ is periodic with respect to variable t, with periodicity L, and k is real.

This theorem has been discovered independently multiple times by mathematicians and physicists such as Floquet [1], Hill [2], and Bloch [3]. Since the permittivity (and permeability) of the device are periodic in the x-direction and y-direction, the differential equation of the electromagnetic fields in the x-direction

and y-direction takes the form as denoted by Floquet theory. This result will not be proven in this report. Thus, the solutions are of the form given by equation 2.4, with respect to the x-direction and y-direction.

We are, however, only interested in a subset of all solutions; namely, those states which are excited by incident radiation of a specific time harmonic, in other words, monochromatic light. Additionally, since the permittivity and permeability are periodic in the x-direction and y-direction, Floquet's theorem applies in the x-direction and y-direction. Because of the need of continuity of the electromagnetic fields, the phase term is directly inherited from the incident field. Thus, we know that the fields inside the device must be of the following form;

$$\begin{aligned}\vec{E}(x,y,z) &= e^{-i(k_{x,\text{inc}}x+k_{y,\text{inc}}y)}\vec{S}(x,y,z)\\ \vec{\tilde{H}}(x,y,z) &= e^{-i(k_{x,\text{inc}}x+k_{y,\text{inc}}y)}\vec{U}(x,y,z)\end{aligned} \tag{2.5}$$

Where $\vec{S}$ and $\vec{U}$ are periodic in the x-direction and y-direction. The exact behavior of $\vec{S}$ and $\vec{U}$ in the z-direction is not known, and is in fact the unknown which the Plane Wave Expansion Method attempts to find. Since the Maxwell equations are linear (and we assume that the radiation does not alter the permittivity/permeability of the material as is the case in non-linear optics), we ignore all other incident fields.

2.3. Fourier space

The Plane Wave Expansion Method is a numerical method which operates in spatial frequency space, also sometimes referred to as Fourier space. This is the method used to formulate the first-order matrix differential equation in the z-direction.

The fields can be described as Fourier sums, modulated by an incident plane wave, in the x-direction and y-direction. The behavior along the z-direction is not known.

$$E_i(x,y,z) = \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} s_i^{m,n}(z)\Phi^{m,n}(x,y) \tag{2.6}$$

$$\tilde{H}_j(x,y,z) = \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} u_j^{m,n}(z)\Phi^{m,n}(x,y) \tag{2.7}$$

for $i,j \in \{x,y,z\}$, where

$$\Psi^{\text{inc}}(x,y) = e^{-i(k_{x,\text{inc}}x+k_{y,\text{inc}}y)} \tag{2.8}$$

$$\Phi^{m,n}(x,y) = e^{i2\pi(\frac{m}{L_x}x+\frac{n}{L_y}y)} \tag{2.9}$$

This is, in effect, the plane wave expansion being performed on the fields. However, instead of modelling the plane waves as being a Fourier transform of the fields, they instead are the Fourier series of the fields, with discrete intervals in k-space between the Fourier terms. The discrete intervals are due to the (pseudo-)periodic boundary conditions applied to the field, where the size of the x-interval in Fourier space is inversely related to the pitch in the x-direction:

$$T_x = \frac{2\pi}{L_x} \tag{2.10}$$

Similarly, in the y-direction, we have:

$$T_y = \frac{2\pi}{L_y} \tag{2.11}$$

How many of these frequencies are taken into account is kept arbitrary. One might think that the maximum frequency is related to the Nyquist frequency, which in turn is equal to two divided by the real space sampling distance. However, this is not necessarily the case; in some cases, considering maximum frequencies beyond the Nyquist frequency yields a higher accuracy still. In other cases, the maximum frequency considered is much lower than the Nyquist frequency for performance reasons. In other cases still, the sampling in real space may not have regular intervals, at which point a clear Nyquist frequency no longer exists. That said, the fact that the relation between discrete points in

real space and Fourier space is weakened is a serious problem, and could be the source of some of RCWA's odd behavior which Fast Fourier Factorization (a topic which is discussed in Chapter 5) attempts to fix. Alternate formulations which attempt to preserve this property may be less error-prone, and thus present a potential future area of research. The derivative in the x-direction:

$$\begin{aligned}\frac{\partial}{\partial x}E_i(x,y,z) &= \frac{\partial}{\partial x}\left(\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\Phi^{m,n}(x,y)\right)\\ &= \frac{\partial}{\partial x}(\Psi^{\text{inc}}(x,y))\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\Phi^{m,n}(x,y) + \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\frac{\partial}{\partial x}(\Phi^{m,n}(x,y))\\ &= -i\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\left(k_{x,\text{inc}}-2\pi\frac{m}{L_x}\right)\Phi^{m,n}(x,y)\\ &= -i\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)k_x^m\Phi^{m,n}(x,y)\end{aligned} \tag{2.12}$$

where we introduce the following shorthand notation:

$$k_x^m = k_{x,\text{inc}} - 2\pi\frac{m}{L_x} \tag{2.13}$$

We evaluate the derivative in the y-direction in a similar way;

$$\begin{aligned}\frac{\partial}{\partial y}E_i(x,y,z) &= \frac{\partial}{\partial y}\left(\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\Phi^{m,n}(x,y)\right)\\ &= \frac{\partial}{\partial y}(\Psi^{\text{inc}}(x,y))\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\Phi^{m,n}(x,y) + \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\frac{\partial}{\partial y}(\Phi^{m,n}(x,y))\\ &= -i\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)\left(k_{y,\text{inc}}-2\pi\frac{n}{L_y}\right)\Phi^{m,n}(x,y)\\ &= -i\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}s_i^{m,n}(z)k_y^n\Phi^{m,n}(x,y)\end{aligned} \tag{2.14}$$

where we introduce the following shorthand notation:

$$k_y^n = k_{y,\text{inc}} - 2\pi\frac{n}{L_y} \tag{2.15}$$

The derivative in the z-direction is evaluated as follows:

$$\frac{\partial}{\partial z}E_i(x,y,z) = \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{d}{dz}(s_i^{m,n}(z))\Phi^{m,n}(x,y) \tag{2.16}$$

The permittivity and permeability can be described as regular Fourier sums in the x-direction and y-direction. They are assumed to be constant along the z-direction. Since the permittivity and permeability are assumed to be isotropic, they are described as scalar distributions.

$$\epsilon = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}a^{m,n}\Phi^{m,n}(x,y) \tag{2.17}$$

$$\mu = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}b^{m,n}\Phi^{m,n}(x,y) \tag{2.18}$$

The product of the permittivity and an electric field component is evaluated as follows:

$$\begin{aligned}\epsilon(x,y)E_i(x,y,z) &= \left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a^{m,n}\Phi^{m,n}(x,y)\right)\left(\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} s_i^{m,n}(z)\Phi^{m,n}(x,y)\right)\\ &= \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a^{m,n}\Phi^{m,n}(x,y)\sum_{m'=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty} s_i^{m',n'}(z)\Phi^{m',n'}(x,y)\\ &= \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\sum_{m'=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty} a^{m,n}s_i^{m',n'}(z)\Phi^{m+m',n+n'}(x,y)\\ &= \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\sum_{m'=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty} a^{m-m',n-n'}s_i^{m',n'}(z)\Phi^{m,n}(x,y)\end{aligned} \tag{2.19}$$

This product of two Fourier sums is known in literature as a Cauchy product; it is mathematically equivalent to the convolution theorem of Fourier transforms, with the main deviation being its application to discrete Fourier transforms, rather than continuous ones.

Using a similar sequence of steps, we evaluate the product of the permeability and a magnetic field component to the following:

$$\mu(x,y)\tilde{H}_j(x,y,z) = \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\sum_{m'=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty} b^{m-m',n-n'}u_j^{m',n'}(z)\Phi^{m,n}(x,y) \tag{2.20}$$

2.4. Substitution into Maxwell equations

From here on, we will substitute the above-found expressions into the Maxwell equations to derive new matrix equations, which can be evaluated computationally. Since a lot of terms are shared between the Maxwell equations, only the following equation will be explicitly evaluated:

$$\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} = k_0\mu_r\tilde{H}_x \tag{2.21}$$

Where

$$\tilde{H}_x = -i\sqrt{\frac{\mu_0}{\epsilon_0}}H_x \tag{2.22}$$

This normalization is similarly applied to H_y and H_z. The purpose of this normalization is an attempt to introduce some symmetry between the E-fields and H-fields in previous literature, which has been adopted as a convention for this report.

Furthermore, the other five Maxwell equations can be found with a similar sequence of steps.

If we substitute the expressions from equation 2.14, 2.16, and 2.20 into this Maxwell equation, then we get the following expression:

$$\begin{aligned}-i\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} s_z^{m,n}(z)k_y^n\Phi^{m,n}(x,y) - \Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{d}{dz}(s_y^{m,n}(z))\Phi^{m,n}(x,y)\\ = k_0\Psi^{\text{inc}}(x,y)\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\sum_{m'=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty} b^{m-m',n-n'}u_x^{m',n'}(z)\Phi^{m,n}(x,y)\end{aligned} \tag{2.23}$$

Since this equation must hold for all possible values of x and y, this equation becomes an infinite series of equations due to phase matching:

$$-is_z^{m,n}(z)k_y^n - \frac{d}{dz}\big(s_y^{m,n}(z)\big) = k_0\sum_{m'=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty} b^{m-m',n-n'}u_x^{m',n'}(z), \forall m,n \in \mathbb{Z}^+ \tag{2.24}$$

2.5. Truncation

While this formulation is still completely rigorous, it is not solvable, due to two reasons;

- The summation term is an infinite sum;
- There are an infinite number of linear equations to solve.

Both of these can be solved by truncating the problem to only consider a finite number of terms. We do this by instead evaluating the following equations:

$$-is_z^{m,n}(z)k_y^n - \frac{d}{dz}\big(s_y^{m,n}(z)\big) = k_0 \sum_{m'=-M}^{M} \sum_{n'=-N}^{N} b^{m-m',n-n'} u_x^{m',n'}(z),$$
$$\forall m \in \{-M, -M+1, \ldots, M-1, M\}, n \in \{-N, -N+1, \ldots, N-1, N\} \quad (2.25)$$

and M, N are both positive integers.

The reason why m and m' have the same domains, and why n and n' have the same domains is to ensure that the operators have the same input and output dimensions, which is a necessity for any future operators to be invertible. However, the selected orders is arbitrary, and different selections of orders have been shown to have faster computation times, with limited loss of accuracy. [4]

2.6. Vector and matrix definitions

In order for the computer to be able to evaluate these equations, we must express these vectors as some combination of vectors and matrices. We start by introducing a new singular index which, on its own, handles the double indices of m and n simultaneously.

More formally, we unroll a 2d vector which has $(2M+1)$ entries in dimension 1, $(2N+1)$ entries in dimension 2, into a one-dimensional vector ranging from 0 to $J = (2M+1)(2N+1) - 1$, with index j. We can express j as a function of m and n as follows:

$$j = (n+N) * (2M+1) + (m+M) \quad (2.26)$$

It is important to note that this definition of j as a function of m and n is arbitrary; a definition where n is swapped with m and N is swapped with M yields an equally valid formulation. The reason this formulation is chosen is that, conceptually, the selected order starts at the top-left corner, sweeps from left to right, then jumps down one position and starts at the left again, the same way an electron gun in the CRT TV would excite the pixels on the screen. The most important part is that the choice of index is consistent throughout the method.

We can express m and n as function of j as follows:

$$m(j) = j\%(2M+1) - M \quad (2.27)$$
$$n(j) = \lfloor \frac{j}{2M+1} \rfloor - N \quad (2.28)$$

Where $\%$ is the modulo operator. We can actually do the exact same with the auxiliary indices:

$$j' = (n'+N) * (2M+1) + (m'+M) \quad (2.29)$$
$$m'(j') = j'\%(2N+1) - M \quad (2.30)$$
$$n'(j') = \lfloor \frac{j'}{2N+1} \rfloor - N \quad (2.31)$$

With this, we can transform equation 2.25 into a one-dimensional series of equations:

$$-is_z^{m(j),n(j)}(z)k_y^{n(j)} - \frac{d}{dz}\big(s_y^{m(j),n(j)}(z)\big) = k_0 \sum_{j'=0}^{J-1} b^{m(j)-m'(j'),n(j)-n'(j')} u_x^{m'(j'),n'(j')}(z),$$
$$\forall j \in \{0, 1, \ldots, J-1\}, J = (2M+1)(2N+1) \quad (2.32)$$

To finish this equation, we make the following substitution:

$$-is_z^{m(j),n(j)}(z)k_y^{n(j)} = -i\sum_{j'=0}^{J-1} k_y^{n(j)}\delta_{jj'}s_z^{m'(j'),n'(j')}(z) \tag{2.33}$$

where $\delta_{jj'}$ is the Kronecker delta, which is defined as follows:

$$\delta_{jj'} = \begin{cases} 1 & \text{for } j = j' \\ 0 & \text{else} \end{cases} \tag{2.34}$$

Thus, we get:

$$-i\sum_{j'=0}^{J-1} k_y^{n'(j')}\delta_{jj'}s_z^{m(j),n(j)}(z) - \frac{d}{dz}\left(s_y^{m(j),n(j)}(z)\right) = k_0\sum_{j'=0}^{J-1} b^{m(j)-m'(j'),n(j)-n'(j')}u_x^{m'(j'),n'(j')}(z),$$
$$\forall j \in \{0,1,\ldots,J-1\}, J = (2M+1)(2N+1) \tag{2.35}$$

The reason for all of this is is done is that each equation has terms which are either scalars representing the Fourier terms of the fields, or dot products between two vectors, one of which represents all the Fourier terms of a field. If we then treat this system of equations as one vector equation, the terms are either vectors containing all the Fourier coefficients, or matrix-vector products, where the matrices are square and the vectors have the same number of Fourier coefficients. These operators being square is what makes operations such as inversion, and by extension eigendecomposition, possible, so this property is crucial for the Transfer Matrix Method further down the line.

In matrix form, we can express equation 2.35 as follows:

$$-ik_0\mathbf{K_y}\mathbf{s_z} - \frac{d}{dz}\mathbf{s_y} = k_0[\![\mu]\!]\mathbf{u_x} \tag{2.36}$$

where the vectors have the following entries:

$$(\mathbf{s_i})_j = s_i^{m(j),n(j)} \tag{2.37}$$

$$(\mathbf{u_i})_j = u_i^{m(j),n(j)} \tag{2.38}$$

and the operators are defined as follows:

$$(\mathbf{K_y})_{jj'} = k_y^{n'(j')}\delta_{jj'} \tag{2.39}$$

$$[\![\mu]\!]_{jj'} = b^{m(j)-m'(j'),n(j)-n'(j')} \tag{2.40}$$

We also define the following matrix operations not used in this equation, but used in the other ones:

$$(\mathbf{K_x})_{jj'} = k_x^{m'(j')}\delta_{jj'} \tag{2.41}$$

$$[\![\epsilon]\!]_{jj'} = a^{m(j)-m'(j'),n(j)-n'(j')} \tag{2.42}$$

2.6.1. Illustrative case

From here on, the process of evaluating the Plane Wave Expansion Method for the case where $M = N = 1$ will be illustrated. These serve only as a demonstration; they serve as a visual aid to show how equations previously derived are transformed into matrix equations. They provide no insight in cases with a different number of orders, and should not serve as instructions for implementation. In all of these cases, the primary index denotes the order in x, and the secondary index denotes the order in y.

To start, in such a case, we have for instance that $J = 9$, and thus j ranges from 0 to 8. The m and n indices thus evaluate to the following:

j	$m(j)$	$n(j)$
0	-1	-1
1	0	-1
2	1	-1
3	-1	0
4	0	0
5	1	0
6	-1	1
7	0	1
8	1	1

Likewise, for the auxiliary indices:

j'	$m'(j')$	$n'(j')$
0	-1	-1
1	0	-1
2	1	-1
3	-1	0
4	0	0
5	1	0
6	-1	1
7	0	1
8	1	1

The Fourier coefficient vectors contain the following:

$$\mathbf{s}_i = \begin{bmatrix} s_i^{-1,-1}(z) \\ s_i^{0,-1}(z) \\ s_i^{1,-1}(z) \\ s_i^{-1,0}(z) \\ s_i^{0,0}(z) \\ s_i^{1,0}(z) \\ s_i^{-1,1}(z) \\ s_i^{0,1}(z) \\ s_i^{1,1}(z) \end{bmatrix}, \forall i \in \{x, y, z\} \tag{2.43}$$

$$\mathbf{u}_i = \begin{bmatrix} u_i^{-1,-1}(z) \\ u_i^{0,-1}(z) \\ u_i^{1,-1}(z) \\ u_i^{-1,0}(z) \\ u_i^{0,0}(z) \\ u_i^{1,0}(z) \\ u_i^{-1,1}(z) \\ u_i^{0,1}(z) \\ u_i^{1,1}(z) \end{bmatrix} \forall i \in \{x, y, z\} \tag{2.44}$$

the system of equations 2.25 for the case where $M = N = 1$ can be expressed as a single vector equation as follows:

$$\begin{bmatrix} is_z^{-1,-1}(z)k_y^{-1} - \frac{d}{dz}\left(s_y^{-1,-1}(z)\right) \\ is_z^{0,-1}(z)k_y^{-1} - \frac{d}{dz}\left(s_y^{0,-1}(z)\right) \\ is_z^{1,-1}(z)k_y^{-1} - \frac{d}{dz}\left(s_y^{1,-1}(z)\right) \\ is_z^{-1,0}(z)k_y^{0} - \frac{d}{dz}\left(s_y^{-1,0}(z)\right) \\ is_z^{0,0}(z)k_y^{0} - \frac{d}{dz}\left(s_y^{0,0}(z)\right) \\ is_z^{1,0}(z)k_y^{0} - \frac{d}{dz}\left(s_y^{1,0}(z)\right) \\ is_z^{-1,1}(z)k_y^{1} - \frac{d}{dz}\left(s_y^{-1,1}(z)\right) \\ is_z^{0,1}(z)k_y^{1} - \frac{d}{dz}\left(s_y^{0,1}(z)\right) \\ is_z^{1,1}(z)k_y^{1} - \frac{d}{dz}\left(s_y^{1,1}(z)\right) \end{bmatrix} = \begin{bmatrix} k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{-1-m',-1-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{-m',-1-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{1-m',-1-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{-1-m',-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{-m',-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{1-m',-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{-1-m',1-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{-m',1-n'} u_x^{m',n'}(z) \\ k_0 \sum_{m'=-1}^{1}\sum_{n'=-1}^{1} b^{1-m',1-n'} u_x^{m',n'}(z) \end{bmatrix} \tag{2.45}$$

The matrix operators will be square matrices, where the number of rows and columns are both equal to $J = 9$. Their entries are given as follows:

$$\mathbf{K}_y = \begin{bmatrix} k_y^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k_y^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_y^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_y^{0} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_y^{0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k_y^{0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k_y^{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_y^{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_y^{1} \end{bmatrix} \tag{2.46}$$

$$[\![\mu]\!] = \begin{bmatrix} b^{0,0} & b^{-1,0} & b^{-2,0} & b^{0,-1} & b^{-1,-1} & b^{-2,-1} & b^{0,-2} & b^{-1,-2} & b^{-2,-2} \\ b^{1,0} & b^{0,0} & b^{-1,0} & b^{1,-1} & b^{0,-1} & b^{-1,-1} & b^{1,-2} & b^{0,-2} & b^{-1,-2} \\ b^{2,0} & b^{1,0} & b^{0,0} & b^{2,-1} & b^{1,-1} & b^{0,-1} & b^{2,-2} & b^{1,-2} & b^{0,-2} \\ b^{0,1} & b^{-1,1} & b^{-2,1} & b^{0,0} & b^{-1,0} & b^{-2,0} & b^{0,-1} & b^{-1,-1} & b^{-2,-1} \\ b^{1,1} & b^{0,1} & b^{-1,1} & b^{1,0} & b^{0,0} & b^{-1,0} & b^{1,-1} & b^{0,-1} & b^{-1,-1} \\ b^{2,1} & b^{1,1} & b^{0,1} & b^{2,0} & b^{1,0} & b^{0,0} & b^{2,-1} & b^{1,-1} & b^{0,-1} \\ b^{0,2} & b^{-1,2} & b^{-2,2} & b^{0,1} & b^{-1,1} & b^{-2,1} & b^{0,0} & b^{-1,0} & b^{-2,0} \\ b^{1,2} & b^{0,2} & b^{-1,2} & b^{1,1} & b^{0,1} & b^{-1,1} & b^{1,0} & b^{0,0} & b^{-1,0} \\ b^{2,2} & b^{1,2} & b^{0,2} & b^{2,1} & b^{1,1} & b^{0,1} & b^{2,0} & b^{1,0} & b^{0,0} \end{bmatrix} \tag{2.47}$$

Finally, while the below operators are not included in the example matrix equation, they are found in others, so are important to mention:

$$\mathbf{K}_x = \begin{bmatrix} k_x^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k_x^{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_x^{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_x^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_x^{0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k_x^{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k_x^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_x^{0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_x^{1} \end{bmatrix} \tag{2.48}$$

$$[\![\epsilon]\!] = \begin{bmatrix} a^{0,0} & a^{-1,0} & a^{-2,0} & a^{0,-1} & a^{-1,-1} & a^{-2,-1} & a^{0,-2} & a^{-1,-2} & a^{-2,-2} \\ a^{1,0} & a^{0,0} & a^{-1,0} & a^{1,-1} & a^{0,-1} & a^{-1,-1} & a^{1,-2} & a^{0,-2} & a^{-1,-2} \\ a^{2,0} & a^{1,0} & a^{0,0} & a^{2,-1} & a^{1,-1} & a^{0,-1} & a^{2,-2} & a^{1,-2} & a^{0,-2} \\ a^{0,1} & a^{-1,1} & a^{-2,1} & a^{0,0} & a^{-1,0} & a^{-2,0} & a^{0,-1} & a^{-1,-1} & a^{-2,-1} \\ a^{1,1} & a^{0,1} & a^{-1,1} & a^{1,0} & a^{0,0} & a^{-1,0} & a^{1,-1} & a^{0,-1} & a^{-1,-1} \\ a^{2,1} & a^{1,1} & a^{0,1} & a^{2,0} & a^{1,0} & a^{0,0} & a^{2,-1} & a^{1,-1} & a^{0,-1} \\ a^{0,2} & a^{-1,2} & a^{-2,2} & a^{0,1} & a^{-1,1} & a^{-2,1} & a^{0,0} & a^{-1,0} & a^{-2,0} \\ a^{1,2} & a^{0,2} & a^{-1,2} & a^{1,1} & a^{0,1} & a^{-1,1} & a^{1,0} & a^{0,0} & a^{-1,0} \\ a^{2,2} & a^{1,2} & a^{0,2} & a^{2,1} & a^{1,1} & a^{0,1} & a^{2,0} & a^{1,0} & a^{0,0} \end{bmatrix} \tag{2.49}$$

This concludes the illustrative example.

2.7. Fast Fourier Factorization (FFF)

The numerical scheme, as implemented now, defines the ϵ and μ operators as a matrix convolution with $\epsilon(x, y)$ and $\mu(x, y)$, respectively. This is, however, not the only way to define such an operator. An alternate way of defining this operator is by evaluating the deconvolution of $\frac{1}{\epsilon(x,y)}$ and $\frac{1}{\mu(x,y)}$. If a convolution operator is expressed as a matrix, the inverse of that matrix serves as the deconvolution. As such, the operator

$$[\![\frac{1}{\epsilon}]\!]^{-1} \tag{2.50}$$

is a valid approximation for the convolution with ϵ in Fourier space.

While a convolution with a function is mathematically equivalent to a deconvolution with the inverse of said function, this is generally not true for truncated matrices.

$$[\![\epsilon]\!] \neq [\![\frac{1}{\epsilon}]\!]^{-1} \tag{2.51}$$

Thus, we have multiple approximations for the convolution operator as a matrix, and if they are dissimilar, one must perform better as an approximation than the other. To know which one is best, we must first explain how the fields in question interact with a boundary.

A boundary is defined as a surface, across which exists a change in permittivity and/or permeability. The normal vector to this surface is denoted as $\vec{n}_{12}$. In the case of no charge and no current on the surface, the continuity equations can be described as such:

$$(\vec{E_1} - \vec{E_2}) \times \vec{n}_{12} = \vec{0} \tag{2.52}$$

$$(\vec{H_1} - \vec{H_2}) \times \vec{n}_{12} = \vec{0} \tag{2.53}$$

$$(\vec{D_1} - \vec{D_2}) \cdot \vec{n}_{12} = 0 \tag{2.54}$$

$$(\vec{B_1} - \vec{B_2}) \cdot \vec{n}_{12} = 0 \tag{2.55}$$

In short, the E-fields and H-fields perpendicular to the surface normal are continuous, while the D-fields and B-fields parallel to the surface normal are continuous. By extension, the D-fields and B-fields perpendicular to the surface normal are discontinuous, while the E-fields and H-fields parallel to the surface normal are discontinuous.

Continuous periodic functions have higher-frequency terms that are relatively small. The approximation of such a function, as a Fourier sum with a finite number of terms, is a decent approximation. Such functions are referred to as ”type 1” functions, and are said to ”converge quickly”.

Discontinuous periodic functions can also be described by the use of Fourier sums. However, owing to the discontinuity, the Fourier sum contains relatively large higher-frequency exponentials. The approximation of such a function, as a Fourier sum with a finite number of terms, is poorer. Such functions are referred to as ”type 2” functions, and are said to ”converge slowly”.

Below is a general table outlining the types of the field components in the most general case:

Field component	x direction	y direction	z direction
E_x	2	1	1
E_y	1	2	1
E_z	1	1	2
H_x	2	1	1
H_y	1	2	1
H_z	1	1	2
D_x	1	2	2
D_y	2	1	2
D_z	2	2	1
B_x	1	2	2
B_y	2	1	2
B_z	2	2	1

Operators acting on poor approximations will yield outputs that themselves are also poor approximations. [5]

The solution to this problem is to frame all operators as acting on type 1 field components. Any convolutions with function $f(x)$ acting on type 1 functions instead become convolutions with function $\frac{1}{f(x)}$ acting on type 2 functions. Once these operators are turned into matrices, the deconvolution can be found by taking the inverse of the convolution matrix. The process of implementing the correct method is referred to as "Fast Fourier Factorization".

The "Laurent rule" convolution operator is given as the following:

$$d_x = [\epsilon]_x s_x \tag{2.56}$$

This operator works best if the field component $\mathbf{s}_x$ is type 1 in the x-direction. By contrast, the "Li rule" is given as the following:

$$d_x = [\frac{1}{\epsilon}]_x^{-1} s_x \tag{2.57}$$

This works best if the field component $\mathbf{s}_x$ is type 2 in the x-direction.

2.7.1. Fast Fourier Factorization in two dimensions

One issue found in RCWA is that, in the case of any device slice with a 2D patterned structure such that it is not homogeneous in both directions, then the only field components that are type 1 in both the x-direction and y-direction are field components made implicit by the numerical method. $E_x, E_y, D_x, D_y, H_x, H_y, B_x,$ and B_y are all type 2 in at least one direction. The E-fields and H-fields in the z-direction are not continuous.

Since the problem does not actually evaluate the Fourier transform in the z-direction (and the fields in the z-direction have been made implicit by the method), only the types in the x-direction and y-direction of the transverse fields are of interest.

In order to accurately deal with these components according to FFF, we need to find a form of permittivity operators that act in one direction as a convolution, and in the other direction as a deconvolution. To do this, we first need to first introduce the concept of a hybrid real/Fourier space.

2.7.2. Hybrid real/Fourier space

In real space, the permittivity, permeability, and field components are given for the Cartesian coordinates. (the z-coordinate is omitted, as it is unimportant for this section.)

The size of the Fourier terms can be determined from finding the projection of the permittivity/permeability in question onto the base functions. Since this is just a plane wave, the projection simplifies to the following surface integral:

$$a^{m,n} = \frac{1}{L_x L_y} \int \epsilon(x,y) e^{-i2\pi(\frac{m}{L_x}x + \frac{n}{L_y}y)} dxdy \tag{2.58}$$

Here we introduce the hybrid real/Fourier space; this is as simple as applying only a one-dimensional Fourier transform in the y-direction.

$$a^n(x) = \frac{1}{L_y} \int \epsilon(x,y) e^{-i2\pi\frac{n}{L_y}y} dy \tag{2.59}$$

For the case of $M = 1$, we can describe the epsilon convolution matrices as follows:

$$[\epsilon]_y(x) = \begin{bmatrix} a^0(x) & a^{-1}(x) & a^{-2}(x) \\ a^1(x) & a^0(x) & a^{-1}(x) \\ a^2(x) & a^1(x) & a^0(x) \end{bmatrix} \tag{2.60}$$

Each entry of this matrix is itself still a function of x. Because of this, it is possible to evaluate the convolution matrix of each matrix entry by applying the Fourier transform in the x-direction:

$$a^{m,n} = \frac{1}{L_x} \int a^n(x) e^{-i2\pi \frac{m}{L_x} x} dx \tag{2.61}$$

Substituting each entry with its respective convolution matrix in the y-direction yields the following block matrix:

$$[[\epsilon]_y]_x = \left[\begin{array}{ccc|ccc|ccc} a^{0,0} & a^{-1,0} & a^{-2,0} & a^{0,-1} & a^{-1,-1} & a^{-2,-1} & a^{0,-2} & a^{-1,-2} & a^{-2,-2} \\ a^{1,0} & a^{0,0} & a^{-1,0} & a^{1,-1} & a^{0,-1} & a^{-1,-1} & a^{1,-2} & a^{0,-2} & a^{-1,-2} \\ a^{2,0} & a^{1,0} & a^{0,0} & a^{2,-1} & a^{1,-1} & a^{0,-1} & a^{2,-2} & a^{1,-2} & a^{0,-2} \\ \hline a^{0,1} & a^{-1,1} & a^{-2,1} & a^{0,0} & a^{-1,0} & a^{-2,0} & a^{0,-1} & a^{-1,-1} & a^{-2,-1} \\ a^{1,1} & a^{0,1} & a^{-1,1} & a^{1,0} & a^{0,0} & a^{-1,0} & a^{1,-1} & a^{0,-1} & a^{-1,-1} \\ a^{2,1} & a^{1,1} & a^{0,1} & a^{2,0} & a^{1,0} & a^{0,0} & a^{2,-1} & a^{1,-1} & a^{0,-1} \\ \hline a^{0,2} & a^{-1,2} & a^{-2,2} & a^{0,1} & a^{-1,1} & a^{-2,1} & a^{0,0} & a^{-1,0} & a^{-2,0} \\ a^{1,2} & a^{0,2} & a^{-1,2} & a^{1,1} & a^{0,1} & a^{-1,1} & a^{1,0} & a^{0,0} & a^{-1,0} \\ a^{2,2} & a^{1,2} & a^{0,2} & a^{2,1} & a^{1,1} & a^{0,1} & a^{2,0} & a^{1,0} & a^{0,0} \end{array}\right] \tag{2.62}$$

When the block lines are erased, this is equivalent to the matrix in equation 2.49. As such, we can state the following:

$$[[\epsilon]_y]_x = [\![\epsilon]\!] \tag{2.63}$$

In a similar vein, if one is careful with indices, performing a one-dimensional Fourier transform in the x-direction first, and the y-direction second, also yields the same result.

$$[[\epsilon]_x]_y = [\![\epsilon]\!] \tag{2.64}$$

This begs the question; if these results are the same, why even bother introducing this alternate way of calculating the two-dimensional convolution matrix? The reason is that the transforms make it possible to evaluate the following block matrices:

$$[[\frac{1}{\epsilon}]_x^{-1}]_y \tag{2.65}$$

$$[[\frac{1}{\epsilon}]_y^{-1}]_x \tag{2.66}$$

This is because the entries of matrix $[\frac{1}{\epsilon}]_x^{-1}$ are still functions of y, so determining the convolution matrix in y for every entry is possible. Likewise, determining the convolution matrix in x for every entry of $[\frac{1}{\epsilon}]_y^{-1}$ is also possible.

By evaluating these two new matrices, we have a choice between two additional operators which conceptually do the same thing as $[\![\epsilon]\!]$, but we know that generally, in the case of finite operators,

$$[[\frac{1}{\epsilon}]_x^{-1}]_y \neq [[\frac{1}{\epsilon}]_y^{-1}]_x \neq [\![\epsilon]\!] \tag{2.67}$$

Outlined below is a table on which these operators act best:

Operator	input type in x	input type in y	field component
$[\![\epsilon]\!]$	1	1	E_z
$[[\frac{1}{\epsilon}]_x^{-1}]_y$	2	1	E_x
$[[\frac{1}{\epsilon}]_y^{-1}]_x$	1	2	E_y
$[\![\mu]\!]$	1	1	H_z
$[[\frac{1}{\mu}]_x^{-1}]_y$	2	1	H_x
$[[\frac{1}{\mu}]_y^{-1}]_x$	1	2	H_y

With this, we can make an informed choice on which operator is most appropriate for which field com-

ponent. This leads to the following substitutions in the matrix equations:

$$\begin{aligned}
\mu\tilde{H}_x &\to [[\frac{1}{\mu}]_x^{-1}]_y \mathbf{u}_x \\
\mu\tilde{H}_y &\to [[\frac{1}{\mu}]_y^{-1}]_x \mathbf{u}_y \\
\mu\tilde{H}_z &\to [\![\mu]\!] \mathbf{u}_z \\
\epsilon E_x &\to [[\frac{1}{\epsilon}]_x^{-1}]_y \mathbf{s}_x \\
\epsilon E_y &\to [[\frac{1}{\epsilon}]_y^{-1}]_x \mathbf{s}_y \\
\epsilon E_z &\to [\![\epsilon]\!] \mathbf{s}_z
\end{aligned}$$

As an addendum, these rules work based on the assumption that all material interfaces are oriented either perfectly in the x-direction, or perfectly in the y-direction. In situations where these assumptions are founded, Fast Fourier Factorization clearly improves the numeric performance of the method. In cases where this assumption is not perfectly founded, the improvement is less clear-cut if present at all.

2.8. New Maxwell equation notation

With these altered operators, we express the Maxwell equations as follows:

$$\begin{aligned}
-ik_0\mathbf{K_y}\mathbf{s_z} - \frac{d}{dz}\mathbf{s_y} &= k_0[[\frac{1}{\mu}]_x^{-1}]_y \mathbf{u_x} \\
\frac{d}{dz}\mathbf{s_x} + ik_0\mathbf{K_x}\mathbf{s_z} &= k_0[[\frac{1}{\mu}]_y^{-1}]_x \mathbf{u_y} \\
-ik_0\mathbf{K_x}\mathbf{s_y} + ik_0\mathbf{K_y}\mathbf{s_x} &= k_0[\![\mu]\!] \mathbf{u_z} \\
-ik_0\mathbf{K_y}\mathbf{u_z} - \frac{d}{dz}\mathbf{u_y} &= k_0[[\frac{1}{\epsilon}]_x^{-1}]_y \mathbf{s_x} \\
\frac{d}{dz}\mathbf{u_x} + ik_0\mathbf{K_x}\mathbf{u_z} &= k_0[[\frac{1}{\epsilon}]_y^{-1}]_x \mathbf{s_y} \\
-ik_0\mathbf{K_x}\mathbf{u_y} + ik_0\mathbf{K_y}\mathbf{u_x} &= k_0[\![\epsilon]\!] \mathbf{s_z}
\end{aligned} \tag{2.68}$$

Here, we make the z-fields implicit with the third and sixth equation:

$$\begin{aligned}
\mathbf{u_z} &= [\![\mu]\!]^{-1}(-i\mathbf{K_x}\mathbf{s_y} + i\mathbf{K_y}\mathbf{s_x}) \\
\mathbf{s_z} &= [\![\epsilon]\!]^{-1}(-i\mathbf{K_x}\mathbf{u_y} + i\mathbf{K_y}\mathbf{u_x})
\end{aligned} \tag{2.69}$$

Substituting this expression into the remaining equations and isolating the z-derivative terms gives us the following:

$$\begin{aligned}
\frac{d}{dz}\mathbf{s_x} &= k_0[[\frac{1}{\mu}]_y^{-1}]_x \mathbf{u_y} - ik_0\mathbf{K_x}[\![\epsilon]\!]^{-1}(-i\mathbf{K_x}\mathbf{u_y} + i\mathbf{K_y}\mathbf{u_x}) \\
\frac{d}{dz}\mathbf{s_y} &= -k_0[[\frac{1}{\mu}]_x^{-1}]_y \mathbf{u_x} - ik_0\mathbf{K_y}[\![\epsilon]\!]^{-1}(-i\mathbf{K_x}\mathbf{u_y} + i\mathbf{K_y}\mathbf{u_x}) \\
\frac{d}{dz}\mathbf{u_x} &= k_0[[\frac{1}{\epsilon}]_y^{-1}]_x \mathbf{s_y} - ik_0\mathbf{K_x}[\![\mu]\!]^{-1}(-i\mathbf{K_x}\mathbf{s_y} + i\mathbf{K_y}\mathbf{s_x}) \\
\frac{d}{dz}\mathbf{u_y} &= -k_0[[\frac{1}{\epsilon}]_x^{-1}]_y \mathbf{s_x} - ik_0\mathbf{K_y}[\![\mu]\!]^{-1}(-i\mathbf{K_x}\mathbf{s_y} + i\mathbf{K_y}\mathbf{s_x})
\end{aligned} \tag{2.70}$$

This finally leads to the following overall matrix equation:

$$\frac{d}{dz}\begin{bmatrix}\mathbf{s}_x \\ \mathbf{s}_y \\ \mathbf{u}_x \\ \mathbf{u}_y\end{bmatrix} = k_0 \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{K_x}[\![\epsilon]\!]^{-1}\mathbf{K_y} & [[\frac{1}{\mu}]_y^{-1}]_x - \mathbf{K_x}[\![\epsilon]\!]^{-1}\mathbf{K_x} \\ \mathbf{0} & \mathbf{0} & \mathbf{K_y}[\![\epsilon]\!]^{-1}\mathbf{K_y} - [[\frac{1}{\mu}]_x^{-1}]_y & -\mathbf{K_y}[\![\epsilon]\!]^{-1}\mathbf{K_x} \\ \mathbf{K_x}[\![\mu]\!]^{-1}\mathbf{K_y} & [[\frac{1}{\epsilon}]_y^{-1}]_x - \mathbf{K_x}[\![\mu]\!]^{-1}\mathbf{K_x} & \mathbf{0} & \mathbf{0} \\ \mathbf{K_y}[\![\mu]\!]^{-1}\mathbf{K_y} - [[\frac{1}{\epsilon}]_x^{-1}]_y & -\mathbf{K_y}[\![\mu]\!]^{-1}\mathbf{K_x} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix}\mathbf{s}_x \\ \mathbf{s}_y \\ \mathbf{u}_x \\ \mathbf{u}_y\end{bmatrix} \tag{2.71}$$

From which we get the altered P and Q matrices:

$$\mathbf{P} = k_0 \begin{bmatrix} \mathbf{K_x}[\![\epsilon]\!]^{-1}\mathbf{K_y} & [[\frac{1}{\mu}]_y^{-1}]_x - \mathbf{K_x}[\![\epsilon]\!]^{-1}\mathbf{K_x} \\ \mathbf{K_y}[\![\epsilon]\!]^{-1}\mathbf{K_y} - [[\frac{1}{\mu}]_x^{-1}]_y & -\mathbf{K_y}[\![\epsilon]\!]^{-1}\mathbf{K_x} \end{bmatrix} \tag{2.72}$$

$$\mathbf{Q} = k_0 \begin{bmatrix} \mathbf{K_x}[\![\mu]\!]^{-1}\mathbf{K_y} & [[\frac{1}{\epsilon}]_y^{-1}]_x - \mathbf{K_x}[\![\mu]\!]^{-1}\mathbf{K_x} \\ \mathbf{K_y}[\![\mu]\!]^{-1}\mathbf{K_y} - [[\frac{1}{\epsilon}]_x^{-1}]_y & -\mathbf{K_y}[\![\mu]\!]^{-1}\mathbf{K_x} \end{bmatrix} \tag{2.73}$$

and, introducing the following shorthand:

$$\mathbf{e} = \begin{bmatrix} \mathbf{s}_x \\ \mathbf{s}_y \end{bmatrix}, \mathbf{h} = \begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \end{bmatrix} \tag{2.74}$$

We get the matrix differential equation in its most compact form.

$$\frac{d}{dz}\begin{bmatrix} \mathbf{e} \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{P}(z) \\ \mathbf{Q}(z) & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{e} \\ \mathbf{h} \end{bmatrix} \tag{2.75}$$

This concludes the Plane Wave Expansion Method.

