The afmm 1.4.5 user manual

A 3D full vectorial mode solver and propagator

Davide Bucci

October 3, 2018

Contents

Acknowledgments

I am deeply thankful to Bruno Martin for his friendly help through the detailed comprehension of the Aperiodic Fourier Modal Method. The philosophy of afmm as well as this manual were initially shaped during the frequent discussions we had during the last months of his Ph.D. at IMEP-LAHC, and during his post-doc at IPAG, in Grenoble, France. Alain Morand also participated to intensive comparison sessions with FDTD simulations.

Another person to whom I am deeply indebted is Jérôme Michallon, who carefully checked all my calculations and corrected several of the mistakes I did. He developed a very detailed understanding of the implementation details, and this allowed him to be active in the code development. Thanks to his work, afmm can now take into account symmetries present in the calculated structure. Jérôme also participated to an accurate revision of this document. Kudos to Elise Ghibaudo; she provided a very detailed feedback and spotted some bugs and errors in this manual.

I would also thank all the colleagues and the students who provided a very welcomed feedback about this software: Thomas Nappez, Marco Casale, Elsa Jardinier, Sandie de Bonnault.

Introduction

One of the most important things to be considered while studying photonic devices is knowing of their electromagnetic field propagation characteristics. For this reason, several numerical methods have been implemented, in order to study more or less arbitrary wave-guiding or scattering structures. An interesting development is the Aperiodic Fourier Modal Method (AFMM), proposed by Lalanne and Silberstein[27, 37], in order to apply to waveguides the Rigorous Coupled Wave Method (RCWA), originally developed for the study of diffraction gratings[34, 30]. Thanks to the refinements proposed through the years, this method has proven to be effective to treat in a very general way waveguides made with different technologies. It considers structures composed by successive sections of a given length, inside which the refractive index distribution is invariant in the propagation axis. The approach used by the AFMM consists in applying an artificial transverse spatial periodisation to each section. Its effects on the fields are then represented thanks an operator based on a truncated Fourier development and this truncation is the sole approximation done for solving Maxwell equations. Derivatives thus become algebraic operations and the propagation modes are identified as the eigenvectors of a matrix. The complete 3D propagation can finally be studied by imposing the fields continuity conditions at each interface between different sections.

Afmm is a 3D full vectorial mode solver and propagator based on the Aperiodic Fourier Modal Method. The afmm propagator is able to consider the case of straight and bent waveguides, since all the developments can be carried out in rectangular and in cylindrical coordinates as well. It should be noted that, even if the AFMM has been initially intended for the study of waveguides, the implemented method is quite general and can be used to study diffraction gratings, micro-structured solar cells [17] and many other things.

This manual begins with chapter 1, containing a detailed description of the technical and mathematical background. Even if a comprehension of each detail is probably not needed to use afmm, it is useful to have a global understanding of the Aperiodic Fourier Modal Method. In fact, the user must clearly know how to cope with the limitations of the method when treating a particular problem. Chapter 2 describes how to install and use afmm as well as the syntax used in the input and output files. A few examples are finally provided, in order to ease the understanding of the structure definition mechanism provided by afmm. In the annexes, examples of construction are given for several important matrices used by afmm calculations, the program interaction with Gnuplot is discussed, a description of the file formats used by afmm, a detailed change-log of this software is given and we list all publications related to afmm.

Chapter 1

Technical background

In this chapter, we present the mathematical development of the Fourier Modal Method in cylindrical coordinates which is the core of afmm. At first, we manipulate Maxwell equations to describe the longitudinal evolution of transverse magnetic and electric fields as the effect of a differential propagation operator. We begin by discussing a 2D mode solver, well suited even for bent waveguides having an high core/substrate index contrast. We then see how to deal with convergence issues appearing when discontinuities in the refractive index distribution affects the fields for the transverse magnetic (TM) polarisation. The Cartesian coordinate calculations can be then seen as a special case of the more general cylindrical coordinates analysis. For this reason, only in a second time we see how the Cartesian coordinates can be treated with a simple limit. The complete 3D propagation is tackled, via the formalism of the S matrix. We finally extend calculations to Bloch/Floquet modes of periodical structures.

1.1 Calculation of the transverse fields

We consider a structure composed by several sections, through which we want to propagate electromagnetic fields. *Each section has a refractive index distribution which is invariant along the propagation axis.* The aim of this paragraph is rewrite curl Maxwell equations in such a way we obtain a (differential) propagation operator which can be applied to the transverse components of the electromagnetic fields in a given section. The developments are done in cylindrical coordinates, following the approach described in [12]. When the components of the electric and magnetic fields $\mathbf{E}$ and $\mathbf{H}$ have an harmonic time dependance, developing time derivatives can be done in a simple way. First, we consider a structure whose cross section in the propagation direction does not change. Curl Maxwell equations can thus be written as follows, using the complex representation of $\mathbf{E}$ and $\mathbf{H}$ vectors, in an uncharged dielectric material:

What is a structure and a section in afmm.

$$\begin{cases} \nabla \wedge \mathbf{E} = & -\mathrm{j}\omega\mu\mathbf{H} \\ \nabla \wedge \mathbf{H} = & \mathrm{j}\omega\epsilon\mathbf{E} \end{cases} \tag{1.1}$$

where $\omega = 2\pi f$ and f is the time frequency of the fields, ϵ and μ are respectively the tensors of permittivity and permeability. For simplicity we consider them both diagonal in our analysis, and spatially varying inside the region of interest.

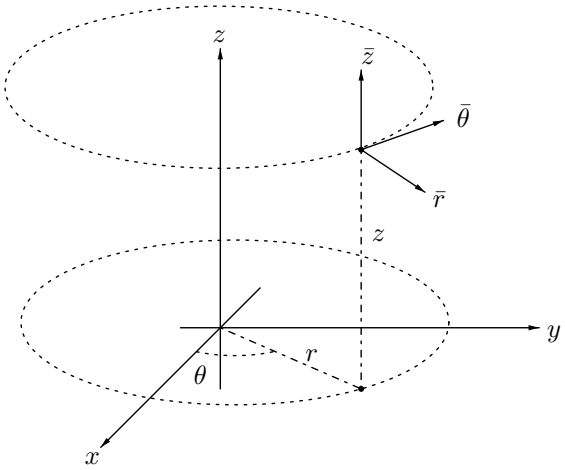

Figure 1.1: The cylindrical coordinate system.

Considering the cylindrical coordinate system shown in figure 1.1, we consider that the propagation axis is θ. Separating the r, z and θ components in equations (1.1), we obtain what follows:

$$\begin{cases} \frac{1}{r}\frac{\partial E_z}{\partial \theta} - \frac{\partial E_\theta}{\partial z} = & -\mu_r \mathrm{j}\omega H_r \\ \frac{\partial E_r}{\partial z} - \frac{\partial E_z}{\partial r} = & -\mu_\theta \mathrm{j}\omega H_\theta \\ \frac{1}{r}\left(\frac{\partial (rE_\theta)}{\partial r} - \frac{\partial E_r}{\partial \theta}\right) = & -\mu_z \mathrm{j}\omega H_z \\ \frac{1}{r}\frac{\partial H_z}{\partial \theta} - \frac{\partial H_\theta}{\partial z} = & \epsilon_r \mathrm{j}\omega E_r \\ \frac{\partial H_r}{\partial z} - \frac{\partial H_z}{\partial r} = & \epsilon_\theta \mathrm{j}\omega E_\theta \\ \frac{1}{r}\left(\frac{\partial (rH_\theta)}{\partial r} - \frac{\partial H_r}{\partial \theta}\right) = & \epsilon_z \mathrm{j}\omega E_z \end{cases} \tag{1.2}$$

Each one of the sections composing the structure must have the bending radius and the transverse distribution of the refractive index invariant along the propagation axis θ. We notice that the radial coordinate r appears in equations (1.2) and it is a spatially variable term. This implies that it should be treated in the same way as the other spatially variable terms, such as the permeabilities and permittivities.

The terms H_θ and E_θ can be extracted from equations (1.2) and injected again in the remaining expressions, to obtain a system of only 4 equations instead of 6, which describe the behaviour of the fields components transverse to the propagation axis (E_r, E_z, H_r and H_z). Those equations constitute a propagation operator giving the evolution of the fields while propagation. Regrouping in the first member the derivatives calculated with respect to θ

yields the following equations:

$$\begin{cases} \dfrac{\partial E_r}{\partial \theta} = & \dfrac{\partial}{\partial r}\left[\dfrac{r}{\mathrm{j}\omega\epsilon_\theta}\left(\dfrac{\partial H_r}{\partial z}-\dfrac{\partial H_z}{\partial r}\right)\right]+\mathrm{j}\omega\mu_z r H_z \\ \dfrac{\partial E_z}{\partial \theta} = & r\dfrac{\partial}{\partial z}\left[\dfrac{1}{\mathrm{j}\omega\epsilon_\theta}\left(\dfrac{\partial H_r}{\partial z}-\dfrac{\partial H_z}{\partial r}\right)\right]-\mathrm{j}\omega\mu_r r H_r \\ \dfrac{\partial H_r}{\partial \theta} = & \dfrac{\partial}{\partial r}\left[-\dfrac{r}{\mathrm{j}\omega\mu_\theta}\left(\dfrac{\partial E_r}{\partial z}-\dfrac{\partial E_z}{\partial r}\right)\right]-\mathrm{j}\omega\epsilon_z r E_z \\ \dfrac{\partial H_z}{\partial \theta} = & r\dfrac{\partial}{\partial z}\left[-\dfrac{1}{\mathrm{j}\omega\mu_\theta}\left(\dfrac{\partial E_r}{\partial z}-\dfrac{\partial E_z}{\partial r}\right)\right]+\mathrm{j}\omega\epsilon_r r E_r. \end{cases} \tag{1.3}$$

Those equations may be used in this form, or alternatively some derivatives may be developed further, in this way:

$$\begin{cases} \dfrac{\partial E_r}{\partial \theta} = \mathrm{j}\omega\mu_z r H_z + \dfrac{1}{\mathrm{j}\omega}\dfrac{1}{\epsilon_\theta}\left(\dfrac{\partial H_r}{\partial z}-\dfrac{\partial H_z}{\partial r}\right)+\dfrac{1}{\mathrm{j}\omega}r\dfrac{\partial}{\partial r}\left(\dfrac{1}{\epsilon_\theta}\right)\left(\dfrac{\partial H_r}{\partial z}-\dfrac{\partial H_z}{\partial r}\right)+ \\ \qquad +\dfrac{1}{\mathrm{j}\omega}\dfrac{r}{\epsilon_\theta}\left(\dfrac{\partial^2 H_r}{\partial r\partial z}-\dfrac{\partial^2 H_z}{\partial r^2}\right) \\ \dfrac{\partial E_z}{\partial \theta} = -\mathrm{j}\omega\mu_r r H_r + \dfrac{1}{\mathrm{j}\omega}r\dfrac{\partial}{\partial z}\left(\dfrac{1}{\epsilon_\theta}\right)\left(\dfrac{\partial H_r}{\partial z}-\dfrac{\partial H_z}{\partial r}\right)+\dfrac{1}{\mathrm{j}\omega}r\dfrac{1}{\epsilon_\theta}\left(\dfrac{\partial^2 H_r}{\partial z^2}-\dfrac{\partial^2 H_z}{\partial z\partial r}\right) \\ \dfrac{\partial H_r}{\partial \theta} = -\mathrm{j}\omega\epsilon_z r E_z - \dfrac{1}{\mathrm{j}\omega}\dfrac{1}{\mu_\theta}\left(\dfrac{\partial E_r}{\partial z}-\dfrac{\partial E_z}{\partial r}\right)-\dfrac{1}{\mathrm{j}\omega}r\dfrac{\partial}{\partial r}\left(\dfrac{1}{\mu_\theta}\right)\left(\dfrac{\partial E_r}{\partial z}-\dfrac{\partial E_z}{\partial r}\right)+ \\ \qquad -\dfrac{1}{\mathrm{j}\omega}\dfrac{r}{\mu_\theta}\left(\dfrac{\partial^2 E_r}{\partial r\partial z}-\dfrac{\partial^2 E_z}{\partial r^2}\right) \\ \dfrac{\partial H_z}{\partial \theta} = -\mathrm{j}\omega\epsilon_r r E_r - \dfrac{1}{\mathrm{j}\omega}r\dfrac{\partial}{\partial z}\left(\dfrac{1}{\mu_\theta}\right)\left(\dfrac{\partial E_r}{\partial z}-\dfrac{\partial E_z}{\partial r}\right)-\dfrac{1}{\mathrm{j}\omega}r\dfrac{1}{\mu_\theta}\left(\dfrac{\partial^2 E_r}{\partial z^2}-\dfrac{\partial^2 E_z}{\partial z\partial r}\right) \end{cases} \tag{1.4}$$

In this context, a propagation mode might be seen as a configuration of the electromagnetic field propagating without changing its shape, but *it can also be seen as an eigenfunction of the differential operator described by equations* (1.3). We try to represent fields via a base constituted by eigenfunctions, whose propagation is therefore relatively simple to being taken into account. Before doing that, we must represent all fields and spatially varying terms in a convenient way to perform algebraic calculations instead of calculating space derivatives. This can be done by exploiting the properties of the Fourier series, as it is described in the next paragraphs.

What is a mode in **afmm**.

1.2 Fourier series field development

1.2.1 Developing terms

The Fourier Modal Method is based on the *application of a periodization of the transverse permeability, permittivity and field distributions, by replicating a chosen calculation window, whose span is T_r and T_z in the r and z directions.* This is represented schematically in figure 1.2 and allows to describe each spatially varying term by means of Fourier series. Of course, this may give an infinite number of terms to be considered for solving equation (1.3).

Fourier series and **afmm**.

By taking into account only $2S_r - 1$ complex Fourier coefficients in the r axis and $2S_z - 1$ in the z axis, the expansion is truncated to a finite number of harmonics. Taking for example ϵ_r, the r component of the permittivity, we can

Truncation.

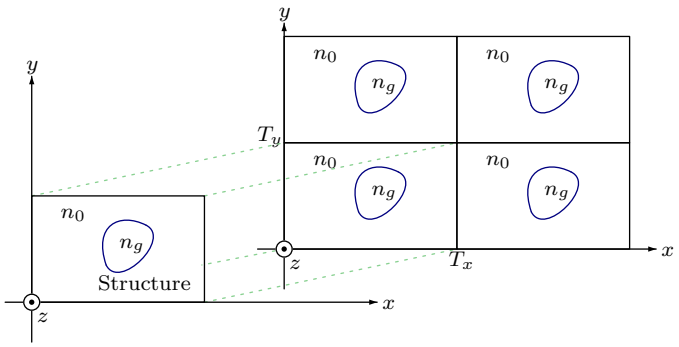

Figure 1.2: The periodization technique applied to develop the permittivity and permeability in Fourier series.

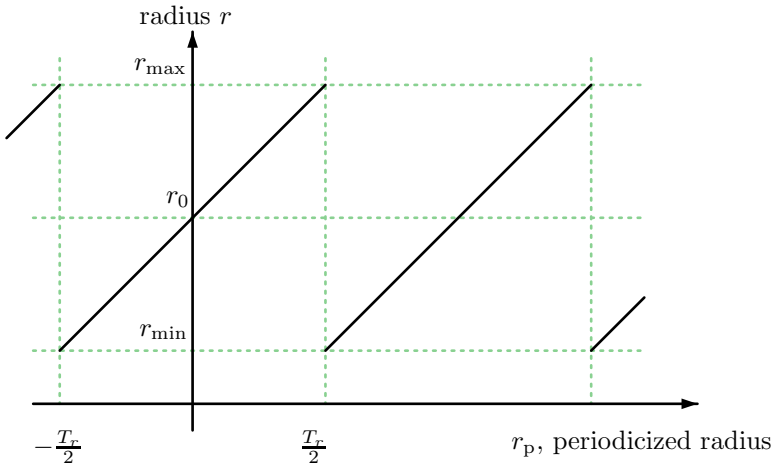

Figure 1.3: The effect of the periodization on the radius: it becomes a sawtooth.

write:

$$\epsilon_r(r,z) = \sum_{m=-(S_r-1)}^{S_r-1} \sum_{n=-(S_z-1)}^{S_z-1} P_{m,n} \mathrm{e}^{\mathrm{j}(m\nu_r r + n\nu_z z)}, \tag{1.5}$$

where $\nu_r = 2\pi/T_r$ is the fundamental spatial frequency on the r axis, $\nu_z = 2\pi/T_z$ is the fundamental space frequency on the z axis and $P_{m,n}$ is the m-order along r and n-order along z Fourier coefficient of the ϵ_r expansion.