3

Transfer Matrix Method

The Transfer Matrix Method (or TMM for short) is the part of the method which works to turn the following differential equation:

$$\frac{d}{dz}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{P}(z)\\ \mathbf{Q}(z) & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix} \tag{3.1}$$

into a relation between incoming and outgoing light.

Let us first take the differential equation for a single device slice. Inside this slice, we make the assumption that **P** and **Q** are continuous and uniform in a range going from l to $l + \Delta l$. Within this range, the differential equation simplifies to:

$$\frac{d}{dz}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix} \tag{3.2}$$

We want to turn this into an equation which relates the fields just above the slice (at $z = l$) to just below the slice (at $z = l + \Delta l$).

$$\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l+\Delta l} = \begin{bmatrix}\mathbf{T}_{11} & \mathbf{T}_{12}\\ \mathbf{T}_{21} & \mathbf{T}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l} \tag{3.3}$$

One way to find this relation is to evaluate the Taylor series of the fields;

$$\begin{aligned}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l+\Delta l} &= \begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l} + \Delta l\frac{d}{dz}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l} + \frac{(\Delta l)^2}{2}\frac{d^2}{dz^2}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l} + \dots\\ &= \sum_{n=0}^{\infty}\frac{(\Delta l)^n}{n!}\frac{d^n}{dz^n}\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l}\end{aligned} \tag{3.4}$$

Using equation 3.1, this becomes:

$$\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l+\Delta l} = \sum_{n=0}^{\infty}\frac{1}{n!}\left(\Delta l\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\right)^n\begin{bmatrix}\mathbf{e}\\ \mathbf{h}\end{bmatrix}_{z=l} \tag{3.5}$$

where $\left(\Delta l\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\right)^0 = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}$. This function is known as the matrix exponential. As this sum has a radius of convergence of n, and n approaches infinity, this matrix series converges unconditionally, just like the regular exponential Taylor series. It is defined via the following relation:

$$expm(\mathbf{A}) = \sum_{n=0}^{\infty}\frac{1}{n!}\mathbf{A}^n \tag{3.6}$$

Thus, an unconditional equation relating the fields at $z = l$ and $z = l + \Delta l$ is given by the following relation:

$$\begin{bmatrix} \mathbf{e} \\ \mathbf{h} \end{bmatrix}_{z=l+\Delta l} = expm\left(\Delta l \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} \right) \begin{bmatrix} \mathbf{e} \\ \mathbf{h} \end{bmatrix}_{z=l} \tag{3.7}$$

In theory, this equation suffices for evaluating the transfer matrix method.

In practice, if $\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$ has positive eigenvalues and Δl becomes large enough, then values of the exponential matrix can become very large. In effect, the growing evanescent fields will start to dominate all other fields. If the growing factor exceeds the relative rounding error, then this poses the problem that the contributions of non-evanescent fields are simply rounded to zero, making the resulting matrix useless for further calculations. If the growing factor even exceeds the maximum number size possible, then overflow makes further calculations impossible.

3.1. Constructing a causal system

The reason why these relations are ill-posed is because it is not a causal relation. The field at one z-slice is not a function of the field at another z-slice. The fields at both z-slices are functions of the incoming sources of electromagnetic radiations. If one just happens to be very small while the other is not, describing the other as a function of the one will become very ill-posed. If one happens to be zero while the other is not, describing the other as a function of the one is equivalent to dividing by zero.

Instead, we describe the outgoing waves as functions of the incoming waves. This is physically well-posed; A situation where the outgoing waves will be very big while the incoming waves are very small will not happen, as this disobeys the principle of conservation of energy. This begs the question; just what are the incoming and outgoing waves?

3.1.1. Eigenvalues

Multiple schemes have been devised to deal with this problem. In this paper, only two will be discussed; the ”enhanced transmittance matrix” approach, and the ”scattering matrix” approach. Both of these rely on a property of $\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$; its eigenvalues and eigenvectors come in pairs. To make use of this property, however, we first need to diagonalize it.

Suppose that a certain vector is an eigenvector of this matrix, with a certain eigenvalue:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} \tag{3.8}$$

Then the following must also be true;

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{-v} \end{bmatrix} = -\lambda \begin{bmatrix} \mathbf{w} \\ \mathbf{-v} \end{bmatrix} \tag{3.9}$$

This is because the former is an eigenvector with eigenvalue λ if and only if the following equations hold:

$$\mathbf{Pv} = \lambda \mathbf{w} \tag{3.10}$$

$$\mathbf{Qw} = \lambda \mathbf{v} \tag{3.11}$$

if λ is substituted with $-\lambda$ and **v** with **-v**, both of these equations still hold, therefore $\begin{bmatrix} \mathbf{w} \\ \mathbf{-v} \end{bmatrix}$ must be an eigenvector as well, with eigenvalue $-\lambda$.

$$\begin{aligned} \mathbf{P}(\mathbf{-v}) &= -\lambda \mathbf{w} \\ -\mathbf{Pv} &= -\lambda \mathbf{w} \\ \mathbf{Pv} &= \lambda \mathbf{w} \\ \mathbf{Qw} &= -\lambda(\mathbf{-v}) \\ -\mathbf{Qw} &= \lambda(\mathbf{-v}) \\ -\mathbf{Qw} &= -\lambda \mathbf{v} \\ \mathbf{Qw} &= \lambda \mathbf{v} \end{aligned} \tag{3.12}$$

Since both $\mathbf{Qw} = \lambda\mathbf{v}$ and $\mathbf{Pv} = \lambda\mathbf{w}$ are shown to hold, $\begin{bmatrix}\mathbf{w}\\ \mathbf{-v}\end{bmatrix}$ is also an eigenvector with eigenvalue $-\lambda$. From this, we can also get the following:

$$\mathbf{PQw} = \lambda\mathbf{Pv} = \lambda^2\mathbf{w} \tag{3.13}$$

From a physical standpoint, the fact that the eigenvectors come in pairs leads us naturally to the conclusion that one of them is travelling from $z = l$ to $z = l + \Delta l$, and the other $z = l$ to $z = l + \Delta l$. The former is referred to as propagating in the positive z-direction, and the latter is referred to as propagating in the negative z-direction.

If they are evanescent, eigenvectors propagating in the positive z direction become smaller in amplitude as z-increases, and eigenvectors propagating in the negative z-direction become smaller in amplitude as z increases. To ensure this behavior, we determine that the modes propagating in the positive z-direction have eigenvalues with non-positive real parts, and modes propagating in the negative z-direction have eigenvalues with non-negative real parts. Additionally, as per sign convention, the modes propagating in the positive z-direction have eigenvalues with non-positive imaginary parts, and modes propagating in the negative z-direction have eigenvalues with non-negative imaginary parts.

There are some cases where eigenvalues with a positive real component but negative imaginary component exist, and vice versa. These modes can neither be described as travelling purely in the positive z-direction nor purely in the negative z-direction. In such a case, the convention is that the real component sign takes precedence. There are some reservations about this distinction from a physical standpoint; Such a situation implies the existence of electromagnetic waves with negative imaginary but positive real parts, which are neither purely causal nor purely anticausal. Accepting such modes as ”waves” uncritically might worsen our understanding of these electromagnetic systems. However, from a computational standpoint, it works regardless of dubious physical motivation.

but from a numerical analysis standpoint it works.

3.1.2. Diagonalization

We define the **W** as a set of eigenvectors of the matrix **PQ**, with eigenvalues of $\boldsymbol{\lambda^2}$;

$$\mathbf{PQ} = \mathbf{W}\boldsymbol{\lambda^2}\mathbf{W}^{-1} \tag{3.14}$$

where $\boldsymbol{\lambda^2}$ is a diagonal matrix containing the eigenvalues of **PQ**. Additionally, let $\boldsymbol{\lambda}$ be a diagonal matrix which satisfies

$$(\boldsymbol{\lambda})^2 = \boldsymbol{\lambda^2} \tag{3.15}$$

Finally, we also define **V** as follows:

$$\mathbf{V} = \mathbf{QW}\boldsymbol{\lambda}^{-1} \tag{3.16}$$

From this, we can diagonalize $\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}$ as the following:

$$\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}\begin{bmatrix}\lambda & \mathbf{0}\\ \mathbf{0} & -\lambda\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}^{-1} \tag{3.17}$$

Proof: If $\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}$ is invertible, this relation can be rewritten as:

$$\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix} = \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}\begin{bmatrix}\lambda & \mathbf{0}\\ \mathbf{0} & -\lambda\end{bmatrix} \tag{3.18}$$

This simplifies to the two following unique submatrix equations:

$$\mathbf{PV} = \mathbf{W}\boldsymbol{\lambda} \tag{3.19}$$

$$\mathbf{QW} = \mathbf{V}\boldsymbol{\lambda} \tag{3.20}$$

The latter simplifies to the expression of **V** as defined before;

$$\mathbf{V} = \mathbf{Q}\mathbf{W}\boldsymbol{\lambda}^{-1} \tag{3.21}$$

using this, the former equation becomes the following:

$$\mathbf{P}\mathbf{Q}\mathbf{W}\boldsymbol{\lambda}^{-1} = \mathbf{W}\boldsymbol{\lambda} \tag{3.22}$$

$$\mathbf{P}\mathbf{Q} = \mathbf{W}\boldsymbol{\lambda}^{2}\mathbf{W}^{-1} \tag{3.23}$$

Therefore, the relation holds.

This results in a valid diagonalization only if the matrix $\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}$ is invertible, otherwise this is a proof by division-by-zero. It is for this reason that $\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & \mathbf{V}\end{bmatrix}$ is an invalid diagonalization.

This result has some interesting implications;

- The matrix **PQ** can be diagonalized as a substitute for $\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}$, saving on computation time. In a worst-case scenario, the computational cost of diagonalization is $\mathcal{O}(n^3)$ where n is the number of rows, so this step cuts down the computational cost of diagonalization by a factor of 8.
- as $\begin{bmatrix}\boldsymbol{\lambda} & \mathbf{0}\\ \mathbf{0} & -\boldsymbol{\lambda}\end{bmatrix}$ is a diagonal matrix, the matrix exponent of this matrix is the same as the term-by-term exponent;
- $\boldsymbol{\lambda}$ is not uniquely defined. This is because flipping the sign for every diagonal entry also satisfies equation 3.15. As a result, if $\boldsymbol{\lambda}^{\mathbf{2}}$ has n rows and columns, there are a total of 2^n valid solutions for $\boldsymbol{\lambda}$. However, we can pick the solution most convenient for our calculations, which in this case means picking the solution which contains no entries with positive real parts. If so, $\boldsymbol{\lambda}$ contains all the eigenvalues of the decaying evanescent modes, and $-\boldsymbol{\lambda}$ contains all the values of the growing evanescent modes. As a result, we know that $\lim_{\Delta l\to\infty} expm(\Delta l\boldsymbol{\lambda})$ is finite.
- due to the following property of the matrix exponent

  $$expm(-A) = expm(A)^{-1} \tag{3.24}$$

  We know that the matrix exponent of $-\Delta l\boldsymbol{\lambda}$ can be expressed as:

  $$expm(-\Delta l\boldsymbol{\lambda}) = expm(\Delta l\boldsymbol{\lambda})^{-1} = \mathbf{X}^{-1} \tag{3.25}$$

  and thus

  $$expm(-\Delta l\boldsymbol{\lambda})^{-1} = expm(\Delta l\boldsymbol{\lambda}) = \mathbf{X} \tag{3.26}$$

With this, we also know that:

$$\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}^n = \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda} & \mathbf{0}\\ \mathbf{0} & -\boldsymbol{\lambda}\end{bmatrix}^n\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}^n & \mathbf{0}\\ \mathbf{0} & -\boldsymbol{\lambda}^n\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}^{-1} \tag{3.27}$$

Which finally leads us to the alternate formulation of 3.7:

$$\begin{aligned} expm\left(\Delta l\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\right) &= \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix} expm\left(\Delta l\begin{bmatrix}\boldsymbol{\lambda} & \mathbf{0}\\ \mathbf{0} & -\boldsymbol{\lambda}\end{bmatrix}\right)\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}^{-1}\\ &= \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}\begin{bmatrix}expm(\Delta l\boldsymbol{\lambda}) & \mathbf{0}\\ \mathbf{0} & expm(-\Delta l\boldsymbol{\lambda})\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}^{-1}\\ &= \begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}\begin{bmatrix}\mathbf{X} & \mathbf{0}\\ \mathbf{0} & \mathbf{X}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{W}\\ \mathbf{V} & -\mathbf{V}\end{bmatrix}^{-1} \end{aligned} \tag{3.28}$$

Therefore, any method which takes the inverse of $expm(-\Delta l\boldsymbol{\lambda})$ but not of $expm(\Delta l\boldsymbol{\lambda})$ circumvents the ”growing evanescent field” problem, and is therefore numerically stable. This brings us to the concept of a two-port network.

3.1.3. Two-port network

We can model this as a two-port network; Two inputs on either side, two outputs at either side.

Figure 3.1: Illustration of a tube as a two-port network.

3.1.4. S-parameters

Scattering parameters (or S-parameters for short) are a framework copied from electrical engineering. This framework is typically known as ”cable theory”, and is readily applicable to multiple-in-multiple-out (MIMO) systems. They linearly relate multiple inputs of a black box to multiple outputs. In our most simplistic case, they relate two inputs to two outputs. This is visualised according to the two-port network.

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{3.29}$$

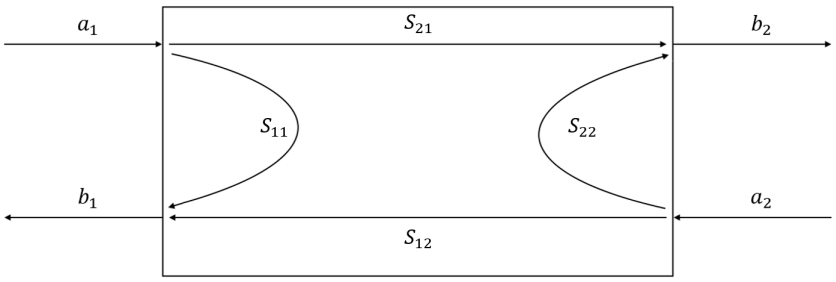


Figure 3.2: Illustration of how S-parameters relate inputs and outputs as a linear process.

It’s important to note that, in this case, the inputs are the coefficients of incident eigenmodes, and the outputs are the coefficients of the outgoing eigenmodes, and not the electromagnetic fields themselves.

The relation between the incident and outgoing waves is causal; Outgoing radiation only occurs if incident radiation is present. Thus, the phenomenon of an ill-posed relation does not occur if all outgoing light is described as a function of all incoming light. As it turns out, the incoming and outgoing modes have already been described and calculated; these are the ”eigenmodes” calculated during the eigendecomposition of $\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$.

An S-parameter matrix is a matrix describing the in-out relations of a purely causal two-port network, where the inputs and outputs describe the complex coefficients (containing both phase and magnitude) of the eigenmodes of the material present at either interface.

3.1.5. Principal pivot transform

Suppose we have a following matrix in block form:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{3.30}$$

where A_{11} is an invertible matrix. Consider the following matrix:

$$B = \begin{bmatrix} A_{11}^{-1} & -(A_{11}^{-1})A_{12} \\ A_{21}(A_{11}^{-1}) & A_{22} - A_{21}(A_{11}^{-1})A_{12} \end{bmatrix} \tag{3.31}$$

Then the relation holds

$$A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{3.32}$$

if and only if

$$B \begin{bmatrix} y_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_2 \end{bmatrix} \tag{3.33}$$

[6] In this paper, B will be referred to as "The principal pivot transform of A about A_{11}". If we were to perform a principal pivotal transform of B about B_{11} (call this C), this yields us A again:

$$C = \begin{bmatrix} (A_{11}^{-1})^{-1} & -((A_{11}^{-1})^{-1})(-(A_{11}^{-1})A_{12}) \\ A_{21}(A_{11}^{-1})(A_{11}^{-1})^{-1} & A_{22} - (A_{21}(A_{11}^{-1})A_{12}) - ((A_{11}^{-1})^{-1})(-(A_{11}^{-1})A_{12})(A_{11}^{-1})A_{12}) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{3.34}$$

Likewise, the "The principal pivot transform of A about A_{22}" can be given as

$$B' = \begin{bmatrix} A_{11} - A_{12}(A_{22}^{-1})A_{21} & A_{12}(A_{22}^{-1}) \\ -(A_{11}^{-1})A_{21} & A_{22}^{-1} \end{bmatrix} \tag{3.35}$$

where

$$B' \begin{bmatrix} x_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ x_2 \end{bmatrix} \tag{3.36}$$

This is but a very basic explanation of principal pivots. In reality, principal pivots can be performed on any selection of diagonal elements; an interesting case is that performing the principal pivot around every diagonal element is the same as evaluating the matrix inverse. This is, in turn, reflected in the fact that B' and B are inverses of each other; from B' to A, the lower half of the variables are pivoted, and from A to B, the upper half of the variables are pivoted, which together combine to be a pivot of every variable, and thus a matrix inverse.

Principal pivot transforms are useful, because matrices are ultimately a linear relation between two sets of variables, where one set is defined as the input and the other as the output. In cases where such matrices are ill-posed, a principal pivot allows finding alternate formulations which might be better-posed. This is not guaranteed, however; the alternate formulation might also be more ill-posed than the original. A well-posed alternate formulation can only be guaranteed if that formulation is causal.

Going back to our two-port network, if we know for sure that we are in a forward-backward mathematical basis, then pivoting about the backward-propagating half leaves us with a block matrix which has incoming waves as inputs and outgoing waves as outputs. In other words, such a formulation is causal, and the conceptualization as a two-port network of a pivoted matrix is a scattering matrix.

A Transfer matrix is a matrix which can be related to an S-parameter matrix, as defined above, with the following transforms:

$$\begin{aligned} \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{S}_{21} - \mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} & \mathbf{S}_{22}\mathbf{S}_{12}^{-1} \\ -\mathbf{S}_{12}^{-1}\mathbf{S}_{11} & \mathbf{S}_{12}^{-1} \end{bmatrix} \\ \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} &= \begin{bmatrix} -\mathbf{T}_{22}^{-1}\mathbf{T}_{21} & \mathbf{T}_{22}^{-1} \\ \mathbf{T}_{11} - \mathbf{T}_{12}\mathbf{T}_{22}^{-1}\mathbf{T}_{21} & \mathbf{T}_{12}\mathbf{T}_{22}^{-1} \end{bmatrix} \end{aligned} \tag{3.37}$$

These transforms are, in effect, a combination of a principal pivot and a block row shift. We can make this clear by introducing a new intermediate formulation, which we call **S'**:

$$\mathbf{S'} = \begin{bmatrix} \mathbf{S'}_{11} & \mathbf{S'}_{12} \\ \mathbf{S'}_{21} & \mathbf{S'}_{22} \end{bmatrix} \tag{3.38}$$

S' is related to **T** by performing a principal pivot about the lower-right block:

$$\begin{aligned} \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{S'}_{11} - \mathbf{S'}_{12}\mathbf{S'}_{22}^{-1}\mathbf{S'}_{21} & \mathbf{S'}_{12}\mathbf{S'}_{22}^{-1} \\ -\mathbf{S'}_{22}^{-1}\mathbf{S'}_{21} & \mathbf{S'}_{11}^{-1} \end{bmatrix} \\ \begin{bmatrix} \mathbf{S'}_{11} & \mathbf{S'}_{12} \\ \mathbf{S'}_{21} & \mathbf{S'}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{T}_{11} - \mathbf{T}_{12}\mathbf{T}_{22}^{-1}\mathbf{T}_{21} & \mathbf{T}_{12}\mathbf{T}_{22}^{-1} \\ -\mathbf{T}_{22}^{-1}\mathbf{T}_{21} & \mathbf{T}_{11}^{-1} \end{bmatrix} \end{aligned} \tag{3.39}$$

and **S'** is related to **S** with a block row shift:

$$\begin{aligned} \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{S'}_{21} & \mathbf{S'}_{22} \\ \mathbf{S'}_{11} & \mathbf{S'}_{12} \end{bmatrix} \\ \begin{bmatrix} \mathbf{S'}_{11} & \mathbf{S'}_{12} \\ \mathbf{S'}_{21} & \mathbf{S'}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{S}_{21} & \mathbf{S}_{22} \\ \mathbf{S}_{11} & \mathbf{S}_{12} \end{bmatrix} \end{aligned} \tag{3.40}$$

Thus, we can map out the overall transform between T-matrices and S-matrices with the following:

$$\mathbf{T} \overset{\text{eq.3.39}}{\longleftrightarrow} \mathbf{S'} \overset{\text{eq.3.40}}{\longleftrightarrow} \mathbf{S}$$

Any formulation which describes a relation to any part of any field cannot be causal, does not have the guaranteed stability property, and therefore can neither be a T-parameter matrix nor an S-parameter matrix.

Here are some examples of Transfer matrices:

- $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$
- $\begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}^{-1} \end{bmatrix}$
- $\left(\begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_j & \mathbf{W}_j \\ \mathbf{V}_j & -\mathbf{V}_j \end{bmatrix} \right)$
- $\left(\begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_j & \mathbf{W}_j \\ \mathbf{V}_j & -\mathbf{V}_j \end{bmatrix} \right)^{-1}$
- any matrix product of the above.

The following are not Transfer matrices, since at least one side of the relations is the electromagnetic fields and its principal pivot is therefore not an S-parameter matrix:

- $expm\left(\Delta l \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} \right)$
- $\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}$
- $\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}^{-1}$

As an example, the following ”propagation” T-matrix:

$$\mathbf{T}_{prop} = \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}^{-1} \end{bmatrix} \tag{3.41}$$

will have the following corresponding S-matrix:

$$\mathbf{S}_{prop} = \begin{bmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{X} & \mathbf{0} \end{bmatrix} \tag{3.42}$$

3.1.6. Concatenation & the Redheffer Star product

Concatenation is the process of connecting two two-port networks as follows:

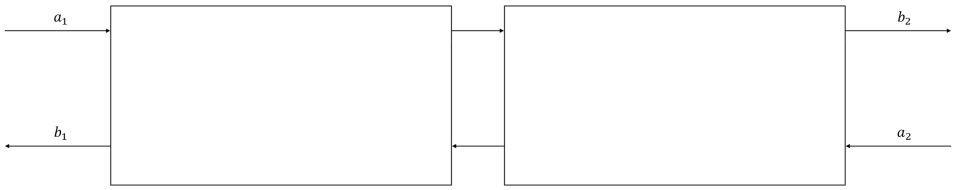


Figure 3.3: Illustration of two two-port networks concatenated.

As one can see, the resulting construct is itself a two-port network. As it turns out, the matrix defining the new two-port network are a function of the matrices of the two constituent two-port networks. In the case of T-matrices, the equation to determine the overall T-matrix is as simple as a matrix multiplication:

Suppose that a transfer matrix is given as a product between two transfer matrices:

$$\mathbf{T}_{AB} = \mathbf{T}_B \mathbf{T}_A = \begin{bmatrix} \mathbf{T}_{11}^B \mathbf{T}_{11}^A + \mathbf{T}_{12}^B \mathbf{T}_{21}^A & \mathbf{T}_{11}^B \mathbf{T}_{12}^A + \mathbf{T}_{12}^B \mathbf{T}_{22}^A \\ \mathbf{T}_{21}^B \mathbf{T}_{11}^A + \mathbf{T}_{22}^B \mathbf{T}_{21}^A & \mathbf{T}_{21}^B \mathbf{T}_{12}^A + \mathbf{T}_{22}^B \mathbf{T}_{22}^A \end{bmatrix} \tag{3.43}$$

In the case of S-parameter matrices, the concatenation is determined by what is known as the Redheffer star product:

$$\mathbf{S}_{AB} = \mathbf{S}_A \otimes \mathbf{S}_B = \begin{bmatrix} \mathbf{S}_{11}^A + \mathbf{S}_{12}^A [\mathbf{I} - \mathbf{S}_{11}^B \mathbf{S}_{22}^A]^{-1} \mathbf{S}_{11}^B \mathbf{S}_{21}^A & \mathbf{S}_{12}^A [\mathbf{I} - \mathbf{S}_{11}^B \mathbf{S}_{22}^A]^{-1} \mathbf{S}_{12}^B \\ \mathbf{S}_{21}^B [\mathbf{I} - \mathbf{S}_{22}^A \mathbf{S}_{11}^B]^{-1} \mathbf{S}_{21}^A & \mathbf{S}_{22}^B + \mathbf{S}_{21}^B [\mathbf{I} - \mathbf{S}_{22}^A \mathbf{S}_{11}^B]^{-1} \mathbf{S}_{22}^A \mathbf{S}_{12}^B \end{bmatrix} \tag{3.44}$$

The S-parameter matrix which describes this can be expressed as the following: $\mathbf{S}_{AB}$ is referred to as the concatenation of $\mathbf{S}_A$ and $\mathbf{S}_B$. This is known in literature as the Redheffer star product. A proof of this identity can be found in the appendix.

It is important to note that the ordering of T-matrices reads right-to-left, while that of S-matrices reads left-to-right.

This product requires the evaluation of at least one inverse. Because of this, evaluating the Redheffer star product is slower than the evaluation of a regular matrix product.

3.2. Gap medium

Both the conventional T-matrix formulation and the S-matrix formulation are not relations between electromagnetic fields; instead, these both describe relations of propagation modes to one another. We have to consider presents the problem that different slices have different eigenmodes, so information about the eigenmodes in both materials is needed to construct an S-parameter matrix across the interface. As a result, information about the materials at both interfaces is needed for an S-parameter matrix across one slice. This makes the mathematics less modular, while modularity is a desireable property for practical reasons.

One solution to this is the gap medium. [7] A gap medium is a theoretical medium which exists between the slices. Since this gap medium has a width of zero between each slice, the presence (or absence) of this material does not alter the electromagnetic properties of the device.

For the sake of completeness, Assuming the differential matrix is invertible, the first-order differential equation of a slice can be diagonalized as follows:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}^{-1} \tag{3.45}$$

Thus, the matrix relating the fields on each side of a slice with width Δl becomes:

$$expm\left(\Delta l \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix} \begin{bmatrix} expm(\Delta l \boldsymbol{\lambda}) & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l \boldsymbol{\lambda}) \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}^{-1} \tag{3.46}$$

if $\Delta l = 0$, then:

$$\pm \Delta l \boldsymbol{\lambda} = \mathbf{0} \tag{3.47}$$

$$expm(\pm \Delta l \boldsymbol{\lambda}) = \mathbf{I} \tag{3.48}$$

$$\begin{aligned} \mathbf{T}_g &= \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \end{aligned} \tag{3.49}$$

Since the Matrix relating the fields on either side of a slice with thickness of zero solves to the identity matrix, its presence has no analytical impact on the final result. Because of this, the choice of gap medium is arbitrary, so long as the behavior in the gap medium itself is causal. This puts some limitations on the possible gap media one can use. The convention is to have either homogeneous vacuum as the gap layer, or homogeneous layers with high permittivity to avoid the gap layer matrix inverse becoming ill-posed.

A typical formulation of an overall system of equations, linking the Electromagnetic fields just above the substrate and just below the superstrate, is as follows:

$$\mathbf{T} = \begin{bmatrix} \mathbf{W}_r & \mathbf{W}_r \\ \mathbf{V}_r & -\mathbf{V}_r \end{bmatrix}^{-1} \prod_i \left(\begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix} \begin{bmatrix} expm(\Delta l \boldsymbol{\lambda})_i & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l \boldsymbol{\lambda}_i) \end{bmatrix} \begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix}^{-1} \right) \begin{bmatrix} \mathbf{W}_t & \mathbf{W}_t \\ \mathbf{V}_t & -\mathbf{V}_t \end{bmatrix} \tag{3.50}$$

If we insert the gap medium between every layer (which entails a gap layer between the superstrate and first intermediate layer, a gap layer between all the intermediate layers, and the a gap layer between the last intermediate layer and substrate), that results in the following:

$$\mathbf{T} = \begin{bmatrix} \mathbf{W}_r & \mathbf{W}_r \\ \mathbf{V}_r & -\mathbf{V}_r \end{bmatrix}^{-1} \prod_i \left(\mathbf{T}_g \begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix} \begin{bmatrix} expm(\Delta l \boldsymbol{\lambda})_i & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l \boldsymbol{\lambda}_i) \end{bmatrix} \begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix}^{-1} \right) \mathbf{T}_g \begin{bmatrix} \mathbf{W}_t & \mathbf{W}_t \\ \mathbf{V}_t & -\mathbf{V}_t \end{bmatrix} \tag{3.51}$$

$$\begin{aligned} \mathbf{T} = &\begin{bmatrix} \mathbf{W}_r & \mathbf{W}_r \\ \mathbf{V}_r & -\mathbf{V}_r \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix} * \\ &\prod_i \left(\begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix} \begin{bmatrix} expm(\Delta l \boldsymbol{\lambda})_i & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l \boldsymbol{\lambda}_i) \end{bmatrix} \begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix} \right) * \\ &\begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_t & \mathbf{W}_t \\ \mathbf{V}_t & -\mathbf{V}_t \end{bmatrix} \end{aligned} \tag{3.52}$$

The matrix product between the inverse of one eigenmode base and another is usually denoted in the following form:

$$\begin{bmatrix} \mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_j & \mathbf{W}_j \\ \mathbf{V}_j & -\mathbf{V}_j \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{ij} & \mathbf{B}_{ij} \\ \mathbf{B}_{ij} & \mathbf{A}_{ij} \end{bmatrix} \tag{3.53}$$

Where

$$\mathbf{A}_{ij} = \frac{1}{2}(\mathbf{W}_i^{-1}\mathbf{W}_j + \mathbf{V}_i^{-1}\mathbf{V}_j), \mathbf{B}_{ij} = \frac{1}{2}(\mathbf{W}_i^{-1}\mathbf{W}_j - \mathbf{V}_i^{-1}\mathbf{V}_j) \tag{3.54}$$

With this notation, we get the inverse of this matrix as the following:

$$\begin{bmatrix} \mathbf{A}_{ij} & \mathbf{B}_{ij} \\ \mathbf{B}_{ij} & \mathbf{A}_{ij} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}_{ji} & \mathbf{B}_{ji} \\ \mathbf{B}_{ji} & \mathbf{A}_{ji} \end{bmatrix} \tag{3.55}$$

And with this, the overall system can be expressed as such;

$$\mathbf{T} = \begin{bmatrix} \mathbf{A}_{tg} & \mathbf{B}_{tg} \\ \mathbf{B}_{tg} & \mathbf{A}_{tg} \end{bmatrix} \prod_i \left(\begin{bmatrix} \mathbf{A}_{gi} & \mathbf{B}_{gi} \\ \mathbf{B}_{gi} & \mathbf{A}_{gi} \end{bmatrix} \begin{bmatrix} expm(\Delta l \boldsymbol{\lambda})_i & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l \boldsymbol{\lambda}_i) \end{bmatrix} \begin{bmatrix} \mathbf{A}_{ig} & \mathbf{B}_{ig} \\ \mathbf{B}_{ig} & \mathbf{A}_{ig} \end{bmatrix} \right) \begin{bmatrix} \mathbf{A}_{gr} & \mathbf{B}_{gr} \\ \mathbf{B}_{gr} & \mathbf{A}_{gr} \end{bmatrix} \tag{3.56}$$

The introduction of gap media achieves the following results simultaneously:

1. The total T-parameter matrix can be calculated for each layer modularly;
2. The total system of equations, formulated this way, is always one between eigenmodes and not between electromagnetic fields. While a principal pivot formulation exists for the electromagnetic fields [8], the performance and stability of such a formulation is relatively poorly understood. By contrast, the relation between eigenmodes is much better understood; a principal pivot transform between eigenmodes is known to be stable, if all the anticausal modes are pivoted simultaneously, and the causal modes are not.

Without the presence of gap media, only one of these results can be achieved at any one time; either one has an intermediate step which relates to the electromagnetic fields, or transfer to the eigenmodes of the adjacent layer need to be calculated, which requires information about the adjacent layers. The gap layer serves as a substitute for the adjacent layers. In this sense, information about the gap layer is still needed to compose the T-parameter matrix, but calculating $\mathbf{W}_g$ and $\mathbf{V}_g$ is fast and only needs to be done once as all gaps are identical, after which the result can be stored in global memory. From there on out, with this stored information, the T-matrix of a layer can be evaluated without knowing the $\mathbf{W}$ and $\mathbf{V}$ matrices of the adjacent non-gap layers, achieving modular computation.

The choice of gap medium is arbitrary; the medium of the gap medium can be as complex or as simple as we desire. Arguably the simplest implementation conceptually is assuming that the gap medium is vacuum. In such a case, the calculations become much easier: [9]

$$\begin{aligned} \mathbf{W}_{vac} &= \mathbf{I} \\ \mathbf{Q} &= k_0 \begin{bmatrix} \mathbf{K_x K_y} & \mathbf{I} - \mathbf{K_x^2} \\ \mathbf{K_y^2} - \mathbf{I} & -\mathbf{K_x K_y} \end{bmatrix} \\ \boldsymbol{\lambda} &= i(\mathbf{I} - \mathbf{K_x^2} - \mathbf{K_y^2})^{1/2} \\ \mathbf{V}_{vac} &= \mathbf{Q} \begin{bmatrix} \boldsymbol{\lambda}^{-1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\lambda}^{-1} \end{bmatrix} \end{aligned} \tag{3.57}$$

Since $\boldsymbol{\lambda}$ is a diagonal matrix, the diagonal terms can be evaluated on a scalar basis in parallel, and all non-diagonal terms will be zero. This is the gap matrix that is used in our formulation.

However, there are other formulations; it is possible to, for example, have a gap medium with a permittivity many times higher than anywhere else in the material. This would let all modes, including evanescent ones, be non-evanescent in the gap medium. This might offer some benefits with numerical stability.

One might even decouple the gap medium from the idea it stems from in physics and try the impact of the following gap matrix and its inverse:

$$\frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ i\mathbf{I} & -i\mathbf{I} \end{bmatrix}, \left(\frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ i\mathbf{I} & -i\mathbf{I} \end{bmatrix} \right)^{-1} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I} & -i\mathbf{I} \\ \mathbf{I} & i\mathbf{I} \end{bmatrix} \tag{3.58}$$

This presents a potential area of future research.

3.3. Final formulation

The final formulation for T-matrices is denoted as follows:

$$\mathbf{T}_{total} = \mathbf{T}_{substrate} \Big(\mathbf{T}_n \mathbf{T}_{n-1} ... \mathbf{T}_1 \Big) \mathbf{T}_{superstrate} \tag{3.59}$$

where

$$\mathbf{T}_{substrate} = \begin{bmatrix} \mathbf{A}_{tg} & \mathbf{B}_{tg} \\ \mathbf{B}_{tg} & \mathbf{A}_{tg} \end{bmatrix} \tag{3.60}$$

$$\mathbf{T}_{superstrate} = \begin{bmatrix} \mathbf{A}_{gr} & \mathbf{B}_{gr} \\ \mathbf{B}_{gr} & \mathbf{A}_{gr} \end{bmatrix} \tag{3.61}$$

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{A}_{gi} & \mathbf{B}_{gi} \\ \mathbf{B}_{gi} & \mathbf{A}_{gi} \end{bmatrix} \begin{bmatrix} \mathbf{X}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_i^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{ig} & \mathbf{B}_{ig} \\ \mathbf{B}_{ig} & \mathbf{A}_{ig} \end{bmatrix} \tag{3.62}$$

This, rewritten to S-parameter formulation, is denoted as the following:

$$\mathbf{S}_{total} = \mathbf{S}_{superstrate} \otimes (\mathbf{S}_1 \otimes \mathbf{S}_2 \otimes ... \otimes \mathbf{S}_n) \otimes \mathbf{S}_{substate} \tag{3.63}$$

Where

$$\mathbf{S}_{superstrate} = \begin{bmatrix} -\mathbf{A}_{rg}^{-1}\mathbf{B}_{rg} & 2\mathbf{A}_{rg}^{-1} \\ 0.5(\mathbf{A}_{tg} - \mathbf{A}_{rg}^{-1}\mathbf{B}_{rg}\mathbf{A}_{rg}) & \mathbf{B}_{rg}\mathbf{A}_{rg}^{-1} \end{bmatrix} \tag{3.64}$$

$$\mathbf{S}_{substate} = \begin{bmatrix} \mathbf{B}_{tg}\mathbf{A}_{tg}^{-1} & 0.5(\mathbf{A}_{tg} - \mathbf{A}_{tg}^{-1}\mathbf{B}_{tg}\mathbf{A}_{tg}) \\ 2\mathbf{A}_{tg}^{-1} & -\mathbf{A}_{tg}^{-1}\mathbf{B}_{tg} \end{bmatrix} \tag{3.65}$$

$$\begin{aligned} \mathbf{S}_i &= \begin{bmatrix} \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1} & 0.5(\mathbf{A}_{ig} - \mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}\mathbf{A}_{ig}) \\ 2\mathbf{A}_{ig}^{-1} & -\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{0} & \mathbf{X}_i \\ \mathbf{X}_i & \mathbf{0} \end{bmatrix} \otimes \begin{bmatrix} -\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig} & 2\mathbf{A}_{ig}^{-1} \\ 0.5(\mathbf{A}_{ig} - \mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}\mathbf{A}_{ig}) & \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}(\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{B}_{ig}) & (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{X}_i(\mathbf{A}_{ig} - \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}) \\ (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{X}_i(\mathbf{A}_{ig} - \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}) & (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}(\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{B}_{ig}) \end{bmatrix} \end{aligned} \tag{3.66}$$

[7]

3.4. Result interpretation

The overall S-matrix outlines a relation between the incident eigenmodes on the reflection-side and transmission-side with the outgoing eigenmodes of the reflection-side and transmission-side. Because the materials of the substrate and superstrate are considered homogeneous, the eigenmodes are plane waves.

As outlined in source [7], the process of finding the reflectance and transmittance starts with defining the source field:

$$\mathbf{e}_r^{src} = \begin{bmatrix} \boldsymbol{\delta}_{0,pq} p_x \\ \boldsymbol{\delta}_{0,pq} p_y \end{bmatrix} \tag{3.67}$$

where p_x, p_y are the electric field polarization vectors. This is turned into the eigenmode vector by projecting this onto the eigenvectors as follows:

$$\mathbf{c}_r^{src} = \mathbf{W}^{-1}\mathbf{e}_r^{src} \tag{3.68}$$

However, for homogeneous materials, the eigenmodes become regular plane waves. As a result, **W** becomes an identity matrix, so this step can be skipped. As this enters only on the transmission side, the overall incident field vector is given as:

$$\mathbf{c}^{src} = \begin{bmatrix} \boldsymbol{\delta}_{0,pq} p_x \\ \boldsymbol{\delta}_{0,pq} p_y \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{3.69}$$

Thus, the magnitude of all the outgoing fields are given by the following relation:

$$\mathbf{c}^{out} = \mathbf{S}_{total}\mathbf{c}^{src} \tag{3.70}$$

This can be split into the outgoing fields on the reflection and transmission side:

$$\mathbf{c}^{out} = \begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{t}_x \\ \mathbf{r}_y \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_{0,pq} p_x \\ \boldsymbol{\delta}_{0,pq} p_y \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}^{\mathsf{T}} \tag{3.71}$$

Considering there is no incident light on the transmission side, this equation simplifies to the following:

$$\mathbf{c}^{out} = \begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{t}_x \\ \mathbf{r}_y \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_{0,pq} p_x \\ \boldsymbol{\delta}_{0,pq} p_y \end{bmatrix} \tag{3.72}$$

So, we get

$$\begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_y \end{bmatrix} = \mathbf{S}_{11} \begin{bmatrix} \boldsymbol{\delta}_{0,pq} p_x \\ \boldsymbol{\delta}_{0,pq} p_y \end{bmatrix} \tag{3.73}$$

$$\begin{bmatrix} \mathbf{t}_x \\ \mathbf{t}_y \end{bmatrix} = \mathbf{S}_{21} \begin{bmatrix} \boldsymbol{\delta}_{0,pq} p_x \\ \boldsymbol{\delta}_{0,pq} p_y \end{bmatrix} \tag{3.74}$$

The longitudinal components are calculated as follows:

$$\mathbf{r}_z = -\mathbf{K}_{z,ref}^{-1}(\mathbf{K}_x r_x + \mathbf{K}_y r_y) \tag{3.75}$$

$$\mathbf{t}_z = -\mathbf{K}_{z,trn}^{-1}(\mathbf{K}_x t_x + \mathbf{K}_y t_y) \tag{3.76}$$

finally, the energy efficiencies are calculated as follows:

$$\mathbf{R} = \frac{Re[-\mathbf{K}_{z,ref}/\mu_{r,inc}]}{Re[k_{z,inc}/\mu_{r,inc}]}(|\mathbf{r}_x|^2 + |\mathbf{r}_y|^2 + |\mathbf{r}_z|^2) \tag{3.77}$$

$$\mathbf{T} = \frac{Re[-\mathbf{K}_{z,trn}/\mu_{r,inc}]}{Re[k_{z,inc}/\mu_{r,inc}]}(|\mathbf{t}_x|^2 + |\mathbf{t}_y|^2 + |\mathbf{t}_z|^2) \tag{3.78}$$

4

Short note on optimization problems for 3D-profilometry

RCWA predicts the optical reflection and transmission for a specific object, which depends on the geometric structure of the object, as well as the optical constants n and k of its constituent materials.As there is a large range of structures that the model can have, so is there a large range of possible reflection and transmission profiles. The goal of this project is to find the specific structure, whose reflection and transmission match measured data as closely as possible. In other words, we want to optimize the model's structure to make its predictions behave as measured indirectly through its far-field diffraction apttern. In the semiconductor industry, this process is oftentimes referred to as 3D-profilometry or as critical dimension (CD) reconstruction.

Optimization problems like this are regularly seen in the training of Artificial Neural Networks, which are used to replicate behavior of systems. Neural networks are machine learning models whose internal data structure imitates the human brain as a dense series of neurons. Since human pattern recognition is an emergent property from the structure of our brains, the goal of replicating the human brain is to also inherit this pattern recognition property.

As this technology has continued to mature, neural networks training processes have become both more sophisticated and efficient, resulting in trained neural networks that are increasingly competent with decreasing computational cost, vastly expanding the scope of application of these technologies.

However, this technology is not without its flaws. In replicating the human brain, it also has inherited some less desireable aspects of human intelligence, the most critical of which is the ability to be confidently wrong. Furthermore, since the parts of training the model are separate pieces of software from the neural network itself, the network lacks the ability of meta-cognition, in the form of being unable to recognise its own gaps in knowledge or guide its own learning. Our plan for this thesis was to adapt this optimization technology to train the model to replicate outside data, while maintaining RCWA's rigorous nature by not applying the neural networks themselves, thereby avoiding some of the largest pitfalls of this technology.

Despite its increasing sophistication, optimization problems can still be described as collection of a few elements:

- A loss function, which quantifies the performance of a model;
- A scheme to evaluate the global gradient of the loss function with respect to the internal variables of the model, typically through Automatic Differentiation;
- A gradient descent method, which iteratively updates the internal variables of the model based on the global gradient, to improve the score on the loss function.

4.1. Loss function

A loss function is a function which quantifies how closely the current predictions by the model replicate the measured data. A typical approach to a loss function is the L2 norm, evaluated as the squared difference between the two datasets, also referred to as ”least squares”. [10]

$$L = \sum_{i=1}^{N} |v_i^{:} - v_i|^2 \tag{4.1}$$

Historically, least squares has often been the loss function of choice, as it finds the maximum likelihood based on a gaussian noise process. However, other loss functions exist, just as well as other noise processes do.

A lower value of the loss function intuitively means better agreement between the predictions and the measurements, which further implies a closer fit of the model variables to the true values.

4.2. Automatic differentiation

Automatic differentiation is a set of techniques used to more easily evaluate the gradient of a composite function. [11]

A composite function is any function which can be created by concatenation of two or more simpler functions; that is, if we have two functions

$$f(x) = y, g(y) = z \tag{4.2}$$

then a function which maps x to z can be composed in the following way:

$$h(x) = g(f(x)) = g(y) = z \tag{4.3}$$

The global derivative of this function can then be found by applying chain rule to the local derivatives.

$$\frac{dh(x)}{dx} = \frac{dg(y)}{dy}\frac{df(x)}{dx} \tag{4.4}$$

The differential equation does not need to be known, only the order of operations. The fact that no differential equation is evaluated makes automatic differentiation distinct from symbolic differentiation.

In the case where the base functions have multiple inputs, the derivatives become gradients; in the case where the base functions have both multiple inputs and outputs, the derivatives become Jacobians.

The power of automatic differentiation is that, since functions composed of base functions have their derivative known, such functions can serve as building blocks for other composite functions. This way, the gradient of even highly composite functions can be calculated automatically, beyond what is realistically possible with either symbolic or numeric differentiation.

In the optimization problem, automatic differentiation is used to find the global gradient of the loss function with respect to the internal parameters of the model. These a priori unknown parameters comprise all relevant geometrical and material properties of the device structure that affect the measured data.

4.3. Gradient descent

Gradient descent is an optimization algorithm, used to find a local optimum (be it minimum or maximum) of a function. Local extrema are determined by the fact that the gradient at the extreme point is zero. [12]

$$\nabla F(\vec{r}) = \begin{bmatrix} \frac{\partial F}{\partial r_1} \\ \frac{\partial F}{\partial r_2} \\ \vdots \\ \frac{\partial F}{\partial r_n} \end{bmatrix} = \vec{0} \tag{4.5}$$

If the gradient of a function at a certain point is non-zero, the point $\vec{r} - \lambda\nabla F$ in parameter space will have a lower value than $\vec{r}$ if λ is small enough (the parameter lambda can be found by a line-search).

These new parameters characterize a model with a modified structure, which ought to yield a smaller number on the loss function and thus more closely replicate the measured data.

Applying the gradient-descent method to the loss function allows us to iteratively approach the local optimum of the loss function. The value of this local minimum and its proximity to the global minimum depend on a range of factors, such as the initial guesses of the model variables, the set of measurements, and optimization settings. The problem of local minima and their proximity to the global minimum lies beyond the scope of this thesis.

5

Unit cell modelling for 3D-profilometry

5.1. Underfitting and Overfitting

Statistical models can exhibit two types of undesired behavior depending on their degrees of freedom. The first is underfitting; if there are not enough parameters to a model to properly explain the observed behavior, then this is typically referred to as “underfitting”. As a result, models which are underfit are not able to properly predict behavior of any dataset, both the training dataset as well as the validation dataset.

In machine learning terms, the degree to which a model is “underfit” is referred to as “bias”. This is because assumptions are made during the modelling process by the model designer, and thus that designer’s biases are introduced into the model thereby introducing bias error, or bias for short.

The second is “overfitting”. In essence, an overfit model is a model which attempts to explain more detail in behavior than is there in reality. In most cases, this is due to the presence of noise. The model attempts to model the noise of specific cases to the general dataset; this added noise only worsens the predictive behavior of the model. An overfit dataset will have considerably better performance on the data it is trained on than any other similar dataset.

In machine learning terms, this is referred to as “variance”. This is the statistical variance from the mean. A too low variance implies the model does not really incorporate deviations from the mean and is probably underfitted; a too high variance implies the model incorporates the variance present from noise as well, so it is overfitted. For our use, variance will only refer to excessive variance i.e. variance which increases error in the validation dataset.

As such, what we see is that, in the case of machine learning models, making the model more complex has the effect of reducing bias and increasing variance. This naturally leads us to the conclusion that there must be a “best fit” model, which is neither underfit nor overfit, which optimizes performance.

In reality, however, such a model is actually simultaneously underfit and overfit. It still has bias. Anything other than a model derived from first principles, and first principles only, is biased. Any simplifications are a source of bias. Attempting to deal with this bias by increasing the complexity of the model is therefore an attempt to offset bias error with variance, thereby inducing extra variance error. As an example, if one attempts to model the population of bacteria in a petri dish over time with a polynomial, no matter how many terms are added, extra datapoints set in the future will have worse predictive power than modelling the population as an exponential.

RCWA has the benefit that the predictions are derived from first principles. It still makes assumptions, such as the absence of unbound charges and currents, as well as assumptions about the shape of the model’s structure. However, in the case where these assumptions hold, RCWA works with few simplifications. Thus, a model which calculates diffraction patterns based on RCWA has lower implicit

bias with the same number of variables, and therefore a lower optimal error. In essence, by building a model from first principles, we hope to have created a model which can reach accuracies that are otherwise unreachable with artificial neural networks.

Still, despite this, the possibilities of overfitting and underfitting still exist, so we must lay some groundwork to ensure neither of these cases happen. To do so, we must first outline how the convolution operators are calculated, as this defines the degrees of freedom of the model and, by extension, its underfitting and overfitting behavior.

5.2. Spatial permittivity model

The terms in the convolution operators originate from the Fourier coefficients of the permittivity, which we evaluate using the following equations:

$$a^{m,n} = \frac{1}{L_x L_y} \int \epsilon(x,y) e^{-i2\pi(\frac{m}{L_x}x + \frac{n}{L_y}y)} dxdy \tag{5.1}$$

$$a^{n}(x) = \frac{1}{L_y} \int \epsilon(x,y) e^{-i2\pi\frac{n}{L_y}y} dy \tag{5.2}$$

$$a^{m}(y) = \frac{1}{L_x} \int \epsilon(x,y) e^{-i2\pi\frac{m}{L_x}x} dx \tag{5.3}$$

The need to apply Fast Fourier Factorization, combined with our choice of Cartesian coordinates for our curvilinear coordinate system, requires that all gradients are perfectly aligned either with the x-direction, or y-direction.

This naturally leads to modelling the permittivity distribution as a rectilinear grid: the unit cell is divided into a number of rectangles (henceforth referred to as ”pixels”) by lines running parallel to the x-axis or y-axis, where no restriction is put on where exactly these lines are located.

Each of the pixels has a permittivity (and likewise a permeability) value, which is constant throughout the whole pixel. The parameters which define the shape are the following:

- The y-position of the lines running parallel to the x-axis, two of which constitute the boundaries of the unit cell, and the remainder separate the pixels;
- The x-position of the lines running parallel to the y-axis, two of which constitute the boundaries of the unit cell, and the remainder separate the pixels;
- the permittivity and permeability of the pixels.

All models considered for the modelling of permittivity and permeability distribution in this thesis are variations of the rectilinear grid, with varying levels of simplification. As such, we start with the Fourier transform of the most general rectilinear grid.

5.2.1. Rect function

The mathematical description of a pixel is done with the use of a rectangular function. The one-dimensional rectangular function is defined as the following:

$$rect(x) = \begin{cases} 1 & \text{for } |x| \leq 1/2 \\ 0 & \text{for } |x| > 1/2 \end{cases} \tag{5.4}$$

Using this, we can define a two-dimensional pixel using two rectangular functions as the following:

$$\epsilon_{pixel}[p,q](x,y) = rect(\frac{x - x_c[p]}{x_w[p]}) rect(\frac{y - y_c[q]}{y_w[q]}) \epsilon[p,q] \tag{5.5}$$

The index p denotes which row the specific pixel is in, and the index q denotes which column the specific pixel is in. The width and center values are dependent on but one of these indices; all pixels in the

same row share the same p index, and all pixels in the same column share the same q index. This function evaluates to the following:

$$\epsilon_{pixel} = \begin{cases} \epsilon[p,q] & \text{for } |\frac{x-x_c}{x_w}| \leq 1/2, |\frac{y-y_c}{y_w}| \leq 1/2 \\ 0 & \text{else} \end{cases} \tag{5.6}$$

The total epsilon distribution is a superposition of all these pixel epsilon functions:

$$\epsilon(x,y) = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} rect(\frac{x - x_c[p]}{x_w[p]}) rect(\frac{y - y_c[q]}{y_w[q]}) \epsilon[p,q] \tag{5.7}$$

In theory, this equation is a two-dimensional summation. In practice, the rectangular functions have no overlap with one another, nor do they leave empty space, so we know that within the unit cell only one term at a time contributes to the summation. As a result, this summation in effect becomes a lookup table.

This allows us to model the continuous permittivity with discrete values, which is essential to implementing Fast Fourier Factorization. Additionally, in the case of a cartesian grid, it allows for the use of the discrete Fourier transform to evaluate this. More on that later.

5.2.2. Fourier transform of rect funtion

The Fourier transform of this pixel in the x-direction is evaluated as:

$$\begin{aligned} \frac{1}{L_x} \int \epsilon_{pixel}(x,y) e^{-i2\pi \frac{m}{L_x} x} dx &= \frac{1}{L_x} \epsilon[p,q] rect(\frac{y-y_c}{y_w}) \int rect(\frac{x-x_c}{x_w}) e^{-i2\pi \frac{m}{L_x} x} dx \\ &= \frac{\epsilon[p,q]}{L_x} rect(\frac{y-y_c}{y_w}) \int_{x_c - x_w/2}^{x_c + x_w/2} e^{-i2\pi \frac{m}{L_x} x} dx \\ &= \frac{\epsilon[p,q]}{L_x} rect(\frac{y-y_c}{y_w}) e^{-i2\pi \frac{m}{L_x} x_c} \int_{-x_w/2}^{x_w/2} e^{-i2\pi \frac{m}{L_x} x} dx \\ &= \frac{\epsilon[p,q]}{L_x} rect(\frac{y-y_c}{y_w}) e^{-i2\pi \frac{m}{L_x} x_c} \left(\frac{e^{-i\pi \frac{m}{L_x} x_w} - e^{i\pi \frac{m}{L_x} x_w}}{-i2\pi \frac{m}{L_x}} \right) \\ &= \frac{\epsilon[p,q]}{L_x} rect(\frac{y-y_c}{y_w}) e^{-i2\pi \frac{m}{L_x} x_c} x_w \left(\frac{e^{-i\pi \frac{m}{L_x} x_w} - e^{i\pi \frac{m}{L_x} x_w}}{2(-i\pi \frac{m}{L_x} x_w)} \right) \end{aligned} \tag{5.8}$$

Here, we define the sinc function (as used in this thesis) as the following real-valued function:

$$sinc(x) = \frac{e^{ix} - e^{-ix}}{2ix} = \frac{sin(x)}{x} \tag{5.9}$$

With this, we can simplify the Fourier transform to the following:

$$\frac{1}{L_x} \int \epsilon_{pixel}(x,y) e^{-i2\pi \frac{m}{L_x} x} dx = \frac{\epsilon[p,q]}{L_x} rect(\frac{y-y_c}{y_w}) e^{-i2\pi \frac{m}{L_x} x_c} x_w sinc(\frac{m}{L_x} x_w) \tag{5.10}$$

And, with this, we get the formula to find the one-dimensional Fourier terms by summing over the contribution of every single pixel:

$$\begin{aligned} a^m(y) &= \frac{1}{L_x} \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} \epsilon[p,q] rect(\frac{y - y_c[q]}{y_w[q]}) e^{-i2\pi \frac{m}{L_x} x_c[p]} x_w[p] sinc(\frac{m}{L_x} x_w[p]) \\ &= \frac{1}{L_x} \sum_{q=0}^{Q-1} rect(\frac{y - y_c[q]}{y_w[q]}) \left(\sum_{p=0}^{P-1} \epsilon[p,q] e^{-i2\pi \frac{m}{L_x} x_c[p]} x_w[p] sinc(\frac{m}{L_x} x_w[p]) \right) \end{aligned} \tag{5.11}$$

In a similar manner, the Fourier terms in the x-direction are given by:

$$\begin{aligned} a^n(x) &= \frac{1}{L_y}\sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\epsilon[p,q]rect(\frac{x-x_c[p]}{x_w[p]})e^{-i2\pi\frac{n}{L_y}y_c[q]}y_w[q]sinc(\frac{n}{L_y}y_w[q]) \\ &= \frac{1}{L_y}\sum_{p=0}^{P-1}rect(\frac{x-x_c[p]}{x_w[p]})\left(\sum_{q=0}^{Q-1}\epsilon[p,q]e^{-i2\pi\frac{n}{L_y}y_c[q]}y_w[q]sinc(\frac{n}{L_y}y_w[q])\right) \end{aligned} \tag{5.12}$$

We introduce an additional shorthand as follows (note the difference here between square and curved brackets):

$$a^m[q] = \frac{1}{L_x}\sum_{p=0}^{P-1}\epsilon[p,q]e^{-i2\pi\frac{m}{L_x}x_c[p]}x_w[p]sinc(\frac{m}{L_x}x_w[p]) \tag{5.13}$$

$$a^n[p] = \frac{1}{L_y}\sum_{q=0}^{Q-1}\epsilon[p,q]e^{-i2\pi\frac{n}{L_y}y_c[q]}y_w[q]sinc(\frac{n}{L_y}y_w[q]) \tag{5.14}$$

Which lets us express the one-dimensional Fourier terms as follows:

$$a^m(y) = \sum_{q=0}^{Q-1}rect(\frac{y-y_c[q]}{y_w[q]})a^m[q] \tag{5.15}$$

$$a^n(x) = \sum_{p=0}^{P-1}rect(\frac{x-x_c[p]}{x_w[p]})a^n[p] \tag{5.16}$$

Thus, we have found a model of the Fourier terms which is theoretically continuous, but in practice is discrete for every region of a pixel in Fourier space. Just like the permittivity is in theory 2D-continuous but in practice a grid of 2D pixels, the Fourier terms themselves are in theory 1D-continuous but in practice a discretized line of 1D pixels.

Because of this, the terms of the hybrid real-Fourier space can be evaluated in one dimension just like the permittivity 2D pixels can. We can get the two-dimensional Fourier Transform by applying the Fourier transform on the untransformed axis:

$$\begin{aligned} a^{m,n} &= \sum_{q=0}^{Q-1}\frac{1}{L_y}a^m[q]e^{-i2\pi\frac{n}{L_y}y_c[q]}y_w[q]sinc(\frac{n}{L_y}y_w[q]) \\ &= \sum_{p=0}^{P-1}\frac{1}{L_x}a^n[p]e^{-i2\pi\frac{m}{L_x}x_c[p]}x_w[p]sinc(\frac{m}{L_x}x_w[p]) \end{aligned} \tag{5.17}$$

As one can see, these terms are equivalent, which is the origin of the previously-mentioned identity for the convolution operator:

$$[[\epsilon]_y]_x = [[\epsilon]_x]_y = [\![\epsilon]\!] \tag{5.18}$$

This begs the question; how are the convolution operators determined from these discrete hybrid Fourier-real terms?

5.2.3. Convolution matrix Fourier transform & two-dimensional convolution matrix

We can formulate the one-dimensional convolution matrix from these one-dimensional Fourier transform terms as follows:

$$[\epsilon]_x[q] = \begin{bmatrix} a^0[q] & a^1[q] & \dots & a^M[q] \\ a^{-1}[q] & a^0[q] & \ddots & \vdots \\ \vdots & \ddots & \ddots & a^1[q] \\ a^{-M}[q] & \dots & a^{-1}[q] & a^0[q] \end{bmatrix} = \begin{bmatrix} a^0 & a^1 & \dots & a^M \\ a^{-1} & a^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a^1 \\ a^{-M} & \dots & a^{-1} & a^0 \end{bmatrix}[q] \tag{5.19}$$

$[\epsilon]_x$ becomes a three-dimensional data structure; two indices for selecting the Toeplitz matrix element, each with index size M, and another one for every unique y-value, with index size Q. $[\epsilon]_x[q]$ is a Toeplitz matrix.

The trick which makes Fast Fourier Factorization work is that, instead of evaluating the Fourier terms of $a^m[q]$ along the y-axis, we evaluate the Fourier terms of the matrix elements of $[\epsilon]_x[q]$ element by element along the y-axis. Due to shared terms, this simplifies to the following:

$$[\epsilon]_x^n = \sum_{q=0}^{Q-1} \frac{1}{L_y} e^{-i2\pi \frac{n}{L_y} y_c[q]} y_w[q] sinc(\frac{n}{L_y} y_w[q]) \begin{bmatrix} a^0 & a^1 & \dots & a^M \\ a^{-1} & a^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a^1 \\ a^{-M} & \dots & a^{-1} & a^0 \end{bmatrix} [q] \tag{5.20}$$

Using this, we can build the two-dimensional convolution matrix in block form:

$$[[\epsilon]_x]_y = \begin{bmatrix} [\epsilon]_x^0 & [\epsilon]_x^1 & \dots & [\epsilon]_x^N \\ [\epsilon]_x^{-1} & [\epsilon]_x^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & [\epsilon]_x^1 \\ [\epsilon]_x^{-N} & \dots & [\epsilon]_x^{-1} & [\epsilon]_x^0 \end{bmatrix} \tag{5.21}$$

This is, in reality, a four-dimensional data structure; two indices are for selecting the block in the block matrix, each with index size N, and two are for selecting the element in the specific block, each with index size M. The lines are "erased" by unrolling axis 3 into axis 1, and unrolling axis 4 into axis 2.

While this is much more computationally expensive, as this requires M^2 Fourier transforms to be evaluated instead of $2M + 1$, this method no longer makes the assumption that the convolution matrix at the intermediate step is a Toeplitz matrix. It is the removal of this assumption which allows us to apply Fast Fourier Factorization.

5.2.4. Fast Fourier Factorization revisited

Here, we introduce the Fourier terms of a related function as follows:

$$(a^m)'[q] = \sum_{p=0}^{P-1} \frac{1}{L_x} \frac{1}{\epsilon}[p,q] e^{-i2\pi \frac{m}{L_x} x_c[p]} x_w[p] sinc(\frac{m}{L_x} x_w[p]) \tag{5.22}$$

Using this, we build the following convolution matrix:

$$[\frac{1}{\epsilon}]_x[q] = \begin{bmatrix} (a^0)'[q] & (a^1)'[q] & \dots & (a^M)'[q] \\ (a^{-1})'[q] & (a^0)'[q] & \ddots & \vdots \\ \vdots & \ddots & \ddots & (a^1)'[q] \\ (a^{-M})'[q] & \dots & (a^{-1})'[q] & (a^0)'[q] \end{bmatrix} = \begin{bmatrix} (a^0)' & (a^1)' & \dots & (a^M)' \\ (a^{-1})' & (a^0)' & \ddots & \vdots \\ \vdots & \ddots & \ddots & (a^1)' \\ (a^{-M})' & \dots & (a^{-1})' & (a^0)' \end{bmatrix} [q] \tag{5.23}$$

Next, we find the deconvolution with $\frac{1}{\epsilon}$ along x by evaluating the inverse of this matrix:

$$[\frac{1}{\epsilon}]_x^{-1}[q] = \begin{bmatrix} (a^0)' & (a^1)' & \dots & (a^M)' \\ (a^{-1})' & (a^0)' & \ddots & \vdots \\ \vdots & \ddots & \ddots & (a^1)' \\ (a^{-M})' & \dots & (a^{-1})' & (a^0)' \end{bmatrix}^{-1} [q] \tag{5.24}$$

From this, we can determine the matrix term-by-term Fourier transform:

$$([\frac{1}{\epsilon}]_x^{-1})^n = \sum_{q=0}^{Q-1} \frac{1}{L_y} e^{-i2\pi \frac{n}{L_y} y_c[q]} y_w[q] sinc(\frac{n}{L_y} y_w[q]) \begin{bmatrix} (a^0)' & (a^1)' & \dots & (a^M)' \\ (a^{-1})' & (a^0)' & \ddots & \vdots \\ \vdots & \ddots & \ddots & (a^1)' \\ (a^{-M})' & \dots & (a^{-1})' & (a^0)' \end{bmatrix}^{-1} [q] \tag{5.25}$$

And, finally, we can build the nested FFF convolution matrix as follows:

$$[[\frac{1}{\epsilon}]_x^{-1}]_y = \begin{bmatrix} ([\frac{1}{\epsilon}]_x^{-1})^0 & ([\frac{1}{\epsilon}]_x^{-1})^1 & \cdots & ([\frac{1}{\epsilon}]_x^{-1})^N \\ ([\frac{1}{\epsilon}]_x^{-1})^{-1} & ([\frac{1}{\epsilon}]_x^{-1})^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & ([\frac{1}{\epsilon}]_x^{-1})^1 \\ ([\frac{1}{\epsilon}]_x^{-1})^{-N} & \cdots & ([\frac{1}{\epsilon}]_x^{-1})^{-1} & ([\frac{1}{\epsilon}]_x^{-1})^0 \end{bmatrix} \tag{5.26}$$

As before, this is a four-dimensional data structure; two indices are for selecting the block in the block matrix, each with index size N, and two are for selecting the element in the specific block, each with index size M. The lines are ”erased” by unrolling axis 3 into axis 1, and unrolling axis 4 into axis 2.

This is a convolution operator which acts as a deconvolution in x and a convolution in y. In order to get a convolution operator which acts as a convolution in x and a deconvolution in y, we do the following:

- We swap the p- and q indices of $\epsilon[p, q]$, so that it calculates a convolution in x and a deconvolution in y;
- We repeat the above calculations, until we have the four-dimensional data structure. This structure has axes 1 and 2 associated with the x-direction, with size M, and axes 3 and 4 associated with the y-direction, with size N;
- we swap axis 1 with axis 3, and axis 2 with axis 4, so that the index orders agree with the other four-dimensional block convolution matrices;
- The lines are ”erased” by unrolling axis 3 into axis 1, and unrolling axis 4 into axis 2.

5.3. Limiting parameter space

With this, let us now outline all the free parameters available to the system;

- There are a total of P x-lines,
- There are a total of Q y-lines,
- There are a total of $P * Q$ pixels, each of which has two values, one for the relative permittivity, and one for the relative permeability.

These variables define the cross-section of the intermediate layer. Together with the layer depth Δl, they define the entire intermediate layer.

In a case of a 100-by-100 grid, we know that $P = 100$, and $Q = 100$. The relative permittivity and permeability grid alone provide a total of 20,000 free parameters, not even considering the other free parameters. Unless we have access to a total of 20,000 unique diffraction intensity values, this is too large a parameter space, and will quickly result in an overfit model. As such, we need ways to reduce the overall size of parameter space, so as to avoid overfitting.

At the same time, we need to also avoid swinging to the other extreme, and produce a biased model with improper choices in restricting parameter space. Because of this, the choice of how to limit the parameter space while still maintaining the desired resolution is ultimately an engineering problem, and is thus dependent on the nature of application.

Rather than outlining a best practice implementation, we outline a series of methods to reduce parameter space. With these, we outline a series of parameter space configurations, explore some of the strengths and drawbacks of such configurations, as well as theorising about potential use cases for such configurations.

5.3.1. Pixel contrast

Suppose that we know there are two available media in one layer. One is vacuum, where $\epsilon_r = 1$, $\mu_r = 1$. The second is the resist, for which $\epsilon_r = 2.5$, $\mu_r = 1$. If we assume that no other material is present, and the grid is small enough, we make the assumption that r is binary-valued; it either is 0 or 1. We can model the permittivity and permeability inside each pixel as:

$$\begin{aligned} \epsilon[p, q] &= (\epsilon_{res}) * r[p, q] + (\epsilon_{vac}) * (1 - r[p, q]) \\ \mu[p, q] &= (\mu_{res}) * r[p, q] + (\mu_{vac}) * (1 - r[p, q]) \end{aligned} \tag{5.27}$$

This has the following benefits:

- Instead of every pixel having mutually-independent permittivity and permeability values, the permittivity and permeability are both linked to the same value
- In this case (and many other cases), the permeability of the material is one, which actually makes the permeability not a variable which needs to be backpropagated at all. This saves not just on computing needed to be done for backpropagation, but in the regular calculation as well.
- Since $r[p, q]$ is bounded to never be lower than 0 or higher than 1, this adds an additional boundary on the remaining half of the space.

Because this limitation is physics-informed, it seems to have no drawbacks in the validity of the resulting model, so this reduction of parameter space is treated as being on by default. However, other restrictions on r can exist, which can be evaluated using effective-medium theory. This presents a possible area of future research.

5.3.2. Restrictions on the grid lines

What can also be restricted is the lines which form the grid.

The first restriction to be mentioned is that the grid positions must be in ascending order. This restriction might seem so banal at first that it ought not be mentioned, but it is nonetheless a restriction. This could result in potentially negative widths, which results in contrasts that might drop below zero. There is, thus, a good case to be made that this limit is justified, but it is a restriction nonetheless.

The second is that the lines must be contained in the unit cell. Again, this might seem banal, but this restriction forces features to not pass from one unit cell into the next neighbor. This puts real limitations on the model resulting from an arbitrary choice of unit cell, so there is a case to be made that this restriction arbitrarily limits parameter space. One way to fix this is to limit y-lines to a range of $[a, a + L_x]$ and x-lines to a range of $[b, b + L_y]$ where a and b are arbitrary, but this in practice is just redefining the unit cell boundaries. On the other hand, removing this restriction might result in features stretching over more than even two unit cells, thereby overlapping with itself, and moving the effective contrast beyond its allowed range.

The third is to force equidistance; the lines must be equally spaced. In literature, this is referred to as a cartesian grid. Because of this restriction, we can determine the following:

$$x_w[p] = \frac{L_x}{P}, y_w[q] = \frac{L_y}{Q} \tag{5.28}$$

$$x_c[p] = \frac{L_x p}{P}, y_c[q] = \frac{L_x q}{Q} \tag{5.29}$$

one might give two more degrees of freedom by again introducing a and b;

$$x_c[p] = \frac{L_x p}{P} + a, y_c[q] = \frac{L_x q}{Q} + b \tag{5.30}$$

where

$$0 \leq a \leq \frac{1}{L_x}, 0 \leq b \leq \frac{1}{L_y} \tag{5.31}$$

We have found our model to be translation-invariant, so this ought to have no impact on the end result of far-field RCWA; it might, however, be of interest in the case of near-field measurements, or in the case of two intermediate layers in which one wants to detect relative alignment. This restriction also forces the previous two restrictions on the grid lines; the lines are all limited to a (shifted) unit cell, and are all forced into ascending order.

5.3.3. Restriction on the pixel contrast

Finally, the pixel contrast can be limited as well.

One method of doing this is to force the contrast to be a certain value for certain pixels. In an area where one knows is no resist, forcing $r = 0$ is a valid limitation which limits contrast. Likewise, in an area where we know is only resist, forcing $r = 1$ is also valid.

A second, more elegant, implementation is the presence of an additional ”shape” object. For example, if one has a circle in the unit cell with a radius ρ, and we want the contrast inside the circle to be 1 and outside the circle to be 0, we can force this with the following definition of $r[p, q]$:

$$r[p,q] = \begin{cases} 1 & \text{for } \sqrt{(x_c[p] - x_{c,circ})^2 + (y_c[q] - y_{c,circ})^2} < \rho, \\ 0 & \text{else} \end{cases} \tag{5.32}$$

In this case, the parameter space of r for every pixel is instead limited to three variables; $x_{c,circ}$, $y_{c,circ}$, and ρ. However, the shapes introduced need not be as limited as this example; they can be any shape which we can describe mathematically, such as ellipses or polygons. There can also be more than one shape; the only extra added consideration with extra shapes is how one deals with intersecting shapes.

5.4. Chosen configurations

With these parameter space limitations discussed, we now highlight the configuration spaces used for our testing, as well as other promising configurations.

5.4.1. Rectilinear grid, ascending order, forced-pixel contrast

This configuration was primarily chosen because this was effectively the scheme used by the ”external” RCWA-model which was run to generate our reference data. Thus, in order to validate our implementation of RCWA, this scheme was required.

This scheme was found to be very adept at edge alignment; since the edge is saved as a floating point value with double precision, we can have a relative precision that is 10^{-16}. This puts the possible precision far beyond what is physically determinable. Because of this, this scheme was determined to be valid for our use case.

Below, we have proposed some other schemes which might have advantages in different situations; however, only the first mentioned scheme has been implemented, and that scheme was not further tested for validation, due to its poor edge alignment ability relative to the first scheme.

5.4.2. Fine Cartesian grid, shape object pixel contrast

As mentioned before, the Cartesian grid (this time with no alignment parameters) gives us the following definition of widths and centers:

$$x_w[p] = \frac{L_x}{P}, y_w[p] = \frac{L_y}{Q} \tag{5.33}$$

$$x_c[p] = \frac{L_x p}{P}, y_c[q] = \frac{L_x q}{Q} \tag{5.34}$$

With this, the expressions evaluating $a^m[q]$ and $a^n[p]$ become a lot simpler:

$$a^m[q] = \sum_{p=0}^{P-1} \frac{1}{L_x} \frac{L_x}{P} \epsilon[p,q] e^{-i2\pi \frac{m}{L_x} \frac{L_x p}{P}} sinc(\frac{m}{L_x} \frac{L_x}{P}) \tag{5.35}$$

$$= sinc(\frac{m}{P}) \frac{1}{P} \sum_{p=0}^{P-1} (\epsilon[p,q] e^{-i2\pi \frac{mp}{P}}) \tag{5.36}$$

Likewise, we get:

$$a^n[p] = sinc(\frac{n}{Q}) \frac{1}{Q} \sum_{q=0}^{Q-1} (\epsilon[p,q] e^{-i2\pi \frac{nq}{Q}}) \tag{5.37}$$

The bracketed term is equivalent to the standard formulation of the discrete Fourier Transform, which can easily be calculated computationally. This leaves us with the following expression:

$$a^m[q] = sinc(\frac{m}{P}) \frac{1}{P} \mathcal{F}_p\{\epsilon[p,q]\}[m,q] \tag{5.38}$$

Likewise,

$$a^n[p] = sinc(\frac{n}{Q})\frac{1}{Q}\mathcal{F}_q\{\epsilon[p,q]\}[n,p] \tag{5.39}$$

However, this is not just a regular Cartesian grid, but a ”fine” Cartesian grid; in a fine cartesian grid we assume that $M << P$ and $N << Q$. In such a case, we also know that $|m| << P, \forall m \in [-M, M]$ and $|n| << Q, \forall n \in [-N, N]$. Since

$$\lim_{x\to 0} sinc(x) = 1 \tag{5.40}$$

we can make the following simplifications:

$$\lim_{M/P\to 0} a^m[q] = \frac{1}{P}\mathcal{F}_p\{\epsilon[p,q]\}[m,q] \tag{5.41}$$

$$\lim_{N/Q\to 0} a^n[p] = \frac{1}{Q}\mathcal{F}_q\{\epsilon[p,q]\}[n,p] \tag{5.42}$$

Thus, for a fine Cartesian grid, the calculations become simpler. However, this is not the primary reason of its use.

By determining the contrast of the material for each pixel using shapes, increasing the pixel density has no effect on the size of the total parameter space. Additionally, since the overall **P** and **Q** matrices depend on the size of M, N and not P, Q, the computational cost of the Transfer Matrix Method overall is also independent of the total pixel density.

The only real impact increasing pixel density has on the overall algorithm is that it makes evaluating the convolution matrices more expensive, as it increases primarily memory costs of storing the intermediate step as well as the computational cost of the inverses required. However, since the eigenvalues of the convolution matrices are well understood (The eigenvalues of the permittivity convolution operator, for example, are bounded to be between the permittivity of vacuum and the resist), the evaluation of these inverses can readily be evaluated on parallel computation multiplication-friendly hardware.

Increasing the pixel numbers also has great benefits; it allows any arbitrary shape to be modelled to an arbitrary degree of accuracy by increasing the pixel density to a certain amount. This, in turn, has the added effect that formulating the convolution operators is extremely memory intensive; for this reason, the method was not developed further.

5.4.3. Rectilinear grid, ascending order, forced-pixel contrast, two lines loop back

Why would one implement this? Well, by arranging the forced-pixel contrast as follows: It effectively allows one to force a discontinuity in the grid, effectively creating two separate grids on two separate parts of the unit cell. Not only this, but it allows these grids to move with respect to one another, both laterally as well as moving apart or overlapping. This makes it a useful tool for perhaps measuring errors generated by stitching; if two parts of the unit cell were printed at different steps of manufacturing or were printed simultaneously by two different print apparatuses (such as seen in electron beam printing), such a unit cell may have misalignment between the two segments. If testing specifically for this manufacturing defect is of interest, such a scheme could be very applicable.

5.4.4. Cartesian grid, no pixel contrast limit

One model which is of interest for future research is the (fine) cartesian grid, in which the permittivity/permeability of each pixel is not restricted by object parameters. Instead, the permittivity and permeability of each pixel are the free parameters of the layer, whose values are bounded by the known materials. Having this many free variables potentially allows for the overall system to become overfitted. Despite this, such an overall system might converge to the correct value anyway. Such a scheme might work to detect the exact fault of printing failures which arises from shot noise. However, shot noise in general is ”averaged out” if looking at a periodic structure. As such, detecting shot noise can only be done to a single period at a time, and requires removing that period from the wider periodic structure and modelling it as a supercell.

5.5. Summary

In the end, we have chosen to proceed with one model. This is the rectilinear grid, where the grid line locations and pixel permittivity and permeability are the free parameters. This method is most useful for testing alignments of grid-like distributions. Since this description also accurately describes the training data provided to us, this method is most adapt at replicating the training data, so this method was implemented also for validation's sake.

The second model we had chosen to develop is the fine cartesian grid, where the permittivity/permeability of layers is impacted by shape objects, and the parameters defining the these shapes are the free parameters of the layer. This method has the advantage that it is theoretically the least restrictive in distributions it can model. The drawback is that it has trouble converging on sub-pixel detail, and increasing pixel density also results in increasing memory and processing costs in evaluating the convolution matrices. Additionally, it has limits on edge alignment which the rectilinear grid dues not. For these reasons, further testing of this scheme was discontinued.

The two other schemes were deemed too ill-suited for our use case, and thus not developed further.

6

Fast Square Root

6.1. Issues with RCWA algorithm

The purpose of this project was to implement this already-existing RCWA algorithm into Tensorflow, with the following two expectations:

- The Automatic Differentiation functionality native to the Keras module would make implementing the gradient descent method relatively easy;
- The so-called ”GPU acceleration” of Tensorflow would allow us to calculate many steps on Graphics processing units, which are designed with highly-parallelized operations (including matrix operations) in mind, so as to hopefully gain a significant increase in calculation speed.