In the same way, by developing in complex Fourier series all space variable terms of equations (1.3), over the periods $T_r = 1/\nu_r$ and $T_z = 1/\nu_z$:

$$E_r = \sum_{a_r} \sum_{a_z} A_{a_r,a_z} \mathrm{e}^{\mathrm{j}(a_r\nu_r r + a_z\nu_z z)} \tag{1.6}$$

$$E_z = \sum_{b_r} \sum_{b_z} B_{b_r,b_z} \mathrm{e}^{\mathrm{j}(b_r\nu_r r + b_z\nu_z z)} \tag{1.7}$$

$$H_r = \sum_{c_r} \sum_{c_z} C_{c_r,c_z} \mathrm{e}^{\mathrm{j}(c_r\nu_r r + c_z\nu_z z)} \tag{1.8}$$

$$H_z = \sum_{d_r} \sum_{d_z} D_{d_r,d_z} \mathrm{e}^{\mathrm{j}(d_r\nu_r r + d_z\nu_z z)} \tag{1.9}$$

$$\mu_r = \sum_{m_r} \sum_{m_z} M_{m_r,m_z} \mathrm{e}^{\mathrm{j}(m_r\nu_r r + m_z\nu_z z)} \tag{1.10}$$

$$\mu_z = \sum_{n_r} \sum_{n_z} N_{n_r,n_z} \mathrm{e}^{\mathrm{j}(n_r\nu_r r + n_z\nu_z z)} \tag{1.11}$$

$$\frac{1}{\mu_\theta} = \sum_{o_r} \sum_{o_z} O_{o_r,o_z} \mathrm{e}^{\mathrm{j}(o_r\nu_r r + o_z\nu_z z)} \tag{1.12}$$

$$\epsilon_r = \sum_{p_r} \sum_{p_z} P_{p_r,p_z} \mathrm{e}^{\mathrm{j}(p_r\nu_r r + p_z\nu_z z)} \tag{1.13}$$

$$\epsilon_z = \sum_{q_r} \sum_{q_z} Q_{q_r,q_z} \mathrm{e}^{\mathrm{j}(q_r\nu_r r + q_z\nu_z z)} \tag{1.14}$$

$$\frac{1}{\epsilon_\theta} = \sum_{r_r} \sum_{r_z} R_{r_r,r_z} \mathrm{e}^{\mathrm{j}(r_r\nu_r r + r_z\nu_z z)} \tag{1.15}$$

Here, symbols A, B, C and D represent the complex amplitudes of the transverse electric and magnetic fields, while M, N, O, P, Q and R are related to the permeability and permittivity.

Thanks to the artificial periodisation, the evolution of the radial coordinate $r_\mathrm{p}(r)$ appears to be a sawtooth wave, as shown in figure 1.3. Its Fourier m-th order term r_m can hence be analytically calculated as follows:

$$r_m = \begin{cases} r_0 & \text{if } m = 0, \\ \mathrm{j}\frac{T_r}{2\pi m} & \text{if } m \neq 0. \end{cases} \tag{1.16}$$

where r_0 and T_r are respectively the bending radius of the centre and the total size in r of the calculation window.

Considering as usual θ as propagation axis, the modal fields have a dependance on θ on the form $\exp(-\mathrm{j}\beta_\mathrm{s}\theta)$, where β_s is unknown and should be determined. This is advantageous to calculate all the partial derivatives with respect to θ appearing in equation 1.3. Equations (1.3) being linear, it is possible to exploit that with Fourier series, and translate equations term by term into matrix operations in the Fourier space. The goal of the following paragraphs will be to write equations (1.3) as an operator applicable to the fields as a matrix vector multiplication.

1.2.2 Convolutions

In equations (1.3), written in the spatial domain, some terms contain a product between space-varying quantities. If we consider the last part of the very first equation:

$$\mathrm{j}\omega\mu_z r H_z, \tag{1.17}$$

the $\mathrm{j}\omega$ does not pose any problem, since it is a constant (the wavelength at which the calculation is done, hence ω, is chosen at the very beginning). More complex is the product between μ, r and H_z, since all those terms are space-varying. The well-known convolution theorem states that they must be handled with convolutions in the Fourier domain. Hence, the problem is knowing how to translate that into convenient matrix operations.

Let us in particular focus on how the product r times H_z may be written. The first thing to do is calculate the Fourier coefficients for r, from equation (1.16). Then, we arrange them in a block-Toeplitz matrix which we will call S_T, constructed with the rules shown in equations (A.6) and (A.7), as described in paragraph A.1. At the same time, the harmonics $D_{dr,dz}$ constituting the z component of the magnetic field (still unknown, for the moment) are arranged[1] in a column vector which will be called $[H_z]$. It can be shown that the convolution corresponding to the two last terms of equation (1.17) is written as follows in the Fourier domain:

$$S_\mathrm{T}[H_z] \tag{1.18}$$

More complicated is now to consider the double convolution resulting by the presence of a μ_z. If possible, the product $\mu_z r$ can be calculated in the spatial domain, before taking the Fourier transform of the result. In some cases, however, an approximate result of the convolution can be conveniently written directly in the Fourier domain, in the following way:

$$\mathrm{j}\omega N_\mathrm{T} S_\mathrm{T}[H_x]. \tag{1.19}$$

In other words, the convolution between coefficients for representing μ_z and r is translated (quite brutally, indeed, as this is only asymptotically true [19]) into a product of Block-Toeplitz matrix. If this strategy is adopted, the matrix product being non commutative, there is not an unique way to write matrix order. For this and other more subtile reasons presented below, afmm offers a choice of different strategies via the `matdev` command.

[1] Packing them into a usable column vector is a little tricky. Details are given in paragraph A.1 in Appendix A.

1.2.3 Derivatives

The reason for which each space-varying term has been represented as a Fourier series is that derivatives are translated into a purely algebrical operator. Let us consider the following derivative:

$$\frac{\partial H_r}{\partial z}, \tag{1.20}$$

which can be thought as a differential operator applied to the H_z component of the field. In Fourier developments, it is well known that taking the derivatives translates into a multiplication with a frequency term. If D_{d_r,d_z} is the term corresponding to the d_r harmonics in r and d_z harmonics in z (see equation (1.8)), the corresponding term of the r derivative of H_z is $\mathrm{j}d_r\nu_r D_{d_r,d_z}$. The situation becomes more intricate when dealing with terms which are more complicated.

Two different strategies are applied in afmm:

- The one which is a legacy of the beginnings consists in preparing appropriate bloc-matrices containing the derivative terms ($d_r\nu_r$) and perform Hadamard multiplications (term by term, not line by column) with the other terms needing to be derived. Considering the propagation operator in the developed form, from equation (1.4), we have terms such as the following:

$$\frac{1}{\mathrm{j}\omega}\frac{\partial}{\partial r}\left(\frac{1}{\epsilon_\theta}\right)\frac{\partial H_r}{\partial z}. \tag{1.21}$$

  A convolution is represented between the terms coming from the development of r, the derivative of ϵ_θ^{-1} and the z derivative of H_r. In matrix form, this may be written with a modified-Toeplitz matrix formalism, as follows[2]:

$$-\frac{1}{\mathrm{j}\omega}R^{(z)}_{\mathrm{T},r}[H_z]. \tag{1.22}$$

  The minus sign comes from the two multiplications by the imaginary unit j coming out from the derivatives. All terms can be written using this notation. For example:

$$\frac{1}{\mathrm{j}\omega\epsilon_\theta}\frac{\partial^2 H_r}{\partial z^2} \leftrightarrow -\frac{1}{\mathrm{j}\omega}R^{(zz)}_{\mathrm{T}}[H_r], \tag{1.23}$$

$$-\frac{1}{\mathrm{j}\omega}r\frac{\partial}{\partial z}\left(\frac{1}{\mu_\theta}\right)\frac{\partial E_z}{\partial r} \leftrightarrow \frac{1}{\mathrm{j}\omega}S_{\mathrm{T}}O^{(r)}_{\mathrm{T},r}[E_z]. \tag{1.24}$$

  Paragraph A.4 gives some details about how the modified bloc-Toeplitz matrices are constructed.

- The second possibility, much more widespread in the literature, as well as probably cleaner yet equivalent, is to observe that the matrix times vector multiplication leads to a derivative representation, if the matrix contains

[2]The modified-Toeplitz formalism has been invented during the development of afmm and it is used in [11]. No one seems to understand it and it is undeniably heavy. The description is still in this document, because of the traces it has left in the source code. In the future, we will probably work more with the compact matrix notation which is commonly found in the literature...

only appropriate terms such as $d_r\nu_r$ in its diagonal. Most of the times, this is applied to equations (1.3). Matrices constructed in such a way (K_r stands for a r-derivative, K_z for a z-derivative) can be seen as operator which derive what it is put at their right. Things are probably clearer with an example of how a term in equation (1.3) can be translated:

$$\frac{\partial}{\partial r}\frac{r}{\epsilon_\theta}\frac{\partial H_r}{\partial z} \leftrightarrow K_r S_{\mathrm{T}} R_{\mathrm{T}} K_z [H_r] \tag{1.25}$$

1.2.4 A naive example

Considering the development rules described in paragraphs 1.2.2 and 1.2.3, we give an example of how equations (1.3) can be written with Fourier coefficients in a matrix form, using the second strategy for developing derivatives. We will see in a second time that those developments are a little bit naive, but they help understanding the general picture. Proceeding mechanically, we can write:

$$\begin{cases} \dfrac{\partial}{\partial\theta}[E_r] = \dfrac{1}{\mathrm{j}\omega}\mathrm{j}K_r S_{\mathrm{T}} R_{\mathrm{T}}(\mathrm{j}K_z[H_r] - \mathrm{j}K_r[H_z]) + \mathrm{j}\omega N_{\mathrm{T}} S_{\mathrm{T}}[H_z] \\ \dfrac{\partial}{\partial\theta}[E_z] = \dfrac{1}{\mathrm{j}\omega} S_{\mathrm{T}}\mathrm{j}K_z R_{\mathrm{T}}(\mathrm{j}K_z[H_r] - \mathrm{j}K_r[H_z]) - \mathrm{j}\omega M_{\mathrm{T}} S_{\mathrm{T}}[H_r] \\ \dfrac{\partial}{\partial\theta}[H_r] = -\dfrac{1}{\mathrm{j}\omega}\mathrm{j}K_r S_{\mathrm{T}} O_{\mathrm{T}}(\mathrm{j}K_z[E_r] - \mathrm{j}K_r[E_z]) - \mathrm{j}\omega Q_{\mathrm{T}} S_{\mathrm{T}}[H_r] \\ \dfrac{\partial}{\partial\theta}[H_z] = -\dfrac{1}{\mathrm{j}\omega} S_{\mathrm{T}}\mathrm{j}K_z O_{\mathrm{T}}(\mathrm{j}K_z[E_r] - \mathrm{j}K_r[E_z]) + \mathrm{j}\omega R_{\mathrm{T}} S_{\mathrm{T}}[H_r] \end{cases} \tag{1.26}$$

Apart from the introduction of coordinate-transform PML's, this development corresponds to the one set by the `matdev nd` command in afmm's jargon.

1.2.5 Eigenvalues and eigenvectors as propagation modes

In equations (1.26), a derivative towards the θ propagation axis still appears. If the field configuration is the one corresponding to a particular propagation mode, the derivative becomes just a phase term:

$$\frac{\partial}{\partial\theta} \leftrightarrow -\mathrm{j}\beta_{\mathrm{s}} \tag{1.27}$$

It is now possible to rewrite equations (1.3) using the Fourier developments in a compact matrix notation, as follows:

$$\frac{\omega\beta_{\mathrm{s}}}{r_0}\begin{pmatrix}[E_r]\\ [E_z]\\ [H_r]\\ [H_z]\end{pmatrix} = A_\theta \begin{pmatrix}[E_r]\\ [E_z]\\ [H_r]\\ [H_z]\end{pmatrix} \tag{1.28}$$

where each vector $[E_r]$, $[E_z]$, $[H_r]$ and $[H_z]$ is formed by unrolling the r and z space harmonics in order to obtain a column vector from $S_r S_z$ coefficients. The matrix A_θ can be seen as an algebraic operator which can be written:

$$A_\theta = \begin{pmatrix} 0 & 0 & X_1 & X_2 \\ 0 & 0 & X_3 & X_4 \\ Y_1 & Y_2 & 0 & 0 \\ Y_3 & Y_4 & 0 & 0 \end{pmatrix} \tag{1.29}$$

In fact, all space derivatives become algebraic operations which might be represented using the standard matrix product. As seen previously, there is not an univoque way of writing $X_1 \dots X_4$ and $Y_1 \dots Y_4$ and afmm offers several different strategies which can be chosen by the user. However, me we just have to know that *the matrix A_θ can be written*. In paragraph 1.4, we will see the details of alternative developments.

In fact, the problem of finding the modes of the waveguide is thus strongly related to the search of eigenvectors and eigenvalues of A. This is computationally intensive. In order to reduce the size of the matrices, it can be convenient to solve the equivalent second order problem, as follows:

$$\frac{\omega^2 \beta_s^2}{r_0^2} \begin{pmatrix} [E_r] \\ [E_z] \end{pmatrix} = B \begin{pmatrix} [E_r] \\ [E_z] \end{pmatrix} \tag{1.30}$$

where B is the following block matrix:

$$B = \begin{pmatrix} X_1 Y_1 + X_2 Y_3 & X_1 Y_2 + X_2 Y_4 \\ X_3 Y_1 + X_4 Y_3 & X_3 Y_2 + X_4 Y_4 \end{pmatrix} \tag{1.31}$$

Working on the second order problem implies a more compact representation of the propagation operator, using the B matrix. Finding eigenvalues and eigenvectors of that matrix is considerably faster than for the A matrix. Furthermore, this technique allows to reduce the memory occupation of the matrices, which is one of the main practical limitations to the number of spatial harmonics which can be taken into account.

1.3 Perfectly Matched Layers (PML)

A practical consequence of the periodisation described in section 1.2 is that we cannot consider each single structure comprised in the calculation window as completely independent to its neighbouring copies. For this reason, the introduction of Periodically Matched Layers has been proposed in order to reduce (or ideally eliminate) the influence of each repetition of the calculation window.

We will describe two different kinds of PML's:

- The first one implies the definition of an anisotropic material able to perfectly (or quite so...) absorb field.
- The second one makes use of a coordinate transform eventually coupled with an absorbing layer, mapping the finite calculation region into an infinite domain.

1.3.1 Anisotropic PMLs

The strategy adopted in the mode solver is to introduce layers of a certain thickness of a lossy and anisotropic material, in order to absorb as much as possible the radiated fields[37]. Those layers are placed at the borders of the calculation window, as it can be seen in figure 1.4 The structure of the PMLs differs depending on their position. There are three different types of PMLs:

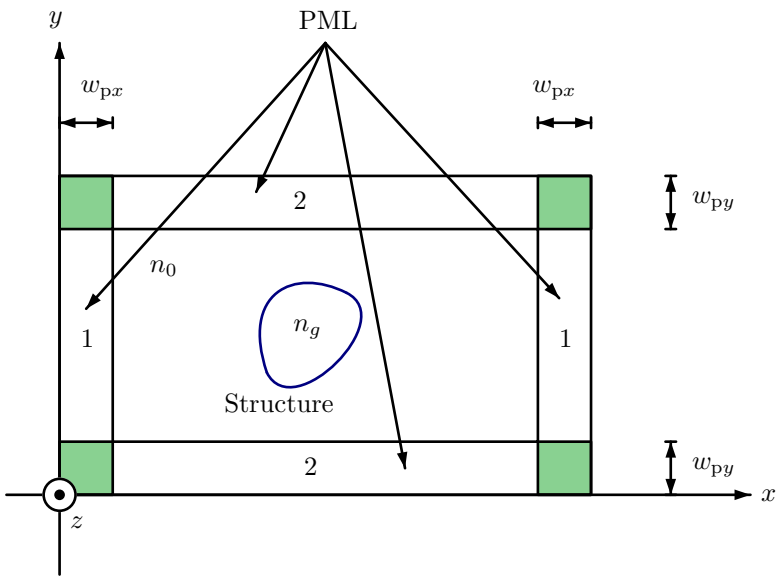

Figure 1.4: A representation of the calculation window completed with perfectly mathed layers, as proposed in [37].