In reality, however, not all operations programmed had the same degree of acceleration. While matrix multiplication was in the range of hundreds of times faster, matrix inversion was more in the range of ten times faster. Most crucially, the most computationally expensive part of the entire RCWA algorithm (the eigendecomposition of **PQ**) had a speedup of only 13%. This led the eigendecomposition to go from approximately two-thirds of the total computation time to 97% in some cases.

While implementing the conventional formulation as-is would technically satisfy the goals of this project as outlined in the research proposal, it would betray the research proposal goal in spirit if finding a solution to this problem was not at least attempted.

6.2. GPU acceleration of Eigendecomposition

Here, we go somewhat into the theory of parallel computing.

Most current research on GPU application for RCWA attempts to parallelize the eigendecomposition as best as possible. This typically involves a CPU available (optimized for high serial throughput) to compute the serial steps, a GPU available (optimized for high parallel throughput) to compute the parallel steps, and an interface between the two, in the form of high-throughput data connection or shared memory. [13] [14] [15]

6.2.1. Amdahl's law

Amdahl models programs as a share of operations which must be performed serially, and a share which can be computed in parallel. Increasing the number of processors increases the speed of the parallel portion, while keeping the speed of the serial portion the same. As such, the limit of the potential speedup is the inverse of the serial fraction.

In short, evaluating an eigendecomposition of a matrix is an inherently serial algorithm in which, according to Amdahl’s law bottlenecks appear earlier than other matrix operations.

6.2.2. Gustafson's law

Gustafson's law serves to work as a complement to Amdahl's law. While Amdahl's law presupposes that the scale of the problem does not increase, Gustafson observed that the scope of the problem tended to increase as the computing power of a system increases. The size of a numerical simulation, for example, tends to be bigger if run on a supercomputer than on a single laptop. Additionally, Gustafson observed that the number of serial steps of an algorithm did not grow with the overall size of the problem.

As such, if Amdahl's law assumes the scope of the problem (and thereby the serial fraction) to be constant, Gustafson's law assumes the serial portion of the problem to be constant while the parallel portion grows alongside the number of processors. In such a case, the serial fraction of the problem shrinks inversely to the number of processors, causing the maximum possible speedup as given by Amdahl's law to shrink. As a result, if you increase the size of the problem a hundredfold but also increase the parallel processes a hundredfold, in theory the total computation time should be the same.

One large problem with Gustafson's law is that it assumes the number of serial steps does not grow with the problem size. Simply put, this is not the case for eigendecomposition. Bigger matrices require a larger amount of serial steps to evaluate their eigendecompositions, which slows down the speed. If the parallel portion grows one hundredfold but the linear portion grows tenfold (so the square root of the problem size), and you increase the number of processors one hundredfold, the new total computation time can range from anywhere between the same time to taking ten times as long, depending on how much time was spent on the linear versus parallel portion previously. The higher the initial time portion of the linear section, the larger the slowdown of the larger problem. Since, in such a case, the share of time spent on the serial section only increases, sooner or later the problem becomes bottlenecked by the linear section and the total problem calculation time will converge to the serial calculation time.

If we want an algorithm which most effectively makes use of parallel computing, it needs to keep the serial portion of that algorithm to a minimum. A lower amount of serial computation means an algorithm will ultimately be faster, even if it requires more parallel processors to reach that point.

Arguably the golden standard of parallel-friendly operations is the matrix multiplication. This is what AI is built on; the massively-parallel processes are designed such that the lion's share of computational cost is matrix multiplication. The layer propagation of neural networks are abstractions which are ultimately calculated with matrix multiplications. These are so parallel-friendly that hardware is being designed which is specifically built to go as fast as possible for calculating matrix multiplications. One example is Google's Tensor Processing Unit, a proprietary integrated circuit designed solely for AI implementations. There is also the likes of NVIDIA implementing tensor cores in their latest Graphics Processing Units for tensor product acceleration, a feature which was available to us during the creation of this thesis.

6.3. Alternate formulation

The differential equation of RCWA is typically decomposed as follows:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{V} & -\mathbf{V} \end{bmatrix}^{-1} \tag{6.1}$$

Here, we substitute $\mathbf{V}$ with its composition as given in equation 3.21:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix}^{-1} \tag{6.2}$$

We can find an alternate formulation by inserting the following:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A}\mathbf{A}^{-1} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \mathbf{A}\mathbf{A}^{-1} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix}^{-1} \tag{6.3}$$

Where $\mathbf{A}$ is an arbitrary, as-of-yet undefined matrix which is invertible, and which has the same dimensions as the other block matrices. The reason we can do this is that $\mathbf{A}$ and $\mathbf{A}^{-1}$ multiply to the identity matrix, and thus should have no effect on the overall matrix beyond rounding error.

However, since matrix multiplications are associative, this expression can be re-evaluated as follows:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right) \left(\mathbf{A}^{-1} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \mathbf{A} \right) \left(\mathbf{A}^{-1} \begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix}^{-1} \right) \tag{6.4}$$

Which further simplifies to

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right) \left(\mathbf{A}^{-1} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \mathbf{A} \right) \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right)^{-1} \tag{6.5}$$

Owing to the fact that the first and last matrices are inverses of each other, the following is true:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}^n = \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right) \left(\mathbf{A}^{-1} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \mathbf{A} \right)^n \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right)^{-1} \tag{6.6}$$

and

$$expm\left(\Delta l \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} \right) = \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right) expm\left(\Delta l (\mathbf{A}^{-1} \begin{bmatrix} \lambda & \mathbf{0} \\ \mathbf{0} & -\lambda \end{bmatrix} \mathbf{A}) \right) \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\lambda^{-1} & -\mathbf{QW}\lambda^{-1} \end{bmatrix} \mathbf{A} \right)^{-1} \tag{6.7}$$

This new formulation is what is known as a similarity transform. [16]

6.3.1. Similarity Transform & Eigendecomposition

Suppose that there are two invertible matrices **A** (which is in no way related to the arbitrary matrix mentioned beforehand) and **C**. In short, if another invertible matrix **B** exists so that the following is true:

$$\mathbf{A} = \mathbf{BCB}^{-1} \tag{6.8}$$

then the matrices **A** and **C** are known as similar. Two similar matrices share the same set of eigenvalues, and the transformation

$$\mathbf{A} \mapsto \mathbf{B}^{-1}\mathbf{AB} \tag{6.9}$$

is referred to as a similarity transform of **A**.

There are many possible similarity transforms of **A**, as the only criteria for such a transform is that the matrix **B** is invertible. However, not all such similarity transforms are equally useful.

Arguably the most useful, used, and best-understood type of similarity transform is the diagonalization:

$$\mathbf{A} = \mathbf{PDP}^{-1} \tag{6.10}$$

Where **D** is a diagonal matrix. Since this diagonal matrix shares the same eigenvalues as **A**, and the eigenvalues of a diagonal matrix are the same as the diagonal values of said matrix, the diagonal entries of **D** are also the eigenvalues of **A**. It is for this reason that the diagonalization is also known as the eigendecomposition of a matrix. Just knowing the eigenvalues of a matrix can give insight beyond just looking at the matrix entries.

Additionally, the diagonalization makes many matrix operations (such as the matrix exponential, logarithm, sine, cosine, square root etc) trivial to compute. [16] Since the matrix exponential is defined as:

$$expm(\mathbf{A}) = \sum_{n=0}^{\infty} \frac{\mathbf{A}^n}{n!} \tag{6.11}$$

and we know that

$$(\mathbf{PDP}^{-1})^n = \mathbf{PD}^n\mathbf{P}^{-1} = (\mathbf{P} \begin{bmatrix} \lambda_1^n & 0 & \dots & 0 \\ 0 & \lambda_2^n & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_m^n \end{bmatrix} \mathbf{P}^{-1}) \tag{6.12}$$

It naturally leads to the following way of calculating the matrix exponential:

$$expm(\mathbf{A}) = \mathbf{P}\begin{bmatrix} \sum_{n=0}^{\infty}\frac{\lambda_1^n}{n!} & 0 & \dots & 0 \\ 0 & \sum_{n=0}^{\infty}\frac{\lambda_2^n}{n!} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sum_{n=0}^{\infty}\frac{\lambda_m^n}{n!} \end{bmatrix}\mathbf{P}^{-1} = \mathbf{P}\begin{bmatrix} e^{\lambda_1} & 0 & \dots & 0 \\ 0 & e^{\lambda_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & e^{\lambda_m} \end{bmatrix}\mathbf{P}^{-1} \tag{6.13}$$

As the sine and cosine functions can be described as a sum of two exponential functions, the matrix sine and cosine also can be described as a sum of two matrix exponentials, and thus inherit this property:

$$sinm(\mathbf{A}) = \mathbf{P}\begin{bmatrix} sin(\lambda_1) & 0 & \dots & 0 \\ 0 & sin(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & sin(\lambda_m) \end{bmatrix}\mathbf{P}^{-1} \tag{6.14}$$

$$cosm(\mathbf{A}) = \mathbf{P}\begin{bmatrix} cos(\lambda_1) & 0 & \dots & 0 \\ 0 & cos(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & cos(\lambda_m) \end{bmatrix}\mathbf{P}^{-1} \tag{6.15}$$

Same for the matrix logarithm, as the definition of the natural logarithm is an inverse operator of the exponential:

$$logm(\mathbf{A}) = \mathbf{P}\begin{bmatrix} ln(\lambda_1) & 0 & \dots & 0 \\ 0 & ln(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & ln(\lambda_m) \end{bmatrix}\mathbf{P}^{-1} \tag{6.16}$$

And, finally, since the principal square root can be evaluated as a combination of the matrix logarithm and matrix exponent:

$$z^{1/2} = exp(\frac{1}{2}ln(z)) \tag{6.17}$$

the matrix principal square root inherits this property as well.

$$\mathbf{A}^{\frac{1}{2}} = \mathbf{P}\begin{bmatrix} \lambda_1^{\frac{1}{2}} & 0 & \dots & 0 \\ 0 & \lambda_2^{\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_m^{\frac{1}{2}} \end{bmatrix}\mathbf{P}^{-1} \tag{6.18}$$

This method of calculating the matrix square root is so intuitive and elegant that it becomes difficult to imagine alternate methods of calculation which don't make use of the eigendecomposition. In RCWA, the following eigendecomposition is calculated:

$$\mathbf{PQ} = \boldsymbol{\Omega}^{\mathbf{2}} = \mathbf{W}\boldsymbol{\lambda}^{\mathbf{2}}\mathbf{W}^{-1} \tag{6.19}$$

where

$$\boldsymbol{\lambda}^{\mathbf{2}} = \begin{bmatrix} \lambda_1^2 & 0 & \dots & 0 \\ 0 & \lambda_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_m^2 \end{bmatrix} \tag{6.20}$$

from this, the following diagonal matrix is calculated:

$$\boldsymbol{\lambda} = \begin{bmatrix} (\lambda_1^2)^{\frac{1}{2}} & 0 & \dots & 0 \\ 0 & (\lambda_2^2)^{\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & (\lambda_m^2)^{\frac{1}{2}} \end{bmatrix} \tag{6.21}$$

Seeing this all together, one can only come to the conclusion that this leads to the following identity:

$$(\mathbf{PQ})^{\frac{1}{2}} = \boldsymbol{\Omega} = \mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} \tag{6.22}$$

However, from any of the literature seen, no mention of $\boldsymbol{\Omega}$ was found. Presumably, this is because one needs to turn this to an eigendecomposition anyway, the step of calculating $\boldsymbol{\Omega}$ became unnecessary, and the name of $\boldsymbol{\Omega^2}$ became vestigial. The eigendecomposition is so easy to work with that making use of it at one stage leads to trivial implementation at other stages, at which point it becomes increasingly difficult to imagine a way of working which does not make use of the eigendecomposition.

The only problem with this, again, is that eigendecomposition is bottlenecked on a parallel computer. Because of this, an alternate formulation requires not only a substitute for computing $\boldsymbol{\Omega}$, but of every computational step which makes use of an eigendecomposition.

Below, we outline an an alternate formulation which provides a substitute for every computational step which makes use of an eigendecomposition.

6.3.2. Alternate formulation

Here we introduce the following matrix:

$$\mathbf{A} = \begin{bmatrix} \mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{-1} \end{bmatrix} \tag{6.23}$$

Its inverse evaluates to the following:

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix} \tag{6.24}$$

Substituting these expressions into equation 6.5 yields the following expression:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\boldsymbol{\lambda}^{-1} & -\mathbf{QW}\boldsymbol{\lambda}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{-1} \end{bmatrix} \right) \left(\begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\lambda} \end{bmatrix} \begin{bmatrix} \mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{-1} \end{bmatrix} \right) \left(\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\boldsymbol{\lambda}^{-1} & -\mathbf{QW}\boldsymbol{\lambda}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{-1} \end{bmatrix} \right)^{-1} \tag{6.25}$$

$$\begin{bmatrix} \mathbf{W} & \mathbf{W} \\ \mathbf{QW}\boldsymbol{\lambda}^{-1} & -\mathbf{QW}\boldsymbol{\lambda}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{QW}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1} & -\mathbf{QW}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1} \end{bmatrix}$$
$$\begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\lambda} \end{bmatrix} \begin{bmatrix} \mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} \end{bmatrix} \tag{6.26}$$

In its most verbose, we can write this new formulation in the two following forms:

$$\begin{aligned} \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} &= \begin{bmatrix} \mathbf{WW}^{-1} & \mathbf{WW}^{-1} \\ \mathbf{QW}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1} & -\mathbf{QW}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{WW}^{-1} & \mathbf{WW}^{-1} \\ \mathbf{QW}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1} & -\mathbf{QW}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{WW}^{-1} & \mathbf{WW}^{-1} \\ \mathbf{VW}^{-1} & -\mathbf{VW}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{WW}^{-1} & \mathbf{WW}^{-1} \\ \mathbf{VW}^{-1} & -\mathbf{VW}^{-1} \end{bmatrix}^{-1} \end{aligned} \tag{6.27}$$

We know that

$$\left(\mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1}\right)^2 = \mathbf{W}\boldsymbol{\lambda}^{\mathbf{2}}\mathbf{W}^{-1} = \mathbf{PQ} \tag{6.28}$$

therefore $\mathbf{W}\boldsymbol{\lambda}\mathbf{W}^{-1}$ must be a square root of $\mathbf{PQ}$. Likewise,

$$\left(\mathbf{W}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1}\right)^2 = \mathbf{W}\boldsymbol{\lambda}^{-2}\mathbf{W}^{-1} = \left(\mathbf{W}\boldsymbol{\lambda}^{\mathbf{2}}\mathbf{W}^{-1}\right)^{-1} = \mathbf{PQ}^{-1} \tag{6.29}$$

so $\mathbf{W}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1}$ is the square root of the inverse of $\mathbf{PQ}$, and due to commutativity of these power operators, $\mathbf{W}\boldsymbol{\lambda}^{-1}\mathbf{W}^{-1}$ is the inverse of the square root of $\mathbf{PQ}$. Here we define $(\mathbf{PQ})^{\frac{1}{2}}$ as a matrix, whose eigenvalues are shared with $\boldsymbol{\lambda}$, with the following property:

$$((\mathbf{PQ})^{\frac{1}{2}})^2 = \mathbf{PQ} \tag{6.30}$$

Using this new matrix, we simplify our similarity transform to the following:

$$\begin{bmatrix}\mathbf{0} & \mathbf{P}\\ \mathbf{Q} & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \mathbf{I}\\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\end{bmatrix}\begin{bmatrix}(\mathbf{PQ})^{\frac{1}{2}} & \mathbf{0}\\ \mathbf{0} & -(\mathbf{PQ})^{\frac{1}{2}}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{I}\\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\end{bmatrix}^{-1} \tag{6.31}$$

This new formulation avoids the need to directly evaluate the diagonalization of $\mathbf{PQ}$, while simultaneously maintaining the ”coupled-wave” property which is needed in order to properly implement an S-matrix or enhanced T-matrix approach. In fact, these provide direct analogues to the matrices required for the S-parameter matrix method:

$$\begin{aligned}&\mathbf{W} \to \mathbf{I}\\ &\mathbf{V} \to \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\\ &\mathbf{X} \to expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}})\end{aligned} \tag{6.32}$$

The only thing needed at this stage is a fast and accurate algorithm to evaluate the square root and exponential of a matrix.

6.3.3. Proof of equivalence

Here, we prove that the analogues provided result in the exact same S-matrix.

We reiterate equation 3.66 , outlining the layer S-parameter matrix between two gap media:

$$\mathbf{S}_i = \begin{bmatrix}(\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}(\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{B}_{ig}) & (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{X}_i(\mathbf{A}_{ig} - \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig})\\ (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{X}_i(\mathbf{A}_{ig} - \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}) & (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}(\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{B}_{ig})\end{bmatrix} \tag{6.33}$$

We replace $\mathbf{W}$,$\mathbf{V}$, and $\mathbf{X}$ with the following:

$$\mathbf{W} \to \mathbf{I}, \mathbf{V} \to \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}, \mathbf{X} \to expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}}) \tag{6.34}$$

To make the proof easier, we use the notation for these ”analogues” given as per equation 6.27:

$$\mathbf{W}' = \mathbf{W}\mathbf{W}^{-1}, \mathbf{V}' = \mathbf{V}\mathbf{W}^{-1}, \mathbf{X}' = \mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1} \tag{6.35}$$

The AB-matrices then become

$$\mathbf{A}'_{ig} = \frac{1}{2}\mathbf{W}_i(\mathbf{W}_i^{-1}\mathbf{W}_g + \mathbf{V}_i^{-1}\mathbf{V}_g) = \mathbf{W}_i\mathbf{A}_{ig}, \mathbf{B}'_{ig} = \frac{1}{2}\mathbf{W}_i(\mathbf{W}_i^{-1}\mathbf{W}_g - \mathbf{V}_i^{-1}\mathbf{V}_g) = \mathbf{W}_i\mathbf{B}_{ig} \tag{6.36}$$

$$(\mathbf{A}'_{ig})^{-1} = \mathbf{A}_{ig}^{-1}\mathbf{W}_i^{-1} \tag{6.37}$$

From this, we get

$$\mathbf{S}'_{i,11} = (\mathbf{A}'_{ig} - \mathbf{X}'_i\mathbf{B}'_{ig}(\mathbf{A}'_{ig})^{-1}\mathbf{X}'_i\mathbf{B}'_{ig})^{-1}(\mathbf{X}'_i\mathbf{B}'_{ig}(\mathbf{A}'_{ig})^{-1}\mathbf{X}'_i\mathbf{A}'_{ig} - \mathbf{B}'_{ig}) \tag{6.38}$$

Which, when expanding every term, gives us

$$\begin{aligned}\mathbf{S}'_{i,11} =&(\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{B}_{ig})^{-1}\\ &\star(\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{B}_{ig})\end{aligned} \tag{6.39}$$

Factoring away a lot of products of $\mathbf{W}_i$ with its inverse gives:

$$\mathbf{S}'_{i,11} = (\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1} \star (\mathbf{W}_i\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{B}_{ig}) \tag{6.40}$$

This further simplifies to:

$$\begin{aligned}\mathbf{S}'_{i,11} &= (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{W}_i^{-1}\mathbf{W}_i(\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{B}_{ig}) \\ &= (\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}(\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{A}_{ig} - \mathbf{B}_{ig}) \\ &= \mathbf{S}_{i,11}\end{aligned} \tag{6.41}$$

We can similarly show that for $\mathbf{S}'_{i,12}$:

$$\begin{aligned}\mathbf{S}'_{i,12} =&(\mathbf{A}'_{ig} - \mathbf{X}'_i\mathbf{B}'_{ig}(\mathbf{A}'_{ig})^{-1}\mathbf{X}'_i\mathbf{B}'_{ig})^{-1} \\ &\star \mathbf{X}'_i(\mathbf{A}'_{ig} - \mathbf{B}'_{ig}(\mathbf{A}'_{ig})^{-1}\mathbf{B}'_{ig}) \\ =&(\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{B}_{ig})^{-1} \\ &\star \mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}(\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{B}_{ig}) \\ =&(\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}(\mathbf{W}_i\mathbf{A}_{ig} - \mathbf{W}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}) \\ =&(\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{W}_i(\mathbf{A}_{ig} - \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}) \\ =&(\mathbf{A}_{ig} - \mathbf{X}_i\mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{X}_i\mathbf{B}_{ig})^{-1}\mathbf{X}_i(\mathbf{A}_{ig} - \mathbf{B}_{ig}\mathbf{A}_{ig}^{-1}\mathbf{B}_{ig}) \\ =&\mathbf{S}_{i,12}\end{aligned} \tag{6.42}$$

Thus, making use of symmetry:

$$\mathbf{S}'_i = \begin{bmatrix}\mathbf{S}'_{i,11} & \mathbf{S}'_{i,12} \\ \mathbf{S}'_{i,12} & \mathbf{S}'_{i,11}\end{bmatrix} = \begin{bmatrix}\mathbf{S}_{i,11} & \mathbf{S}_{i,12} \\ \mathbf{S}_{i,12} & \mathbf{S}_{i,11}\end{bmatrix} = \mathbf{S}_i \tag{6.43}$$

This concludes the proof that the S-matrix generated using $\mathbf{W}'$,$\mathbf{V}'$, and $\mathbf{X}'$ is identical as the S-matrix generated using $\mathbf{W}$,$\mathbf{V}$, and $\mathbf{X}$.

A second proof replicates this result with the Transmittance matrix:

$$\begin{aligned}\mathbf{T}'_i = &\begin{bmatrix}\mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{I} & \mathbf{I} \\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\end{bmatrix}\begin{bmatrix}expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}}) & \mathbf{0} \\ \mathbf{0} & -expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}})^{-1}\end{bmatrix} \\ &\star\begin{bmatrix}\mathbf{I} & \mathbf{I} \\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g\end{bmatrix}\end{aligned} \tag{6.44}$$

$$\mathbf{T}_i = \begin{bmatrix}\mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i\end{bmatrix}\begin{bmatrix}expm(\Delta l\boldsymbol{\lambda}_i) & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l\boldsymbol{\lambda}_i)\end{bmatrix}\begin{bmatrix}\mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g\end{bmatrix} \tag{6.45}$$

Since

$$\begin{aligned}&expm\left(\Delta l\begin{bmatrix}\mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\right) \\ =&\begin{bmatrix}\mathbf{I} & \mathbf{I} \\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\end{bmatrix}\begin{bmatrix}expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}}) & \mathbf{0} \\ \mathbf{0} & -expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}})^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{I} \\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}}\end{bmatrix}^{-1} \\ =&\begin{bmatrix}\mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i\end{bmatrix}\begin{bmatrix}expm(\Delta l\boldsymbol{\lambda}_i) & \mathbf{0} \\ \mathbf{0} & expm(-\Delta l\boldsymbol{\lambda}_i)\end{bmatrix}\begin{bmatrix}\mathbf{W}_i & \mathbf{W}_i \\ \mathbf{V}_i & -\mathbf{V}_i\end{bmatrix}^{-1}\end{aligned} \tag{6.46}$$

Thus, we know that

$$\begin{bmatrix}\mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g\end{bmatrix}^{-1} expm\left(\Delta l\begin{bmatrix}\mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0}\end{bmatrix}\right)\begin{bmatrix}\mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g\end{bmatrix} = \mathbf{T}'_i = \mathbf{T}_i \tag{6.47}$$

and since the S-matrices are defined as principal pivotal transforms solely of the T-matrices, we can thus asser the following:

$$\mathbf{S}'_i = \mathbf{S}_i \tag{6.48}$$

6.4. Matrix spectra

The matrix spectrum is the set of eigenvalues of a square matrix.

Why is this important? Well, as outlined previously, many matrix functions can be described as their equivalent scalar functions acting on every eigenvalue. Thus, if you know the spectrum of a matrix, you know what the spectrum of a function acting on that matrix would also look like. This is the principle which makes the use of eigendecomposition trivialise other matrix operations.

While we will not be making use of this property, the spectrum is still very useful for illustration purposes, as it allows us a way to visualise what is otherwise a grid of complex values. A lot of these illustrative plots will be scatter plots in the complex plane, showing where all the eigenvalues fall (without saying anything about the eigenvectors).

The following illustrations are based ont he spectrum of a sample **PQ** matrix. This is the same setup as provided by ”Case 2” in the results, with some small deviations:

- $\theta = -5°$
- $\phi = 2°$
- $M = 5$
- $N = 4$

In the case of ”lossy” material, the additional adjustment is made:

- $\epsilon_{res} = (1.5 - 0.5i)^2$

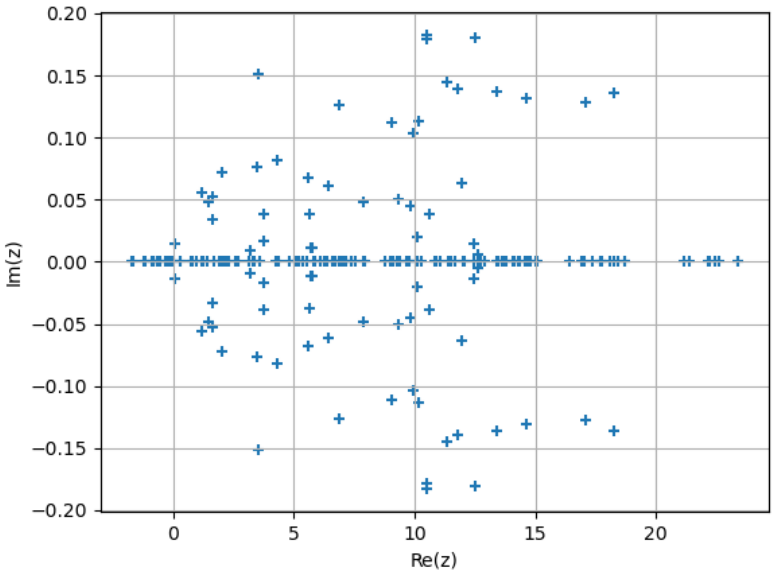


Figure 6.1: Spectrum of **PQ** in the complex plane, $M = 5$, $N = 4$, lossless.

Important to note: The imaginary axis is stretched to put more emphasis on the imaginary part of the spectrum. What we see here is the following:

- On the left, there are some eigenvalues close to -1; these are the regular plane waves.
- On the right, you have eigenvalues with large, positive values only; these are your deeply evanescent waves.
- Since there is no absorptive medium, all the eigenvalues stay very close to the real number line, but there are some terms with complex parts. These complex roots come in pairs. This is the complex conjugate root theorem in action; the eigenvalues are the roots of the characteristic

polynomial of the matrix, and since (in the case of non-absorptive media) all the terms of **PQ** are real, the roots come in conjugate pairs.

This last point has been a primary source of headache; because of this, we know that generally, in the case of non-absorptive media, there are eigenvalues in every quadrant of the complex plane, or at least we cannot assume that this is not the case. The spectrum of the principal matrix square root of **PQ** is the exact same as the principal scalar square root of the spectrum of **PQ**. Our own matrix square root algorithm (which will be explained in detail later on in this chapter) confirms this behavior:

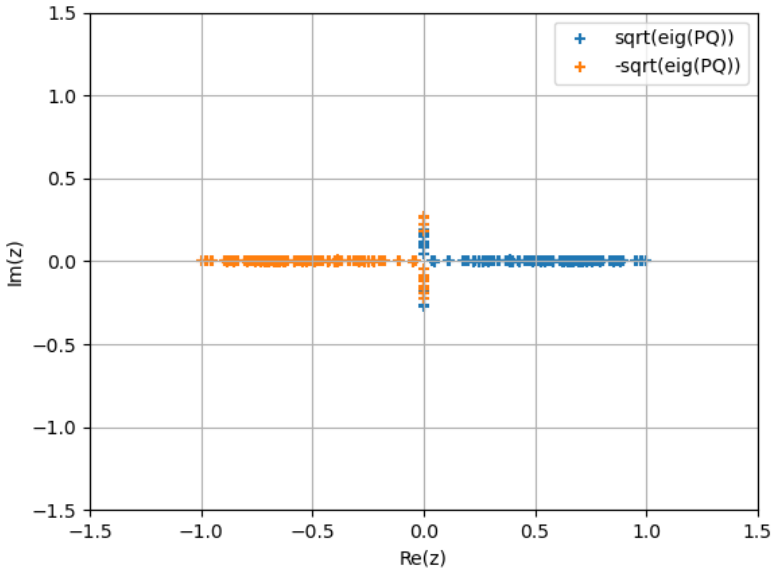


Figure 6.2: Positive and negative square roots of spectrum of **PQ** in the complex plane, $M = 5$, $N = 4$, lossless.

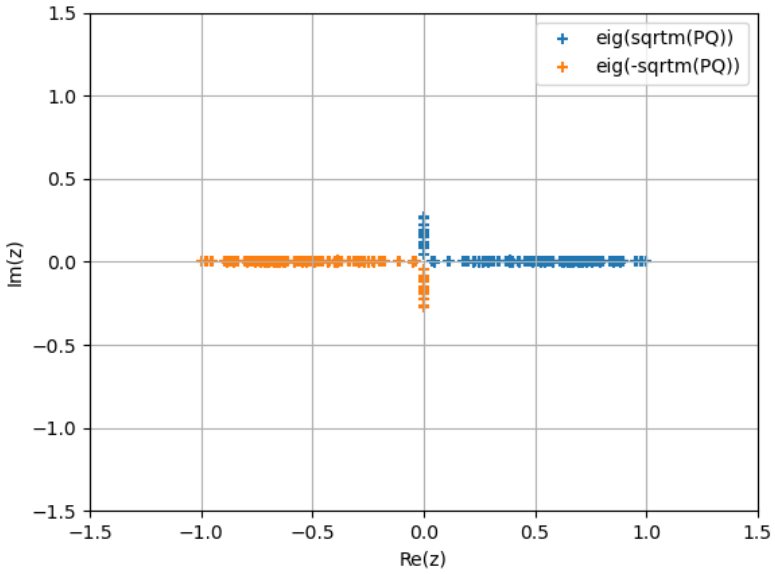


Figure 6.3: Spectrums of $sqrtm$(**PQ**) and $-sqrtm$(**PQ**) in the complex plane, $M = 5$, $N = 4$, lossless.

In this case, the ”break line” which considers in which set the eigenvalues is the imaginary axis; what

determines whether an eigenvalue is in $sqrt(eig(\mathbf{PQ}))$ or $-sqrt(eig(\mathbf{PQ}))$ is the sign of the imaginary part of an eigenvalue. In many cases, this sign is determined basically by rounding error. To perfectly replicate this in $eig(sqrtm(\mathbf{PQ}))$ would require some values to sit on the branch cut (which have no rounding error), thus the method will not converge. This error is primarily attributed to trying to find the principal square root (the root with spectra wholly in the right half of the complex plane), which in this case does not exist. Thus, we need an offset to not go through all the eigenvalues, which gets us a slightly different one. More on this later on. Equation 6.31 tells us that the $\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$ is linked by a similarity transform to $\begin{bmatrix} (\mathbf{PQ})^{\frac{1}{2}} & \mathbf{0} \\ \mathbf{0} & -(\mathbf{PQ})^{\frac{1}{2}} \end{bmatrix}$, so they should have the same eigenvalues. Therefore, the spectrum is the same as the union of the spectrum of $(\mathbf{PQ})^{\frac{1}{2}}$ and $-(\mathbf{PQ})^{\frac{1}{2}}$ (since it is a block matrix). This is exactly what we see:

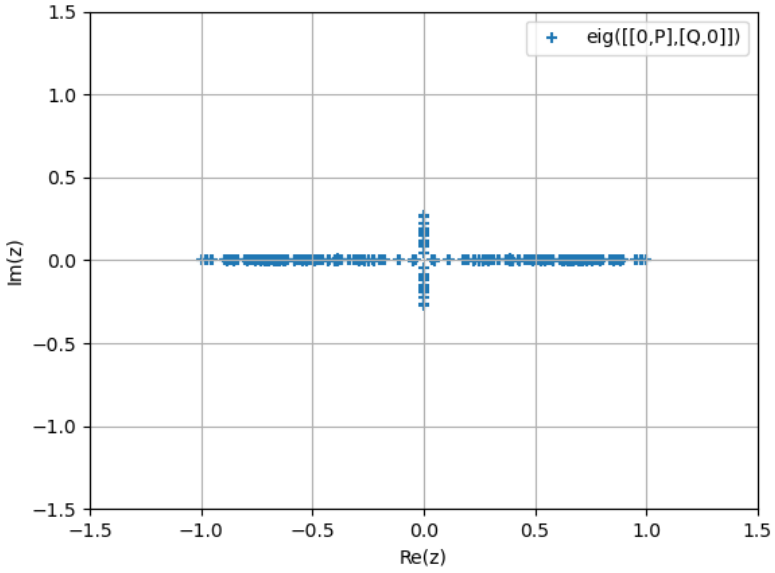


Figure 6.4: Spectrum of $\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$ in the complex plane, $M = 5$, $N = 4$, lossless.

6.4.1. Lossy media

No validation data for lossy data was available, so the following insights are inferred from a potentially invalid formulation.

If we take $n_{res} = 1.5 - 0.5i$, **PQ** gets the following spectrum for $M = 5$,$N = 4$:

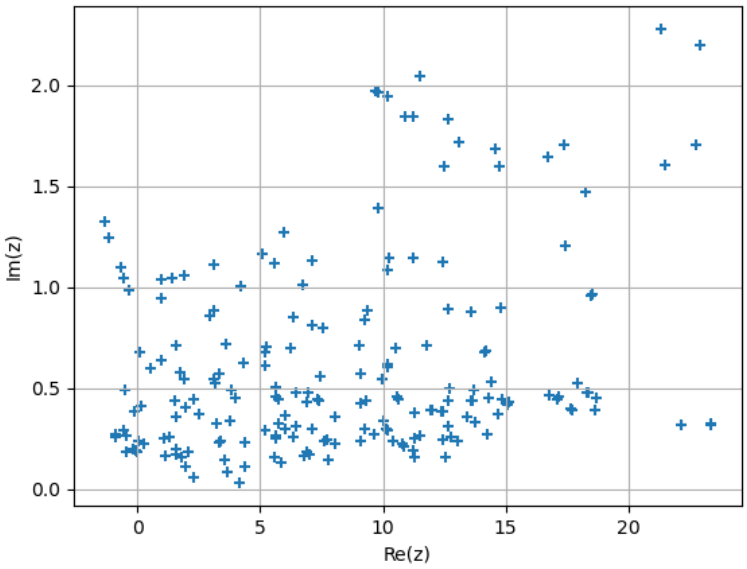


Figure 6.5: Spectrum of **PQ** in the complex plane, $M = 5$, $N = 4$, lossy.

Important to note: In this case, there are no eigenvalues of **PQ** with a negative imaginary part. Because of this, $-i\mathbf{PQ}$ would have no eigenvalues with negative real part. In the future, lossy layers where the eigenvalues of **PQ** have no negative imaginary part are referred to as ”sufficiently lossy”.

Also, in this case, there is no issue with the dividing line going straight through some eigenvalues of $\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$, so the issue of which eigenvalues for the square root becomes effectively moot.

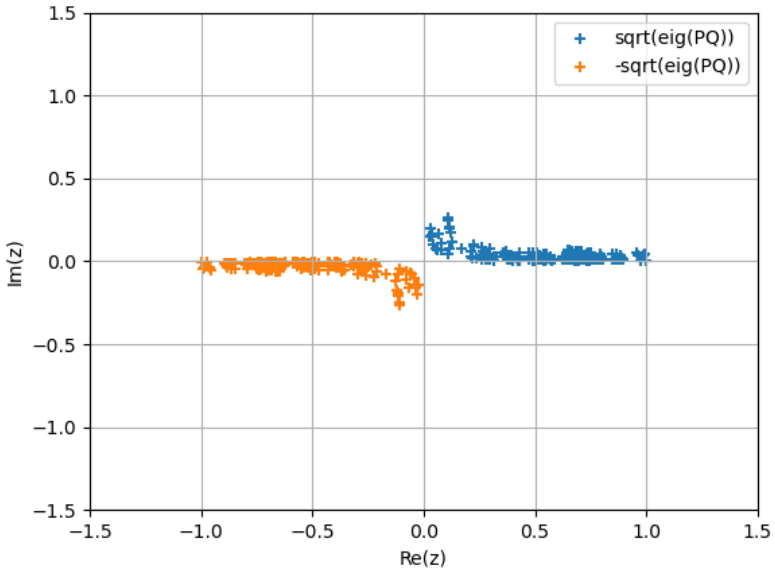


Figure 6.6: Positive and negative square roots of spectrum of **PQ** in the complex plane, $M = 5$, $N = 4$, lossy.

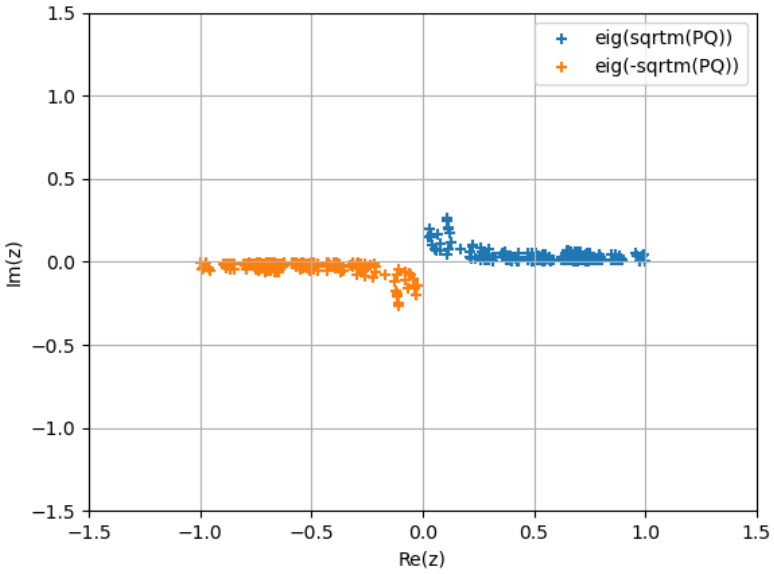


Figure 6.7: Spectrums of $sqrtm(\mathbf{PQ})$ and $-sqrtm(\mathbf{PQ})$ in the complex plane, $M = 5$, $N = 4$, lossy.

6.4.2. Preface: Spectral radius

It also begs the question; what exactly is the norm of a matrix?

There are multiple different norm formulas. One of the most well-known norms is the Frobenius norm:

$$|\mathbf{A}|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2} \tag{6.49}$$

All matrix norms have the property that:

$$\lim_{n\to\infty} |\mathbf{A}^n|^{1/n} = \rho(\mathbf{A}) \tag{6.50}$$

where ρ is the spectral radius of a matrix.

$$\rho(\mathbf{A}) = max|\lambda_i| \tag{6.51}$$

The spectral radius is the radius of the circle about the origin in the complex plane, which contains all eigenvalues. As an example:

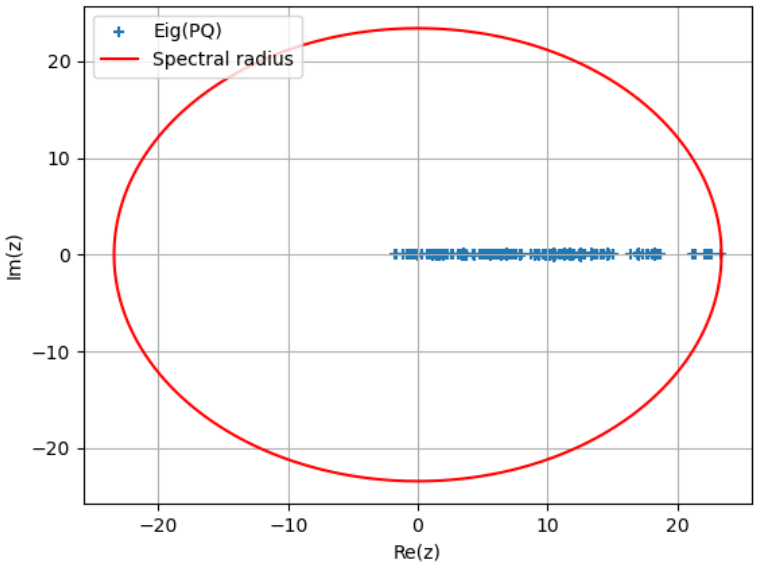


Figure 6.8: Spectrum of **PQ** in the complex plane, $M = 5$, $N = 4$, lossless, with spectral radius.