- PMLs of type 1, placed parallel to the y direction, having a thickness of w_{px} and the following relative permeability and permittivity tensors:

$$\epsilon_r = \mu_r = \begin{pmatrix} 1/\alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix} \tag{1.32}$$

- PMLs of type 2, placed parallel to the x direction, having a thickness of w_{py} and the following relative permeability and permittivity tensors:

$$\epsilon_r = \mu_r = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & 1/\alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix} \tag{1.33}$$

- PMLs placed on the corners, with relative permeability and permittivity tensors:

$$\epsilon_r = \mu_r = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \alpha^2 \end{pmatrix} \tag{1.34}$$

Each PML layer thus depends on the complex parameter α which should be empirically chosen in order to achieve a good convergence speed.

1.3.2 Coordinate transform PMLs

Introduced by S.J. Hewlett and F. Ladouceur [21] and extended by J.P. Hugonin and P. Lalanne [22, 23] for the case of a vectorial mode solver, the idea consists

in using a nonlinear coordinate transform in order to map the finite calculation window in an infinite region. The main advantages of this technique consists in achieving a better extinction of the field far from the waveguide, thus allowing to obtain useful results even with a smaller calculation window in comparison with the anisotropic PMLs as described in 1.3.1.

Using the same mapping described in detail by Hugonin and Lalanne in [22], for the r axis we might define a matrix F, which is a block-Toeplitz matrix formed by the Fourier coefficients given by:

$$\begin{aligned} f_m = \delta_m - \frac{q_r}{2T_r}\left[\left(1+\frac{\gamma}{4}\right)\mathrm{sinc}\frac{mq_r}{T_r} + \frac{1}{2}\mathrm{sinc}\frac{mq_r}{T_r} - 1\right. \\ \left. + \frac{1}{2}\mathrm{sinc}\frac{mq_r}{T_r} + 1 - \frac{\gamma}{8}\mathrm{sinc}\frac{mq_r}{T_r} - 2 - \frac{\gamma}{8}\mathrm{sinc}\frac{mq_r}{T_r} + 2\right], \end{aligned} \tag{1.35}$$

where m is the order of the Fourier coefficient to be calculated, γ is a complex parameter, q_r is the total thickness of the PML in the r axis, δ_m is equal to 1 if $m = 0$ and to zero otherwise and $\mathrm{sinc}x = \sin(x)/x$. In an equivalent way, we can define a G matrix built with the same coefficients (1.35), but transposed on the z axis.

1.4 Convergence and discontinuities

Several strategies of writing the blocks $X_1 \ldots X_4$ and $Y_1 \ldots Y_4$ exist, showing different convergence behaviours. Two problems must be taken into account. The first one is how to develop derivatives and convolutions in the truncated case. As seen in paragraph 1.2.3, the very first versions of afmm employed a special notation, which we called modified Toeplix matrices. Successively, we adopted diagonal matrices K_r and K_z and ordinary matrix product. In our observations, no relevant difference could be pointed out in the convergence behaviour. The latter solution is however much simpler to be applied, leading to shorter and much more readable equations. On these, it is easier to implement what the solutions proposed below, improving the convergence in delicate cases.

The second problem is more serious and concerns the way the convolution is done when there are permittivity discontinuities in the structure being calculated. In the literature, it is well known that a naive development of the equations leads to an unsatisfactory convergence for modes in which the electric field is not continuous at interfaces (this is sometimes called a "TM" case). This affects equations (1.26). While this has long been considered as a Gibbs effect intrinsic to the truncated Fourier series, in reality it has ben observed that a wise development of the matrix leads to a far better convergence [26, 29].

Discontinuity of TM fields.

Let's consider the case of a discontinuous ϵ_z term in equation (1.3). In this equation, ϵ_z appears to be multiplied times E_z in the term $\mathrm{j}\omega\epsilon_z r E_z$ of $\partial H_z/\partial\theta$. Let neglect here the r term, considering that it is almost a constant in this context. The naive approach consists in writing matrices as follows (direct, or Laurent's rule):

$$\epsilon_z E_z \leftrightarrow Q_{\mathrm{T}}[E_z] \tag{1.36}$$

This yield to unsatisfactory convergence when E_z is discontinuous. In fact, Maxwell equations require that $\epsilon_z E_z$ remains continuous. In this case, an alter-

native approach may be used (inverse, or Li's rule[26, 29]):

$$\epsilon_z E_z \leftrightarrow Q^{-1}_{\mathrm{T,m1}}[E_z] \tag{1.37}$$

where $Q_{\mathrm{T,m1}}$ is the Toeplitz matrix formed by the Fourier coefficients of $1/\epsilon_z$:

$$\frac{1}{\epsilon_z} = \sum_{q_r}\sum_{q_z} Q_{\mathrm{m1},q_r,q_z}\, \mathrm{e}^{\mathrm{j}(q_r\nu_r r + q_z\nu_z z)} \tag{1.38}$$

This formulation ensures that the term $\epsilon_z E_z$ is continuous, even if, taken separately, terms ϵ_z and E_z are not.

Dealing with the orientation of discontinuities requires different developments via the "matdev" command

While it can be shown that for an infinite number of Fourier coefficients the formulations of equations (1.36) and (1.37) are equivalent[19], the differences in the truncated case can be relevant. *There is not a unique way in which a problem can be formulated using the direct or inverse rule just presented. Therefore, the appropriate strategy must be chosen by the user on a case-by-case basis.* The calculation core of afmm lets the user choose which matrix development should be used, in order that the convergence is optimum for the particular problem treated. In the following paragraphs, we describe the development strategy followed for the matrices. The title of the paragraphs reflects the terminology used by the afmm commands. Matdev an, af and be reflects a study which has been done to determine whether the multiplication order of the matrix when taking into account the PML's yielded different results[3].

1.4.1 Matdev an (obsolete)

This development strategy indicates a multiplication after matrix construction. This was the standard in in afmm up to version 1.2.3. A modified-Toeplitz matrix notation is used in the developments in order to calculate them in the Fourier space. For more information about the modified-Toeplitz notation, see paragraph 1.2.3. No attempt at coding the Li's rule has been done. The matrix definitions are the following:

$$\begin{aligned}
X_1 &= \mathrm{j}R^{(z)}_{\mathrm{i,T}}G - S_{\mathrm{T}}\left(R^{-1,(z)}_{\mathrm{i,T}r} + R^{-1,(rz)}_{\mathrm{i,T}}\right)FG\\
X_2 &= -\mathrm{j}R^{-1,(r)}_{\mathrm{i,T}}F + S_{\mathrm{T}}\left[-\omega^2 N_{\mathrm{T}} + \left(R^{-1,(r)}_{\mathrm{i,T}r} + R^{-1,(rr)}_{\mathrm{i,T}}\right)F^2\right]\\
X_3 &= S_{\mathrm{T}}\left[\omega^2 M_{\mathrm{T}} - \left(R^{-1,(z)}_{\mathrm{i,T}z} + R^{-1,(zz)}_{\mathrm{i,T}}\right)G^2\right]\\
X_4 &= S_{\mathrm{T}}\left(R^{-1,(r)}_{\mathrm{T}z} + R^{-1,(zr)}_{\mathrm{i,T}}\right)FG\\
Y_1 &= -\mathrm{j}O^{(z)}_{\mathrm{T}}G + S_{\mathrm{T}}\left(O^{(z)}_{\mathrm{T}r} + O^{(rz)}_{\mathrm{T}}\right)FG\\
Y_2 &= \mathrm{j}O^{(r)}_{\mathrm{T}}F + S_{\mathrm{T}}\left[\omega^2 Q_{\mathrm{T}} - \left(O^{(r)}_{\mathrm{T}r} + O^{(rr)}_{\mathrm{T}}\right)F^2\right]\\
Y_3 &= S_{\mathrm{T}}\left[-\omega^2 P_{\mathrm{T}} + \left(O^{(z)}_{\mathrm{T}z} + O^{(zz)}_{\mathrm{T}}\right)G^2\right]\\
Y_4 &= -S_{\mathrm{T}}\left(O^{(r)}_{\mathrm{T}z} + O^{(zr)}_{\mathrm{T}}\right)FG
\end{aligned} \tag{1.39}$$

The developments `matdev an` are very similar to `matdev af`, except that some minor tweaking and optimisations have been implemented in `matdev af`.

[3] I could not find a remarkable difference in the convergence rates. The big difference comes instead from the application of the Li's rule.

1.4.2 Matdev af (obsolete)

This development strategy indicates still a multiplication after construction as in afmm versions up to 1.2.3. Second order derivatives are developed as equations (1.4) and matrix developments are the same as those shown in equations (1.39). No attempt at coding the Li's rule has been done. Some optimizations have been put in place and the global memory occupation should be more efficient than `matdev an`.

1.4.3 Matdev be (obsolete)

This development strategy indicates still a multiplication before construction as in afmm versions up to 1.3. Second order derivatives are developed as equation (1.4) and a modified-Toeplitz matrix notation is used in the developments in order to calculate them in the Fourier space. No attempt at coding the Li's rule has been done. For more information about the modified-Toeplitz notation, see paragraph 1.2.3.

$$
\begin{aligned}
X_1 &= \mathrm{j}GR_{\mathrm{i,T}}^{-1,(z)} - S_{\mathrm{T}}FG\left(R_{\mathrm{i,T}r}^{-1,(z)} + R_{\mathrm{i,T}}^{-1,(rz)}\right)\\
X_2 &= -\mathrm{j}FR_{\mathrm{i,T}}^{-1,(r)} + S_{\mathrm{T}}\left[-\omega^2 N + F^2\left(R_{\mathrm{i,T}r}^{-1,(r)} + R_{\mathrm{i,T}}^{-1,(rr)}\right)\right]\\
X_3 &= S_{\mathrm{T}}\left[\omega^2 M - G^2\left(R_{\mathrm{i,Tz}}^{-1,(z)} + R_{\mathrm{i,T}}^{-1,(zz)}\right)\right]\\
X_4 &= S_{\mathrm{T}}FG\left(R_{\mathrm{i,T}z}^{(r)} + R_{\mathrm{i,T}}^{(rz)}\right)\\
Y_1 &= -\mathrm{j}GO^{(z)} + S_{\mathrm{T}}FG\left(O_{\mathrm{T}r}^{(z)} + O_{\mathrm{T}}^{(rz)}\right)\\
Y_2 &= \mathrm{j}FO^{(r)} + S_{\mathrm{T}}\left[\omega^2 Q_{\mathrm{T}} - F^2\left(O_{\mathrm{T}r}^{(r)} + O_{\mathrm{T}}^{(rr)}\right)\right]\\
Y_3 &= S_{\mathrm{T}}\left[-\omega^2 P_{\mathrm{T}} + G^2\left(O_{\mathrm{T}z}^{(z)} + O_{\mathrm{T}}^{(zz)}\right)\right]\\
Y_4 &= -S_{\mathrm{T}}FG\left(O_{\mathrm{T}z}^{(r)} + O_{\mathrm{T}}^{(rz)}\right)
\end{aligned}
\tag{1.40}
$$

1.4.4 Matdev nd

In this strategy, double derivative are not developed, thus less terms are present in the developments, done directly from equation (1.3). Derivatives are calculated via the matrix multiplication by diagonal matrices K_r and K_z. In the `nd` strategy, *no form of inverse-rule formulation is used for improved convergence of the TM (discontinuous) field components.* On the other hand, this formulation is perfectly symmetrical: rotating the structure of multiples of 90 degrees does not yield different results.

$$\begin{aligned}
X_1 &= -FK_r S_{\mathrm{T}} R_{\mathrm{i,T}} GK_z \\
X_2 &= FK_r S_{\mathrm{T}} R_{\mathrm{i,T}} FK_r - \omega^2 S_{\mathrm{T}} N \\
X_3 &= -S_{\mathrm{T}} GK_z R_{\mathrm{i,T}} GK_z + \omega^2 S_{\mathrm{T}} M \\
X_4 &= S_{\mathrm{T}} GK_z R_{\mathrm{i,T}} FK_r \\
Y_1 &= FK_r S_{\mathrm{T}} OGK_z \\
Y_2 &= -FK_r S_{\mathrm{T}} OFK_r + \omega^2 S_{\mathrm{T}} Q \\
Y_3 &= S_{\mathrm{T}} GK_z OGK_z - \omega^2 S_{\mathrm{T}} P \\
Y_4 &= -S_{\mathrm{T}} FK_r OGK_z
\end{aligned} \tag{1.41}$$

1.4.5 Matdev la (Lalanne-type developments)

In this strategy, like the one described in paragraph 1.4.4, double derivative are not developed. Derivatives are calculated via the matrix multiplication by diagonal matrices K_r and K_z. The issue concerning the poor convergence behaviour for discontinuous electric field components is tackled thanks to the strategy Philippe Lalanne developed in [24]. Even if it is not adapted to represent arbitrarily oriented index discontinuities, this approach is well fit to situations where an interface is either horizontal or vertical in the calculation window. A practical example (propagation modes in a slab waveguides) is discussed in paragraph 2.7.

$$\begin{aligned}
X_1 &= -FK_r S_{\mathrm{T}} R_{\mathrm{i,T}} GK_z \\
X_2 &= FK_r S_{\mathrm{T}} R_{\mathrm{i,T}} FK_x - \omega^2 S_{\mathrm{T}} N \\
X_3 &= -S_{\mathrm{T}} GK_z R_{\mathrm{i,T}} GK_z + \omega^2 S_{\mathrm{T}} M \\
X_4 &= S_{\mathrm{T}} GK_z R_{\mathrm{i,T}} FK_r \\
Y_1 &= FK_r S_{\mathrm{T}} OGK_z \\
Y_2 &= -FK_r S_{\mathrm{T}} OFK_r + \omega^2 S_{\mathrm{T}} [\alpha Q + (1-\alpha) Q_{\mathrm{m1}}] \\
Y_3 &= S_{\mathrm{T}} GK_z OGK_z - \omega^2 S_{\mathrm{T}} [(1-\alpha) P + \alpha P_{\mathrm{m1}}] \\
Y_4 &= -S_{\mathrm{T}} FK_r OGK_z
\end{aligned} \tag{1.42}$$

The case of $\alpha = 0$ works well when a single dielectric discontinuity is present parallel with the x axis., whereas $\alpha = 1$ works well when a single dielectric discontinuity is present parallel with the y axis (see paragraph 2.7).

1.4.6 Matdev las (Lalanne developments, with symmetries)

This development is equivalent to the one described in paragraph 1.4.5, but the presence of symmetries is taken into account. The implementation of symmetries is available only in the Cartesian coordinates.

1.4.7 Matdev nf (Normal field)

This type of development let the user take into account a normal vector field which describe in each point of the section the orientation of those index discon-

tinuities potentially leading to non continuous fields. So the correct development rule can be followed in each point of the section.

$$\begin{aligned}
X_1 &= -FK_r S_{\mathrm{T}} R_{\mathrm{i,T}} GK_z \\
X_2 &= FK_r S_{\mathrm{T}} R_{\mathrm{i,T}} FK_x - \omega^2 S_{\mathrm{T}} N \\
X_3 &= -S_{\mathrm{T}} GK_z R_{\mathrm{i,T}} GK_z + \omega^2 S_{\mathrm{T}} M \\
X_4 &= S_{\mathrm{T}} GK_z R_{\mathrm{i,T}} FK_r \\
Y_1 &= FK_r S_{\mathrm{T}} OGK_z - \omega^2 \Delta_z [N_r N_z] S_{\mathrm{T}} \\
Y_2 &= -FK_r S_{\mathrm{T}} OFK_r + \omega^2 S_{\mathrm{T}} (Q - \Delta_z [N_z^2]) \\
Y_3 &= S_{\mathrm{T}} GK_z OGK_z - \omega^2 S_{\mathrm{T}} (P - \Delta_r [N_r^2]) \\
Y_4 &= -S_{\mathrm{T}} FK_r OGK_z + \omega^2 \Delta_r [N_r N_z] S_{\mathrm{T}}
\end{aligned} \tag{1.43}$$

where:

$$\Delta_r = P - P_{\mathrm{m1}}^{-1} \tag{1.44}$$

$$\Delta_z = Q - Q_{\mathrm{m1}}^{-1} \tag{1.45}$$

and N_r and N_z are respectively the Toeplitz matrix of the r and z components of the normal vector field. Normal field developments offer a considerable freedom, coming with the additional cost of generating the normal field descriptions. Two additional files describing the x and y orientation of the normal field components must be specified to afmm.