An algorithm has been implemented to approximate the spectral radius of a matrix as follows:

Algorithm 1 Matrix spectrum approximation

Require: $\mathbf{A}, k$
Ensure: ρ (approximation of matrix spectrum)

1: **function** Spectrum($\mathbf{A}, k$)
2: $\mathbf{X} \leftarrow \mathbf{A}$
3: **for** $i \leftarrow 0$ to k **do**
4: $\mathbf{X} \leftarrow \mathbf{XX}$
5: **end for**
6: $\rho \leftarrow (|\mathbf{X}|_F)^{\frac{1}{2^k}}$
7: **return** ρ
8: **end function**

This algorithm calculates the following:

$$\rho(\mathbf{A}, k) = |\mathbf{A}^{2^k}|_F^{1/2^k} \tag{6.52}$$

This method allows us to approach equation 6.50 for n with $\log_2(n)$ matrix multiplications. In practice, only a relatively rough estimate of $\rho(\mathbf{A})$ is needed, so $k = 4$ was chosen as a standard.

What happens in such a case? Well, we can plot the spectrum of $\mathbf{PQ}^{16}$ for lossless PQ:

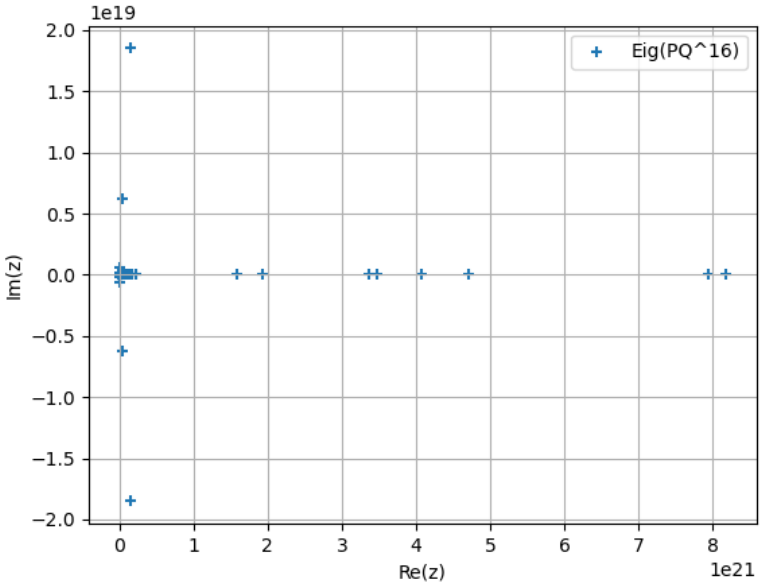


Figure 6.9: Spectrum of $\mathbf{PQ}^{16}$ in the complex plane, $M = 5$, $N = 4$, lossless.

What we see is that the vast majority of eigenvalues relatively approach zero. Because of this, in evaluating the matrix norm, only the biggest eigenvalues matter. If this is taken to its extreme, only the biggest eigenvalue matters, and every other eigenvalue is relatively zero. These relatively small eigenvalues do not contribute as much to the matrix norm as when they were relatively larger, so a matrix norm for $|\mathbf{PQ}^{16}|$ is a better estimate for the spectral radius of $\mathbf{PQ}^{16}$ than the matrix norm for $|\mathbf{PQ}|$ as an estimate for the spectral radius of $\mathbf{PQ}$.

In such a case, the norm of $\mathbf{PQ}^{16}$ converges in relative size to the absolute value of the largest eigenvalue of $\mathbf{PQ}^{16}$. Then, to get the absolute value of the largest eigenvalue of $\mathbf{PQ}$, we take the sixteenth root of $|\mathbf{PQ}^{16}|$, which then becomes our estimate for the spectral radius. This has the added benefit that, if a matrix norm consistently overestimates the spectral radius by a factor α, it will overestimate the spectral radius of $\mathbf{PQ}^{16}$ by a factor of α, and the actual spectrum only by a factor of $\sqrt[16]{\alpha}$. In most situations, as previously outlined, α also begins to approach 1 as the power term of $\mathbf{PQ}^n$ becomes larger.

To start, we attempted to use the stock Tensorflow matrix square root algorithm. This algorithm was almost exactly the same speed of the diagonalization function of Tensorflow, leading us to the suspicion that the matrix square root was evaluated using an eigendecomposition. Again, the eigendecomposition is so elegant in implementation that it was being evaluated even we attempted to circumvent its use. As a result, we needed a different algorithm for evaluating the square root.

The methods landed on are a class of methods called iterative methods; These methods continuously repeat a certain iteration, in which the output of iteration k serves as the input of iteration $k + 1$. The goal is that these iterations approach the desired matrix function, and do so with infinite precision with a finite number of calculations. Before we found methods which actually satisfied these criteria, there were a couple of other methods explored, which we will briefly mention now.

6.4.3. Newton iteration

The second thing tried was the Newton iteration:

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{A} \\ \mathbf{P}_{k+1} &= \frac{\mathbf{P}_k + \mathbf{A}\mathbf{P}_k^{-1}}{2} \\ \lim_{k\to\infty} \mathbf{P}_k &= \mathbf{A}^{1/2} \end{aligned} \tag{6.53}$$

This method had the issue that it was numerically unstable. The reason for this is that, due to rounding errors, $\mathbf{A}$ does not perfectly commute with $\mathbf{P}_k$, and therefore with $\mathbf{P}_k^{-1}$. This causes the "commutation error" (the size of $\mathbf{A}\mathbf{P}^{-1} - \mathbf{P}^{-1}\mathbf{A}$) to grow with every iteration. Eventually, the size of this commutation error will start to dominate the matrix, at which point the method starts to diverge, making the overall method useless.

6.4.4. The complex sign function

The complex sign function is defined as follows:

$$csgn(z) = \begin{cases} -1 & \text{for } \Re(z) < 0 \\ 1 & \text{for } \Re(z) > 0 \end{cases} \tag{6.54}$$

The behavior for $\Re(z) = 0$ is undefined. There are some extensions which define it as $msgn(iz)$, which we can implement with the use of a limit:

$$csgn_{i^+}(z) = lim_{\phi \to 0^+} csgn(e^{i\phi} z) \tag{6.55}$$

The complex sign can actually be approached using the following recursion: [17]

$$\begin{aligned} z_{k+1} &= \frac{1}{2}\left(z_k + \frac{1}{z_k}\right) \\ \lim_{k\to\infty} z_k &= \mathrm{csgn}(z_0) \end{aligned} \tag{6.56}$$

This iteration does not converge if $\Re(z_0) = 0$.

6.4.5. Matrix sign function

The matrix sign function, conceptually, is the sign function of the eigenvalues of a matrix. if we take the eigendecomposition of a matrix

$$\mathbf{A} = \mathbf{W} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_n \end{bmatrix} \mathbf{W}^{-1} \tag{6.57}$$

Then the matrix sign function can be expressed as:

$$msgn(\mathbf{A}) = \mathbf{W} \begin{bmatrix} csgn(\lambda_1) & 0 & \dots & 0 \\ 0 & csgn(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & csgn(\lambda_n) \end{bmatrix} \mathbf{W}^{-1} \tag{6.58}$$

Likewise, the matrix sign can be found with the following recursion:

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{A} \\ \mathbf{P}_{k+1} &= \frac{\mathbf{P}_k + \mathbf{P}_k^{-1}}{2} \\ \lim_{k\to\infty} \mathbf{P}_k &= msgn(\mathbf{A}) \end{aligned} \tag{6.59}$$

This is, in effect, the same as the previously-mentioned Newton iteration for the square root, with the exception that it is not attempting to find the square root of $\mathbf{I}$. As $\mathbf{I}$ commutes with every matrix, the commutation error no longer appears. While each iteration still introduces rounding errors, they no longer have the opportunity to grow. This gives a rounding error which is approximately equal to the sum of the rounding error introduced at every step.

6.4.6. First square root method

With all this, a rudimentary algorithm was formulated to find the approximate matrix square root:

- find msgn($\mathbf{A}$) using the matrix sign iteration:
- find a square root of msgn($\mathbf{A}$) by evaluating

$$\sqrt{\text{msgn}(\mathbf{A})} = \frac{msgn(\mathbf{A}) + i\mathbf{I}}{1+i} \tag{6.60}$$

  using the Newton iteration (the phase offset was needed to ensure convergence);
- find an approximate square root of $\mathbf{B} = \text{msgn}(\mathbf{A})\mathbf{A}$ using the Taylor series:

$$\mathbf{B}^{\frac{1}{2}}_{approx.} = \sqrt{|\mathbf{B}|_F}\left(\sum_{k=1}^{K}\binom{\frac{1}{2}}{k}(\mathbf{I} - \frac{\mathbf{B}}{|\mathbf{B}|_F})^k\right) \tag{6.61}$$

  where K was in the triple digits;
- use commutativity to find an approximate square root using the commutation identity

$$\sqrt{\mathbf{A}}_{approx.} = (\text{msgn}(\mathbf{A}))^{\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}_{approx.} \tag{6.62}$$

- get this as close as possible to the actual square root by using a Newton iteration on the result, until the error started growing again.

This method was slow, convoluted, the value of K was arbitrarily chosen yet impactful on the end result, the size of the error was still relatively large, and it did not even evaluate the principal square root. It had flaws, to say the least. However, it did work in finding square roots of $\mathbf{PQ}$ sometimes. It made use of a stable Newton iteration in some form, while finding the matrix sign. The only thing it lacked is a way to use this stable iteration to calculate the square root directly.

From here on, we will discuss iterative methods which are stable, and approach the matrix square root with theoretically infinite precision with a finite number of calculations.

6.5. Iterative methods for the Matrix Square root

6.5.1. Stable Newton's method

Eventually, the following method was found:[18]

$$\begin{aligned}
\mathbf{Y}_0 &= \mathbf{A}, \mathbf{Z}_0 = \mathbf{I} \\
\mathbf{Y}_{k+1} &= \frac{\mathbf{Y}_k + \mathbf{Z}_k^{-1}}{2} \\
\mathbf{Z}_{k+1} &= \frac{\mathbf{Z}_k + \mathbf{Y}_k^{-1}}{2} \\
\lim_{k\to\infty} \mathbf{Y}_k &= \mathbf{X} \\
\lim_{k\to\infty} \mathbf{Z}_k &= \mathbf{X}^{-1} \\
\mathbf{X}^2 &= \mathbf{A}
\end{aligned} \tag{6.63}$$

It turns out that it makes use of the following property of the matrix sign function: [19]

$$msgn(\begin{bmatrix}\mathbf{0} & \mathbf{A} \\ \mathbf{I} & \mathbf{0}\end{bmatrix}) = \begin{bmatrix}\mathbf{0} & \mathbf{A}^{1/2} \\ \mathbf{A}^{-1/2} & \mathbf{0}\end{bmatrix} \tag{6.64}$$

In essence, it uses the iterative method as outlined by equation 6.59, but since the diagonal blocks are always zero (and the inverse of the two remaining blocks can be computed independently), the remaining non-trivial calculations are the same as those outlined in the ”Stable Newton’s Method”.

This method converges unconditionally, so long as all eigenvalues are not on the discontinuity line in the complex plane, which goes from $0 + 0i$ to $-\infty + 0i$. This method guarantees that all eigenvalues of

the square root fall on the right-hand side of the complex plane. This criteria is useful for convergence reasons.

This is unlike many other iterative methods, which eventually start to diverge due to compounding rounding error. This method avoids this by not including the matrix of interest in the iteration, but only in the initial conditions.

Below, we illustrate the number of iterations required for this iteration to converge:

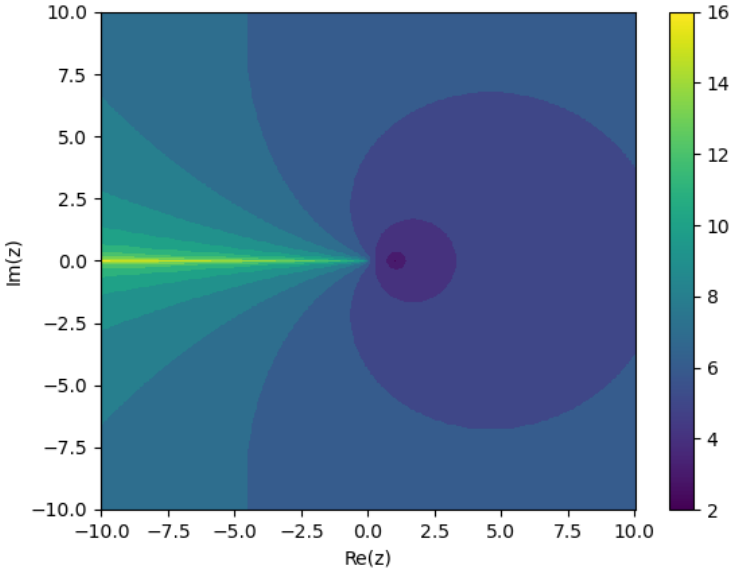


Figure 6.10: Number of iterations required for the eigenvalues of **PQ** to converge, depending on their value.

This figure is actually somewhat incomplete; the figure actually places no values on the real number line. This is because those values would not converge. For generating this graph, a total of 100 steps are done, and assumes that all values would converge by that point. For values on the negative real line, these would all have the value of 100, ruining the contrast for all other values. Furthermore, the data one would see is a lie, because if the maximum number of iterations was set at any other value, it would be that value. Below is this figure repeated, but with values on the negative real line, with a maximum of 100 iterations.

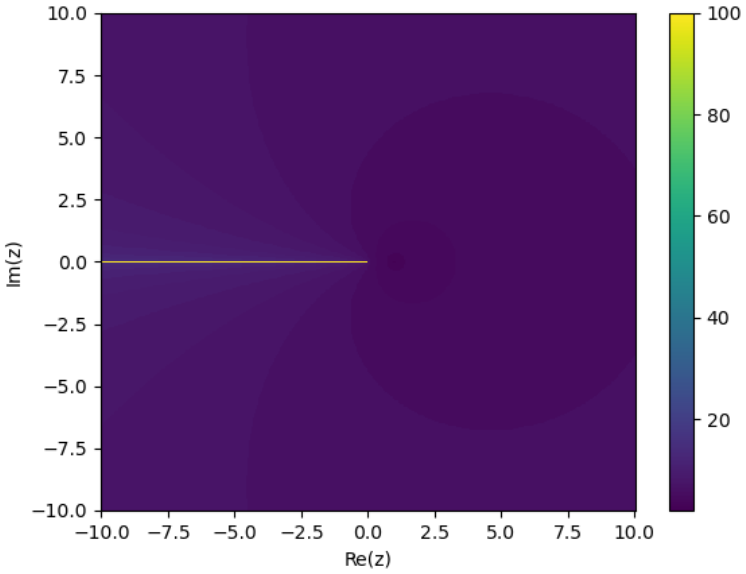


Figure 6.11: Number of iterations required for the eigenvalues of **PQ** to converge, depending on their value, with some values on the negative real number line.

6.5.2. Spectral rotation

The problem is that there are many cases in RCWA where an eigenvalue lies on on the discontinuity line in the complex plane, which goes from $0+0i$ to $-\infty+0i$. In the case of non-absorptive media, this is almost always the case.

To circumvent this problem, we instead find the root of $(e^{i\phi}\mathbf{PQ})$, which we multiply with $e^{-i\phi/2}$, where ϕ is real. While this result is still a square root of **PQ**, the root found is possibly not the same one; its eigenvalues are no longer guaranteed to be in the right-half plane. Additionally, to reduce the magnitude of rounding errors, the matrix is first normalized to its spectral radius.

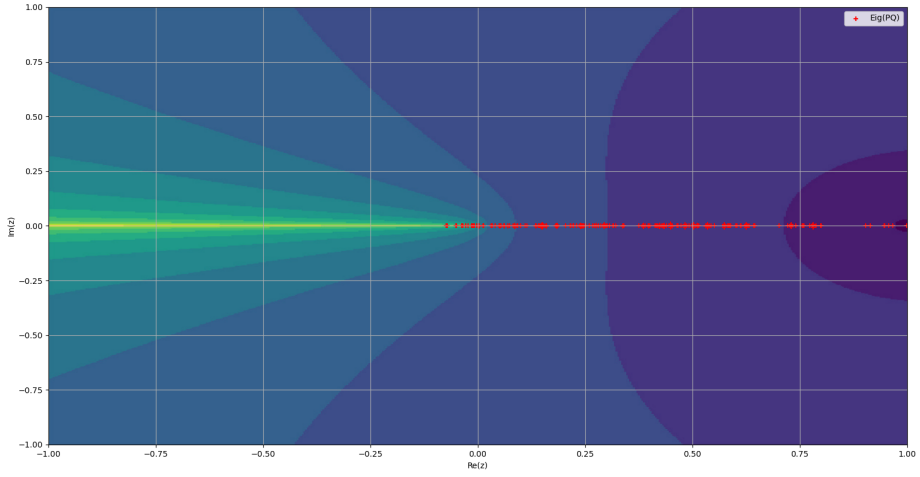


Figure 6.12: Normalized spectrum of $e^{i\phi}\mathbf{PQ}$, $M=5, N=4, \phi=0$, superimposed on figure 6.10.

Since some values are perfectly on the negative real number line, this will not converge.

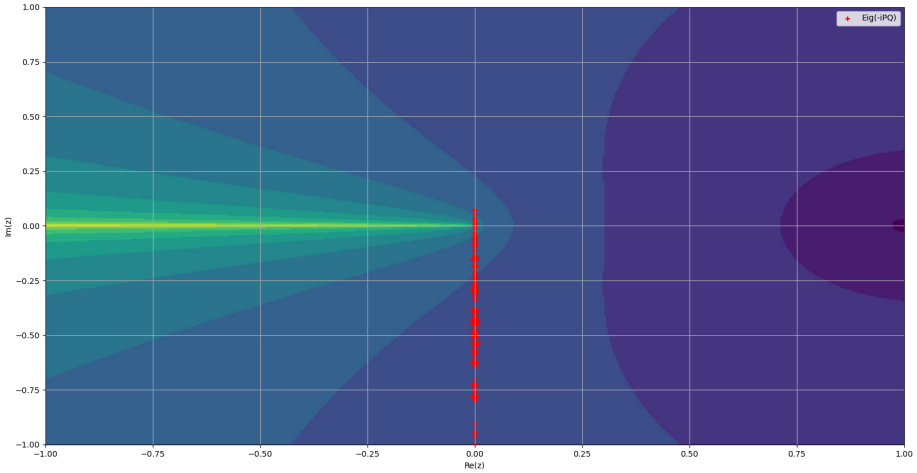


Figure 6.13: Normalized spectrum of $e^{i\phi}$**PQ**, $M = 5, N = 4, \phi = \pi/2$, lossy, superimposed on figure 6.10.

This will converge, due to the rotation.

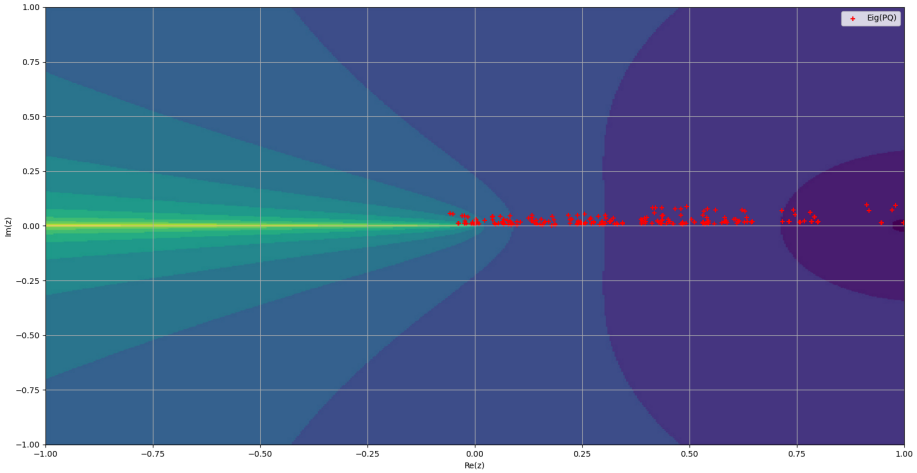


Figure 6.14: Normalized spectrum of $e^{i\phi}$**PQ**, $M = 5, N = 4, \phi = 0$, lossy, superimposed on figure 6.10.

This will converge, so rotation is not strictly necessary, but only slowly.

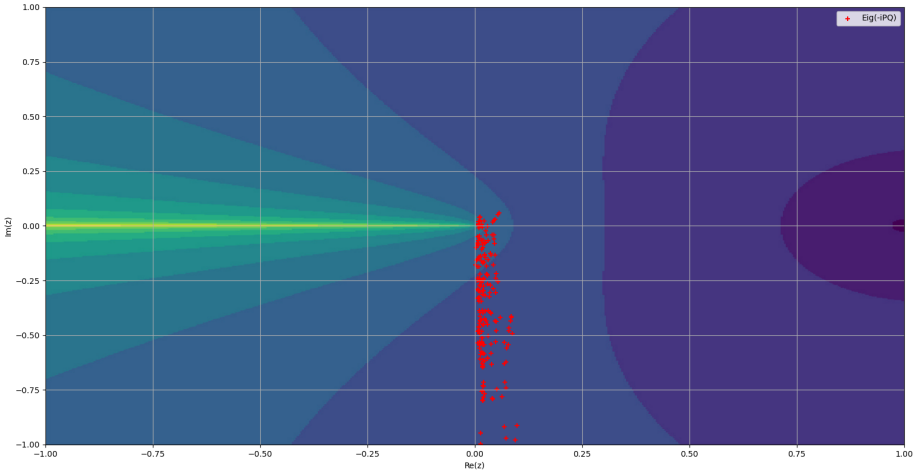


Figure 6.15: Normalized spectrum of $e^{i\phi}$**PQ**, $M = 5$,$N = 4$,$\phi = \pi/2$, lossy, superimposed on figure 6.10.

This will converge faster than if not rotated.

The approach as implemented now takes $phi = \pi/2$, so that $e^{i\phi} = i$. This does not alter the eigenvalues for primarily-evanescent eigenmodes, but might alter the eigenvalues for primarily propagating eigenmodes.

Alternatively, one can pick $e^{i\phi} = 1 + \zeta i$, where ζ is very small. This method has almost no impact on the roots selected. It does, however, have the downside that convergence becomes much slower. This is because values converge to their root more slowly the closer they are to the negative real line, to the point where they theoretically never converge if on the negative real line.

While the effect of different eigenvalues for further calculations is not wholly explored, they are estimated to be only minimal. The choice of $e^{i\phi} = i$ is made on the basis that it probably converges fastest, and the potential downsides are seen as negligible.

6.5.3. Other iterative methods for complex sign function

While this method does avoid the eigendecomposition, it is not perfect. It requires two matrix inversions per iteration (which can be speeded up to one inverse per iteration with some reformulation), and has quadratic convergence. While this is good enough for the HPC to have serious gains in computation speed, it is not multiplication-friendly; that is, it does not make use of solely multiplication and addition operations. Finding methods which compute a smaller number of matrix inversions in exchange for a larger number of matrix multiplications will perform faster on the most cutting-edge parallelized hardware. As such, the search for alternate iterative methods continued, as outlined see below.

Here we introduce the scalar Newton-Shultz Iteration; it takes the first-order Taylor approximation for the inverse, which is

$$\frac{1}{z_k} \approx z_k(1 - z_k^2) \tag{6.65}$$

And places it in equation 6.56:

$$z_{k+1} = \frac{1}{2}\left(z_k + \frac{1}{z_k}\right)$$

Thus giving the iteration

$$z_{k+1} = \frac{z_k + z_k(1 - z_k^2)}{2} = \frac{z_k}{2}(3 - z_k^2) \tag{6.66}$$

As given by [17], theorem 3.1, we can guarantee converges if the following is true:

$$|(1 - z_0^2)| < 1 \tag{6.67}$$

Below, we illustrate the region of convergence of iteration 6.66 evaluated numerically, as well as the region of convergence guaranteed by condition 6.67:

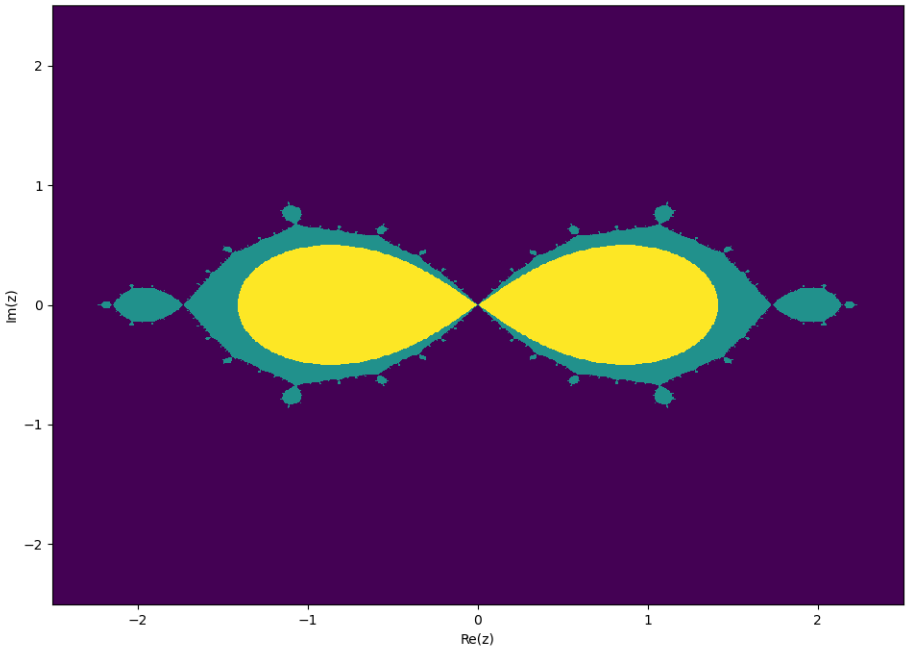


Figure 6.16: Region of convergence for scalar Newton-Shultz iteration for z_0. Yellow is region of guaranteed convergence by equation 6.67, green is the actual of the region of convergence.

With this groundwork, we now introduce the matrix Newton-Shultz iteration:

$$\begin{aligned} \mathbf{Y}_0 &= \mathbf{A}, \mathbf{Z}_0 = \mathbf{I} \\ \mathbf{Y}_{k+1} &= \frac{1}{2}\mathbf{Y}_k(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k) \\ \mathbf{Z}_{k+1} &= \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k)\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k &= \mathbf{X} \\ \lim_{k\to\infty} \mathbf{Z}_k &= \mathbf{X}^{-1} \\ \mathbf{X}^2 &= \mathbf{A} \end{aligned} \tag{6.68}$$

The convergence condition is the worst-case scenario of the eigenvalues of $\mathbf{A}$. All the eigenvalues need to satisfy

$$|1 - \lambda_i^2| < 1, \forall i\{1, 2, \ldots, N\} \tag{6.69}$$

Or, more easily tested;

$$|\mathbf{I} - \mathbf{YZ}| = |\mathbf{I} - \mathbf{A}| < 1 \tag{6.70}$$

This principle is applicable to any appropriate matrix norm.

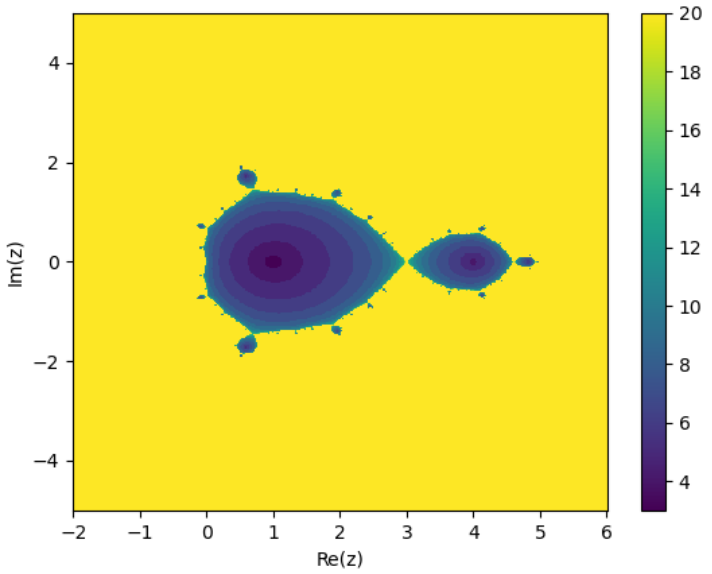


Figure 6.17: Number of iterations required for the eigenvalues of **PQ** to converge. Yellow region does not converge.

What is interesting is that, in the case where **A** only has eigenvalues with positive real parts, then we know that a value a exists such that $|\mathbf{I} - \frac{\mathbf{A}}{a}| < 1$. In the case of highly-absorptive media, we actually know that $i\mathbf{PQ}$ satisfies this condition. With proper normalization, this actually allows an iterative method for the square root with no matrix inversions. This is the closest we can get to a method to evaluate the square root without inversions.

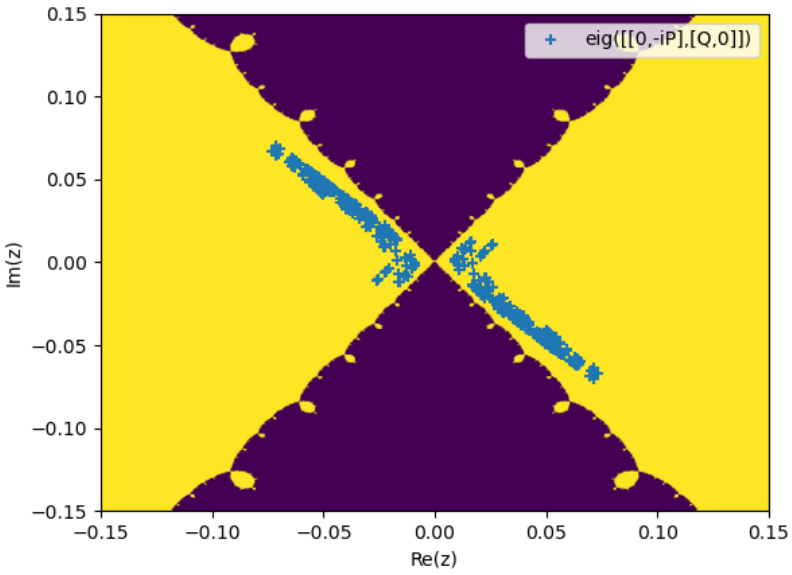


Figure 6.18: Spectrum of $\begin{bmatrix} \mathbf{0} & i\mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$, normalized and divided by 10, for $\epsilon_{res} = 1.5 + 0.5i$, $M = 5$,$N = 4$, superimposed on the convergence region of the Newton-Shultz iteration. Notice all the terms are squarely inside the region of convergence.

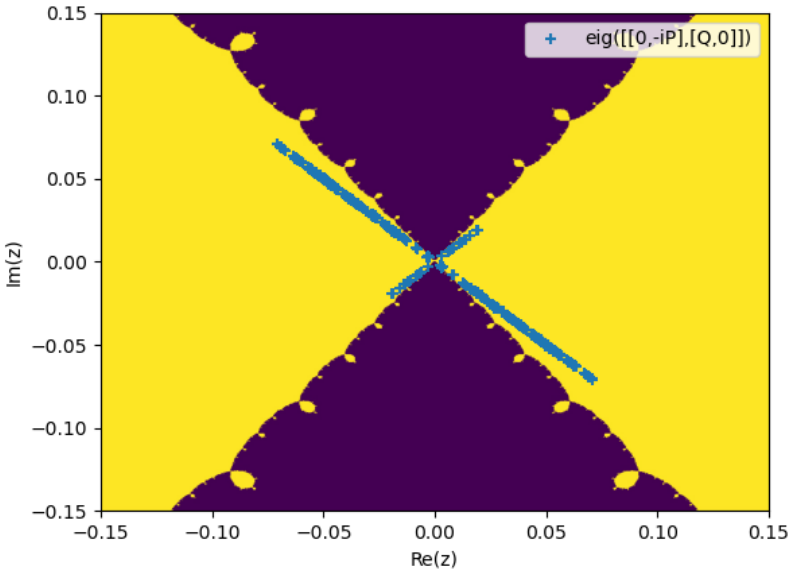


Figure 6.19: Spectrum of $\begin{bmatrix} \mathbf{0} & i\mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$, normalized and divided by 10, for $\epsilon_{res} = 1.5$, $M = 5$,$N = 4$, superimposed on the convergence region of the Newton-Shultz iteration. It is hard to discern, but the terms just barely fall inside the region of convergence.

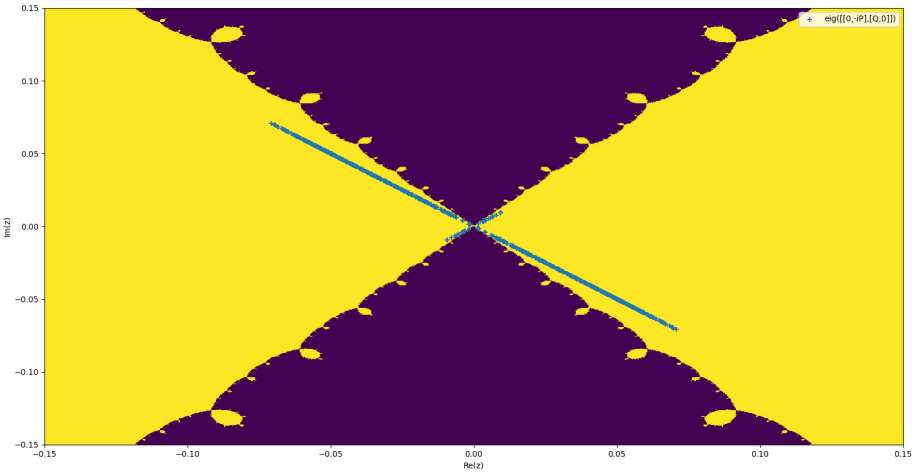


Figure 6.20: Spectrum of $\begin{bmatrix} \mathbf{0} & i\mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}$, normalized and divided by 10, for $\epsilon_{res} = 1.5$, $M = 10$,$N = 8$, superimposed on the Convergence region of the Newton-Shultz iteration. It is hard to discern, but only a few terms just barely fall outside the region of convergence, so this will not converge.

The fact that a situation exists where $i\mathbf{PQ}$ does not satisfy this condition should be a future field of research. This is because such eigenmodes are aphysical. It suggests the presence of eigenvalues inside the second and fourth quadrant of the complex plane, where the sign of the imaginary and real parts are not the same; such eigenmodes are neither purely causal nor purely anti-causal, causing the whole notion of coupled-waves to break down. The existence of these modes has up to now primarily been only a conceptual problem, but now presents a serious practical obstacle in the implementation

of a ”multiplication-friendly” algorithm (that is, an algorithm which is primarily based on matrix multiplications) for RCWA.[20]

6.5.4. Padé iteration

This finally leads us to the following family of scalar Padé approximants: [17]

$$z_{k+1} = \frac{(1-z_k)^p - (1+z_k)^p}{(1-z_k)^p + (1+z_k)^p} \tag{6.71}$$

From which we get the following class of Padé iterations: [21] [22]

$$\begin{aligned} \mathbf{Y}_0 &= \mathbf{A}, \mathbf{Z}_0 = \mathbf{I} \\ \mathbf{Y}_{k+1} &= \mathbf{Y}_k \left(\sum_{n=0}^{m} \binom{2m+1}{2n+1} (\mathbf{Z}_k\mathbf{Y}_k)^n\right)\left(\sum_{n=0}^{m} \binom{2m+1}{2n} (\mathbf{Z}_k\mathbf{Y}_k)^n\right)^{-1} \\ \mathbf{Z}_{k+1} &= \left(\sum_{n=0}^{m} \binom{2m+1}{2n+1} (\mathbf{Z}_k\mathbf{Y}_k)^n\right)\left(\sum_{n=0}^{m} \binom{2m+1}{2n} (\mathbf{Z}_k\mathbf{Y}_k)^n\right)^{-1}\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k &= \mathbf{X} \\ \lim_{k\to\infty} \mathbf{Z}_k &= \mathbf{X}^{-1} \\ \mathbf{X}^2 &= \mathbf{A} \end{aligned} \tag{6.72}$$

In these equations, $\binom{2m+1}{2n+1}$ and $\binom{2m+1}{2n}$ are binomial coefficients which evaluate to:

$$\binom{2m+1}{2n+1} = \frac{(2m+1)!}{(2n+1)!(2m-2n)!} \tag{6.73}$$

$$\binom{2m+1}{2n} = \frac{(2m+1)!}{(2n)!(2m-2n+1)!} \tag{6.74}$$

This method is globally convergent if the block matrix $\begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}$ has no eigenvalues on the imaginary number line in the complex plane ([17], theorem 4.2), which is satisfied if $\mathbf{A}$ has no eigenvalues on the negative real line. ([21], lemma 4.3)

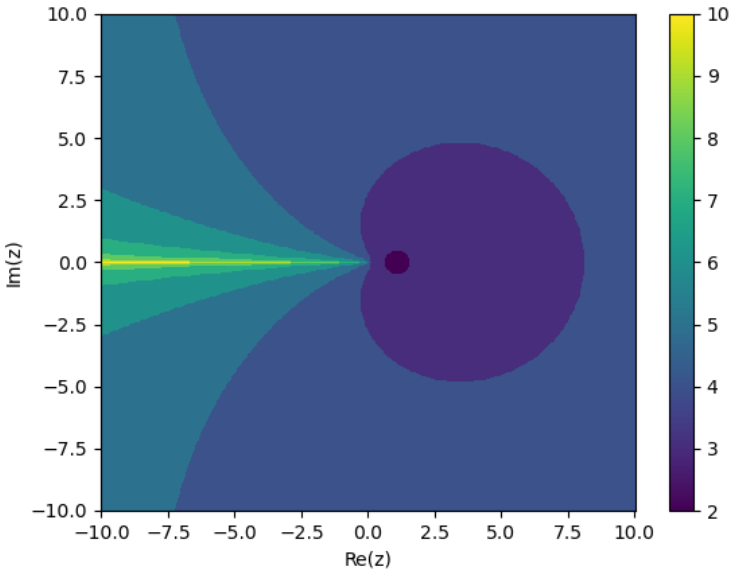


Figure 6.21: Number of iterations required for the eigenvalues of **PQ** to converge, depending on their value, using the $m = 1$ Padé iteration.