1.4.8 Matdev nfs (Normal-field development, with symmetries)

This development is equivalent to the one described in paragraph 1.4.7, but the presence of symmetries is taken into account. The implementation of symmetries is available only in the Cartesian coordinates.

1.5 Characteristics of bent waveguides

While in the rectangular case it is straightforward to define an effective index, this definition in cylindrical coordinates is somewhat arbitrary. Our choice has been to normalise the propagation constant on r_0, the bending radius of the centre of the calculation window. The following definition of the effective index of bent waveguides is therefore adopted:

$$n_{\mathrm{eff}} = \frac{\lambda_{\mathrm{s}}}{r_0 k_0} \tag{1.46}$$

where $k_0 = 2\pi/\lambda$ is the wave number in the vacuum.

A micro-ring resonator can be seen as a guide having a constant bending radius which makes an angle of 2π. There is an interesting interpretation which relates the resonances of such a structure[11]. The real part of the complex effective index n_{eff} calculated for a guided mode allows to calculate the azimuthal order m at a given wavelength λ:

$$m = \frac{2\pi}{\lambda} r_0 \mathbb{R}\{n_{\mathrm{eff}}\} \tag{1.47}$$

where r_0 is the bending radius of the centre of our calculation window. If the azimuthal order m is integer, there is a resonance at that particular wavelength. The quality factor Q associated to this resonance can also be determined from the complex effective index:

$$Q = -\frac{\mathbb{R}\{n_{\text{eff}}\}}{2\mathbb{I}\{n_{\text{eff}}\}} \tag{1.48}$$

Since bent waveguides are intrinsically lossy, the ability of calculating correctly a quality coefficient depends strongly on the correct setup of the calculation window and the PMLs[11].

1.6 From cylindrical to Cartesian coordinates

Rectangular coordinates can be seen as a particular case of cylindrical coordinates, when the bending radius is sufficiently high to be considered infinite. There is however a small price to pay: the rectangular coordinate system which can be obtained in this way is left-handed. It can thus be interesting to see if our physical intuition is confirmed by our mathematical developments. In other terms, we want to see how equations are transformed when the average bending radius r_0 tends towards infinity.

Equation(1.16) shows an interesting thing. If the average bending radius r_0 increases, only the very first term of the Fourier development changes. The higher order harmonics depend from the width of the calculation region, but not from r_0. Thanks to the Toeplitz matrix structure, this means that the principal diagonal of the matrix S_{T} tends to become preponderant as r_0 increases. In other words, if we take a matrix norm, we can write:

$$\lim_{r_0 \leftrightarrow +\infty} \frac{\|S_{\text{T}} - r_0 I\|}{r_0} = 0 \tag{1.49}$$

where I is the unit matrix with the same size of S_{T}. These considerations allow us to substitute S_{T} with $r_0 I$ into equations, when the radius r_0 is sufficiently large. By simplifying the r_0 which comes from our definition of effective index,

Cylindrical	Rectangular
r	$-x$
z	y
θ	z

Table 1.1: Correspondance between the axis between the cylindrical and a right-hand rectangular coordinates systems.

we can write:

$$\begin{aligned}
\frac{X_1}{r_0} &= \frac{\mathrm{j}R_{\mathrm{i,T}}^{(z)}}{r_0} - \left(R_{\mathrm{i,T}r}^{-1,(z)} + R_{\mathrm{i,T}}^{-1,(rz)}\right) \\
\frac{X_2}{r_0} &= -\frac{\mathrm{j}R_{\mathrm{i,T}}^{-1,(r)}}{r_0} + \left(-\omega^2 N_{\mathrm{T}} + R_{\mathrm{i,T}r}^{-1,(r)} + R_{\mathrm{i,T}}^{-1,(rr)}\right) \\
\frac{X_3}{r_0} &= \omega^2 M_{\mathrm{T}} + R_{\mathrm{i,T}z}^{-1,(z)} - R_{\mathrm{i,T}}^{-1,(zz)} \\
\frac{X_4}{r_0} &= R_{\mathrm{T}z}^{-1,(r)} + R_{\mathrm{i,T}}^{-1,(zr)} \\
\frac{Y_1}{r_0} &= -\frac{\mathrm{j}O_{\mathrm{T}}^{(z)}}{r_0} + \left(O_{\mathrm{T}r}^{(z)} + O_{\mathrm{T}}^{(rz)}\right) \\
\frac{Y_2}{r_0} &= \frac{\mathrm{j}O_{\mathrm{T}}^{(r)}}{r_0} + \left(\omega^2 Q_{\mathrm{T}} - O_{\mathrm{T}r}^{(r)} - O_{\mathrm{T}}^{(rr)}\right) \\
\frac{Y_3}{r_0} &= -\omega^2 P_{\mathrm{T}} + O_{\mathrm{T}z}^{(z)} + O_{\mathrm{T}}^{(zz)} \\
\frac{Y_4}{r_0} &= O_{\mathrm{T}z}^{(r)} + O_{\mathrm{T}}^{(zr)}
\end{aligned} \tag{1.50}$$

By injecting these equations in the definition of the A matrix, and by adopting the definition of effective index proposed in equation (1.46), the bending radius r_0 can be simplified. By the way, this is why we put a term $1/r_0$ in front of the first term of equation (1.28). If the radius r_0 is big enough, an analogy between axis in rectangular and cylindrical coordinates can be traced and it is shown in table 1.1. Unfortunately, the coordinate system which come out is left handed, hence the minus signs appearing.

1.7 Implementation of symmetries

Maxwell equations have a certain degree of symmetry conservation. The case which is interesting in our context is to calculate electromagnetic field propagation into a structure whose cross section is symmetrical. In fact, if the excitation is symmetrical (respectively anti-symmetrical), the resulting field components will be symmetrical (respectively anti-symmetrical). Dealing only with these fields is advantageous, since their spatial Fourier transform can truncated to a certain frequency contains less terms. We develop here the theoretical background of the symmetry implementation in afmm as done by [9].

1.7.1 Symmetry and anti-symmetry

This section is yet to be completed...

A detailed description of the implementation of symmetries can be found in [32].

1.8 Propagation

A structure is a sequence of sections with specific characteristics.

The developments shown in the previous paragraphs have shown how to treat one section with a constant finite or infinite curvature radius. At first, we begin by giving a geometrical interpretation of the discussion seen above. Then, we adopt it in order to propagate the fields by imposing the continuity conditions at the interfaces between sections. *All sections of the structure share the number of harmonics retained in the calculations as well as the size of the calculation window, but can have different bending radiuses, PMLs, as well as a different refractive index distribution.*

1.8.1 Geometrical interpretation

The equation (1.30) represents an eigenvalue problem. The diagonalisation of the propagation operator in the Fourier space allows to change the base for the representation of the fields. Through the eigenvalues and eigenvectors (physically, the propagation modes of the structure), we can switch from the Fourier base to the modal base and vice versa. The former base is convenient to express the continuity of fields at the interface between two sections of the structure. The latter base is useful for the propagation, since the propagation operator is diagonal and each mode is associated to an eigenvector that can be propagated knowing its propagation constant.

In a given section of the structure, if we call W the matrix formed by the eigenvectors and B is the matrix representing the truncated propagation operator in the Fourier space, we can write:

$$B = WDW^{-1} \tag{1.51}$$

where D is the diagonal matrix formed by the eigenvalues. Each eigenvalue is associated to a propagative and a counter-propagative mode. In a given point of the section, at a given coordinate θ_0, we can thus write:

$$\bar{s}_{\mathrm{w}} = P^{+}_{\theta_0}\bar{s}^{+}_{\mathrm{w}} + P^{-}_{\theta_0}\bar{s}^{-}_{\mathrm{w}} \tag{1.52}$$

where $\bar{s}_{\mathrm{w}}$ is vector containing the amplitudes of the total field represented in the modal base, $\bar{s}^{+}_{\mathrm{w}}$ is the vector of amplitudes of the propagative modes, $\bar{s}^{-}_{\mathrm{w}}$ is the vector of amplitudes of the counter-propagative modes. We use the the letter w as an index, in order to indicate that we are dealing with the modal base. The matrices P^{+} and P^{-} are the matrices representing the phase shifts for the propagation in the two directions. We indicate with θ_0 the origin of the propagation axis and with θ the point considered inside the section.

$$P^{+}_{\theta_0}(\theta) = \begin{pmatrix} \mathrm{e}^{-\mathrm{j}\beta_0(\theta-\theta_0)} & 0 & 0 & \cdots \\ 0 & \mathrm{e}^{-\mathrm{j}\beta_1(\theta-\theta_0)} & 0 & \cdots \\ 0 & 0 & \mathrm{e}^{-\mathrm{j}\beta_2(\theta-\theta_0)} & \cdots \\ \cdots & & & \cdots \end{pmatrix} \tag{1.53}$$

$$P^-_{\theta_0}(\theta) = \begin{pmatrix} e^{j\beta_0(\theta-\theta_0)} & 0 & 0 & \cdots \\ 0 & e^{j\beta_1(\theta-\theta_0)} & 0 & \cdots \\ 0 & 0 & e^{j\beta_2(\theta-\theta_0)} & \cdots \\ \cdots & & & \cdots \end{pmatrix} \tag{1.54}$$

Expressed in the Fourier base, the total field amplitude vector is thus given by a simple coordinate change:

$$\begin{pmatrix} [E_r] \\ [E_z] \end{pmatrix} = W\bar{s}_{\mathrm{w}} = W\left(P^+_{\theta_0}\bar{s}^+_{\mathrm{w}} + P^-_{\theta_0}\bar{s}^-_{\mathrm{w}}\right) \tag{1.55}$$

To summarise, in each section of our structure we have a vector space represented by the field of the structure and we use two orthogonal bases which are the Fourier representation and the modal representation. Once the eigenvalues and eigenvectors of the propagation matrix B are calculated, we can switch from the former representation to the latter by simple matrix algebra.

1.8.2 Continuity of the fields through two adjacent sections

If we consider two adjacent sections t and $t+1$, they share an interface in which the transverse electric and magnetic fields must be continuous. In other words, we have:

$$\begin{pmatrix} [E_r] \\ [E_z] \\ [H_r] \\ [H_z] \end{pmatrix}(\theta_{t+1}) = \begin{pmatrix} [E_r] \\ [E_z] \\ [H_r] \\ [H_z] \end{pmatrix}(\theta_t) \tag{1.56}$$

Since as we have seen in paragraph 1.2 we express our problem for the electric field components $\hat{A}$ and $\hat{B}$, we must find a way to reconstruct the magnetic fields. From equation (1.56), we can write:

$$\frac{\partial}{\partial\theta}\begin{pmatrix} [E_r] \\ [E_z] \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}\begin{pmatrix} [H_r] \\ [H_z] \end{pmatrix} \tag{1.57}$$

and thus:

$$\begin{pmatrix} [H_r] \\ [H_z] \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}^{-1}\frac{\partial}{\partial\theta}\begin{pmatrix} [E_r] \\ [E_z] \end{pmatrix} \tag{1.58}$$

The derivative can be explicitly calculated by using equation (1.55):

$$\begin{pmatrix} [H_r] \\ [H_z] \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}^{-1} W\Lambda\left(P^+_{\theta_0}\bar{s}^+_{\mathrm{w}} - P^-_{\theta_0}\bar{s}^-_{\mathrm{w}}\right) \tag{1.59}$$

where:

$$\Lambda = \begin{pmatrix} -j\beta_0 & 0 & \cdots & 0 \\ 0 & -j\beta_1 & \cdots & 0 \\ \cdots & & & \\ 0 & 0 & \cdots & -j\beta_{N-1} \end{pmatrix} \tag{1.60}$$

where β_i, $i \in \{0, \cdots, N-1\}$ are the propagation constants of the modes of the section being considered. We can define the matrix of the eigenvectors of the magnetic field, as follows:

$$V = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}^{-1} W\Lambda \tag{1.61}$$

At the interface, the continuity of the transverse electric field can be expressed as follows:

$$W^{[t]}\left(P_{\theta_t}^{+[t]}(\theta_{t+1})\bar{s}_{\mathrm{w}}^{+[t]}+P_{\theta_t}^{-[t]}(\theta_{t+1})\bar{s}_{\mathrm{w}}^{-[t]}\right)=W^{[t+1]}\left(\bar{s}_{\mathrm{w}}^{+[t+1]}+\bar{s}_{\mathrm{w}}^{-[t+1]}\right) \quad (1.62)$$

While for the transverse magnetic field we can write:

$$V^{[t]}\left(P_{\theta_t}^{+[t]}(\theta_{t+1})\bar{s}_{\mathrm{w}}^{+[t]}-P_{\theta_t}^{-[t]}(\theta_{t+1})\bar{s}_{\mathrm{w}}^{-[t]}\right)=V^{[t+1]}\left(\bar{s}_{\mathrm{w}}^{+[t+1]}-\bar{s}_{\mathrm{w}}^{-[t+1]}\right) \quad (1.63)$$

The idea is that the Fourier base is very practical when imposing continuity conditions at the interfaces between two adjacent section, while the modal base is useful when dealing with propagation. The continuity conditions expressed in the modal base are given by equations (1.62) and (1.63). We must now find a way to calculate the different contributions given by propagative and counter-propagative fields, by avoiding the convergence problems which may arise.

1.8.3 The scattering matrix formalism

An useful strategy for calculating all the propagative and counter-propagative components of the field at each interface (and then propagate them through the sections) is given by the scattering matrix (often called S-matrix) formalism. We follow here the notations used by [31] and [28]. Our structure is composed by several adjacent sections and in each one the refractive index distribution is invariant through all the length of the section. For each section, we define the following matrix:

$$\begin{pmatrix}\bar{s}_{\mathrm{w}}^{+[t+1]}\\ \bar{s}_{\mathrm{w}}^{-[t]}\end{pmatrix}=\begin{pmatrix}t_{++}^{[t]} & r_{+-}^{[t]}\\ r_{-+}^{[t]} & t_{--}^{[t]}\end{pmatrix}\begin{pmatrix}\bar{s}_{\mathrm{w}}^{+[t]}\\ \bar{s}_{\mathrm{w}}^{-[t+1]}\end{pmatrix}=S^{[t]}\begin{pmatrix}\bar{s}_{\mathrm{w}}^{+[t]}\\ \bar{s}_{\mathrm{w}}^{-[t+1]}\end{pmatrix} \quad (1.64)$$

The name and the definition of this matrix derives directly from the classical scattering matrix used for the description of radio frequency devices. With matrix algebra, it can be shown (see for example [31] for detailed calculations) that the different blocks of the $S^{[T]}$ matrix can be calculated from the eigenvectors and eigenvalues matrices associated to the two adjacent sections t and $t+1$:

$$t_{++}^{[t]}=2\left(W^{[t]-1}W^{[t+1]}+V^{[t]-1}V^{[t+1]}\right)^{-1}P^{[t]} \quad (1.65)$$

$$r_{+-}^{[t]}=\left(W^{[t]-1}W^{[t+1]}+V^{[t]-1}V^{[t+1]}\right)^{-1}\left(-W^{[t]-1}W^{[t+1]}+V^{[t]-1}V^{[t+1]}\right)P^{[t+1]} \quad (1.66)$$

$$r_{-+}^{[t]}=\left(W^{[t+1]-1}W^{[t]}+V^{[t+1]-1}V^{[t]}\right)^{-1}\left(-W^{[t+1]-1}W^{[t]}+V^{[t+1]-1}V^{[t]}\right)P^{[t]} \quad (1.67)$$

$$t_{--}^{[t]}=2\left(W^{[t+1]-1}W^{[t]}+V^{[t+1]-1}V^{[t]}\right)^{-1}P^{[t+1]} \quad (1.68)$$

This situation is depicted in figure 1.5 If we have N sections in our structure, we should consider a global excitation composed by the vectors $\bar{s}_{\mathrm{w}}^{+[0]}$ and $\bar{s}_{\mathrm{w}}^{-[N-1]}$ for the input of our calculations. The former is the propagative excitation at the beginning of the structure and the latter is the counter-propagative excitation at the end of the structure. We then begin by calculating all the propagative and

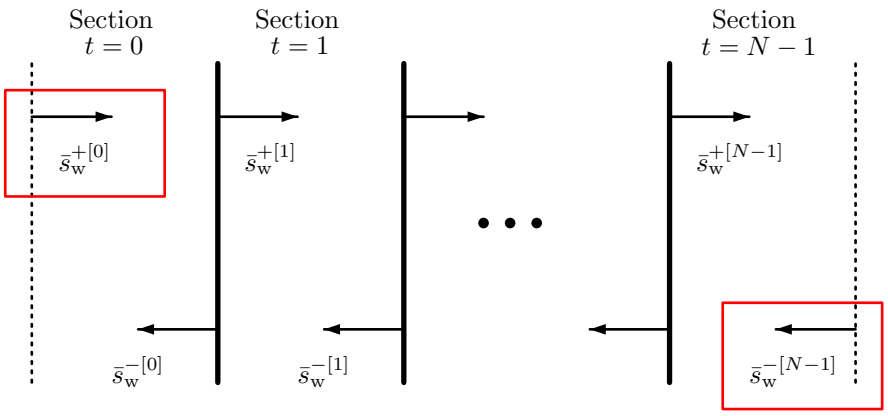

Figure 1.5: Schematic representation of the structure and the field excitations at each interface between N adjacent sections. The vectors surrounded by rectangles are the global propagative and counter-propagative excitations of the structure.

counter-propagative excitations for each sections, from the global excitations $\bar{s}_{\mathrm{w}}^{+[0]}$ and $\bar{s}_{\mathrm{w}}^{-[N-1]}$:

$$\begin{pmatrix} \bar{s}_{\mathrm{w}}^{+[t+1]} \\ \bar{s}_{\mathrm{w}}^{-[0]} \end{pmatrix} = \begin{pmatrix} T_{++}^{[t]} & R_{+-}^{[t]} \\ R_{-+}^{[t]} & T_{--}^{[t]} \end{pmatrix} \begin{pmatrix} \bar{s}_{\mathrm{w}}^{+[0]} \\ \bar{s}_{\mathrm{w}}^{-[t+1]} \end{pmatrix} \tag{1.69}$$

where the blocks are defined by recurrence:

$$T_{++}^{[t]} = t_{++}^{[t]} \left(I - R_{+-}^{[t-1]} r_{-+}^{[t]} \right)^{-1} T_{++}^{[t-1]} \tag{1.70}$$

$$R_{+-}^{[t]} = t_{++}^{[t]} \left(I - R_{+-}^{[t-1]} r_{-+}^{[t]} \right)^{-1} R_{+-}^{[t-1]} t_{--}^{[t]} + r_{+-}^{[t]} \tag{1.71}$$

$$R_{-+}^{[t]} = R_{-+}^{[t-1]} + T_{--}^{[t-1]} \left(I - r_{-+}^{[t]} R_{+-}^{[t-1]} \right)^{-1} r_{-+}^{[t]} T_{++}^{[t-1]} \tag{1.72}$$

$$T_{--}^{[t]} = T_{--}^{[t-1]} \left(I - r_{-+}^{[t]} R_{+-}^{[t-1]} \right)^{-1} t_{--}^{[t]} \tag{1.73}$$

where $t = 1, \cdots, N-1$. Now we can determine the excitations of the structure:

$$\bar{s}_{\mathrm{w}}^{-[t]} = \left(I - r_{-+}^{[t]} R_{+-}^{[t-1]} \right)^{-1} \left(r_{-+}^{[t]} T_{++}^{[t-1]} \bar{s}_{\mathrm{w}}^{+[0]} + t_{--}^{[t]} \bar{s}_{\mathrm{w}}^{-[t+1]} \right) \tag{1.74}$$

$$\bar{s}_{\mathrm{w}}^{+[t]} = t_{++}^{[t-1]} \bar{s}_{\mathrm{w}}^{+[t-1]} + r_{+-}^{[t-1]} \bar{s}_{\mathrm{w}}^{-[t]} \tag{1.75}$$

where $t = N-1, N-2, \cdots, 1$. And the field in each point of the structure can be calculated by propagating all the eigenmodes of the structure and by summing up all the contributions in the point in which the field should be calculated with equation (1.55).

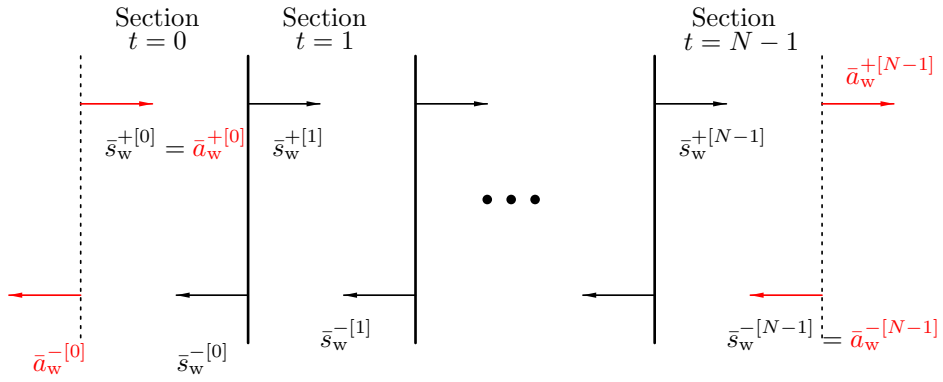

Figure 1.6: Calculations of excitation of Bloch-modes in the structure.

1.9 Bloch-mode calculation

Following the ideas developed in [13], Bloch-like modes of a structure can be calculated by supposing that the sections defined in afmm are repeated an infinite number of times, thus forming a periodic structure. This can be useful for the analysis of Bragg gratings or photonic crystal structures. In fact, in figure 1.6 calculating a Bloch-like mode consists in applying a sort of continuity between the vectors representing modes at the left and at the right of the structure. We called those vectors $\bar{a}_{\mathrm{w}}^{-[0]}$, $\bar{a}_{\mathrm{w}}^{+[0]}$ and $\bar{a}_{\mathrm{w}}^{+[N-1]}$ as well as $\bar{a}_{\mathrm{w}}^{-[N-1]}$. There is a difference between the vectors here called $\bar{a}_{\mathrm{w}}$ and the vectors employed above for the propagation, i.e. $\bar{s}_{\mathrm{w}}$. In fact, they are not always referred to the same plane and in certain cases we have to apply the appropriate propagation matrix. We can write:

$$\begin{aligned} \bar{a}_{\mathrm{w}}^{+[0]} &= \bar{s}_{\mathrm{w}}^{+[0]} \\ \bar{a}_{\mathrm{w}}^{-[0]} &= P^{[0]}\bar{s}_{\mathrm{w}}^{-[0]} \\ \bar{a}_{\mathrm{w}}^{+[N-1]} &= P^{[N-1]}\bar{s}_{\mathrm{w}}^{+[N-1]} \\ \bar{a}_{\mathrm{w}}^{-[N-1]} &= \bar{s}_{\mathrm{w}}^{-[N-1]} \end{aligned}. \tag{1.76}$$

By recalling the definition of the global scattering matrix seen in equation (1.69), we can write:

$$\begin{aligned} P^{[N-1]}\bar{s}_{\mathrm{w}}^{+[N-1]} &= P^{[N-1]}T_{++}^{[N-1]}\bar{s}_{\mathrm{w}}^{+[0]} + P^{[N-1]}R_{+-}^{[N-1]}\bar{s}_{\mathrm{w}}^{-[N-1]} \\ P^{[0]}\bar{s}_{\mathrm{w}}^{+[0]} &= P^{[0]}R_{-+}^{[N-1]}\bar{s}_{\mathrm{w}}^{+[0]} + P^{[0]}T_{--}^{[N-1]}\bar{s}_{\mathrm{w}}^{-[N-1]} \end{aligned}. \tag{1.77}$$

By putting at the first member all terms containing s referred to section 0 and by recognizing a vectors with definitions seen in equation (1.76), we can rewrite equation (1.77) in the following matrix form:

$$\begin{pmatrix} P^{[N-1]}T_{++}^{[N-1]} & 0 \\ P^{[0]}R_{-+}^{[N-1]} & -I \end{pmatrix} \begin{pmatrix} \bar{a}_{\mathrm{w}}^{+[0]} \\ \bar{a}_{\mathrm{w}}^{-[0]} \end{pmatrix} = \begin{pmatrix} I & -P^{[N-1]}R_{+-}^{[N-1]} \\ 0 & -P^{[0]}T_{--}^{[N-1]} \end{pmatrix} \begin{pmatrix} \bar{a}_{\mathrm{w}}^{+[N-1]} \\ \bar{a}_{\mathrm{w}}^{-[N-1]} \end{pmatrix}. \tag{1.78}$$

A Bloch-mode is characterised by a quasi-periodic behaviour. In other words, the propagative and counter-propagative fields at the structure's boundaries can

be related as follows:

$$\begin{pmatrix} \bar{a}_{\mathrm{w}}^{+[N-1]} \\ \bar{a}_{\mathrm{w}}^{-[N-1]} \end{pmatrix} = \exp(\mathrm{j}\beta_{\mathrm{b}} L) \begin{pmatrix} \bar{a}_{\mathrm{w}}^{+[0]} \\ \bar{a}_{\mathrm{w}}^{-[0]} \end{pmatrix}, \tag{1.79}$$

where L is the total lenght of the structure. By combining equations (1.78) with (1.79) we can relate the Bloch-mode search with a well known generalised eigenvalue problem, as visible in equation (1.80):

$$\begin{pmatrix} P^{[N-1]} T_{++}^{[N-1]} & 0 \\ P^{[0]} R_{-+}^{[N-1]} & -I \end{pmatrix} \begin{pmatrix} \bar{a}_{\mathrm{w}}^{+[0]} \\ \bar{a}_{\mathrm{w}}^{-[0]} \end{pmatrix} = \alpha_{\mathrm{b}} \begin{pmatrix} I & -P^{[N-1]} R_{+-}^{[N-1]} \\ 0 & -P^{[0]} T_{--}^{[N-1]} \end{pmatrix} \begin{pmatrix} \bar{a}_{\mathrm{w}}^{+[0]} \\ \bar{a}_{\mathrm{w}}^{-[0]} \end{pmatrix}, \tag{1.80}$$

where $\alpha_{\mathrm{b}} = \exp(\mathrm{j}\beta_{\mathrm{b}} L)$. Standard numerical techniques can be exploited for the calculation of the generalised eigenvalues and eigenvectors. The calculation of the propagation constant β_{b} and therefore of an equivalent effective index $n_{\mathrm{eff}} = \beta_{\mathrm{b}}/k_0$ involves the evaluation of a complex logarithm and encounters a difficulty. In fact, the complex logarithm is a multivalued function and it is difficult to determine the correct branch of the Riemann-surface to be used for the calculation. This problem is related to the well known issue of phase unwrapping which arises in several interferometric systems. If α_{b} is a generalised eigenvalue, we can write:

$$n_{\mathrm{eff}} = \frac{\mathrm{Log}(\alpha_{\mathrm{b}}) - \mathrm{j}2\pi p}{k_0 L} \tag{1.81}$$

where p is an integer which determines on which branch of the Riemann surface the calculation is done and $\mathrm{Log}(\alpha_{\mathrm{b}})$ is the principal value of the complex logarithm. Another more intuitive interpretation of p is how many integer multiples of 2π should be considered for the phase, while propagating the field in the structure. In fact, the principal part of the logarithm gives only the fractionary part of the phase.

1.10 Conclusion

In this chapter, we presented the main mathematical developments used for the implementation of a 2D Fourier Modal Method mode solver implementation. In particular, we introduced a shorthand notation useful to write derivatives and products in a truncated 2D Fourier space, using modified block-Toeplitz matrices. This notation has been used to derive a version applicable to cylindrical coordinates. The following chapter will be dedicated to the use of the afmm program, the tool which has been developed to perform complete 3D analysis of the field propagation using the Aperiodic Fourier Modal Method.

Chapter 2

Installing and using afmm

The afmm tool is an Aperiodic Fourier Modal Method 2D mode solver and 3D electromagnetic field propagator. As it can be seen in the detailed theoretical description given in chapter 1, it is based on the representation of the fields and refractive index distribution in a truncated Fourier series. The derivatives present in the Maxwell equations are thus translated to algebraic operations. Afmm defines a certain number of commands which allow to specify the input structure, to launch the calculations and handle the results. This chapter first illustrates how to install and use the program. Afmm is a command-driven software which can be used interactively or with script files. A detailed description of each available command is given and then a few examples are commented, to show how to handle some classic simulation problems.

2.1 Compiling and installing afmm

You may skip all those geeky details if you have afmm already preinstalled on a computer.

The afmm source code is written in C++, using the GCC compiler and the GNU Make utility. It should run on every platform on which a standard compliant compiler is available, provided that all the needed libraries are given during the compilation.

If you are lucky, a binary distribution of afmm is available for your operating system. In this case, you may just need to copy the provide file somewhere in your disk and start using afmm. In other cases, you may need to compile it from sources. This may allow to fine-tune the optimisation options for your particular machine and somehow increase the overall performances. All what follows refers to a typical UNIX or compatible system such as Linux, Solaris or MacOSX, but afmm can be compiled and installed on Microsoft Windows thanks to tools like Cygwin[15] or MinGW[33].

Afmm makes use of the LAPACK[8, 5] numerical library, which should have been installed in the host system, as well as the BLAS[10, 2] library, needed by LAPACK. If an optimised version of BLAS is not available for the host system, we strongly recommend using ATLAS[38, 1] or OpenBLAS[7]. Installing those libraries may require a Fortran compiler on your computer. The libf2c[6] library (Fortran to C adapter) should be included when linking afmm object files with LAPACK and BLAS. Since version 1.0.2 of afmm, the FFTW (a nice acronym for the "Fastest Fourier Transform in the West"[18, 3]) is used for calculating

the structure and mode profiles. You may refer to the documentation provided with each of those libraries to know how to configure and compile them.

Afmm adopts Gnu autotools and a `./configure` script is provided, which automatically tries to determine the best software configuration useful for compiling the code, as well as how to find the required libraries.

Once the makefile has been configured correctly for your system, type `make all` at the terminal prompt, in the directory in which you have copied afmm's sources.

In order to compile afmm, you should perform the following steps:

1. Download and compile LAPACK on your system, take note of the path of the file `liblapack.a`

2. Download and compile ATLAS on your system, take note of the path of the files `libatlas.a`, `libf77blas.a`[1]

3. Download and compile FFTW3; you need the file `libfftw3.a`

4. Download and compile the f2c library; you need the file `libf2c.a`

5. Launch `./configure` to determine the best compilation configuration for afmm. If the configuration fails, due to missing libraries, you can explicitely specify where to find them via explicit linker options:

   ```
   ./configure "LDFLAGS=-Ldir1␣-Ldir2␣-Ldir3..."
   ```

   You might also find useful to use the CXXFLAGS option to specify compiler options. For example, with the following call the `-g` option is passed to the compiler (this means that some debug information is retained):

   ```
   ./configure CXXFLAGS=-g
   ```

   Launching `./configure --help` shows some useful information concerning other configuration flags.

6. From the build directory you should type `make all` in order to compile all the files.

7. If everything has gone ok, you may launch afmm with the command `./afmm` from the build directory.

In some cases, you may have to manually specify the compilation conditions, the position of the different libraires. Here is an example which worked on a Linux machine:

```
export "LDFLAGS=-L/users/labos/imep/bucci/libs";export \
"LIBS=-lgfortran";export "BLAS=-lptf77blas␣-lptcblas␣-latlas";export\
"LAPACK=-lptlapack";export LAPACK_ADD_FINAL_UNDERSCORE=1; export\
BLAS_ADD_FINAL_UNDERSCORE=1;export CXXFLAGS="-O3";./configure\
 CC=c++ CPPFLAGS=-I/users/labos/imep/bucci/libs/include

make
```

[1] You may need `libptf77blas.a` instead of `libf77blas.a` if you chose to compile ATLAS in order to use more than one thread in a multi-core environment.

Different conventions exist when calling Fortran routines from C/C++. Some compilers add an underscore to the function names before or after the name of the function itself. For example, the BLAS routine `zgetri` may become `_zgetri` or `zgetri_` depending on the conventions of the Fortran compiler. Therefore, the following variables may be put equal to 1 depending on the particular convention adopted:

- `LAPACK_ADD_FINAL_UNDERSCORE`
- `LAPACK_ADD_LEADING_UNDERSCORE`
- `LAPACK_ADD_BOTH_UNDERSCORES`
- `BLAS_ADD_FINAL_UNDERSCORE`
- `BLAS_ADD_LEADING_UNDERSCORE`
- `BLAS_ADD_BOTH_UNDERSCORES`

Once a successful compilation has been done, there are some automatic tests available in the directory `test`. They will be performed if you launch the `run_all_tests.sh` Bash script present there. A successful report looks as follows:

```
   A F M M

   Automatic test suite
   by Davide Bucci 2013-2015

 NOTE: the following tests will run ../../afmm

Testing for loops:            OK
Testing for command (2 loops) OK
Testing layer decomposition:  OK
Testing if command            OK
Testing load command          OK
Testing simple calculations:  OK
Testing simple mode search:   OK
Testing Bloch mode calc. TE:  OK
Testing Bloch mode calc. TM:  OK
Testing pl. wave exc. angles  OK
Testing symmetries:           OK
Testing multithreading:       OK
Testing madev strategies:     OK
Testing multi-propagation:    OK
Testing multi-propagation 2:  OK
Testing monitor:              OK
Testing orientation of fft:   OK
Testing file read/write :     OK

All is Ok! :-)
```

2.2 Conventions

You must know this.