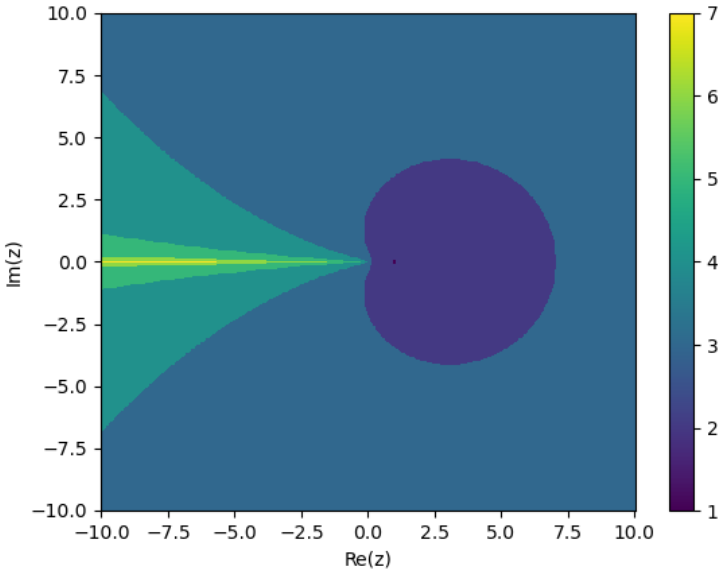


Figure 6.22: Number of iterations required for the eigenvalues of **PQ** to converge, depending on their value, using the $m = 2$ Padé iteration.

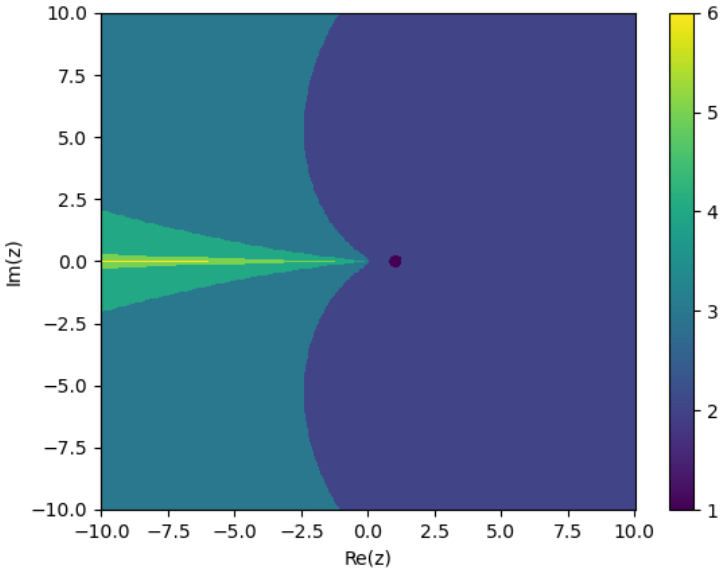


Figure 6.23: Number of iterations required for the eigenvalues of **PQ** to converge, depending on their value, using the $m = 3$ Padé iteration.

6.5.5. Alternate calculation

Another property of the matrix sign function is the following:

$$msgn(\begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{B} & \mathbf{0} \end{bmatrix}) = \begin{bmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^{-1} & \mathbf{0} \end{bmatrix} \tag{6.75}$$

where

$$\mathbf{C} = \mathbf{A}(\mathbf{BA})^{-1/2} \tag{6.76}$$

With this, we can actually calculate the following:

$$msgn(\begin{bmatrix} \mathbf{0} & \mathbf{Q} \\ \mathbf{P} & \mathbf{0} \end{bmatrix}) = \begin{bmatrix} \mathbf{0} & \mathbf{V} \\ \mathbf{V}^{-1} & \mathbf{0} \end{bmatrix} \tag{6.77}$$

where

$$\mathbf{V} = \mathbf{Q}(\mathbf{PQ})^{-1/2} \tag{6.78}$$

Where we can calculate the square root as follows:

$$\mathbf{V}^{-1}\mathbf{Q} = (\mathbf{Q}(\mathbf{PQ})^{-1/2})^{-1}\mathbf{Q} = ((\mathbf{PQ})^{-1/2})^{-1}\mathbf{Q}^{-1}\mathbf{Q} = (\mathbf{PQ})^{1/2} \tag{6.79}$$

Another method is the following:

$$\mathbf{PV} = \mathbf{PQ}(\mathbf{PQ})^{-1/2} = (\mathbf{PQ})^{1/2} \tag{6.80}$$

Calculating the matrix sign of $\begin{bmatrix} \mathbf{0} & \mathbf{Q} \\ \mathbf{P} & \mathbf{0} \end{bmatrix}$ and determining the square root indirectly has the benefit that no matrix inversion is required to calculate $\mathbf{V}^{-1}$. Since the matrix $\begin{bmatrix} \mathbf{0} & \mathbf{Q} \\ \mathbf{P} & \mathbf{0} \end{bmatrix}$ is ”similar” to $\begin{bmatrix} \mathbf{0} & \mathbf{PQ} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}$, it shares the same set of eigenvalues and, by extension, the same convergence behavior.

6.5.6. Matrix exponential

The matrix exponential can be found with the use of the Matrix Taylor series:

$$expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}}) = \sum_{n=0}^{\infty} \frac{(d(\mathbf{PQ})^{\frac{1}{2}})^n}{n!} \tag{6.81}$$

By applying normalization to $d(\mathbf{PQ})^{\frac{1}{2}}$ (for example, $(\Delta l(\mathbf{PQ})^{\frac{1}{2}})_{norm} = \frac{\Delta l(\mathbf{PQ})^{\frac{1}{2}}}{2^m}$), we can bound the absolute value of the eigenvalues to a specific region, for example, to be smaller than one. In such a case, the eigenvalues of each term are bounded by

$$|eig(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})^n}{2^m n!})| < \frac{1}{n!} \tag{6.82}$$

If n=19, then $\frac{1}{n!}$ evaluates to 8.8e-18, which is smaller than the rounding error for double float precision. Terms order higher than $n = 19$ have no contribution to the sum after rounding error. As such, we can approximate the infinite sum with a finite sum instead:

$$\sum_{n=0}^{\infty} \frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})^n}{2^m n!} \approx \sum_{n=0}^{19} \frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})^n}{2^m n!} \tag{6.83}$$

Next, we can use the functional

$$expm(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})}{2^{m-1}}) = expm(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})}{2^m})^2 \tag{6.84}$$

to reconstruct the matrix exponential by multiples of two, until we reach $expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}})$.

The choice of the size of the normalization factor 2^m is dependent on the largest eigenvalues of $(\mathbf{PQ})^{\frac{1}{2}}$, the absolute value of which we can find by approximating the matrix spectrum using the following identity:

$$\rho(\mathbf{A}) = \lim_{k\to\infty} ||(\mathbf{A})^k||^{1/k} \tag{6.85}$$

The method of evaluating which quickly has been outlined previously. [23] [24]

6.6. The radius of convergence

What we find is that, for many methods which attempt to model complex functions of matrices, there are strict restrictions to the spectrum, otherwise the method does not work. These generally all are in the form of $|\mathbf{I} - \mathbf{A}| < 1$, but why?

In essence, many of these functions have a break point in the complex plane at $z = 0$. The reason why comes down to the logarithm of a complex value having a break point at $z = 0$, and then a branch cut following this. Many polynomial orders (which are not positive integers) are determined from the natural logarithm as follows:

$$z^n = \exp(n * \ln(z)) \tag{6.86}$$

A Taylor series of the natural logarithm about 1 becomes the following:

$$ln(1 - z) = -\sum_{n=0}^{\infty} \frac{z^n}{n} \tag{6.87}$$

This series has a radius of convergence of 1. Thus, the condition for convergence becomes

$$|1 - z| < 1 \tag{6.88}$$

The matrix extension of any of these functions, then, also have the restriction:

$$\rho(\mathbf{I} - \mathbf{A}) < 1 \tag{6.89}$$

And, since any conservative matrix norm satisfies

$$|\mathbf{I} - \mathbf{A}| < \rho(\mathbf{I} - \mathbf{A}) \tag{6.90}$$

We get the condition

$$|\mathbf{I} - \mathbf{A}| < 1 \tag{6.91}$$

6.7. Summary

To summarise, we have the following alternate formulation:

$$\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} (\mathbf{PQ})^{\frac{1}{2}} & \mathbf{0} \\ \mathbf{0} & -(\mathbf{PQ})^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} & -\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} \end{bmatrix}^{-1} \tag{6.92}$$

This new formulation has the same structure as the previous eigendecomposition method, providing the following analogues:

$$\begin{aligned} \mathbf{W}' &= \mathbf{I} \\ \mathbf{V}' &= \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} \\ \mathbf{X}' &= expm(\Delta l(\mathbf{PQ})^{\frac{1}{2}}) \end{aligned} \tag{6.93}$$

In the case where $|\mathbf{I} - \mathbf{PQ}| \geq 1$ with no eigenvalues on the negative number line, we use the following method:

$$\begin{aligned} \mathbf{Y}_0 = \mathbf{Q}, \mathbf{Z}_0 = \mathbf{P} \\ \mathbf{Y}_{k+1} = \mathbf{Y}_k(\sum_{n=0}^{m} \binom{2m+1}{2n+1} (\mathbf{Z}_k\mathbf{Y}_k)^n)(\sum_{n=0}^{m} \binom{2m+1}{2n} (\mathbf{Z}_k\mathbf{Y}_k)^n)^{-1} \\ \mathbf{Z}_{k+1} = (\sum_{n=0}^{m} \binom{2m+1}{2n+1} (\mathbf{Z}_k\mathbf{Y}_k)^n)(\sum_{n=0}^{m} \binom{2m+1}{2n} (\mathbf{Z}_k\mathbf{Y}_k)^n)^{-1}\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k = \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} = \mathbf{V}' \\ \lim_{k\to\infty} \mathbf{Z}_k = (\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}})^{-1} = (\mathbf{PQ})^{\frac{1}{2}}(\mathbf{Q}^{-1}) = (\mathbf{V}')^{-1} \end{aligned} \tag{6.94}$$

If there is no convergence, we can ensure convergence by introducing a phase offset. In the case where $|\mathbf{I} - \mathbf{PQ}| < 1$, we use the following method:

$$\begin{aligned} \mathbf{Y}_0 = \mathbf{Q}, \mathbf{Z}_0 = \mathbf{P} \\ \mathbf{Y}_{k+1} = \frac{1}{2}\mathbf{Y}_k(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k) \\ \mathbf{Z}_{k+1} = \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k)\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k = \mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}} = \mathbf{V}' \\ \lim_{k\to\infty} \mathbf{Z}_k = (\mathbf{Q}(\mathbf{PQ})^{-\frac{1}{2}})^{-1} = (\mathbf{PQ})^{\frac{1}{2}}(\mathbf{Q}^{-1}) = (\mathbf{V}')^{-1} \end{aligned} \tag{6.95}$$

From this, we can find the square root as:

$$(\mathbf{PQ})^{\frac{1}{2}} = (\mathbf{V}')^{-1}\mathbf{Q} \tag{6.96}$$

We can find $\mathbf{X}'$ by use of the scaling-and-squaring algorithm:

- find smallest m such that $\rho(\Delta l(\mathbf{PQ})^{\frac{1}{2}}) < 2^m$
- evaluate

$$expm(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})}{2^m}) \approx \sum_{n=0}^{19} \frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})^n}{2^m n!} \tag{6.97}$$

- use the functional

$$expm(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})}{2^{m-1}}) = expm(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})}{2^m})^2 \tag{6.98}$$

  m times to arrive at

$$expm(\frac{(\Delta l(\mathbf{PQ})^{\frac{1}{2}})}{2^0}) = \mathbf{X}' \tag{6.99}$$

7

Faulty RCWA

RCWA, up until now, has required finding a similarity transform of the intermediate T-matrix. From a computational science standpoint, the primary benefit of this is that this minimizes the amount of rounding error while evaluating the intermediate S-matrix. From a physical standpoint, this formulation corresponds closely with causal and anticausal eigenmodes (also referred to as a ”causal basis”), and therefore ought to be numerically stable in as many model configurations as possible by closely adhering to first principles.

This chapter explores a new theoretical method dubbed ”Faulty RCWA”. This method still evaluates a T-matrix with a ”causal basis”, therefore yielding a well-behaved S-matrix, but does so without evaluating a similarity transform of any kind. As a result, this formulation introduces numerical rounding error that the other formulations don’t, and the error scales exponentially with the layer thickness. In exchange, this method avoids both the computational cost and the potential rounding error introduced with the similarity transform. As a result, this method might be both faster and more accurate in the case of sufficiently thin layers.

To quickly recap, the reason why we use the current formulation of implementing S-matrices (and T-matrices by extension) is twofold. Firstly, S-matrices being causal from a physical standpoint is useful for guaranteeing a well-posed transform from a purely mathematical standpoint. There are very few instances of perfectly constructive interference leading one of the fields to become zero, thereby eventually leading to a divide-by-zero error once one tries to evaluate an inversion.

Secondly, the eigendecomposition (and the alternate formulation explored in the previous chapter) lead to an internal ”propagation T-matrix” of the following form:

$$\begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}^{-1} \end{bmatrix}$$

This directly corresponds with the following S-matrix:

$$\begin{bmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{X} & \mathbf{0} \end{bmatrix}$$

Thus, to evaluate the propagation S-matrix, one only needs to evaluate $\mathbf{X}$. This skips the need to calculate and incorporate the very big values of $\mathbf{X}^{-1}$.To understand why this is useful, here we demonstrate another implementation which skips finding the forward and backward modes altogether. In essence, it directly calculates

$$expm\left(\Delta l \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}\right) \tag{7.1}$$

With the scaling-and-squaring algorithm outlined before. Then, it transforms this to a T-matrix relating

the eigenmodes of gap media with the following:

$$\mathbf{T} = \begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix}^{-1} expm\left(\Delta l \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}\right) \begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix} \tag{7.2}$$

This is then finally transformed to an S-matrix with a principal pivot with the following:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} = \begin{bmatrix} -\mathbf{T}_{22}^{-1}\mathbf{T}_{21} & \mathbf{T}_{22}^{-1} \\ \mathbf{T}_{11} - \mathbf{T}_{12}\mathbf{T}_{22}^{-1}\mathbf{T}_{21} & \mathbf{T}_{12}\mathbf{T}_{22}^{-1} \end{bmatrix} \tag{7.3}$$

The drawback is that equation 7.2 lets the growing evanescent modes (which are very big floating point numbers) and the shrinking evanescent modes (which are very small floating point numbers) mix. When adding or subtracting two floating point numbers, a numerical error called ”floating point error” is introduced. The size of this floating point error is proportional to the largest floating point. In the case of double precision, the floating point error is approximately the largest floating point, times 10^{-16}. With this, we can describe the floating point rounding error as follows:

$$\epsilon_{\text{rounding}} \approx 10^{-16}\rho(\mathbf{X}^{-1}) = 10^{-16}\rho(expm(\Delta l(\mathbf{PQ}^{1/2})) \tag{7.4}$$

Here, $\rho(\mathbf{X}^{-1})$ is the spectral radius of the growing propagation matrix; it describes the extent to which the largest evanescent mode grows as it propagates through the layer.

While this seems to have only minimal impact, if the magnitude of the growing evanescent modes reaches approximately 10^{16}, and the regular propagating modes stay at around the normalized magnitude of 1, then the rounding error of adding the growing evanescent modes becomes larger than the magnitude of the regular propagating modes. In effect, the information you want to have is rounded away, and what remains is effectively rounding error noise. The S-matrix implementations previously discussed do all this extra work to effectively add guardrails to the numerical method and thereby guarantee that this never happens.

However, this begs the question; do we always needs these guardrails? We can get a conservative estimate using equation 7.4. If we have a certain error tolerance (say, errors below 10^{-12} are OK), we can find a criteria for the layer thickness doing the following back-of-the-envelope calculations:

$$\begin{aligned} \rho(expm(\Delta l(\mathbf{PQ}^{1/2})) &\leq e^{\rho(\Delta l(\mathbf{PQ}^{1/2}))} \\ 10^{-16}e^{\rho(\Delta l(\mathbf{PQ}^{1/2}))} &< 10^{-12} \\ e^{\rho(\Delta l(\mathbf{PQ}^{1/2}))} &< 10^{4} \\ \rho(\Delta l(\mathbf{PQ}^{1/2})) &< ln(10^4) \approx 9.21 \\ \rho(\Delta l(\mathbf{PQ}^{1/2})) &= \Delta l * \rho((\mathbf{PQ}^{1/2})) = \Delta l * \sqrt{\rho(\mathbf{PQ})} \\ \Delta l &< \frac{9.21}{\sqrt{\rho(\mathbf{PQ})}} \end{aligned} \tag{7.5}$$

Thus, by keeping the layer thickness of a layer smaller than a certain threshold, we can ensure that the error falls within our error tolerance even without evaluating either a matrix square root or an eigendecomposition. This layer thickness depends on three things;

- The size of the floating point errors introduced;
- The spectral radius of **PQ**;
- Our designated error tolerance.

This puts serious limitations on how thick our layers can be. However, there are ways around this; construct a thick layer out of a stack of thin layers.

7.1. Auto-concatenation

Auto-concatenation is largely inspired by the following illustration on a ”doubling procedure”: [7]

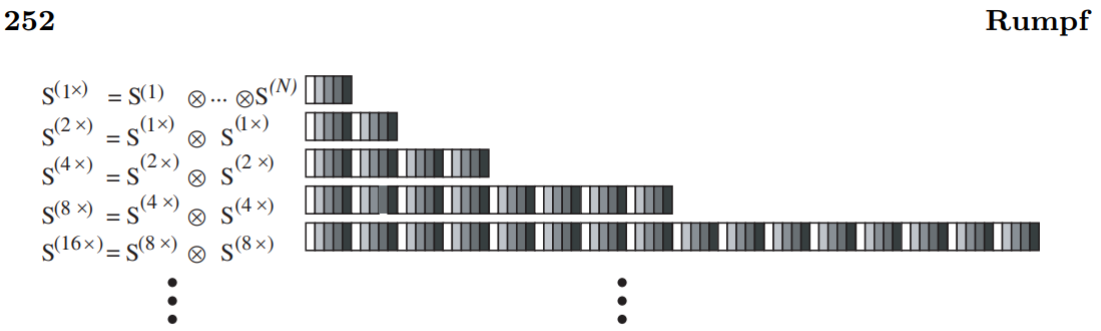


Figure 5. Doubling procedure.

Figure 7.1: Doubling Procedure, as found in paper [7].

This procedure allows one to minimise the concatenations needed to evaluate the S-matrix of a series of layers. We can also apply this concept in reverse; an S-matrix for a layer can be treated as a concatenation of two sublayers with half the thickness, in what can be considered a ”halving procedure”:

$$\begin{aligned} \mathbf{S}^{(\Delta l)} &= \mathbf{S}^{(\Delta l/2)} \otimes \mathbf{S}^{(\Delta l/2)} \\ \mathbf{S}^{(\Delta l/2)} &= \mathbf{S}^{(\Delta l/4)} \otimes \mathbf{S}^{(\Delta l/4)} \\ \mathbf{S}^{(\Delta l/4)} &= \mathbf{S}^{(\Delta l/8)} \otimes \mathbf{S}^{(\Delta l/8)} \\ &\vdots \end{aligned} \tag{7.6}$$

In essence, we are replacing an intermediate layer with two intermediate layers of half the layer thickness. As each of these halves are identical, they yield identical S-matrices, so we only need to store information on one half. Since these half-layers can accurately be described as intermediate layers unto their own, according to equation 3.66, they must also be symmetric. Because of this, we know that:

$$\begin{aligned} \mathbf{S}_{11}^{(\Delta l/2)} &= \mathbf{S}_{22}^{(\Delta l/2)} \\ \mathbf{S}_{12}^{(\Delta l/2)} &= \mathbf{S}_{21}^{(\Delta l/2)} \end{aligned} \tag{7.7}$$

Because of this, we can define the whole system with only two blocks, $\mathbf{S}_{11}^{(\Delta l/2)}$ and $\mathbf{S}^{(\Delta l/2)}$.

If these half layers have a layer thickness small enough to satisfy our error tolerance, then we stop here. If not, we can further halve the halves, and evaluate the quarter layer directly. Again, we can define the whole system with only two blocks, $\mathbf{S}_{11}^{(\Delta l/4)}$ and $\mathbf{S}_{12}^{(\Delta l/4)}$. In effect, this halving procedure is repeated until a sub-layer thickness sufficiently small is reached. For this sub-layer, the S-matrix terms can be calculated directly. For further illustrations, we assume that this appropriate thickness is reached for $\mathbf{S}_{12}^{(\Delta l/8)}$.

With these sub-layer blocks, we can find the S-paramter matrix recursively with the Redheffer star product:

$$\begin{aligned} \mathbf{S}^{(\Delta l/4)} &= \mathbf{S}^{(\Delta l/8)} \otimes \mathbf{S}^{(\Delta l/8)} \\ &= \begin{bmatrix} \mathbf{S}_{11}^{(\Delta l/8)} + \mathbf{S}_{12}^{(\Delta l/8)}[\mathbf{I} - \mathbf{S}_{11}^{(\Delta l/8)}\mathbf{S}_{22}^{(\Delta l/8)}]^{-1}\mathbf{S}_{11}^{(\Delta l/8)}\mathbf{S}_{21}^{(\Delta l/8)} & \mathbf{S}_{12}^{(\Delta l/8)}[\mathbf{I} - \mathbf{S}_{11}^{(\Delta l/8)}\mathbf{S}_{22}^{(\Delta l/8)}]^{-1}\mathbf{S}_{12}^{(\Delta l/8)} \\ \mathbf{S}_{21}^{(\Delta l/8)}[\mathbf{I} - \mathbf{S}_{22}^{(\Delta l/8)}\mathbf{S}_{11}^{(\Delta l/8)}]^{-1}\mathbf{S}_{21}^{(\Delta l/8)} & \mathbf{S}_{22}^{(\Delta l/8)} + \mathbf{S}_{21}^{(\Delta l/8)}[\mathbf{I} - \mathbf{S}_{22}^{(\Delta l/8)}\mathbf{S}_{11}^{(\Delta l/8)}]^{-1}\mathbf{S}_{22}^{(\Delta l/8)}\mathbf{S}_{12}^{(\Delta l/8)} \end{bmatrix} \end{aligned} \tag{7.8}$$

There are multiple degrees of symmetry here:

- The two layers being concatenated have identical S-parameter matrices;
- The S-parameter matrices are block-symmetric;
- Combining these two facts, the resulting Redheffer star product is itself block-symmetric as well.

As a result, the equation to determine the S-submatrices simplify to the following:

$$\begin{aligned}
\mathbf{S}_{11}^{(\Delta l/4)} &= \mathbf{S}_{11}^{(\Delta l/8)} + \mathbf{S}_{12}^{(\Delta l/8)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/8)})^2]^{-1}\mathbf{S}_{11}^{(\Delta l/8)}\mathbf{S}_{12}^{(\Delta l/8)} \\
\mathbf{S}_{12}^{(\Delta l/4)} &= \mathbf{S}_{12}^{(\Delta l/8)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/8)})^2]^{-1}\mathbf{S}_{12}^{(\Delta l/8)} \\
\mathbf{S}_{21}^{(\Delta l/4)} &= \mathbf{S}_{12}^{(\Delta l/4)} \\
\mathbf{S}_{22}^{(\Delta l/4)} &= \mathbf{S}_{11}^{(\Delta l/4)}
\end{aligned} \tag{7.9}$$

Again, this means that half of the blocks are unnecessary information, simplifying the equations to:

$$\begin{aligned}
\mathbf{S}_{11}^{(\Delta l/4)} &= \mathbf{S}_{11}^{(\Delta l/8)} + \mathbf{S}_{12}^{(\Delta l/8)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/8)})^2]^{-1}\mathbf{S}_{11}^{(\Delta l/8)}\mathbf{S}_{12}^{(\Delta l/8)} \\
\mathbf{S}_{12}^{(\Delta l/4)} &= \mathbf{S}_{12}^{(\Delta l/8)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/8)})^2]^{-1}\mathbf{S}_{12}^{(\Delta l/8)}
\end{aligned} \tag{7.10}$$

Now, for the half-layer thickness S-parameter matrix, we can use the same calculations:

$$\begin{aligned}
\mathbf{S}_{11}^{(\Delta l/2)} &= \mathbf{S}_{11}^{(\Delta l/4)} + \mathbf{S}_{12}^{(\Delta l/4)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/4)})^2]^{-1}\mathbf{S}_{11}^{(\Delta l/4)}\mathbf{S}_{12}^{(\Delta l/4)} \\
\mathbf{S}_{12}^{(\Delta l/2)} &= \mathbf{S}_{12}^{(\Delta l/4)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/4)})^2]^{-1}\mathbf{S}_{12}^{(\Delta l/4)}
\end{aligned} \tag{7.11}$$

And the full-layer thickness S-parameter matrix:

$$\begin{aligned}
\mathbf{S}_{11}^{(\Delta l)} &= \mathbf{S}_{11}^{(\Delta l/2)} + \mathbf{S}_{12}^{(\Delta l/2)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/2)})^2]^{-1}\mathbf{S}_{11}^{(\Delta l/2)}\mathbf{S}_{12}^{(\Delta l/2)} \\
\mathbf{S}_{12}^{(\Delta l)} &= \mathbf{S}_{12}^{(\Delta l/2)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/2)})^2]^{-1}\mathbf{S}_{12}^{(\Delta l/2)}
\end{aligned} \tag{7.12}$$

Which we then finally assemble into the full S-matrix formulation:

$$\mathbf{S}^{(\Delta l)} = \begin{bmatrix} \mathbf{S}_{11}^{(\Delta l)} & \mathbf{S}_{12}^{(\Delta l)} \\ \mathbf{S}_{12}^{(\Delta l)} & \mathbf{S}_{11}^{(\Delta l)} \end{bmatrix} \tag{7.13}$$

7.2. General algorithm

- Choose a certain positive integer k, so that the T-matrix is ”accurate enough”.
- Evaluate

$$\mathbf{T}^{(\Delta l/2^k)} = \begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix}^{-1} expm\left(\frac{\Delta l}{2^k}\begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{Q} & \mathbf{0} \end{bmatrix}\right)\begin{bmatrix} \mathbf{W}_g & \mathbf{W}_g \\ \mathbf{V}_g & -\mathbf{V}_g \end{bmatrix} \tag{7.14}$$

- Evaluate the symmetric S-matrix terms

$$\begin{aligned}
\mathbf{S}_{12}^{(\Delta l/2^k)} &= (\mathbf{T}_{22}^{(\Delta l/2^k)})^{-1} \\
\mathbf{S}_{11}^{(\Delta l/2^k)} &= \mathbf{T}_{12}^{(\Delta l/2^k)}\mathbf{S}_{1,2}^{(\Delta l/2^k)}
\end{aligned}$$

- Use the following doubling procedure

$$\begin{aligned}
\mathbf{S}_{11}^{(\Delta l/2^{k-1})} &= \mathbf{S}_{11}^{(\Delta l/2^k)} + \mathbf{S}_{12}^{(\Delta l/2^k)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/2^k)})^2]^{-1}\mathbf{S}_{11}^{(\Delta l/2^k)}\mathbf{S}_{12}^{(\Delta l/2^k)} \\
\mathbf{S}_{12}^{(\Delta l/2^{k-1})} &= \mathbf{S}_{12}^{(\Delta l/2^k)}[\mathbf{I} - (\mathbf{S}_{11}^{(\Delta l/2^k)})^2]^{-1}\mathbf{S}_{12}^{(\Delta l/2^k)}
\end{aligned}$$

  k times to get $\mathbf{S}_{11}^{(\Delta l)}$ and $\mathbf{S}_{12}^{(\Delta l)}$.
- Arrange these into a symmetric S-matrix shape.

This is conceptually the same scheme as the scaling-and-squaring outlined for the matrix exponent, but for entire S-matrices rather than the propagation factors of T-matrices.

7.3. Utility

For a ”sufficiently thin” layer, this method calculates the S-parameter matrix using only a matrix exponential and a principal pivotal transform. This principal pivotal transform can be further simplified to the inverse of one block, and one matrix multiplication. As a result, theoretically this method is probably faster than either the eigendecomposition or the square root method.

For thicker layers, the auto-concatenation adds extra computational overhead which is relatively smaller for the square root method and effectively absent for the eigendecomposition. As such this method is probably not optimal for thicker intermediate layers.

While not rigorously tested, we found that calculating the intermediate S-matrix using faulty RCWA, even for relatively thick layers, yielded an intermediate S-matrix whose entries corresponded more closely to an intermediate S-matrix calculated using the square root method than either method corresponded to an intermediate S-matrix obtained via eigendecomposition. As such, by avoiding a similarity transform, Faulty RCWA might be more accurate than the eigendecomposition method, making the term somewhat a misnomer.

Because of this method’s performance for thin intermediate layers, this method seems most promising for application with staircasing, which approximates z-dependent layers by creating a series of thin, z-independent layers. Such situations typically occur in profilometry of crucial dimension reconstruction for three-dimensional structures.

Additionally, this method can also be used in conjunction with the enhanced T-matrix method. The enhanced T-matrix method diverges from the regular T-matrix method in a couple of ways:

- it works back-to-front, rather than front-to-back, which allows it to express all waves as a linear relation between the intermediate forward modes.
- it normalized all the wave sizes with respect to the intermediate forward modes to keep everything bounded.

Both of these necessitate the finding of a forward-backward basis. However, just like with the S-matrix formulation, this can be achieved with the use of the forward and backward modes of the gap medium, rather than the forward and backward modes of whatever medium one finds itself in at that point in time.

8

Results

There have been two general sets of geometrical models, accompanied with electromagnetic properties n and k of each material and accompanying reflectance & transmittance data, provided to us. The In both of these cases, the reflectance & transmittance data is data calculated by an external RCWA scheme, which was assumed to have been programmed properly. Replication of the datasets thus served to validate our implementation of RCWA.

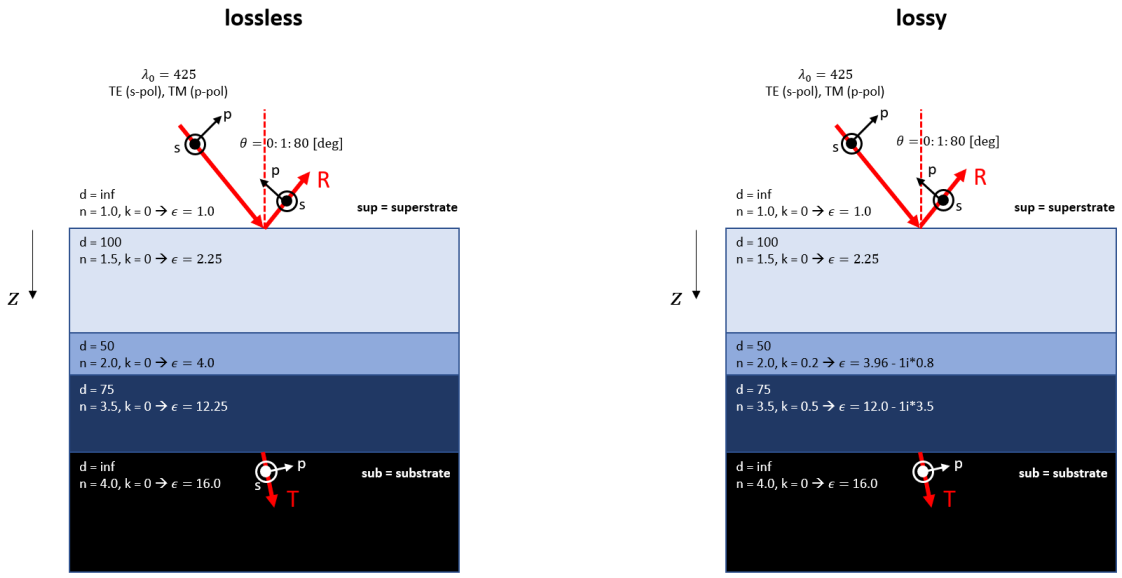
For both cases, we have demonstrated the ability to replicate the behavior of the external RCWA scheme. Additionally, for both cases, we have demonstrated that, in the case of perturbation of at least one variable, our RCWA schemes demonstrate the ability to retrieve the correct value of the perturbed variable.

Finally, there is some information on the performance of the square root method, relative to the performance of the eigendecomposition method.

- Information detailing the

8.1. Case 1

The first case is explained by the following illustration:



$$DER = |R|^2$$

$$DET = |T|^2 \frac{k_z^{sub}}{k_z^{sup}}$$

$$\epsilon = (n - 1i * k)^2$$

Figure 8.1: RCWA configuration, one-dimensional. Reflectance denoted with DER, transmittance denoted with DET.

Here, DER refers to the reflectance, which is the fraction of the intensity which is reflected. DET refers to the transmittance, which is the fraction of the intensity which is reflected. This is the nomenclature of the provided case; later on, the reflectance will be denoted with R, and transmittance with T. With the following solution:

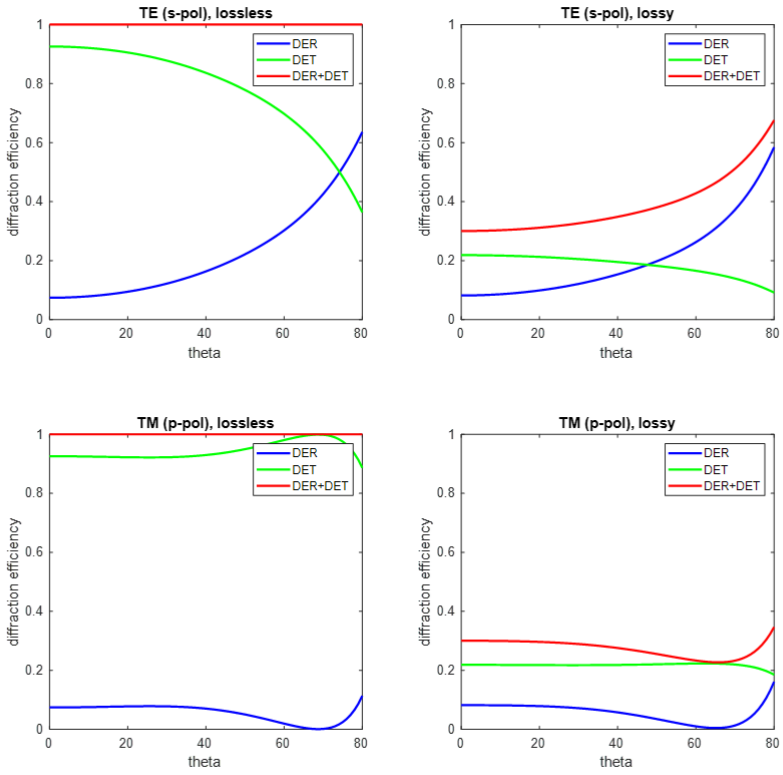


Figure 8.2: RCWA results, one-dimensional.

Owing to the fact that these layers were homogeneous, no scattering to other diffraction orders occurs. As a result, the overall system could be accurately modelled while ignoring any diffraction order other than zero; this is achieved by defining $M = N = 0$. In such a case, the **P** and **Q** matrices simplify to the following:

$$\begin{aligned} \mathbf{P} &= \frac{1}{\epsilon_r} \begin{bmatrix} k_{x,inc}k_{y,inc} & \mu_r\epsilon_r - k_{x,inc}^2 \\ k_{y,inc}^2 - \mu_r\epsilon_r & k_{x,inc}k_{y,inc} \end{bmatrix} \\ \mathbf{Q} &= \frac{1}{\mu_r} \begin{bmatrix} k_{x,inc}k_{y,inc} & \mu_r\epsilon_r - k_{x,inc}^2 \\ k_{y,inc}^2 - \mu_r\epsilon_r & k_{x,inc}k_{y,inc} \end{bmatrix} \end{aligned} \tag{8.1}$$

As such, replicating these results was the initial step of our RCWA implementation.

8.1.1. Validation of RCWA scheme

Below are our replications of figure 8.2.

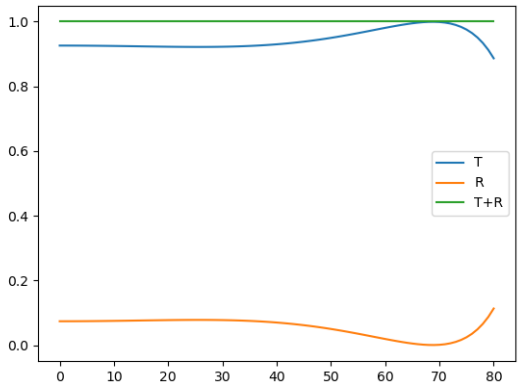


Figure 8.3: RCWA results, Lossless, p-pol. Reflectance denoted with R, transmittance denoted with T.

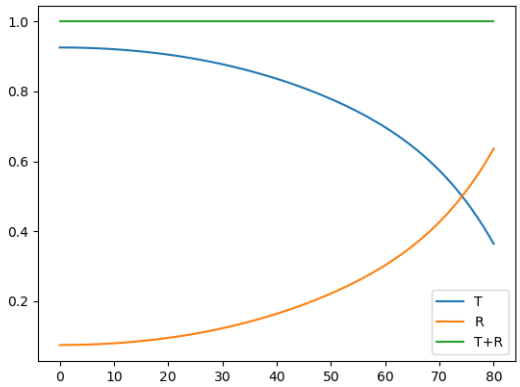


Figure 8.4: RCWA results, Lossless, s-pol. Reflectance denoted with R, transmittance denoted with T.

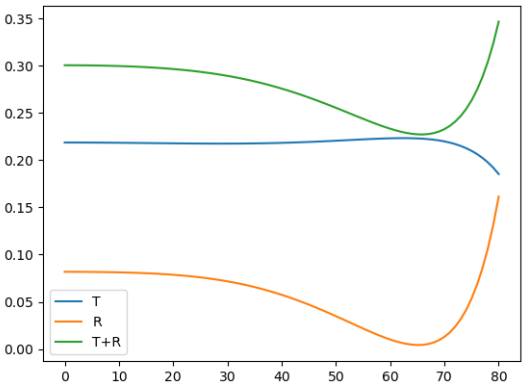


Figure 8.5: RCWA results, Lossy, p-pol. Reflectance denoted with R, transmittance denoted with T.

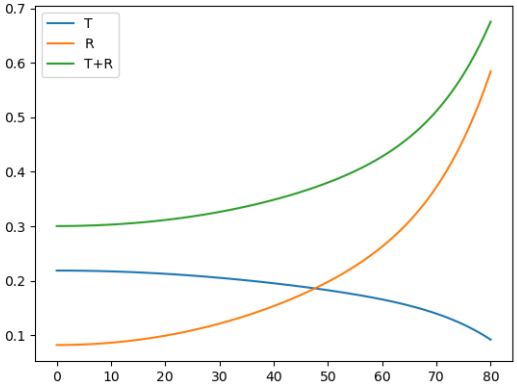


Figure 8.6: RCWA results, Lossy, s-pol. Reflectance denoted with R, transmittance denoted with T.

8.1.2. Perturbation and retrieval of model

For both of these cases, a perturbation was introducted by altering the thickness of intermediate layer 2 from 50 to 40. the thickness of each layer was determined to be ”trainable”; no other variable was determined to be trainable.

The training data consisted of the s-polarization diffraction curves and the p-polarization diffraction curves combined.

In ten epochs, the lossless case did not converge perfectly. It stagnated with a maximum loss around $1.2e-07$. This was most likely oscillatory behavior. Still, it does demonstrate that it reduced the value of the loss function.

Epoch	Loss
1	1.1037e-04
2	1.6722e-07
3	1.2064e-07
4	1.3039e-07
5	1.2054e-07
6	1.2675e-07
7	1.2205e-07
8	1.2238e-06
9	3.5198e-07
10	2.2421e-07

Variable	Original	Perturbed	Retrieved
d_1	0.1000	0.1000	0.1081
d_2	0.0500	0.0400	0.0455
d_3	0.0750	0.0750	0.0795

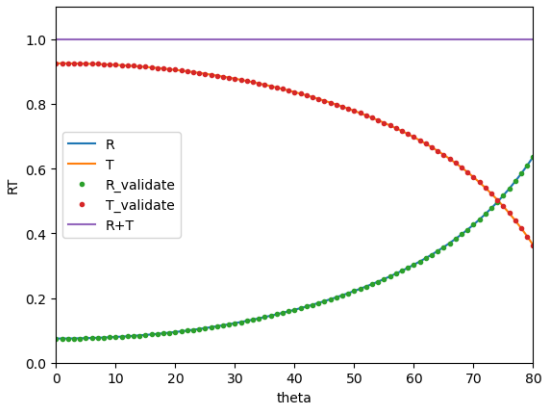


Figure 8.7: RCWA results, Lossless, p-pol. Values calculated with retrieved depth values are superimposed. Reflectance denoted with R, transmittance denoted with T.

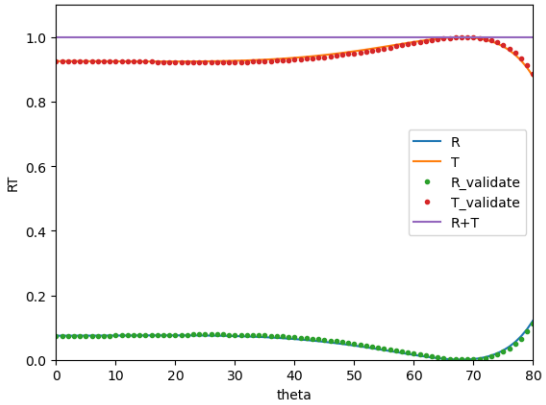


Figure 8.8: RCWA results, Lossless, s-pol. Values calculated with retrieved depth values are superimposed. Reflectance denoted with R, transmittance denoted with T.

The lossy case converged to within rounding error ($L = 5.3813e - 17$) by epoch 6.

Epoch	Loss
1	9.1381e-05
2	5.5145e-08
3	2.6718e-11
4	1.4731e-14
5	1.1017e-16
6	5.3813e-17
7	6.5085e-17
8	5.6538e-17
9	9.9442e-17
10	6.2881e-17

Variable	Original	Perturbed	Retrieved
d_1	0.1000	0.1000	0.1000
d_2	0.0500	0.0400	0.0500
d_3	0.0750	0.0750	0.0750

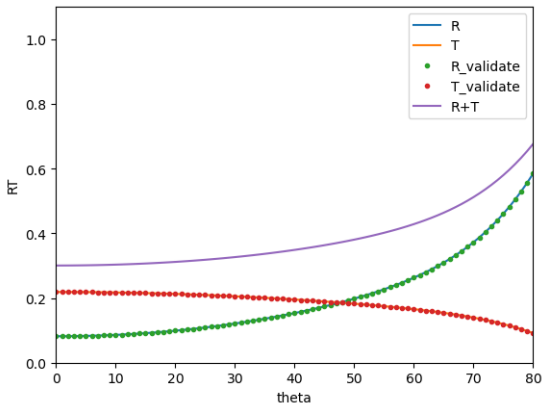


Figure 8.9: RCWA results, Lossy, p-pol. Values calculated with retrieved depth values are superimposed. Reflectance denoted with R, transmittance denoted with T.

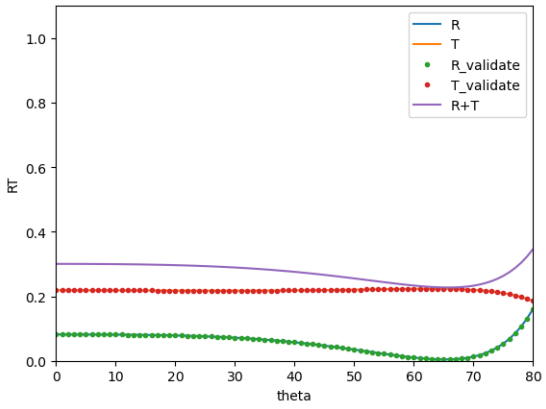


Figure 8.10: RCWA results, Lossy, s-pol. Values calculated with retrieved depth values are superimposed. Reflectance denoted with R, transmittance denoted with T.

8.2. Case 2

The second set of cases had only one structure configuration.

For all materials present in the device model, the relative permeability was was $\mu = 1$.

The structure was constructed with only one intermediate layer. This intermediate layer had a thickness of $\Delta l = 100$. The unit cell had periodicities/pitches in the x-direction of $L_x = 600$, and in the y-direction of $L_y = 500$. The unit cell ranges in the x-direction from -300 to 300, and in the y-direction from -250 to 250.

Within the unit cell, there is a rectangular block consisting of resist. The resist block ranges in the x-direction from -150 to 150, and in the y-direction from -125 to 125. It has a relative permittivity of $\epsilon_{res} = 2.25$. All other volume in the unit cell was occupied by vacuum, which has a relative permittivity of $\epsilon_{vac} = 1$.

The superstrate has a relative permittivity of $\epsilon_{sup} = 1$, and the substrate a relative permittivity of $\epsilon_{sub} = 16$.

In every case, the system was illuminated with an s-polarization plane wave, with azimuthal angle $\phi = 30°$. The incident light was monochromatic, with wavelength $\lambda = 425$.

As not all layers were homogeneous, scattering to other diffraction orders occurs. As a result, the calculated diffraction order intensities are impacted by the total considered harmonics. In total, validation data was provided for the configurations $[M, N] = [[5, 4], [10, 8], [15, 12], [20, 16], [25, 20]]$. Finally, the

incident light either had normal incidence, with angle relative to z-axis of $\theta = 0°$, or conical incidence, with angle relative to z-axis of $\theta = 30°$.

This gave us a total of 10 configurations which we could try out, but in reality, the computational power available to us limited us to $[M, N] = [[5, 4], [10, 8], [15, 12]$; anything else took too long to evaluate for us to get insights on a meaningful schedule.

Implementation of RCWA in TensorFlow consisted of roughly two steps. Firstly, we implemented the steps in a purely functional method. This required, among other things, functions which do the following:

- Evaluate θ and ϕ into appropriate $k_{x,inc}$ and $k_{y,inc}$;
- Evaluate the derivative operators and all flavors of convolution operators;
- Arrange these into **P** and **Q** matrices;
- Use these terms to evaluate **W**, **V** and **X** (or their analogs);
- Evaluate $\mathbf{W}_g$ and $\mathbf{V}_g$;
- Evaluate the S-matrices of every layer, which include the superstrate, substrate, and every intermediate layer;
- Use the Redheffer star product to evaluate the overall S matrix;
- Extract the R and T values from the overall S matrix to get the results.

Secondly, we ported over these functions to an object-oriented framework; this is needed because Keras, the TensorFlow framework which actually does the backpropagation, treats the layers of a neural network as objects, which have their own functionality based on internal parameters. It is these internal parameters which are tweaked to get the desired in-out relation just right. In our case, these layers include primarily the actual layers of the model, where each outputs its own intermediate S-parameter matrix. However, they also include the functions which concatenate the layers, create the incident field, and measure the resulting R and T values.

Because of automatic differentiation, as explained in chapter 4, the inner functionality of these layers primarily consists of the following:

- Save the tuneable parameters internally during object initialization;
- Call the previously-formulated functions, where some of the inputs to the fuctions are inputs to the layer.

The layers themselves have, in effect, the same functionality as the functions they call, with the exception that the Keras layers provide the backpropagation functionality. Since this functionality is not strictly needed during implementation (there is no point providing tuning to a model which can not get the result right in the first place), and the layer objects presented an additional level of abstraction, so implementation of the backpropagation functionality was left aside for a next stage, after proper functioning code for RCWA results was implemented.

8.2.1. Validation of RCWA scheme

Below we have the calculated R and T for conical incidence $\theta = [0°, 30°]$

- $\theta = [0°, 30°]$;
- $[M, N] = [[5, 4], [10, 8], [15, 12]]$

This is with an intermediate layer modelled as a rectilinear grid. For these cases, we have demonstrated that each ”method” implemented lands the same results printed. Furthermore, all these results replicate the provided data for the same structure, down to the accuracy available when printing. For $M = 15$, $N = 12$ conical incidence, there was discrepancy in the digits beyond rounding error in 42 of the 87 cases; the largest relative error was ≈1.2e-6, while the largest absolute error was 2e-7.

In these cases, the only non-agreeing value was the external dataset. It was determined that this was due to an error in Matlab itself; if the file containing the R-values and T-values was imported via Python, the discrepancies were found to be much smaller.

conical	R			
index	provided	Faulty RCWA	Square root Newton method	Eigendecomposition method
388	2.190807E-01	2.19080710E-01	2.19080710E-01	2.19080710E-01
389	1.966798E-02	1.96679809E-02	1.96679809E-02	1.96679809E-02
413	2.298410E-02	2.29841006E-02	2.29841006E-02	2.29841006E-02
414	1.032329E-02	1.03232945E-02	1.03232945E-02	1.03232945E-02

From this, we validated whether the object-oriented framework duplicated the behavior from the functional implementation. Once this was done, we evaluated whether the object-oriented framework could perform parameter retrieval in the case of perturbation.

8.3. Model for case 2

The model itself has effectively the same internal mechanisms as the test script. It is configured as follows:

- It calculates the layer as a rectilinear grid;
- It evaluates the S-matrix using the matrix square root method;

It calculates the square root using the Padé iterative method, with Padé order $m = 3$ set as a default. This default was chosen as we determined through trial-and-error that it was approximately the fastest Padé iteration for the provided hardware.

Unless otherwise specified, the total harmonics considered in the x-direction is $M = 15$ and the total harmonics considered in the y-direction is $N = 12$.

All following tests were performed on an NVIDIA RTX A6000. A link to the datasheet of the GPU can be found in the appendix.

8.3.1. Validation

For the conical incidence case, the found R-values are illustrated in grid form:

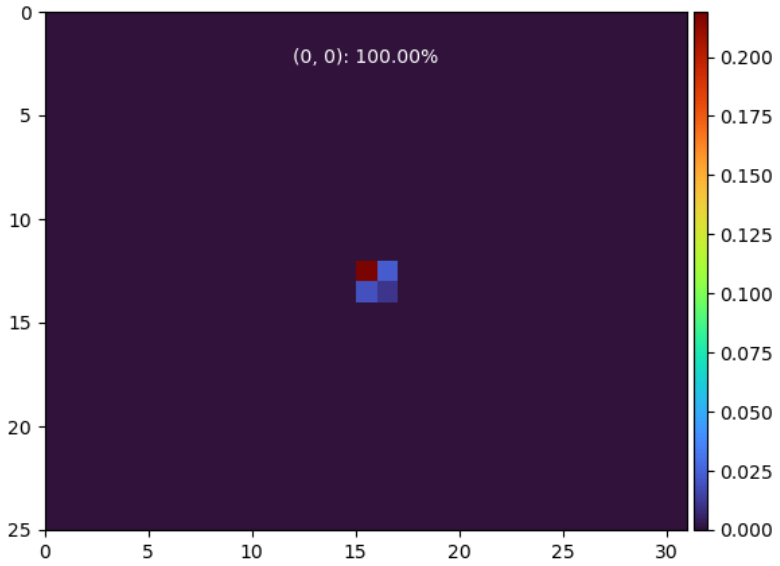


Figure 8.11: R-values found for conical incidence.

conical index	T provided	Faulty RCWA	Square root Newton method	Eigendecomposition method
263	7.501651E-07	7.50165280E-07	7.50165280E-07	7.50165280E-07
286	5.428520E-08	5.42851972E-08	5.42851972E-08	5.42851972E-08
287	1.799077E-06	1.79907711E-06	1.79907711E-06	1.79907711E-06
288	4.842463E-06	4.84246312E-06	4.84246312E-06	4.84246312E-06
289	2.309399E-06	2.30939869E-06	2.30939869E-06	2.30939869E-06
290	6.262498E-08	6.26249812E-08	6.26249812E-08	6.26249812E-08
291	8.496016E-09	8.49602177E-09	8.49602177E-09	8.49602177E-09
310	3.801270E-07	3.80126991E-07	3.80126991E-07	3.80126991E-07
311	8.174938E-07	8.17493781E-07	8.17493781E-07	8.17493781E-07
312	2.477466E-05	2.47746650E-05	2.47746650E-05	2.47746650E-05
313	6.659268E-05	6.65926791E-05	6.65926791E-05	6.65926791E-05
314	3.298786E-05	3.29878609E-05	3.29878609E-05	3.29878609E-05
315	1.218751E-06	1.21875125E-06	1.21875125E-06	1.21875125E-06
316	1.121555E-06	1.12155518E-06	1.12155518E-06	1.12155518E-06
335	1.814619E-07	1.81461938E-07	1.81461938E-07	1.81461938E-07
336	1.572584E-06	1.57258375E-06	1.57258375E-06	1.57258375E-06
337	3.278249E-05	3.27824895E-05	3.27824895E-05	3.27824895E-05
338	1.238228E-04	1.23822814E-04	1.23822814E-04	1.23822814E-04
339	7.750652E-05	7.75065247E-05	7.75065247E-05	7.75065247E-05
340	4.593149E-06	4.59314879E-06	4.59314879E-06	4.59314879E-06
341	8.881200E-07	8.88120032E-07	8.88120032E-07	8.88120032E-07
342	1.389876E-07	1.38987651E-07	1.38987651E-07	1.38987651E-07
359	2.061861E-06	2.06186085E-06	2.06186085E-06	2.06186085E-06
360	1.692031E-05	1.69203121E-05	1.69203121E-05	1.69203121E-05
361	6.736331E-05	6.73633132E-05	6.73633132E-05	6.73633132E-05
362	2.381719E-03	2.38171923E-03	2.38171923E-03	2.38171923E-03
363	1.204216E-02	1.20421628E-02	1.20421628E-02	1.20421628E-02
364	5.464233E-03	5.46423333E-03	5.46423333E-03	5.46423333E-03
365	1.969905E-04	1.96990521E-04	1.96990521E-04	1.96990521E-04
366	5.072769E-05	5.07276937E-05	5.07276937E-05	5.07276937E-05
367	6.739986E-06	6.73998610E-06	6.73998610E-06	6.73998610E-06
384	6.414450E-06	6.41444952E-06	6.41444952E-06	6.41444952E-06
385	5.140714E-05	5.14071404E-05	5.14071404E-05	5.14071404E-05
386	2.439006E-04	2.43900633E-04	2.43900633E-04	2.43900633E-04
387	1.367965E-02	1.36796533E-02	1.36796533E-02	1.36796533E-02
388	6.488402E-01	6.48840170E-01	6.48840170E-01	6.48840170E-01
389	1.299787E-02	1.29978731E-02	1.29978731E-02	1.29978731E-02
390	7.298974E-04	7.29897417E-04	7.29897417E-04	7.29897417E-04
391	1.272724E-04	1.27272419E-04	1.27272419E-04	1.27272419E-04
392	1.708226E-05	1.70822572E-05	1.70822572E-05	1.70822572E-05
409	2.961557E-06	2.96155700E-06	2.96155700E-06	2.96155700E-06

conical	T			
index	provided	Faulty RCWA	Square root Newton method	Eigendecomposition method
410	2.008136E-05	2.00813554E-05	2.00813554E-05	2.00813554E-05
411	1.542083E-04	1.54208285E-04	1.54208285E-04	1.54208285E-04
412	7.219321E-03	7.21932102E-03	7.21932102E-03	7.21932102E-03
413	1.328997E-02	1.32899736E-02	1.32899736E-02	1.32899736E-02
414	6.019298E-03	6.01929796E-03	6.01929796E-03	6.01929796E-03
415	3.797893E-04	3.79789296E-04	3.79789296E-04	3.79789296E-04
416	4.975448E-05	4.97544824E-05	4.97544824E-05	4.97544824E-05
417	7.572382E-06	7.57238222E-06	7.57238222E-06	7.57238222E-06
434	1.624554E-07	1.62455353E-07	1.62455353E-07	1.62455353E-07
435	5.552704E-07	5.55270493E-07	5.55270493E-07	5.55270493E-07
436	1.534496E-05	1.53449561E-05	1.53449561E-05	1.53449561E-05
437	2.890804E-04	2.89080431E-04	2.89080431E-04	2.89080431E-04
438	1.863454E-03	1.86345439E-03	1.86345439E-03	1.86345439E-03
439	2.663216E-04	2.66321611E-04	2.66321611E-04	2.66321611E-04
440	2.712543E-05	2.71254263E-05	2.71254263E-05	2.71254263E-05
441	2.237770E-07	2.23777047E-07	2.23777047E-07	2.23777047E-07
442	3.939751E-07	3.93975059E-07	3.93975059E-07	3.93975059E-07
460	3.113856E-06	3.11385634E-06	3.11385634E-06	3.11385634E-06
461	6.297356E-06	6.29735643E-06	6.29735643E-06	6.29735643E-06
462	2.437255E-04	2.43725462E-04	2.43725462E-04	2.43725462E-04
463	5.535552E-04	5.53555241E-04	5.53555241E-04	5.53555241E-04
464	1.389395E-04	1.38939534E-04	1.38939534E-04	1.38939534E-04
465	6.486021E-06	6.48601989E-06	6.48601989E-06	6.48601989E-06
466	1.705218E-06	1.70521793E-06	1.70521793E-06	1.70521793E-06
467	3.867955E-07	3.86795463E-07	3.86795463E-07	3.86795463E-07
485	7.584534E-08	7.58453441E-08	7.58453441E-08	7.58453441E-08
486	7.003453E-07	7.00345285E-07	7.00345285E-07	7.00345285E-07
487	1.270808E-05	1.27080840E-05	1.27080840E-05	1.27080840E-05
488	2.665410E-05	2.66541050E-05	2.66541050E-05	2.66541050E-05
489	9.228481E-06	9.22848124E-06	9.22848124E-06	9.22848124E-06
490	2.140691E-07	2.14069149E-07	2.14069149E-07	2.14069149E-07
491	1.486752E-07	1.48675258E-07	1.48675258E-07	1.48675258E-07
492	3.092684E-09	3.09268376E-09	3.09268376E-09	3.09268376E-09
511	3.359374E-07	3.35937398E-07	3.35937398E-07	3.35937398E-07
512	9.276603E-06	9.27660371E-06	9.27660371E-06	9.27660371E-06
513	1.579908E-05	1.57990778E-05	1.57990778E-05	1.57990778E-05
514	4.717637E-06	4.71763680E-06	4.71763680E-06	4.71763680E-06
515	3.839638E-07	3.83963819E-07	3.83963819E-07	3.83963819E-07
516	9.615747E-08	9.61574716E-08	9.61574716E-08	9.61574716E-08
537	2.804091E-07	2.80409063E-07	2.80409063E-07	2.80409063E-07
538	1.251929E-06	1.25192929E-06	1.25192929E-06	1.25192929E-06
539	4.124983E-07	4.12498321E-07	4.12498321E-07	4.12498321E-07

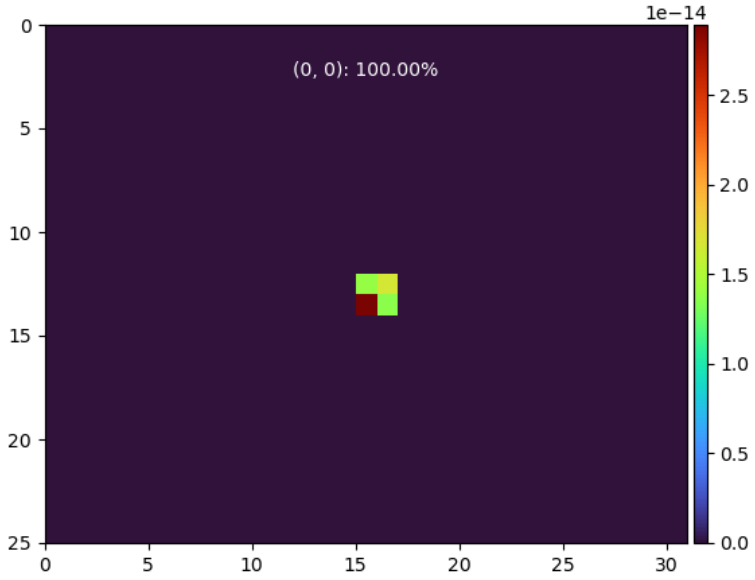


Figure 8.12: L1-error for each R-value found, relative to validation data.

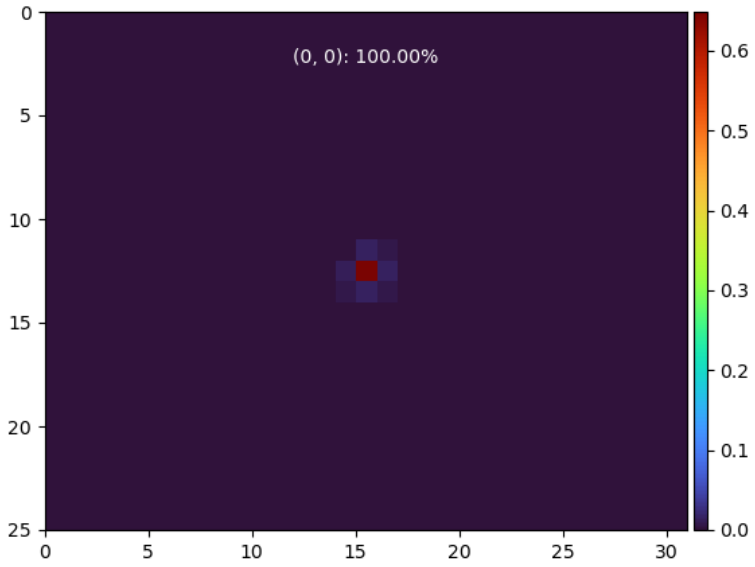


Figure 8.13: T-values found for conical incidence.

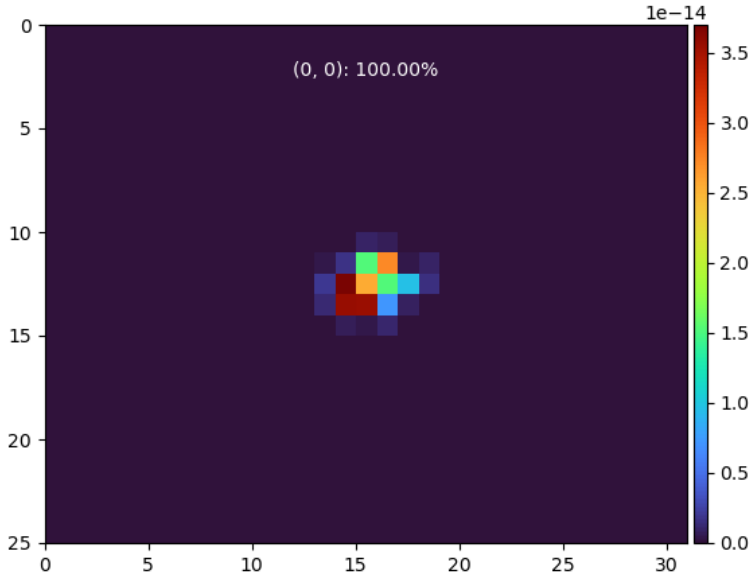


Figure 8.14: L1-error for each T-value found, relative to validation data.

As one can see, the transmittance data have a deviation from the validation data of at most ≈3.7e-14 in magnitude. It is because of this reason that previous discrepancy between measured and found data was considered due to a MATLAB error; reading the values through Python showed a difference much smaller in scope.

8.3.2. Performance

The total time was evaluated by running up to 10 times the RCWA algorithm with the same inputs (The Y20X25 performance was only run thrice, as it took too long). This was done to attempt to standardize timing; as running the same input do

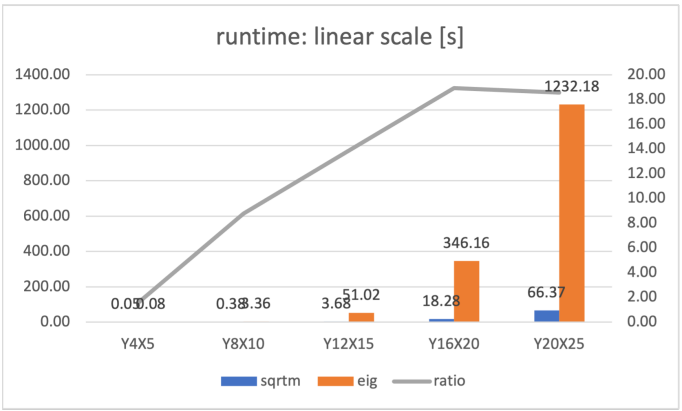


Figure 8.15: RCWA performance, Lossless, s-pol, linear scale.

On the HPC, it shows an increasing relative speedup, until seemingly stabilising at a factor just short of 20.

We can make an educated guess about the curve of the grey line; at the lowest order, the overall system is dominated by linear operations which do not scale; the number of concatenations and initializations do not scale with size.

Later on, the linear operations of the eigendecomposition seem to dominate, and the total computational cost seems to grow. The Padé method also grows, but at a slightly slower pace, so the ratio is still increasing.

Finally, beyond a certain size, this paradigm breaks down.whether this is a new bottleneck in our GPU power being no longer effectively infinite or a bottleneck in data transmission is unclear, and presents an area of future research.

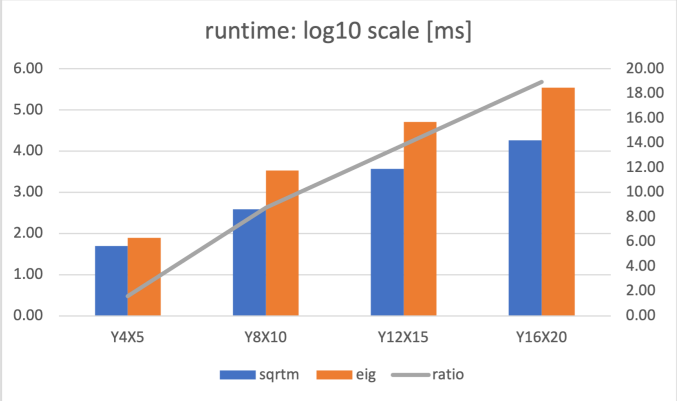


Figure 8.16: RCWA performance, Lossless, s-pol, log10 scale. Ratio shown is still in linear scale.

8.3.3. Perturbation and retrieval

What we found is that all parameters configuring the intermediate layer could be updated using back-propagation. Here we introduced a perturbation to the unit cell, adjusting X_w to 302 and Y_w to 298. We then used two standard optimization configurations; Adam and Stochastic Gradient Descent. No other variables were deemed ”tuneable”, so parameter space in these cases was two-dimensional.

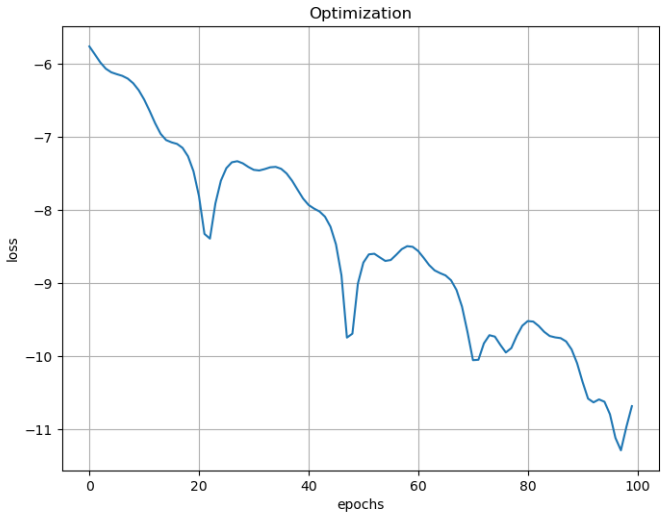


Figure 8.17: Loss function of perturbed system, with Adam training regime. Loss function is L2 norm in log10 scale.

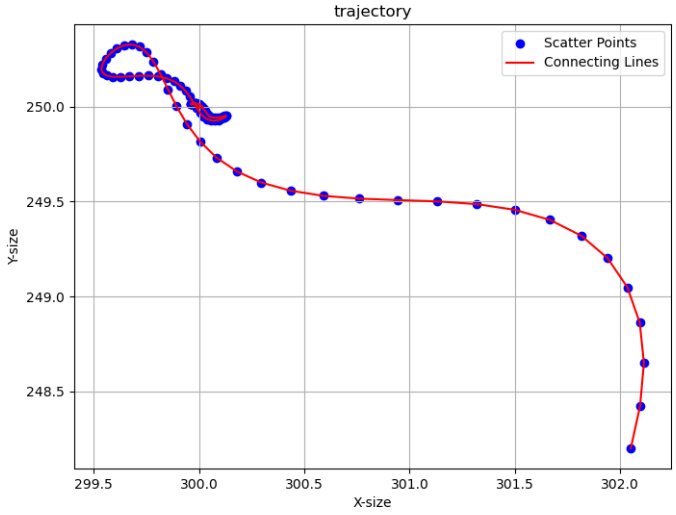


Figure 8.18: Dot plot of the trajectory of the tuneable parameters with Adam training regime. As one can see, it settles around $Y = 250, X = 300$.

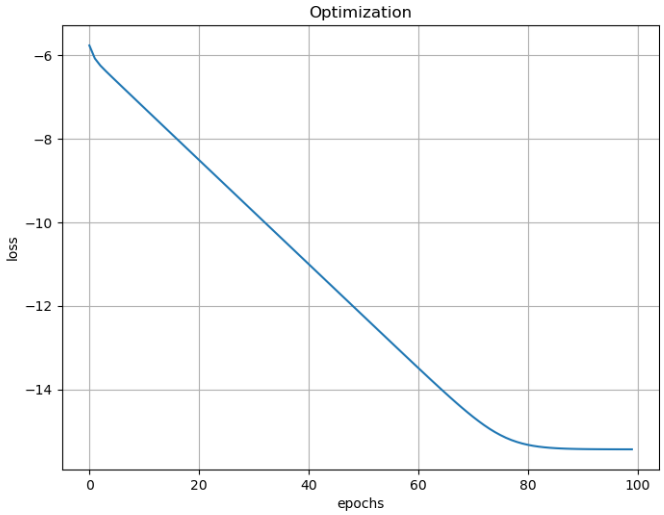


Figure 8.19: Loss function of perturbed system, with SGD training regime. Loss is L2 norm, log10 scale.

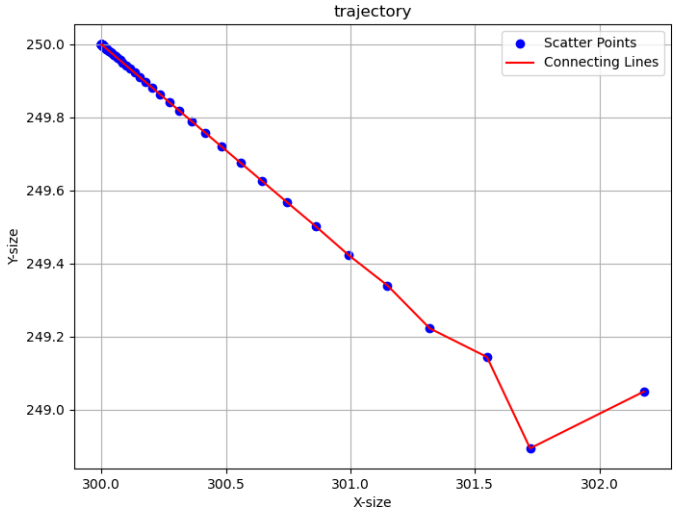


Figure 8.20: Dot plot of the trajectory of the tuneable parameters, with SGD training regime. As one can see, it settles around $Y = 250, X = 300$.

With this, we have found that both methods are able to converge to the right answer, with SGD able to converge to machine-precision within 100 epochs.

9

Discussion

9.1. Some reflection on the results

9.1.1. Validation

Not all aspects were systematically compared. The primary reason for this was time; proper systematic comparison of the entire dataset would have required the implementation of a script which compared found results to the provided dataset. As this was considered a lower priority than demonstrating the capacity to perform parameter retrieval via our new differentiable RCWA algorithm, there is limited systematic validation.

Most of the time, result comparison entailed printing on screen the results of our RCWA algorithm, and visually comparing the values to the corresponding validation data open in MATLAB. Once debugging was completed, we could not spot errors using this method, so the validation was considered ”good enough”.

As we assumed that our algorithm would primarily be used for far-field applications, we only validated the reflectance and transmittance values. The complex field amplitudes have not been compared to those provided by the validation dataset. As such, this method is not validated in the case of near-field measurements or applications such as ptychography, which typically has larger unit cells or supercells. As a result, the validation is currently translation-invariant, which has impacts on the ability to retrieve translation parameters.

Finally, the algorithm has only been evaluated against a case of lossless media. The behavior in the case of lossy media has been validated for homogeneous layers, but not for lossy non-homogeneous layers.

9.1.2. Performance

These results were done against a stock Tensorflow eigendecomposition algorithm. Research has been done on how to accelerate the eigendecomposition method on GPU[13] [14] [15]; this has not been implemented.

The performance comparison is done between the initialization of the intermediate S-matrix only. As the square root method is no longer bottlenecked by the eigendecomposition, the inclusion or exclusion on these extra steps has serious impact on the found performance increase. At that point, the speedup becomes less a measure of the speed of the square root algorithm, and more a measure of the extent to which the rest of the algorithm was bottlenecked by the eigendecomposition.

9.1.3. Optimization for 3D-profilometry

For case two, the retrieval was performed for only two paremeters, x_w and y_w, which converged to sub-nanometer precision. In reality, this is most likely not possible. In a real-world application, one would attempt to retrieve more (most likely all) parameters simultaneously, be comparing a geometrical model of the metrology target to real-world data rather than another geometrical model, the real device would

probably have small imperfections which the model is unable to accurately replicate, and the presence of Poisson noise due to the quantized nature of photon detection would reduce the feasible accuracy further yet. Further, note that ”parameter correlation” when inferring multiple parameters becomes an increasingly large problem.

The results initially did not seek to find the degree to which convergence could be achieved. Instead, the fact that convergence closer to the result at all was possible was already considered a satisfactory result.

We did not demonstrate the impact of a perturbation on the entire parameter space; In separate instances, we did find that these parameters could be updated through the backpropagation scheme, but did not record these results or test further.

We also know for a fact that the RCWA algorithm will not properly retrieve two parameters; since the R-values and T-values are translation-invariant, the absolute position of x_c and y_c cannot be retrieved. However, relative translation between two non-homogeneous layers probably can be retrieved.

9.2. Not fully explored implementations

9.2.1. Properties of matrix sign

There are multiple useful properties of the matrix sign function. Firstly, we know that:

$$msgn(\mathbf{A})^2 = \mathbf{I} \tag{9.1}$$

Secondly, we know that $\mathbf{A}$ and $msgn(\mathbf{A})$ commute:

$$msgn(\mathbf{A})\mathbf{A} - \mathbf{A}msgn(\mathbf{A}) = \mathbf{0} \tag{9.2}$$

This leads naturally from looking at equations 6.57 and 6.58. Combining these, we get the following identity:

$$msgn(\mathbf{A})\mathbf{A}msgn(\mathbf{A}) = \mathbf{A} \tag{9.3}$$

With this, we assert that

$$(msgn(\mathbf{A})\mathbf{A})^{\frac{1}{2}}\sqrt{msgn(\mathbf{A})}$$

is a valid matrix square root of $\mathbf{A}$.

Proof: We know that, if two matrices commute and are diagonalizable, then they are mutually diagonalizable; that is, a matrix $\mathbf{P}$, containing a basis of eigenvectors, exists so that we can express $\mathbf{A}$ and $\mathbf{B}$ as

$$\mathbf{A} = \mathbf{P}\mathbf{D}_1\mathbf{P}^{-1} \tag{9.4}$$

$$\mathbf{B} = \mathbf{P}\mathbf{D}_2\mathbf{P}^{-1} \tag{9.5}$$

Where $\mathbf{D}_1$ and $\mathbf{D}_2$ are both diagonal matrices. These each have the following (not necessarily principal) square roots:

$$\sqrt{\mathbf{A}} = \mathbf{P}\sqrt{\mathbf{D}_1}\mathbf{P}^{-1} \tag{9.6}$$

$$\sqrt{\mathbf{B}} = \mathbf{P}\sqrt{\mathbf{D}_2}\mathbf{P}^{-1} \tag{9.7}$$

With this, let us evaluate the following:

$$(\sqrt{\mathbf{A}}\sqrt{\mathbf{B}})^2 = \sqrt{\mathbf{A}}\sqrt{\mathbf{B}}\sqrt{\mathbf{A}}\sqrt{\mathbf{B}} = \mathbf{P}\sqrt{\mathbf{D}_1}\sqrt{\mathbf{D}_2}\sqrt{\mathbf{D}_1}\sqrt{\mathbf{D}_2}\mathbf{P}^{-1} \tag{9.8}$$

Since $\sqrt{\mathbf{D}_1}$ and $\sqrt{\mathbf{D}_2}$ are diagonal matrices, they must commute, so this evaluates to the following:

$$(\sqrt{\mathbf{A}}\sqrt{\mathbf{B}})^2 = \mathbf{P}\sqrt{\mathbf{D}_1}^2\sqrt{\mathbf{D}_2}^2\mathbf{P}^{-1} = \mathbf{P}\mathbf{D}_1\mathbf{D}_2\mathbf{P}^{-1} = \mathbf{A}\mathbf{B} \tag{9.9}$$

Therefore, if $\mathbf{A}$ and $\mathbf{B}$ are mutually diagonalizable (i.e. both are invertible and they commute), then $(\sqrt{\mathbf{A}}\sqrt{\mathbf{B}})$ is a valid square root of $\mathbf{AB}$, so we can write the following:

$$\sqrt{\mathbf{A}}\sqrt{\mathbf{B}} = \sqrt{\mathbf{A}\mathbf{B}} \tag{9.10}$$

Having said this, this general case does not translate to the specific case of the principal square root, i.e. a matrix square root with eigenvalues with positive real parts:

$$\mathbf{A}^{\frac{1}{2}}\mathbf{B}^{\frac{1}{2}} \neq \mathbf{A}\mathbf{B}^{\frac{1}{2}}$$

As a quick proof, if the first eigenvalue of $\mathbf{A}^{\frac{1}{2}}$ is $1+2i$, and the first eigenvalue of $\mathbf{B}^{\frac{1}{2}}$ is $3+2i$, then the first eigenvalues of $\mathbf{A}^{\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}$ is $-1+8i$. Since this has a negative real part, then $\mathbf{A}^{\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}$ cannot be a principal square root.

Since, if $\mathbf{A}$ is invertible, then $msgn(\mathbf{A})\mathbf{A}$ is positive definite, so we can use multiplication-friendly algorithms evaluating it. Likewise, finding a square root for $msgn(\mathbf{A})$ is trivial.

Because these two matrices commute, the matrix sign seemed as a stepping stone to a matrix with eigenvalues with no negative real parts, for which the literature of finding a matrix square root seemed richer at first.

9.2.2. Square root of matrix sign

The value

$$\frac{msgn(\mathbf{A}) + i\mathbf{I}}{1+i} \tag{9.11}$$

is a matrix square root of $msgn(\mathbf{A})$. Proof:

$$\left(\frac{msgn(\mathbf{A}) + i\mathbf{I}}{1+i}\right)^2 = \frac{msgn(\mathbf{A})^2 + 2i * msgn(\mathbf{A}) - \mathbf{I}}{1+2i-1} = \frac{\mathbf{I} + 2i * msgn(\mathbf{A}) - \mathbf{I}}{1+2i-1} = \frac{2i * msgn(\mathbf{A})}{2i} = msgn(\mathbf{A}) \tag{9.12}$$

9.2.3. Taylor series

Another method tried was the Taylor series for the matrix square root: [25]

$$\mathbf{A}^{\frac{1}{2}} = \sqrt{|\mathbf{A}|_F}\left(\sum_{k=1}^{\infty}\binom{\frac{1}{2}}{k}\left(\mathbf{I} - \frac{\mathbf{A}}{|\mathbf{A}|_F}\right)^k\right) \tag{9.13}$$

This method had multiple issues:

- It did not work if the matrix had eigenvalues with negative real parts;
- It did not work if the matrix was not properly normalized;
- It converged extremely slowly. In fact, the closer one got, the slower it would converge relatively.
- The more aggressive normalization was, the slower it would converge.