Afmm can be used in two ways. If a file is specified in the command line, afmm reads it and processes each line. If not, afmm can be used interactively from the command line. In each case, the simulation is described to the software via commands, often requiring a certain number of arguments. The commands

may be used to specify the calculation window, the wavelength to be used, the structure and all calculations that afmm should do. In afmm is in the interactive mode, those commands are read and executed from the command line[2]. If an input file is provided while launching the software, it will be scanned and if valid commands are found, they will be parsed and executed. In the non-interactive mode, afmm quits just after the last command in the file.

A command is formed by the command name and several parameters separated by spaces. If you want to use a space in a single parameter (for example for a file name), you should use double quotes. All commands and parameters should follow the following conventions:

- All geometrical sizes and distances are given in meters, and so does the wavelength.
- The centre of coordinates is the centre of the calculation window.
- The term *structure* refers to a succession of *sections*. They share the transverse size of the calculation window, but each section may be straight or have a certain curvature radius. The length of each section can be arbitrary.
- If a section is bent, the curvature radius is calculated from the centre of the calculation window and it is positive for a clockwise bent and negative for a counterclockwise bent.
- In order to maintain a coherent notation for straight and curved sections, inside each of them, the transverse coordinates will be always called x and y, whereas the propagation axis will be called z, even if a curvature is considered. In this case, z is a curvilinear coordinate referred to the centre of the calculation window. Be careful that the need to combine cylindrical and Cartesian coordinates leads to a left-hand Cartesian system.

Three classes of commands can be identified:

- *Commands which refer to the whole structure*, specifying the simulation conditions affecting all the structure (for example, the wavelength, the number of retained harmonics in the calculations or the size of the transverse calculation window...).
- *Section-level commands*, referring to parameters of the current section of the structure.
- *Other commands*, which do not specify the structure itself, but control the behaviour of afmm.

The commands of those three classes will be described separately in the following sections.

2.3 General structure-level commands

In this section we will describe the syntax of each structure-level command available with afmm.

[2] I *strongly* recommend the GNU `rlwrap` utility for a much better user experience while using the interactive mode[35].

2.3.1 SIZE: Set calculation window size

Define the size (in meters) of the calculation window. This is the transverse size of the volume in which the field can be calculated.

Usage: `size sx sy`

Parameters:

- `sx`: total x size of the calculation window T_x.
- `sy`: total y size of the calculation window T_y.

For example, the command:
defines a $2\,\mu\text{m} \times 2\,\mu\text{m}$ calculation window.

2.3.2 HARMONICS: Set number of space harmonics to be used

This command specifies the number of harmonics to be considered in the Fourier decomposition of the input structure. The harmonics specified here (including each a positive and a negative term except for DC) are employed for the representation of the structure, i.e. the spatial distribution of the permeability and permittivity. To be more explicit, if you are working in cylindrical coordinates and consider the spatial distribution in the cross-section $epsilon(r, z)$ you get the expression (1.5) in the manual at page 15. The number of harmonics is therefore S_r and S_z in the r and z direction respectively. If you are working in Cartesian coordinates, of course the quantity being represented is $epsilon(x, y)$ and the number of harmonics is S_x and S_y. This yields respectively $2S_x - 1 \times 2S_y - 1$ coefficients for the Fourier representation of the structure, enough to calculate convolutions with S_x times S_y coefficients for the electric and magnetic fields. If S_x and S_y are odd numbers, this yields $(S_x + 1)/2 \times (S_y + 1)/2$ harmonics for the representation of electric and magnetic fields.

Usage: `harmonics Sx Sy`

Parameters:

- `Sx`: number of harmonics on the x axis (or r axis if the structure is bent).
- `Sy`: number of harmonics on the y axis (or z axis if the structure is bent).

Examples:

```
harmonics 43 43
```

Use during calculations 43 harmonics in each direction, which means the continuous term as well as ± 42 harmonics. This gives a result described by a continuous term plus ± 21 harmonics in the x and y direction. The memory occupation needed for the calculations is $O(S_x S_y \times S_x S_y)$. There are no restrictions to the number of harmonics but the finite memory of the computer or workstation which is being used. It is better to use an odd number of harmonics. This command should be placed at the very beginning of the calculation.

Instead, with:

```
harmonics 5 1
```

we obtain that:

- The permeability and permittivity will be represented with 11 coefficients in x (DC+fundamental+harmonics up to order 4) and 1 coefficient in y (DC only).
- The resulting calculated electric and magnetic fields will contain 5 coefficients in x (DC+fundamental+harmonics up to order 2) and 1 coefficient in y (DC only).

2.3.3 WAVELENGTH: Set the wavelength to be used

Usage: `wavelength lambda`

Return value: `ans` is set to `lambda`.

Parameters:

- `lambda`: the wavelength to be used (in vacuum).

For example:

```
wavelength 1550e-9
```

sets to 1550 nm the wavelength (in vacuum) used in the simulations.

2.3.4 SOLVE: Find propagation modes

This command launches the mode solving calculations for all the defined sections. The calculation window and the structure should have been completely defined by the time this command is invoiced. All propagation modes of the structure are obtained, by calculating the eigenvectors of the matrix B defined in equation (1.31), for all sections of the structure. The number of modes obtained is quite high (guided modes as well as a discretization of radiated modes are obtained) and thus a mechanism to select certain propagation modes to be considered "interesting" has been put in place.

The minimum and maximum value of the effective refractive index to consider a mode as interesting can be set by using the `lowindex` and `highindex` commands. In this case, the `solve` command writes some information about the modes whose effective index falls inside the given interval. An alternative way to select interesting modes in bent structures is to perform the selection on the basis of the azimuthal order. See the command `order`. This calculation implies the construction and the diagonalisation of a very big matrix (see paragraph 1.2). The memory occupation is $O(S_xS_y \times S_xS_y)$ where S_x and S_y represent the number of harmonics in the x and y directions. The time needed to perform the operation strongly depends by the machine used, and it is approximatively $O((S_xS_y)^3)$ since it is basically the time needed by the QR factorisation used in the eigenvalue calculation. If a high number of sections are to be considered, a benefit may be obtained by the `parallel` command, in order to launch a multithreaded calculation, considering more than one section in parallel.

Usage: `solve`

Return value: after `solve` is called, `ans` contains an array whose elements are alternatively the real and the imaginary part of the effective index of the "interesting" modes found by `solve`. Therefore, the size of the array will be twice the number of "interesting" modes.

2.3.5 WANTS: Save information for later field calculations

This command should be used before calling `solve` and specify that some information useful for calculating the magnetic field or the propagation will not be erased once the eigenvalues have been calculated at the end of the calculations performed by the `solve` command. This is normally done to optimise the memory requirements when not every field component is needed for the upcoming calculations.

Usage: `wants h`

Parameters:

- `h`: can be `{Hx|Hy|both|propagation}`. Specifies that the user wants to calculate the magnetic field or the propagation after calling `solve`.

2.3.6 PROPAGATION: Propagate fields in the structure

Once all the propagation matrices have been assembled, propagate the excitation fields defined and write on a file the results. In order to calculate the propagation of the field, the user should enter the commands `wants propagation` before `solve` and then `assemble`. The excitation fields should be specified with `excitation`. The output file is organised in a Gnuplot-compatible format (see paragraph D.1).

Usage: `propagation e rimc dz nx ny filename`

Parameters:

- `e`: must be `{Ex|Ey|Ez|Ex2|Ey2|Ez2|Hx|Hy|Hz|Dx|Dy|Dz}` and indicates the field component to be calculated. The notation `2` after the field component stands for an alternative and improved representation of the electric field, which requires the adoption of the normal field technique (see `matdev`). The index 2 refers to the improved representation technique of the fields, as in [25].
- `rimc` must be `{r|i|m|c}` and indicates if the real part should be written (`r`), the imaginary part (`i`), the magnitude (`m`) or the complex value (`c`) of the fields
- `dz` propagation step to be used for the output file. If there are curved sections, this is the increment which is applied to the curvilinear coordinate passing in the centre of the calculation window.
- `nx` number of x points in the output file
- `ny` number of y points in the output file
- `filename` the name of the output file which will contain the field.

Note: more than a field component can be written with a single `propagation` command. You can combine several of them with a comma, like in the following example:

```
propagation Ex,Ey,Ez m 5e-9 101 101 f_x.txt,f_y.txt,f_z.txt
```

you might notice how in this case you must specify several output files with the same technique, namely `f_x.txt`, `f_y.txt` and `f_z.txt`. This single call is formally equivalent to:

```
propagation Ex m 5e-9 101 101 f_x.txt
propagation Ey m 5e-9 101 101 f_y.txt
propagation Ez m 5e-9 101 101 f_z.txt
```

with the important advantage that, if the `parallel` command is used, the generation of the three files will be done exploiting the multithreading. Up to 10 different components can be calculated by a single propagation command.

2.3.7 ASSEMBLE: Prepare all the matrices needed for the propagation

After having given the `solve` command, calculate all the matrices needed for performing the field propagation, as described in paragraphs 1.8.2 and 1.8.3. If a high number of sections are to be considered, a moderate benefit may be obtained by using the `parallel` command before calling `assemble`.

Usage: `assemble`

2.3.8 EXCITATION: Define the excitation of the calculation region

This command allows to specify the excitation of the beginning and the end of the calculation region. In this context, an excitation applied on the first section, oriented towards the positive z axis is called propagative. An excitation applied to the last section described and oriented towards the negative z axis is called counter-propagative.

Usage: `excitation fb t r i a b`

Return value: if `t=cx` or `t=cy`, `ans` is a vector containing the angles of the excitation (see NOTE below).

Parameters:

- `fb` can be `{f|b}` where `f` indicates the forward excitation and `b` stands for backward excitation. The forward excitation is applied to the beginning of the calculation region in section 1 and propagates towards the last section defined. The backward excitation is applied to the end of the calculation region, i.e. after the last section which has been defined and propagates towards the section 1.

- `t` defines the excitation types and can be `{m|s|fx|fy}`.

- `r,i` parameters giving the real and imaginary part of the excitation coefficient for the selected mode or field. This is useful to control the phase or intensity relation between two modes or the forward and backward excitation. In fact, several `excitation` commands can be used to excite for example several propagation modes employing different strategies.

- `a,b` additional parameters whose signification depends to the parameter `t` which has been provided to the command.

NOTE: The syntax of the command is articulate, because its exact structure is determined by the kind of excitation which is selected by the `t` parameter:

– If `t=m`, the command `excitation` employs a fundamental mode excitation. The fundamental mode is the mode recognised as "interesting" (see the `lowindex` and `highindex` commands for a description of the meaning of "interesting" in the context of afmm) which has the highest real part for the effective index. In this case, the parameters `a` and `b` are ignored, but `r` and `i` must be present. For example, the following command:

```
excitation f m 1 0 0 0
```

makes sort that afmm searches for the fundamental mode using the rules described above, then puts an excitation coefficient of $1 + 0\mathrm{j}$ in the vector of modal excitation in the propagative direction.

– If `t=s`, the user selects a particular propagation mode, whose real part of the refractive index is the closest possible to the provided value of `a`. In the following example:

```
excitation f s 1 0 1.476 0
```

it is described an excitation of the mode having its real part of the effective index the closest to 1.476. A coefficient $1+0\mathrm{j}$ is put in the vector of modal excitation in the propagative direction for the selected mode.

– If `t=cx` (resp. `cy`), the excitation field is a plane wave for the E_x (resp. E_y) electric field component. In this case, `r,i` are interpreted be respectively as the real and the imaginary part of the constant to be used. If `a,b` are provided, they specify the (integer) orders h and k of the angles in the plane xz and yz. If they are not provided, a constant excitation is assumed: normal plane wave excitation. When `t=cx` or `t=cy`, it must be considered that in the RCWA/AFMM method there is an close analogy between plane waves coming from certain angles and Fourier coefficients. In fact, an angle α (in the plane xz) corresponds to a particular Fourier harmonic in the x decomposition, when the following condition is matched:

$$h\lambda_s / \sin(\alpha) = T_x \tag{2.1}$$

A similar condition may be written for the angle β in the yz plane:

$$k\lambda_s / \sin(\beta) = T_y \tag{2.2}$$

where λ_s is the wavelength considered in the medium from which the excitation comes. Therefore we have $\lambda_s = \lambda / n_\mathrm{a}$, with n_a being the substrate refractive index of the first or last section of the structure for a forward or backward excitation respectively. T_x and T_y are the calculation window sizes (see the `size` command) and h and k are positive or negative integers. Using `t=cx` or `t=cy` makes sort that the angles α and β are calculated internally and you can access them using `ans`, which is in this case a vector containing 2 elements. The following example specifies a back-excitation with aplane wave having $|E_x| = 1\,\mathrm{V/m}$, with the wave vector parallel to the z axis.

```
excitation b cx 1 0 0 0
```

- If `t=cy`, the excitation field is a plane wave for the E_y electric field component. See `t=cx` for a description of `r,i` and `a,b`.
- If `t=fx`, the excitation field is read from a file and considered as the E_x electric field component. When the field is read from a file, `a` represent the multiplication coefficient (see the `indfile` command) whereas `b` is the file name. The size specified in the excitation file (Optiwave f3d file format) must match exactly with the size of the calculation window. See paragraph D.2.2 for a description of the file format. In the following example:

  ```
  excitation f fx 1 0 1e-6 test_exc_x.f3d
  ```

  the field contained in the file `test_exc_x.f3d` is used for the propagative excitation of the first section.
- If `t=fy`, the program considers the file as the E_y component. See `t=fx` for details.
- If `t=bl`, this indicates a Bloch-mode excitation (see the `bloch` command). The Bloch eigenmodes should have been already calculated and the wanted one is selected via the closest real part to the specified `b` parameter, while `a` is the phase unwrapping parameter (see `outbloch` for more information about this detail).
- If `t=id`, the user selects a particular mode corresponding to the given position index, specified by `a`. In the following example:

  ```
  excitation f id 1 0 25 0
  ```

  the mode having the position index 25 is selected. A coefficient $1 + 0j$ is put in the vector of modal excitation in the propagative direction for the selected mode. In reality, the position index is an internal representation which is subjected to change depending to the number of harmonics, development strategies, LAPACK inner working etc. A position index is guaranteed to remain coherent only within a single simulation session. One can obtain the position indices by means of the `modepos` command, store it and then use it in a second time, as in the following example:

  ```
  # Structure definition...
  ...

  wants propagation

  solve

  select 1
  modepos

  let posA=ans(0)
  let posB=ans(3)

  excitation "f" "id" 1.0 0.0 posA 0
  excitation "f" "id" 1.0 0.0 posB 0

  assemble

  propagation Ex m 100e-9 50 50 propag.txt
  ```

2.3.9 CARPET: Sweep some convergence problems under the carpet

The `carpet` command forces all modes to have a negative imaginary part for the effective index. It is useful when calculating the propagation of the fields, since a positive imaginary part would result in a field which can quickly reach very high value when propagating. This often result in a convergence issue. It should be noted that what it is done by the `carpet` command is somewhat artificial and a rigorous mathematical justification is still missing[20]. It just seems to work. This command should be used before `solve`.

Usage: `carpet`

2.3.10 OUTDATA: Calculate additional output data during propagation

This command sets up for the calculation of additional output data during the propgation of the fields. This command should be used before the `propagation` command.

Usage: `outdata t par1 par2 ...`

Parameters:

- `t` type of the output data which should be recorded
  - if `t=i`, the integral of the field over a rectangular window is calculated. The field component F which is integrated depends on which field component will constitute the output for the `propagation` command:

$$o = \sqrt{\int_{c_x-w/2}^{c_x+w/2} \int_{c_y-h/2}^{c_y+h/2} F^2 \mathrm{d}x\mathrm{d}y} \tag{2.3}$$

    `outdata i cx cy w h filename`

    where:

    * `cx` is the x centre of the window to be used for the calculation
    * `cy` is the y centre of the window to be used for the calculation
    * `w` is the width of the window
    * `h` is the height of the window
    * `filename` is the name of the file on which the results should be recorded
  - if `t=g`, it computes the 2D generation rate cylindrincally interpolated from the 3D generation rate using:

$$G(r,z) = \frac{1}{2\pi} \int_{\theta=0}^{2\pi} G(x = r\cos\theta, y = r\sin\theta, z)\mathrm{d}\theta \tag{2.4}$$

    with

$$G(x,y,z) = \frac{\mathrm{Im}(\epsilon)}{2\hbar} \left(|E_x|^2 + |E_y|^2 + |E_z|^2 \right) \tag{2.5}$$

    Usage: `outdata g z0 z1 r1 dr filename`

x symmetry:	x anti-symmetry:
$E_x(-x,y) = -E_x(x,y)$	$E_x(-x,y) = E_x(x,y)$
$E_y(-x,y) = E_y(x,y)$	$E_y(-x,y) = -E_y(x,y)$
$H_x(-x,y) = H_x(x,y)$	$H_x(-x,y) = -H_x(x,y)$
$H_y(-x,y) = -H_y(x,y)$	$H_y(-x,y) = H_y(x,y)$
y symmetry:	y anti-symmetry:
$E_x(x,-y) = E_x(x,y)$	$E_x(x,-y) = -E_x(x,y)$
$E_y(x,-y) = -E_y(x,y)$	$E_y(x,-y) = E_y(x,y)$
$H_x(x,-y) = -H_x(x,y)$	$H_x(x,-y) = H_x(x,y)$
$H_y(x,-y) = H_y(x,y)$	$H_y(x,-y) = -H_y(x,y)$

Table 2.1: The meaning of symmetry and anti-symmetry in the context of the `symmetry` command.

Return value: `ans` contains the power calculated by integrating the generation rate. The step in z given by the command propagation is employed for calculating the integral (trapezoidal rule).

where:

* `dr` is the step employed for the integration in the radius (trapezoidal rule).
* `filename` is the name of the file on which the interpolated generation rate map should be recorded.
* `z0` lower limit in z for the integration.
* `z1` upper limit in z for the integration.
* `r1` upper limit in r for the integration.

The output file contains three columns: the current radius r, the position in z and finally the generation rate integrated over a 2π angle at the current radius. Every time a step in z is done, a blank line is added to the file.

2.3.11 SYMMETRY: Handle structure symmetries

This command specifies whether the symmetry of the structure should be taken into account in the calculation of the field.

Usage: `symmetry sanx sany`

Parameters:

- `sanx`: Set up the mirror symmetry perpendicular of the x axis, can be `{s,a,n}` where:
  - `s` indicates x symmetry (see tab. 2.1)
  - `a` indicates x anti-symmetry (see tab. 2.1)
  - `n` means that no symmetry is used.
- `sany`: sets up the mirror symmetry perpendicular of the y axis, can be `{s,a,n}` (same as x).

Note: in case of symmetry, `pml` and `bend` commands cannot (yet) be used, except in the special case of `symmetry n n`. Only two kinds of developments are

available (see the description of the `matdev` command in section 2.4.17) when exploiting symmetries: Lalanne developments (`matdev las alpha`) and normal field (`matdev nfs mult fileNx fileNy`). It is not (yet!) possible to have the same kind of symmetry for the two axis.

2.3.12 BLOCH: Calculate Bloch eigenmodes and eigenvectors

The `bloch` command calculates the Bloch modes of the current structure, periodized by an infinite repetition on the propagation axis, as described in paragraph 1.9. This is useful when considering photonic crystal as well as Bragg gratings. The `bloch` command is a direct implementation of the technique described by Cao et. al. in [13].

This command does not provide a direct output beside changing the value of `ans`, see `outbloch` if you want to write Bloch modes on a file.

Usage: `bloch`

Return value: after `bloch` is called, `ans` contains an array whose elements are alternatively the real and the imaginary part of the propagation constants of the Bloch modes found by `bloch`. Therefore, the size of the array will be twice the number of modes.

2.3.13 ANGLES: Define excitation angles different from 0

Define the excitation angles to be used in the xz and yz plane.

Usage: `angles n0 thetax thetay`

Parameters:

- `n0` the refractive index of the excitation section to be used for the angle calculation
- `thetax` the angle of the excitation wave in the xz plane, with respect to the propagation direction (in radians).
- `thetay` the angle of the excitation wave in the yz plane, with respect to the propagation direction (in radians).

2.3.14 OUTBLOCH: Output the results of the Bloch mode calculation

After the Bloch modes of the structure have been calculated with `bloch`, the output on a file can be obtained with `outbloch`

Usage: `outbloch phase selection d`

Parameters:

- `phase` number of complete 2π contributions to be added for phase unwrapping. This is parameter p introduced in equation (1.81) in paragraph 1.9.
- `selection` selection strategy:

  - `il` Imaginary part lower than a given threshold. The maximum absolute value of imaginary part for printing modes must be less than parameter.
  - `cr` Closest real. Just one mode closest to the specified parameter `d`.
  - `fi` File. All effective indices as well as coefficients will be written in a file specified by the parameter `d`.

- `d` depends on selection

2.4 Section-level commands

Here we describe all those commands which affects somehow the current section of the structure.

2.4.1 SECTION: Define a section of the given length

A structure is an assembly of sections of fixed length. This command creates a new section whose length must be given.

Usage: `section tl`

Parameters:

- `tl` total length in meters

2.4.2 SUBSTRATE: Chooses the refractive index of the substrate

Usage: `substrate nr ni`

Parameters:

- `nr`: substrate refractive index, real part
- `ni`: substrate refractive index, imaginary part

Notes: This command should be used once the size of the calculation window (command `size`) and the number of space harmonics to be taken (command `harmonics`) have been defined. Executing this command initialises the refractive index distribution of the current section, so it is frequently used at the very beginning of a section definition. The real part of the given refractive index is used as the lowest bound of the effective index range used consider a mode "interesting" when executing the `solve` command.

For example:

```
substrate 1.44 0
```

sets a 1.44 refractive index substrate (for example, pure silica).

2.4.3 RECTANGLE: Add a rectangular waveguide to the structure

Usage: `rectangle ngr ngi wx wy px py`

Parameters:

- `ngr`: core refractive index (real part of n_c)
- `ngi`: core refractive index (imaginary part of n_c)
- `wx`: waveguide x size
- `wy`: waveguide y size
- `px`: x position of the centre of the waveguide (referred to the centre of the calculation window)
- `py`: y position of the centre of the waveguide (referred to the centre of the calculation window)

Notes: This command should be used once the size of the calculation window (command `size`) and the number of space harmonics to be taken in the calculations (command `harmonics`) have been defined. The substrate index should have already been defined (command `substrate`). The Fourier coefficients are analytically calculated by considering the theoretical cardinal sinus, truncated to the given number of harmonics. This command implicitly defines the maximum value of the effective index for a mode to be considered "interesting" when using the `solve` command. If n_0 is the value of the substrate refractive index, the use of the `rectangle` command in reality adds a rectangle to the current section refractive index distribution, whose value is $n_c - n_0$. It is perfectly legitimate to superimpose several rectangles inside the calculation window, just by having in mind that each one acts this way. For complex situations, the command `inpstruct` may be useful for checking the obtained refractive index distribution.

For example:

```
rectangle 3.46 0.0 500e-9 200e-9 0 0
```

Define a 3.46 refractive index (silicon) rectangular waveguide, 500 nm × 200 nm, centreed in the calculation window.

2.4.4 PML: Add perfectly matched layers to the waveguide

Implement anisotropic perfectly matched layers on the boundaries of the calculation window, as described in [37] and in this manual in section 1.3.1.

Usage: `pml alphar alphai wx wy`

Parameters:

- `alphar`: real part of the alpha parameter
- `alphai`: imaginary part of the alpha parameter

Notes: This command should be used once the size of the calculation window and the number of space harmonics to be taken has been defined. The substrate index should have already been defined. The reference refractive index used is the substrate's one. This command should not be used when loading the refractive index distribution from an input file with the `indfile` command. Note that this type of PML are not available when using symmetries.

For example:

```
pml 1 -1 200e-9 200e-9
```

Define PML layers with $\alpha = 1 - \mathrm{j}$ and with thickness of 200 nm in the x and y direction.

2.4.5 PML_TRANSF: Adopt PML's with coordinate transform

This command defines a coordinate transform section used as a Perfectly Matched Layer, as described in [23].

Usage: `pml_transf qx qy gammar gammai`

Parameters:

- `qx`: size of the transform region in x (should be less than the x size of the calculation window).
- `qy`: size of the transform region in y (should be less than the y size of the calculation window).
- `gammar`: real part of the complex factor γ
- `gammai`: imaginary part of the complex factor γ, as described in [22].

2.4.6 INDFILE: Reads refractive index distribution from an input file

The command `indfile` can be used to read an arbitrary refractive index distribution from an input file. This command overwrites the previous settings regarding the size of the calculation window (command `size`), as well as the substrate (command `substrate`). For this reason, this command can be used just after the `harmonics` command, since the definition of the calculation window is read from the input file. This command does NOT define the minimum and maximum value of the indices to be used for the guided mode search. The user should define them by using the `lowindex` and `highindex` or `order` commands in order to explicitly define the range of "interesting" modes.

Usage: `indfile mult filename`

Parameters:

- `mult`: a multiplier to be considered for length calculations. For example, if the sizes in the input file are specified in micrometers, `mult` should be set to 1×10^{-6}. If they are in meters, it should be 1 and so on. Usually setting `mult` to `1e-6` is what required for file compatible with Optiwave software.
- `filename`: the name of the file to be used, in the Optiwave rid format, described in paragraph D.2.1 (use double quotes if your file name includes spaces).

File format: the refractive index distribution must be coded in the Optiwave `rid` format, described in detail in section D.2.1.

2.4.7 INPSTRUCT: Show the input structure

Write on a file the permittivity or permeability chart of the input structure (stands for INPut STRUCTure). This may be useful to check if it has been defined correctly by the input file, as well as to check how well the structure is represented, given the number of harmonics taken into account, unless the `im` type of representation is chosen. In this case, the represented structure is the one originally specified, without passing by the Fourier transforms.

Usage: `inpstruct type size_x size_y file`
Parameters:

- `type` must be `{i|im|imo|ex|ey|ez|mux|muy|muz}`, `i` indicates the refractive index (it takes ϵ_{rx} squared), while `Ex`, `Ey` or `Ez` indicate respectively the x, y or z component of the permittivity tensor. On the same way, `mux`, `muy` or `muz` indicate respectively the x, y or z component of the permeability tensor. `im` indicates the original refractive index distribution, without having passed by a Fourier transform. `imo` is the same as `im`, but the output is done in the Optiwave format.
- `size_x` number of points to be plot in the x direction
- `size_y` number of points to be plot in the y direction
- `file` name of the file which should be written

File format: Gnuplot-compatible (see appendix D.1).
Note: The structure considered is obtained by an inverse Fourier transform calculated from the vectors containing the permeability and permittivity distribution. If the number of point given by `size_x` and `size_y` is bigger than the corresponding size of the vectors, a zero padding technique is applied before calculating the inverse transform of the matrix obtained. This allows to represent a given distribution with a comfortable number of point even if the number of harmonics taken is much less. No magic: it is just that the results are interpolated.

2.4.8 OUTGMODES: Write on a file the interesting modes found

This command writes on a file the calculated field associated to all "interesting" modes selected by `solve`. The user must specify the output field characteristics and the program will automatically append modal information to the file name provided by the user. This command should be called after only after the modes have been calculated with the `solve` command. The number of points in the x and y directions should be no less than the number of harmonics used in the calculation in each direction.
Usage: `outgmodes type rimco size_x size_y fileb`
Parameters:

- `type` must be `be {Ex|Ey|Ex2|Ey2|Ez|Ez2|Hx|Hy|Hz|Dx|Dy|Dz}` and indicates which transverse field component must be drawn. If you need to output a magnetic field component (and thus `type` is either `Hx` or `Hy`), you must use the `wants` command before calling `solve`. The fields `Ez`, `Ez2`, `Hx`, `Hy` or `Dz` require `wants` command (`wants both`) before calling solve. The index 2 refers to the improved representation technique of the fields, as in [25].
- `rimco` must be `{r|i|m|c|o}` and indicates that the real part should be written (`r`), the imaginary part (`i`), the magnitude (`m`), the complex magnitude (`c`) of the fields or (`o`) the Optiwave f3d format.
- `size_x` minimum number of points to be plot in the x axis. If it is less than the number of harmonics, the latter will be used.

- `size_y` minimum number of points to be plot in the y axis. If it is less than the number of harmonics, the latter will be used.

- `fileb` filename base which should be written. The file name is formed as follows: `fileb_type_rimc_nn` where `nn` is the progressive mode number and `fileb`, `type` and `rimc` are the calling parameters of `outgmodes`.

File format: Gnuplot-compatible (see appendix D.1), or Optiwave f3d format.

2.4.9 SELMODES: output the modes using a rule

The command `selmodes` employs a selection rule on the modes of a given section and and writes on a file their shape. See the `outgmodes` command (paragraph 2.4.8) for a description of the file formats, as well as the way the file names are assembled. This command is often used for performing a modal analysis after a complete propagation of the field has been calculated. For the moment, only the `{exco}` rule is available.

Usage: `selmodes type rimco rule value1 value2 size_x size_y fileb`
Parameters:

- `type` see `outgmodes`

- `rimco` see `outgmodes`

- `rule` must be `{exco}` (mnemonic for EXCitation Order). In this case the modes written by `selmodes` are those carrying most of the power. The parameter value1 in this case can be `{f|b}` to indicate wether the forward power will have to be considered, or the backward power. The power will be calculated by considering the excitation coefficient, times the normalization coefficient (the same calculated by `eigenvc`). `value2` in this case specifies how many modes should be used. For example:

  ```
  selmodes Ex o exco f 3 101 101 modes
  ```

  will write on files `modes_Ex_o_0.f3d`, `modes_Ex_o_1.f3d`, `modes_Ex_o_3.f3d` the E_x component of the three modes which in the current section are carrying the most of the power. Each file produced by this example will be in the Optiwave f3d file format and composed by 101x101 points. Of course, since the excitation coefficients need to be known, the command has to be used only that a propagation has been calculated. After the execution of the command, `ans` will contain an array of effective refractive indices (real and imaginary parts) of all modes found in the analysis. The mode numbering does not have anything to do with the mode numbering produced by `solve` or `outgmodes`. In fact, with `selmodes` and the `{exco}` rule the first mode will be the one carrying the most of the power, followed by the others in a decreasing order. NOTE: This rule must be used with a great care when there are modes which are not orthogonal in the problem being considered.

- `value1` see `rule`

- `value2` see `rule`

2.4.10 LOWINDEX: Set the lowest value allowed for the effective index to be considered "interesting"

If the user is interested in a propagation mode which is comprised in a given (known) interval, this command can be used in conjunction with `highindex` in order to specify the range in which the modes found by `solve` is for example exported using `outgmodes`. If this command is not used, the lowest index allowed for the real part of the guided mode is assumed to be the substrate's real part of the refractive index, as defined by the `substrate` command. The `lowindex` command can be useful when dealing with asymmetric or complex structures, in order to avoid flooding the disk with data about propagation modes which are not relevant to the problem being considered.

Usage: `lowindex lr li`

Parameters:

- `lr` lowest value of the real part of the refractive index for a mode to be considered "interesting"
- `li` lowest value of the imaginary part of the refractive index

2.4.11 HIGHINDEX: Set the highest value allowed for the effective index to be considered "interesting"

See the `lowindex` command. If this command is not used, the highest index value is assumed to be the highest refractive index real part used for waveguide definition by the `rectangle` command. This command can be useful when the automatic calculations are not working, for example because of several waveguide sections overlapping, when using the `indfile` command, or to search for specific modes.

Usage: `highindex hr hi`

Parameters:

- `hr` highest value of the real part of the refractive index for a mode to be considered "interesting"
- `hi` highest value of the imaginary part of the refractive index

2.4.12 BEND: Define the bending radius

This command allows the user to define a certain bending radius of the structure being defined.

Usage: `bend r`

Parameters:

- `r`: bending radius (around y axis) of the *centre* of the x calculation window. This value is used also when calculating the effective index of the modes with equation (1.46). For this reason, the user should choose where to place the waveguide in the calculation window. Different conventional choices are present in the literature. Someone defines the internal radius of the waveguide as the curvature radius, others the central radius, other the external radius. Be aware of the choice you are using when placing the waveguide core!