In hindsight, these methods were outlined for situations in which the exact square root was not required, only an approximation, so the methods outlined in source **??** ultimately would not have sufficed. Looking back, it also mentions the Padé approximant for the matrix square root, but did not extend the concept to iterative methods.

However, it was a method which could provide an accurate value, so it was a start. This led to a search of finding a way to turn a matrix into one without, while simultaneously preserving the information. This search led to the matrix sign function.

9.3. Other multiplication-friendly ideas

The advent of AI has also introduced something new; because it primarily makes use of matrix multiplications, the advent of AI has single-handedly created demand for computing systems which are optimized for but one operation; matrix multiplications. These come in the form of Google's Tensor Processing Unit, but also NVIDIA's industrial GPU's having TPU FLOPs per second multiple times larger than regular float64 FLOPs per second.

This also includes the like of matrix additions and matrix scalar multiplications; each of these are comparably cheap computationally to a matrix multiplication, and are similar highly parallelizable.

In short, this includes every operation used in our RCWA model, save for one; the matrix inversion. Removing the matrix inversion is not feasible in the case of RCWA; for one, it is the matrix inverse which keeps the size of the result bounded, whether this is in the form of the stable Padé iteration, or the inverse found in the Redheffer star product, or the normalization present in the enhanced Transmittance Matrix method.

In this chapter, we will discuss two iterative, matrix multiplication friendly algorithms which converge to the matrix inverse, found in literature. Additionally, we will discuss a newly discovered algorithm which also evaluates the matrix inverse, with improved stability performance over one of the two algorithms found in literature. Finally, we will discuss a new method for evaluating the T-matrix which is matrix multiplication friendly, with the drawback of introducing (limited) rounding error. These all suggest that it is possible to create a matrix multiplication friendly algorithm which is compatible with the most advanced hardware of today.

9.3.1. Multiplication-friendly polar decomposition

The matrix sign function can also be used to calculate the following:

$$msgn(\begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A^H} & \mathbf{0} \end{bmatrix}) = \begin{bmatrix} \mathbf{0} & \mathbf{U} \\ \mathbf{U^H} & \mathbf{0} \end{bmatrix} \tag{9.14}$$

Where

$$\mathbf{A} = \mathbf{UP} \tag{9.15}$$

In which **U** is unitary, and **P** is positive semi-definite. Because of this, **P** can be calculated as follows:

$$\mathbf{P} = \mathbf{AU^H} \tag{9.16}$$

The matrix $\mathbf{AA^H}$ is symmetric positive semi-definite, and if **A** is invertible, $\mathbf{AA^H}$ is symmetric positive definite. Since $\mathbf{AA^H}$ shares the same eigenvalues as $\mathbf{A^HA}$, this also goes for that matrix. Finally, this means that the spectrum of any square root of $\mathbf{A^HA}$ is on the real number line, and if **A** is invertible, excludes the origin in the complex plane.

In other words, the spectrum of $\begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A^H} & \mathbf{0} \end{bmatrix}$ falls within the convergence range of the Newton-Shultz iteration. Thus, we can describe a multiplication-friendly iterative method as follows:

$$\begin{aligned} \mathbf{Y}_0 = \mathbf{A}, \mathbf{Z}_0 = \mathbf{A^H} \\ \mathbf{Y}_{k+1} = \frac{1}{2}\mathbf{Y}_k(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k) \\ \mathbf{Z}_{k+1} = \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k)\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k = \mathbf{U} \\ \lim_{k\to\infty} \mathbf{Z}_k = \mathbf{U^H} \\ \lim_{k\to\infty} \mathbf{AZ}_k = \mathbf{P} \end{aligned} \tag{9.17}$$

This is guaranteed to converge if **A** is properly normalized. Exactly how useful this result is for the finding of the matrix square root is yet unclear; this gives you more information about the matrix $\mathbf{A^HA}$ than it does about **A**. In the case where **A** commutes with $\mathbf{A^H}$ (which is referred to as a normal matrix), this decomposition gives you information about the phase and magnitude of the spectrum of **A** by extension, but in the most general case this is not true.

9.3.2. Multiplication-friendly matrix inverse

$$\text{msgn}(\begin{bmatrix} \mathbf{0} & \mathbf{A^HA} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}) = \begin{bmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{P}^{-1} & \mathbf{0} \end{bmatrix} \tag{9.18}$$

Since $\mathbf{A^H A}$ is positive-definite if invertible, this can be solved with Newton-Shultz iteration. From this, we can get:

$$\mathbf{U} = \mathbf{A P}^{-1} \tag{9.19}$$

$$\mathbf{U}^* = (\mathbf{P}^{-1})^* \mathbf{A^H} = \mathbf{P}^{-1} \mathbf{A^H} \tag{9.20}$$

$$\mathbf{A} = \mathbf{U P} \tag{9.21}$$

$$\mathbf{A}^{-1} = \mathbf{P}^{-1} \mathbf{U}^{-1} = \mathbf{P}^{-1} \mathbf{U}^* \tag{9.22}$$

$$\mathbf{A}^{-1} = \mathbf{P}^{-2} \mathbf{A^H} \tag{9.23}$$

With this, we have the following algorithm:

$$\begin{aligned} \mathbf{Y}_0 = \frac{\mathbf{A^H A}}{\sqrt{\rho(\mathbf{A^H A})}}, \mathbf{Z}_0 = \frac{I}{\sqrt{\rho(\mathbf{A^H A})}} \\ \mathbf{Y}_{k+1} = \frac{1}{2} \mathbf{Y}_k (3\mathbf{I} - \mathbf{Z}_k \mathbf{Y}_k) \\ \mathbf{Z}_{k+1} = \frac{1}{2} (3\mathbf{I} - \mathbf{Z}_k \mathbf{Y}_k) \mathbf{Z}_k \\ \lim_{k \to \infty} \mathbf{Y}_k = \mathbf{P} \\ \lim_{k \to \infty} \mathbf{Z}_k = \mathbf{P}^{-1} \\ \mathbf{A}^{-1} = (\lim_{k \to \infty} \mathbf{Z}_k)^2 \mathbf{A^H} \end{aligned} \tag{9.24}$$

However, this is in practice only a slower, more memory-costly, and seemingly less stable version of the Newton iteration for the matrix inverse:

$$\begin{aligned} \mathbf{Y}_0 = \frac{\mathbf{A^H}}{\rho(\mathbf{A^H A})} \\ \mathbf{Y}_{k+1} = 2\mathbf{Y}_k - \mathbf{Y}_k \mathbf{A} \mathbf{Y}_k \\ \lim_{k \to \infty} \mathbf{Y}_k = \mathbf{A}^{-1} \end{aligned} \tag{9.25}$$

Since $\mathbf{A^H A}$ is positive-definite, its eigenvalues all lie on the positive real axis. Because of this, we can also find the smallest eigenvalue as follows:

$$\lambda_{min} = \rho(\mathbf{A^H A} - \mathbf{I}\rho(\mathbf{A^H A})) + \rho(\mathbf{A^H A}) \tag{9.26}$$

In essence, because they are all on the real positive number line, shifting the eigenvalues leftward in the complex plane far enough (we do this by subtracting $\mathbf{I}\rho(\mathbf{A^H A})$) makes the previously-smallest eigenvalue in absolute terms now the biggest in absolute terms. As a result, we can use the spectrum to find the now biggest in absolute terms eigenvalue, then we shift it back to see its actual size.

The reason this is of interest is because the size of the condition number the largest and smallest eigenvalue is ultimately what determines the following:

- the amount of steps needed for convergence;
- the amount of rounding error introduced at each step;
- by extension of these two, the total error after the iterative method is done;
- whether the method will converge at all.

In cases where eigenvalues have a small spectral range, Newton's method for finding inverses is most likely superior. Seeing how this might be integrated within a GMRES algorithm is a potential field of future research.

9.3.3. Faster Newton-Shultz iteration

[17] As it turns out, both the Newton-Shultz iteration and the Padé iterations are specific cases of a general class of iterative methods for the sign function. All of these are stable for the previously-mentioned criterium:

$$|1 - z_0^2| < 0$$

The Padé cases are special, because the symmetry in the numerator and denominator allows for it to be globally convergent, and thus has a one-size-fits-all nature. The Newton-Shultz case is special, because it is multiplication-friendly. Most other iterative methods outlined in that paper have neither of these properties; they require an inverse to compute, but are not globally convergent. It is for this reason that they are not mentioned in this thesis.

There are two iterations not mentioned which ought to be, however. These are the following iterations [17]:

$$\begin{aligned} \mathbf{Y}_0 &= \mathbf{A}, \mathbf{Z}_0 = \mathbf{I} \\ \mathbf{Y}_{k+1} &= \frac{1}{8}\mathbf{Y}_k(15\mathbf{I} - 10\mathbf{Z}_k\mathbf{Y}_k + 3(\mathbf{Z}_k\mathbf{Y}_k)^2) \\ \mathbf{Z}_{k+1} &= \frac{1}{8}(15\mathbf{I} - 10\mathbf{Z}_k\mathbf{Y}_k + 3(\mathbf{Z}_k\mathbf{Y}_k)^2)\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k &= \mathbf{X} \\ \lim_{k\to\infty} \mathbf{Z}_k &= \mathbf{X}^{-1} \\ \mathbf{X}^2 &= \mathbf{A} \end{aligned} \tag{9.27}$$

and

$$\begin{aligned} \mathbf{Y}_0 &= \mathbf{A}, \mathbf{Z}_0 = \mathbf{I} \\ \mathbf{Y}_{k+1} &= \frac{1}{16}\mathbf{Y}_k(35\mathbf{I} - 35\mathbf{Z}_k\mathbf{Y}_k + 21(\mathbf{Z}_k\mathbf{Y}_k)^2 - 5(\mathbf{Z}_k\mathbf{Y}_k)^3) \\ \mathbf{Z}_{k+1} &= \frac{1}{16}(35\mathbf{I} - 35\mathbf{Z}_k\mathbf{Y}_k + 21(\mathbf{Z}_k\mathbf{Y}_k)^2 - 5(\mathbf{Z}_k\mathbf{Y}_k)^3)\mathbf{Z}_k \\ \lim_{k\to\infty} \mathbf{Y}_k &= \mathbf{X} \\ \lim_{k\to\infty} \mathbf{Z}_k &= \mathbf{X}^{-1} \\ \mathbf{X}^2 &= \mathbf{A} \end{aligned} \tag{9.28}$$

These iterations are, in effect, higher-order Newton-Shultz iterations. They require 1 or 2 extra matrix multiplications per iteration, respectively, to reach a convergence order of 3 or 4, respectively. If we assume that the total calculations needed are proportional to

$$N_{tot} = N_{iter} * N_{calc/iter} \propto \frac{N_{calc/iter}}{\ln(m)} \tag{9.29}$$

The total matrix multiplications per iteration, as a function of the order m, is $m + 1$ (to achieve order 2, you need 3 matrix multiplications per iteration, order 3 needs 4 matrix multiplications, and so on). If we look to find the positive integer m which makes for the smallest value of

$$N_{tot} \propto \frac{m+1}{\ln(m)} \tag{9.30}$$

That would be $m = 4$, with a result

$$\frac{5}{\ln(4)} \approx 3.607 \tag{9.31}$$

For $m = 3$, this instead evaluates to

$$\frac{4}{\ln(3)} \approx 3.641 \tag{9.32}$$

For $m = 2$, this instead evaluates to

$$\frac{3}{\ln(2)} \approx 4.328 \tag{9.33}$$

Thus, this alternative $m = 4$ formulation could give another speedup of

$$\frac{3}{\ln(2)} / \frac{5}{\ln(4)} = \frac{3 * 2}{5} = 1.2 \tag{9.34}$$

Thus, we could see a speedup of 20%, relative to the Newton-Shultz Iteration, with this alternate iteration. for $m = 3$, this would be a speedup of 18.9%. For higher values of m, the total computational cost goes up again.

9.4. Special case for the convolution matrix: homogeneous layer

A "homogeneous layer" is a layer in which there is one pixel, whose area overlaps perfectly with the area of the unit cell. In such a case, we know that:

$$\frac{x_w}{L_x} = 1, \frac{y_w}{L_y} = 1, Q = 1, P = 1 \tag{9.35}$$

In such situations, the act of defining any of these variables is conceptually pointless, but they are necessary for further steps.

The Fourier transforms become:

$$a^m[q] = \epsilon[p, q] e^{-i2\pi \frac{m}{L_x} x_c[p]} sinc(m) \tag{9.36}$$

$$a^n[p] = \epsilon[p, q] e^{-i2\pi \frac{n}{L_y} y_c[q]} sinc(n) \tag{9.37}$$

Because of how we previously defined the sinc function, we know that for integer values:

$$sinc(m) = \begin{cases} 1, & m = 0 \\ 0, & \text{else} \end{cases} \tag{9.38}$$

In such a case, we can then replace this with the Kronecker delta. Because of this, the Fourier terms simplify to only the 0'th order Fourier component having a non-zero value:

$$\begin{aligned} a^m[q] &= \delta_m \epsilon[q] \\ a^n[p] &= \delta_n \epsilon[p] \end{aligned} \tag{9.39}$$

If we write out the one-dimensional convolution matrices, it will look like the following:

$$[\epsilon]_x[q] = \begin{bmatrix} \epsilon[q] & 0 & \dots & 0 \\ 0 & \epsilon[q] & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \epsilon[q] \end{bmatrix} = \epsilon[q] \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix} = \epsilon[q] \mathbf{I} \tag{9.40}$$

The inverse of such a matrix is:

$$[\epsilon]_x[q]^{-1} = \left(\epsilon[q] \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix} \right)^{-1} = \frac{1}{\epsilon}[q] \mathbf{I} = [\frac{1}{\epsilon}]_x^{-1}[q] \tag{9.41}$$

And, in the case of homogeneity in the second direction as well, this gives us:

$$a^{m,n} = \delta_m \delta_n \epsilon \tag{9.42}$$

It is for this reason that, for case 1 in the results which only featured homogeneous layers, that the system could be accurately modelled while keeping $M = N = 0$. In such a case, we can state the following:

$$[[\frac{1}{\epsilon}]_x^{-1}]_y = [[\frac{1}{\epsilon}]_y^{-1}]_x = [\![\epsilon]\!] \tag{9.43}$$

And, as a result, the Li rule is equivalent to the Laurent rule.

9.5. Points of attention with RCWA

The RCWA algorithm itself is not perfect. It contains some inherent flaws. The largest flaw is that it only gives correct results for one incident plane wave at a time. More specifically, the R-values and T-values for the incident field defined in equation 9.44 sum to 1 in case of non-absorptive media. However, if instead the incident field is defined as

$$\mathbf{c}^{src} = \begin{bmatrix} \boldsymbol{\delta}_{1,pq} p_x \\ \boldsymbol{\delta}_{1,pq} p_y \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{9.44}$$

then the resulting R-values and T-values no longer sum to 1 in cases of non-absorptive media. If it did, the S-matrix as evaluated by the entire RCWA model could be repurposed for different incident plane waves, which would allow the computational cost of RCWA to be spread over multiple incidences. Instead, a whole new instance has to be evaluated, where the incident wave k-values are defined as follows:

$$k_{x,\text{inc}} = k_{x,\text{inc,prev}} - \frac{2\pi}{L_x} \tag{9.45}$$

$$k_{y,\text{inc}} = k_{y,\text{inc,prev}} \tag{9.46}$$

Other issues are related to Fast Fourier Factorization; its necessity stems from the fact that the inverse of a convolution operator is not itself a convolution operator. Physically, however, there ought to be no distinction between deconvolution and convolution with a function's inverse. This suggests that the formulation of the convolution operator is flawed. Further research on whether this flaw stems from truncation of the orders or overall incorrect formulation is recommended.

9.6. Further research

Future research in terms of the numerical method to find the matrix square root should focus on:

- Devise a better test for convergence properties of a matrix, which is not as conservative as current methods;
- Devise some scheme for the matrix square root by extrapolation; that is, we can (for example) easily find the matrix square root of $i\mathbf{PQ} + 2\mathbf{I}$ or $i\mathbf{PQ} + \mathbf{I}$. Both of these square roots should commute with the actual square root we want to find. Perhaps the eigenvalues trace some curve in the complex plane which we can follow to the end destination.
- Alternatively, we can iteratively peel off the "most-negative" eigenvalues, thereby getting closer to what we want. I'm not sure this is really all that useful, since we need some inverses for concatenation anyway, but it's a start.
- Attempt to model the logarithm or square root using series different from Taylor series, such as Laurent series.

Future research on RCWA itself can focus on:

- Why is Fast Fourier Factorization needed at all?
- Why do **P** and **Q** each have a trace of zero? Does this not suggest some more elegant quadruple-mode coupling?
- Why does the gap medium provide a causal base for RCWA, even in materials where the eigen-modes of the material are not the same as that of the gap medium? Does it match up because it somehow relates to the Poynting vectors inside the material?

10

Conclusion

We have rewritten the RCWA into a new algorithm, which we have implemented and evaluated in practice. RCWA is a semi-analytical method which is evaluated using the Plane Wave Expansion Method in the x-direction and y-direction. The method is evaluated in the z-direction using the Transfer Matrix Method, making use of Scattering parameters and gap media to make the system well-posed.

We have implemented this algorithm in Tensorflow. For this scheme, we have demonstrated the following:

- It is able to replicate the reference validation datasets (obtained with state-of-the-art RCWA implementation) down to near-rounding error accuracy.
- When the parameters are perturbed, the model is able to perform backpropagation and perform parameter retrieval to regain the correct value for every parameter of the device; this is the typical inverse problem of 3D-profilometry in e.g. semiconductor metrology.

In other words, our new concept has been proven.

Additionally, we have found an alternate formulation for the S-parameter matrix, whose terms serve as analogues to those found in the conventional Scattering parameter matrix method. These are analogous to the **W**, **V**, and **X** matrices. We have demonstrated that using these analogues in lieu of the known values inside the S-parameter matrix algorithm will numerically yield the same values as those found by eigendecomposition, and have demonstrated that the equation for the S-parameter matrix yields the same result with the analogues as with the conventional values.

We have demonstrated that these analogues can be calculated using iterative methods. These methods involve the use of matrix multiplications, scalar multiplications, matrix additions, and matrix solves. We have demonstrated that this algorithm can be up to 20 times faster on the HPC hardware with GPU's provided by TU Delft.

For this method, the concept of using our differentiable RCWA backpropagtion for parameter retrieval in 3D profilometry has also been proven, within the context of automatic differentiation.

We have also found a second alternate formulation, which evaluates the T-matrix in a matrix-multiplication-friendly algorithm. While this method is unstable, the stability of the method is well-understood, and can be converted to an S-matrix formulation at any point limiting the effective error. For this alternate method, the concept of using RCWA backpropagtion for parameter retrieval has not been proven.

The following areas for further research are presented;

- Alternate formulations for the intermediate **P** and **Q** matrices, which might be better-posed for further calculations, and potentially allow a matrix-multiplication friendly method of evaluating the **W**,**V**, and **X** analogues;
- Further explore the potential application of the "Faulty RCWA" approach for the use of staircasing;

- Explore the needed adjustments to the ”overall” RCWA algorithm (beyond the eigendecomposition problem, which was addressed as a main topic in our report) for application on massively-parallelized computer architectures;
- Alternate gradient-descent methods which make use of a Hessian (or pseudo-Hessian) operator for accelerated convergence for parameter retrieval;
- Apply this concept to other numerical methods for parameter retrieval, such as volume-integral methods or finite element methods.

References

[1] Gaston Floquet. “Sur les équations différentielles linéaires à coefficients périodiques”. In: *Annales scientifiques de l’École normale supérieure*. Vol. 12. 1883, pp. 47–88.

[2] George William Hill. “On the part of the motion of the lunar perigee which is a function of the mean motions of the sun and moon”. In: (1886).

[3] Felix Bloch. “Über die quantenmechanik der elektronen in kristallgittern”. In: *Zeitschrift für physik* 52.7-8 (1929), pp. 555–600.

[4] Lifeng Li. “New formulation of the Fourier modal method for crossed surface-relief gratings”. In: *J. Opt. Soc. Am. A* 14.10 (Oct. 1997), pp. 2758–2767. doi: 10.1364/JOSAA.14.002758. url: https://opg.optica.org/josaa/abstract.cfm?URI=josaa-14-10-2758.

[5] Lifeng Li. “4. Mathematical Reflections on the Fourier Modal Method in Grating Theory”. In: *Mathematical Modeling in Optical Science*, pp. 111–139. doi: 10.1137/1.9780898717594.ch4. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717594.ch4. url: https://epubs.siam.org/doi/abs/10.1137/1.9780898717594.ch4.

[6] Michael J. Tsatsomeros. “Principal pivot transforms: properties and applications”. In: *Linear Algebra and its Applications* 307.1 (2000), pp. 151–165. issn: 0024-3795. doi: https://doi.org/10.1016/S0024-3795(99)00281-5. url: https://www.sciencedirect.com/science/article/pii/S0024379599002815.

[7] Raymond C Rumpf. “Improved formulation of scattering matrices for semi-analytical methods that is consistent with convention”. In: *Progress In Electromagnetics Research B* 35 (2011), pp. 241–261.

[8] Eng Leong Tan. “Hybrid-matrix algorithm for rigorous coupled-wave analysis of multilayered diffraction gratings”. In: *Journal of Modern Optics* 53.4 (2006), pp. 417–428.

[9] Raymond Rumpf. *Computational electromagnetics*. Sept. 2022. url: https://empossible.net/academics/emp5337/.

[10] Åke Björck. “Least squares methods”. In: *Handbook of numerical analysis* 1 (1990), pp. 465–652.

[11] Atilim Gunes Baydin et al. “Automatic differentiation in machine learning: a survey”. In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[12] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[13] Jingxiao Xu and Martin D. B. Charlton. “Speed efficiency optimization for GPU accelerated rigorous coupled-wave analysis program”. In: *Physics and Simulation of Optoelectronic Devices XXXI*. Ed. by Bernd Witzigmann, Marek Osiński, and Yasuhiko Arakawa. Vol. 12415. International Society for Optics and Photonics. SPIE, 2023, 124150J. doi: 10.1117/12.2650022. url: https://doi.org/10.1117/12.2650022.

[14] W Iff et al. “Memory sparing, fast scattering formalism for rigorous diffraction modeling”. In: *Journal of Optics* 19.7 (June 2017), p. 075602. doi: 10.1088/2040-8986/aa7012. url: https://dx.doi.org/10.1088/2040-8986/aa7012.

[15] Xvlong Wei and Shuqiang Chen. “Parallel Computing for Application in Rigorous Coupled-Wave Analysis”. In: *2012 Fifth International Symposium on Computational Intelligence and Design*. Vol. 2. 2012, pp. 186–189. doi: 10.1109/ISCID.2012.198.

[16] Brian C. Hall. “Lie Groups, Lie Algebras, and Representations”. In: *Quantum Theory for Mathematicians*. New York, NY: Springer New York, 2013, pp. 333–366. isbn: 978-1-4614-7116-5. doi: 10.1007/978-1-4614-7116-5_16. url: https://doi.org/10.1007/978-1-4614-7116-5_16.

[17] Charles Kenney and Alan J. Laub. “Rational Iterative Methods for the Matrix Sign Function”. In: *SIAM Journal on Matrix Analysis and Applications* 12.2 (1991), pp. 273–291. doi: 10.1137/0612020. eprint: https://doi.org/10.1137/0612020. url: https://doi.org/10.1137/0612020.

[18] Nicholas J Higham. “Newton’s method for the matrix square root”. In: *Mathematics of computation* 46.174 (1986), pp. 537–549.

[19] Nicholas J Higham. “Stable iterations for the matrix square root”. In: *Numerical Algorithms* 15 (1997), pp. 227–242.

[20] John J Hench and ZDENEK Strakoš. “The RCWA method-a case study with open questions and perspectives of algebraic computations”. In: *Electronic Transactions on Numerical Analysis* 31 (2008), pp. 331–357.

[21] Nicholas J Higham et al. “Functions preserving matrix groups and iterations for the matrix square root”. In: *SIAM Journal on Matrix Analysis and Applications* 26.3 (2005), pp. 849–877.

[22] Nicholas J Higham et al. “Computing the polar decomposition and the matrix sign decomposition in matrix groups”. In: *SIAM journal on matrix analysis and applications* 25.4 (2004), pp. 1178–1192.

[23] Nicholas J Higham. “The scaling and squaring method for the matrix exponential revisited”. In: *SIAM Journal on Matrix Analysis and Applications* 26.4 (2005), pp. 1179–1193.

[24] Massimiliano Fasi and Nicholas J. Higham. “An Arbitrary Precision Scaling and Squaring Algorithm for the Matrix Exponential”. In: *SIAM Journal on Matrix Analysis and Applications* 40.4 (2019), pp. 1233–1256. doi: 10.1137/18M1228876. eprint: https://doi.org/10.1137/18M1228876. url: https://doi.org/10.1137/18M1228876.

[25] Yue Song, Nicu Sebe, and Wei Wang. “Fast Differentiable Matrix Square Root”. In: *CoRR* abs/2201.08663 (2022). arXiv: 2201.08663. url: https://arxiv.org/abs/2201.08663.

A

Appendix

A.1. Redheffer star product equivalence

Suppose we take two arbitrary S-matrices:

$$\begin{bmatrix} b_1^A \\ b_2^A \end{bmatrix} = \begin{bmatrix} S_{11}^A & S_{12}^A \\ S_{21}^B & S_{22}^A \end{bmatrix} \begin{bmatrix} a_1^A \\ a_2^A \end{bmatrix} \tag{A.1}$$

$$\begin{bmatrix} b_1^B \\ b_2^B \end{bmatrix} = \begin{bmatrix} S_{11}^B & S_{12}^B \\ S_{21}^B & S_{22}^B \end{bmatrix} \begin{bmatrix} a_1^B \\ a_2^B \end{bmatrix} \tag{A.2}$$

If we convert both of these to T-matrices, we get the following:

$$\begin{bmatrix} T_{11}^A & T_{12}^A \\ T_{21}^A & T_{22}^A \end{bmatrix} = \begin{bmatrix} S_{21}^A - S_{22}^A (S_{12}^A)^{-1} S_{11}^A & S_{22}^A (S_{12}^A)^{-1} \\ -(S_{12}^A)^{-1} S_{11}^A & (S_{12}^A)^{-1} \end{bmatrix} \tag{A.3}$$

$$\begin{bmatrix} T_{11}^B & T_{12}^B \\ T_{21}^B & T_{22}^B \end{bmatrix} = \begin{bmatrix} S_{21}^B - S_{22}^B (S_{12}^B)^{-1} S_{11}^B & S_{22}^B (S_{12}^B)^{-1} \\ -(S_{12}^B)^{-1} S_{11}^B & (S_{12}^B)^{-1} \end{bmatrix} \tag{A.4}$$

Concatenating these matrices gives the following sub-matrices:

$$T_{11}^{BA} = (S_{21}^A - S_{22}^A (S_{12}^A)^{-1} S_{11}^A)(S_{21}^B - S_{22}^B (S_{12}^B)^{-1} S_{11}^B) - (S_{22}^A (S_{12}^A)^{-1})((S_{12}^B)^{-1} S_{11}^B) \tag{A.5}$$

$$T_{12}^{BA} = (S_{21}^A - S_{22}^A (S_{12}^A)^{-1} S_{11}^A)(S_{22}^B (S_{12}^B)^{-1}) + (S_{22}^A (S_{12}^A)^{-1})((S_{12}^B)^{-1}) \tag{A.6}$$

$$T_{21}^{BA} = (-(S_{12}^A)^{-1} S_{11}^A)(S_{21}^B - S_{22}^B (S_{12}^B)^{-1} S_{11}^B) + ((S_{12}^A)^{-1})((S_{12}^B)^{-1} S_{11}^B) \tag{A.7}$$

$$T_{22}^{BA} = (-(S_{12}^A)^{-1} S_{11}^A)(S_{22}^B (S_{12}^B)^{-1}) + ((S_{12}^A)^{-1})((S_{12}^B)^{-1}) \tag{A.8}$$

Most of these expressions we can simplify somewhat by expanding the brackets and grouping common factors:

$$T_{22}^{BA} = (S_{12}^A)^{-1}(I - S_{11}^A S_{22}^B)(S_{12}^B)^{-1} \tag{A.9}$$

$$T_{21}^{BA} = -(S_{12}^A)^{-1} S_{11}^A S_{21}^B - (S_{12}^A)^{-1}(I - S_{11}^A S_{22}^B)(S_{12}^B)^{-1} S_{11}^B \tag{A.10}$$

$$T_{12}^{BA} = S_{21}^A S_{22}^B (S_{12}^B)^{-1} + S_{22}^A (S_{12}^A)^{-1}(I - S_{11}^A S_{22}^B)^{-1}(S_{12}^B)^{-1} \tag{A.11}$$

These somewhat simplified expressions can be used in the expression for the transform from T to S, to give us the overall S-matrix.

$$S_{12}^{BA} = T_{22}^{BA-1} = S_{12}^B (I - S_{11}^A S_{22}^B)^{-1} S_{12}^A \tag{A.12}$$

$$S_{11}^{BA} = -T_{22}^{BA-1}T_{21}^{BA} = -\left(S_{12}^{B}(I-S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A}\right)\left(-(S_{12}^{A})^{-1}S_{11}^{A}S_{21}^{B}-(S_{12}^{A})^{-1}(I-S_{11}^{A}S_{22}^{B})(S_{12}^{B})^{-1}S_{11}^{B}\right) = \tag{A.13}$$

$$= S_{12}^{B}(I - S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A}(S_{12}^{A})^{-1}S_{11}^{A}S_{21}^{B} + S_{12}^{B}(I - S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A}(S_{12}^{A})^{-1}(I - S_{11}^{A}S_{22}^{B})(S_{12}^{B})^{-1}S_{11}^{B} \tag{A.14}$$

$$S_{11}^{BA} = S_{12}^{B}(I - S_{11}^{A}S_{22}^{B})^{-1}S_{11}^{A}S_{21}^{B} + S_{11}^{B} \tag{A.15}$$

$$S_{22}^{BA} = T_{12}^{BA}T_{22}^{BA-1} = -\left(S_{21}^{A}S_{22}^{B}(S_{12}^{B})^{-1} + S_{22}^{A}(S_{12}^{A})^{-1}(I - S_{11}^{A}S_{22}^{B})^{-1}(S_{12}^{B})^{-1}\right)\left(S_{12}^{B}(I - S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A}\right) \tag{A.16}$$

$$S_{22}^{BA} = S_{21}^{A}S_{22}^{B}(I - S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A} + S_{22}^{A} \tag{A.17}$$

Finally, for the last submatrix, we evaluate the following expression:

$$S_{21}^{B}A = T_{11}^{BA} - T_{12}^{BA}T_{22}^{BA-1}T_{21}^{BA} \tag{A.18}$$

$$= \left(S_{21}^{A}S_{21}^{B} - S_{21}^{A}S_{22}^{B}S_{21}^{B-1} - \right) \tag{A.19}$$

$$-\left(S_{12}^{B}(I-S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A}\right)\left(-(S_{12}^{A})^{-1}S_{11}^{A}S_{21}^{B}-(S_{12}^{A})^{-1}(I-S_{11}^{A}S_{22}^{B})(S_{12}^{B})^{-1}S_{11}^{B}\right)\left(S_{12}^{B}(I-S_{11}^{A}S_{22}^{B})^{-1}S_{12}^{A}\right) \tag{A.20}$$

Evaluating all of this solves to the following:

$$S_{21}^{BA} = S_{21}^{A}(I - S_{22}^{B}S_{11}^{A})^{-1}S_{21}^{B} \tag{A.21}$$

Thus, arranging this all in matrix form gives us the following:

$$\mathbf{S}_A \otimes \mathbf{S}_B = \begin{bmatrix} \mathbf{S}_{11}^{A} + \mathbf{S}_{12}^{A}[\mathbf{I} - \mathbf{S}_{11}^{B}\mathbf{S}_{22}^{A}]^{-1}\mathbf{S}_{11}^{B}\mathbf{S}_{21}^{A} & \mathbf{S}_{12}^{A}[\mathbf{I} - \mathbf{S}_{11}^{B}\mathbf{S}_{22}^{A}]^{-1}\mathbf{S}_{12}^{B} \\ \mathbf{S}_{21}^{B}[\mathbf{I} - \mathbf{S}_{22}^{A}\mathbf{S}_{11}^{B}]^{-1}\mathbf{S}_{21}^{A} & \mathbf{S}_{22}^{B} + \mathbf{S}_{21}^{B}[\mathbf{I} - \mathbf{S}_{22}^{A}\mathbf{S}_{11}^{B}]^{-1}\mathbf{S}_{22}^{A}\mathbf{S}_{12}^{B} \end{bmatrix} \tag{A.22}$$

This proves that the converting two S-matrices to T-matrices, concatenating them via a simple matrix product, and converting the resulting T-matrix back to an S-matrix, yields mathematically the same result as the Redheffer star product, so long as the partial inversions are well-posed.

A.2. NVIDIA RTX A6000 datasheet link

https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/proviz-print-nvidia-rtx-a6000-datasheet-us-nvidia-1454980-r9-web%20(1).pdf

A.3. Enhanced Transmittance Matrix for alternate formulation

This proof below is to show that the square root method can be made to work with Enhanced Transmittance Matrices.

The enhanced transmittance matrix iteration scheme goes as follows:

$$\begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{b}_{i-1} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{i-1} & \mathbf{W}_{i-1} \\ -\mathbf{V}_{i-1} & \mathbf{V}_{i-1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_{i} & \mathbf{W}_{i} \\ -\mathbf{V}_{i} & \mathbf{V}_{i} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_{i} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{b}_i\mathbf{a}_i^{-1}\mathbf{X}_i \end{bmatrix} \tag{A.23}$$

If we simplify this, we get:

$$\begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{b}_{i-1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{i-1,i} & \mathbf{B}_{i-1,i} \\ \mathbf{B}_{i-1,i} & \mathbf{A}_{i-1,i} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{X}_i\mathbf{b}_i\mathbf{a}_i^{-1}\mathbf{X}_i \end{bmatrix} \tag{A.24}$$

$$\mathbf{a}_i = \mathbf{A}_{i-1,i} + \mathbf{B}_{i-1,i}\mathbf{X}_i\mathbf{b}_i\mathbf{a}_i^{-1}\mathbf{X}_i \tag{A.25}$$

$$\mathbf{b}_i = \mathbf{B}_{i-1,i} + \mathbf{A}_{i-1,i}\mathbf{X}_i\mathbf{b}_i\mathbf{a}_i^{-1}\mathbf{X}_i \tag{A.26}$$

Thus, with the new formulation, we get:

$$\mathbf{a}'_i = \mathbf{W}_{i-1}\mathbf{A}_{i-1,i}\mathbf{W}_i^{-1} + \mathbf{W}_{i-1}\mathbf{B}_{i-1,i}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{b}'_{i-1}(\mathbf{a}'_{i-1})^{-1}\mathbf{W}_i\mathbf{X}_i\mathbf{W}_i^{-1} \tag{A.27}$$

Simplifying a bit gives us:

$$\mathbf{a}'_i = \mathbf{W}_{i-1}(\mathbf{A}_{i-1,i} + \mathbf{B}_{i-1,i}\mathbf{X}_i\mathbf{W}_i^{-1}\mathbf{b}'_{i-1}(\mathbf{a}'_{i-1})^{-1}\mathbf{W}_i\mathbf{X}_i)\mathbf{W}_i^{-1} \tag{A.28}$$

Here, we take some creative licence to get:

$$\mathbf{W}_{i-1}^{-1}\mathbf{a}'_{i-1}\mathbf{W}_i = \mathbf{A}_{i-1,i} + \mathbf{B}_{i-1,i}\mathbf{X}_i(\mathbf{W}_i^{-1}\mathbf{b}'_i\mathbf{W}_{i+1})(\mathbf{W}_i^{-1}\mathbf{a}'_i\mathbf{W}_{i+1})^{-1}\mathbf{X}_i \tag{A.29}$$

From this, it is clear to see that:

$$\mathbf{b}'_i = \mathbf{W}_i\mathbf{b}_i\mathbf{W}_{i+1}^{-1} \tag{A.30}$$

Likewise, we can get:

$$\mathbf{a}'_i = \mathbf{W}_i\mathbf{a}_i\mathbf{W}_{i+1}^{-1} \tag{A.31}$$

And, from this, the term

$$\mathbf{b}'_i(\mathbf{a}'_i)^{-1} = \mathbf{W}_i\mathbf{b}_i\mathbf{a}_i\mathbf{W}_i^{-1} \tag{A.32}$$

This does bring somewhat of a problem; unlike with the S-matrix implementation, the intermediate step of the enhanced T-matrix approach with analogues is not the same as the regular entries from the enhanced T-matrix approach. In order for the last step to be the same, we actually need to find the eigendecomposition of the last layer.

However, this problem can be solved by propagating the system from the first layer backwards one step further, into the superstrate;

$$\begin{bmatrix}\mathbf{a}'_\mathbf{I}\\ \mathbf{b}'_\mathbf{I}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_{\mathbf{I},1} & \mathbf{B}_{\mathbf{I},1}\\ \mathbf{B}_{\mathbf{I},1} & \mathbf{A}_{\mathbf{I},1}\end{bmatrix}\begin{bmatrix}\mathbf{W}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{W}_1\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{W}_1\mathbf{X}_1\mathbf{b}_1\mathbf{a}_1^{-1}\mathbf{X}_1\mathbf{W}_1^{-1}\end{bmatrix} \tag{A.33}$$

For the superstrate (as well as the substrate), we actually have to compute the eigenvectors, both for the S-matrix approach as well as the enhanced T-matrix approach. However, since these are homogeneous layers, this can be done simply analytically. From this, we get:

$$\begin{bmatrix}\mathbf{I}\\ \mathbf{b}'_\mathbf{I}(\mathbf{a}'_\mathbf{I})^{-1}\end{bmatrix} = \begin{bmatrix}\mathbf{I}\\ \mathbf{b}_\mathbf{I}\mathbf{a}_\mathbf{I}^{-1}\end{bmatrix} \tag{A.34}$$

Since we've propagated all the way into the superstrate, we actually know that the forward-travelling waves are the incident waves, and the backward-travelling waves are the reflected waves, which leads us to the following equation:

$$\begin{bmatrix}\mathbf{I}\\ \mathbf{b}'_\mathbf{I}(\mathbf{a}'_\mathbf{I})^{-1}\end{bmatrix}\mathbf{T}_\mathbf{I} = \begin{bmatrix}\delta_{pq}\\ \mathbf{R}\end{bmatrix} \tag{A.35}$$

Where

$$\mathbf{T} = \prod_{i=0}^{N-1}\left((\mathbf{a}'_{N-i})^{-1}\mathbf{X}_{N-i}\right)(\mathbf{a}'_\mathbf{I})^{-1}\mathbf{T}_\mathbf{I} \tag{A.36}$$

However, no implementation of the Enhanced Transmittance Matrix Method was attempted in this work.