Note: `bend` is not (yet) compatible with the use of symmetries.

2.4.13 SPECTRUM: Obtain all calculated effective indices

Command writing on a file the real and imaginary part of ALL effective indices of ALL modes calculated.

Usage: `spectrum filename`

Parameters:

- `filename`: the name of the file on which to write.

2.4.14 ORDER: Select the "interesting" modes on the basis of azimuthal order

This command specifies that the modes which are considered interesting are selected in terms of azimuthal order (defined as equation (1.47)), instead of of the effective index. This can be particularly useful when searching for modes of bent waveguides and the meaning of the azimuthal order can be more physically profound than using the effective index.

Usage: `order min max`

Parameters:

- `min` the minimum azimuthal order retained
- `max` the maximum azimuthal order retained

2.4.15 COEFFICIENT: Determine the transmission coefficient of a mode in the structure

Calculate the transmission coefficient of a mode at the current section, the mode being selected in two way. The first one is by specifying the real part its effective index. The second one is by giving the mode position index, i.e. its internal position in the vectors and matrices used by afmm. This command must be used only after `propagation` as it returns the excitation coefficient for the selected mode in the current section after that the propagation has been done. In other words, the module squared value of that coefficient times the normalization factor (i.e. the square of the integral of the Poyinting vector of the mode profile in the calculation window) yields the power carried by the mode (see `eigenvc`).

Usage: `coefficient fb a`

Return value: `ans` will contain an array of three elements:

- `ans[0]` is the real part of the coefficient
- `ans[1]` is the imaginary part of the coefficient
- `ans[2]` is the squared module of the coefficient

Parameters:

- `fb` can be `{f|b}` where `f` stands for forward and `b` for backward
- `a` the real part of the refractive index of the mode to analyze

Alternatively, one can specify the mode position index: `coefficient fb id pos`

Parameters:

- `fb` can be `{f|b}` where `f` stands for forward and `b` for backward
- `id` the two letters `id` tag, indicating the alternative of selection by the mode position index.
- `pos` the position index. See the `modepos` command for a description of what is the mode position index.

Examples:

```
# Selection via the real part of effective index
# Backward direction
coefficient b 1.2193

# Selection via the mode position index
coefficient f id pos2
coefficient b id pos2
```

2.4.16 SELECT: Change the current section

Specify which section afmm must consider for section-level calculations.

Usage: `select a`

Parameters:

- `a` the number of the wanted section, numbered from 1 to N, where N is the number of sections currently entered.

2.4.17 MATDEV: Choose the matrix development strategy

This command determines the matrix development to be used during the construction of the propagation operator.

Usage: `matdev type [alpha]`

Parameters:

- `type` can be `{af|an|be|nd|la}` where:
  - `af` stands for multiplication after construction strategy. See paragraph 1.4.2.
  - `an` indicates still a multiplication after construction as in v. 1.2.3. See paragraph 1.4.1.
  - `be` indicates multiplication before construction as in v. 1.3. See paragraph 1.4.3.
  - `nd` adopts a form of compact developments. See paragraph 1.4.4.
  - `la` is Lalanne type developments[24]. The `alpha` parameter is thus required and should be comprised between 0 and 1. See paragraph 1.4.5.
  - `las` is Lalanne type developments[24], taking into account the symmetries. The `alpha` parameter is thus required and should be comprised between 0 and 1. See paragraph 1.4.6. Use it as in the `la` case.

- `nf` implements a normal field development as in[36]. In this case, the syntax is as follows:
  `matdev nf mult fileNx fileNy`
  where `mult` is the size multiplier for data contained in files `fileNx` and `fileNy` (both in the Optiwave format).
- `nfs` implements a normal field development as in[36], taking into account the symmetries. Use it as in the `nf` case.

Note: the default developments used are `af`. This command should be used before `solve`. Only two kinds of developments are available when exploiting symmetries: Lalanne developments (`matdev las alpha`) and normal field (`matdev nfs mult fileNx fileNy`).

2.4.18 NORM: Calculate the L2 norm of the vector of excitation

Calculate the L2 norm of the vector s of excitation coefficients for the current section t:

$$L^2\{s\} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|s(i)|^2} \tag{2.6}$$

Where N is the total number of harmonics.
Usage: `norm fb`
Parameters:

- `fb` can be `{f|b}`. If `f` is given, the forward excitation of the current section is considered and in this case $s = W^{[t]}\bar{s}_W^{+[t]}$. If `b` is given, the backward excitation is calculated and so $s = W^{[t]}\bar{s}_W^{+[t]}$. The matrix $W^{[t]}$ is formed by the eigenvectors calculated for the current section, and thus s represents the excitation vector in the Fourier space.

Return value: `ans` contains the measured norm.

2.4.19 EIGENVC: Report eigenvectors of the current section

Report on a file all the excitation coefficients of the modes of the current section, as well as their effective index. It must be used after `solve`, unless the `fba` parameter is equal to `a`. In this case, `eigenvc` must be called after that a complete propagation has been calculated.
Usage: `eigenvc fba name`
Parameters:

- `fba` can be `{f|b|a}` where `f` stands for forward and `b` for backward. If this parameter is `a`, a more complex set of information is given for each line in the file:
  - real and imaginary part of the effective index
  - real and imaginary part of the forward coefficients
  - real and imaginary part of the backward coefficients

  - the integral of the Poynting vector

- `name` the filename.

2.4.20 EIGENAM: Report eigenvalues of the current section

Write on a file all the Fourier coefficients of the eigenvalues matrices W or V, if available, as defined in equations (1.51) and (1.61). It must be used after `solve`.

Usage: `eigenam eh name`

Parameters:

- `eh` can be `{e|h}` if the electric (`e`) or the magnetic (`h`) field should be considered, writing respectively the W and V matrices on the file.

- `name` the filename.

2.4.21 POWER: Calculate propagated power

This command calculates the power flowing at a boundary between two sections by summing up all the modal contributions. This commands computes the power carried by all the modes propagating in a given section, in a given direction (forward or backward). A mode is considered as propagating when the real part of its effective refractive index exceeds a certain threshold.

Usage: `power fb threshold`

Return value: `ans` contains the measured power.

Parameters:

- `fb` can be `{f|b}` if the direction is forward or backward.

- `threshold` is the threshold for the real part of the refractive index for a propagation mode to be considered as propagating.

2.4.22 POWERZ: Calculate all power flowing in a given quota

This function calculate the power flowing at a given quota z in the structure. This commands takes into account all modes (propagating and counter-propagating). Calculated power is considered positive when flowing in the forward direction (propagating), negative otherwise (counter-propagating). Power is calculated by taking into account the electric and magnetic fields and integrating the Poynting vector in the calculation window. If there power flowing in the two directions, the net result will be calculated and shown by the command as the difference between the integration of the two fluxes.

Usage: `powerz z`

Return value: `ans` contains the measured power.

- `z` coordinate at which compute the power.

2.4.23 MONITOR: Monitors power flow in a rectangular surface

This function calculates the power flowing in a rectangular surface at a given depth z. The surface is transverse to the propagation axis z. The calculation is done as follows. The longitudinal component of the Poyinting vector is calculated from the transverse electric and magnetic fields. Then, the power is calculated as a surface integral of the Poyinting vector. This function can be useful in case of PML's, since one can specify a rectangle which excludes them.

Usage: `monitor z wx wy px py`

Return value: `ans` contains the measured power.

Parameters:

- `z` coordinate at which compute the total power
- `wx`: rectangle width
- `wy`: rectangle height
- `px`: x position of the centre of the surface (referred to the centre of the calculation window)
- `py`: y position of the centre of the surface (referred to the centre of the calculation window)

2.4.24 MODEPOS: Get mode position indices

This command, to be called after SOLVE after having selected the section of interest, shows on the screen and loads in the "ans" variable the list of all position indices of the interesting modes. The position index is a way to know where the eigenvalue corresponding to the mode is stored in the the internal matrices of afmm. Since the internal order appears in such things such as the ordering of the modes in the files produced by the `spectrum` or `eigenvc` commands, this may be useful to know. It can be interesting also for specifying a particular mode in the excitation.

Usage: `modeindex`

Return value: `ans` is loaded with the position indices

2.5 Other commands of general use

Here we describe the commands which are not directly related to the structure or the sections, but are useful for the control of afmm.

2.5.1 #: Start a comment

You can introduce comments in the input file by using the `#` character.

For example:

```
# This is a comment. You should use it to make
# your code easier to read.

size 2e-6 200e-9 # this is a valid comment
```

2.5.2 HELP: List the supported commands

This command shows a very concise help, which may be useful in the interactive mode.

Usage: `help`

2.5.3 LABEL and GOTO: Unconditional jumps

The command `goto` can be used to perform an unconditional jump to a particular point[3] in the script, defined by the command `label`

In the following example:

```
print First part
goto skip
print Second part
label skip
print Third part
```

The output is as follows:

```
First part
Third part
```

2.5.4 IF, ELSE, ENDIF: Perform a conditional test

The script language used in afmminclude some instructions for conditional tests. They are used as follows:

```
if cond
    # This code is executed if cond is different from zero
else
    # This code is executed if cond is equal to zero
endif
```

Note that you might want to do some test with the `==` operator, which gives 1 if the two elements compared are equal, or 0 elsewhere. There is a definite difference with `=` which indicates only the assignment operator.

2.5.5 FOR and NEXT: Define a cycle

Afmm can include cycles in the script.

Usage: `for :variable start stop increment ...next`

Parameters:

- `:variable` is the name of the variable to be incremented preceded by a colon.
- `start` is the beginning of the count, `variable` is set to `start`.
- `stop` is the end of the count, i.e. the cycle will continue while `variable` is less than (but not equal to) `stop`. Note that `stop` is never reached.
- `increment` is the increment of `variable` at each iteration.

Cycles can be nested, as in the following example:

[3] Of course, you know that "Go To Statement Considered Harmful" [16].

```
for :i 1 4 1
    for :t 2 4 1
        print i t
    next :t
next :i
```

Count of `i` starts at 1 and reach 3. Count of `i` starts at 2 and reach 3. You may explicitly indicate the variable in the `next` command, or you left it out.

2.5.6 DO and WHILE: Define a cycle with an exit condition

Afmm scripts can employ the well known do-while construct in the following way:

```
do
    # inner part of the cycle
while condition
```

In the code shown above, `condition` is the exit condition. The cycle repeats itself while the condition is non-zero.

2.5.7 LOAD: Load (or includes) a new file

Execute all the afmm commands contained in the given file. This may be used in the interactive mode and in the non interactive mode.

Usage: `load filename`

Parameters:

- `filename` path of the file to be included (use double quotes if your file name includes spaces).

2.5.8 QUIT: Exit immediately

This command makes afmm quit immediately. This is useful in the interactive mode, as well when debugging a script, to stop its execution before launching time-consuming calculations.

Usage: `quit`

2.5.9 PRINT: Print a line at the output

This command make afmm echo all the given parameters.

Usage: `print whatever you want`

2.5.10 FOPEN: Open a file

The `fopen` command allows to open a file.

Usage: `fopen filename mode`

Return value: `ans` will contain a handle to the opened file.

Parameters:

- `filename` the name of the file to be opened
- `mode` the open mode. The write mode is specified by `w`, the read mode is `r`.

Example: write a message and a number on "test.txt" file (see also `fprint` and `fclose` commands. Read back the number.

```
fopen "test.txt" "w"
    # ans now contains the handle to the file.
    let fileo=ans

    fprint fileo "This is a test."
    fprint fileo 42
fclose fileo

fopen "test.txt" "r"
    let filei=ans
    # Skip the first line.
    fskip filei
    fscan filei
    # Should print 42
    print ans
fclose filei
```

2.5.11 FCLOSE: Close a file

Close a file previously opened via `fopen`

Usage: `fclose handle`

Parameters:

- `handle` handle of the file to be read

Return value:

2.5.12 FPRINT: Print a line on a file

This command make afmm echo all the given parameters in a file.

Usage: `fprint handle whatever you want`

Parameters:

- `handle` the file handle provided by `fopen`.
- the rest of the line will be printed on the file.

2.5.13 FSCAN read a number

This command reads from a file a numeric value and put it in the `ans` variable.

Usage: `fscan handle`

Return: `ans` will contain the read value

2.5.14 FCHECK checks if the last file read was successful

This command reads should be called after `fscan` and the `ans` variable will contain a nonzero value if the read was successful.

Usage: `fcheck handle`

Return: `ans` a nonzero value means that the last `fscan` command was successful.

2.5.15 FSKIP: Skip a line on the file

Numerical values can be read in a file opened by the `fopen` command using the `fscan` command. However, lines containing text or comments can be skipped using `fskip`.

Usage: `fskip handle`

Parameters:

- `handle` the file handle provided by `fopen`.

2.5.16 ADDSPACE: Define the strategy for spaces and newlines in printing

The `print` and `fprint` commands by default add some spaces and newline. With `addspace` you can control this behaviour.

Usage: `addspace activate`

Parameters:

- `activate` select the mode and can be `{i,o}`: `i` (default) means that newlines and spaces are automatically added to printing commands (print and fprint). `o` means that newlines and spaces are not automatically added.

2.5.17 CLEAR: Reset the structure

Reset the wavelength settings, the structure and all the definitions, but does not affect variables.

Usage: `clear`

2.5.18 SYSTEM: Executes a system command

The command `system` may be used to execute some commands of the operating system. Of course, the use will be quite dependent from the operating environment.

Usage: `system command`

Parameters:

- `command` the command to be executed

Note: this command is disabled by default. You may activate it when you know for sure that afmm is being used in a secure environment by launching it with the `-e` option. In the following example, afmm is launched with the `-e` in the interactive mode and the `ls` Unix command is being used:

```
[davidebucci@imep197-231-6]$ ./afmm -e
 ********************************************************************************
 *      Aperiodic Fourier Modal Method full vectorial 3D propagation            *
 *                            version 1.3.3                                     *
 *                                                                              *
 *     Davide Bucci, IMEP-LAHC march 2008 - april 2011                          *
 *     MINATEC-Grenoble INP, 3, parvis Luis Neel                                *
 *     38016, Grenoble CEDEX, France                                            *
 *                                                                              *
 *     bucci@minatec.grenoble-inp.fr                                            *
 *                                                                              *
 ********************************************************************************
Execution of external commands allowed.
```

```
Reading file: stdin (interactive mode)
0 > system ls
CMakeLists.txt    finterface.o      matrixDev.o       section.o
afmm              fortran_types.h   parsefile.cpp     slice
block_matrix.cpp  lib               parsefile.h       slice.cpp
block_matrix.h    main.cpp          parsefile.o       slice.o
block_matrix.o    main.o            parsercl.cpp      structure.cpp
commands.cpp      makefile          parsercl.h        structure.h
commands.h        makefile.standard parsercl.o        structure.o
commands.o        manual            resultats.txt     test
finterface.cpp    matrixDev.cpp     section.cpp
finterface.h      matrixDev.h       section.h
System command ls executed and returned 0
1 > exit
[davidebucci@imep197-231-6]$
```

2.5.19 LET: Perform numerical calculations

Perform numerical calculations on the arguments shown.

Usage: `let expression1 expression2...`

Parameters:

- `expression1` an expression to be evaluated
- `expression2` same as above

Example: `let r=2 pi=3.141592654 V=4/3*pi*r^3`

2.5.20 MEMOCC: Report memory occupation

Print some info about the current memory occupation.

Usage: `memocc`

Note: afmm might have required more memory for storing some intermediate results when matrices are being created. A more realistic estimation might give a peak memory occupation that can exceed by 200% what shown by `memocc`.

2.5.21 PARALLEL: Try to parallelise calculations when possible

Some of the commands of afmm can gain a definite speedup by parallelising calculations. This is done on two levels. The first one is done at a low level and depends on how the BLAS library has been implemented on your machine. The second one is done at an higher level by afmm itself. For example, in the command `solve` calculations can be done in parallel for more than one section at once by running different threads. The maximum number of threads allowable can be set with the `parallel` command and must be tweaked depending on the number of cores you have on the processor you are using. The default settings for afmm is to disable parallelisation by using only one thread (unless BLAS creates and handles his own).

Usage: `parallel n`

Parameters:

- `n` maximum number of threads afmm will try to create when doing calculations which can be done in parallel. As a rule of thumb, the maximum

performance gain might be obtained setting this parameter equal to the number of cores you have on your machine. If your installation of afmm makes use of a threaded version of BLAS (such as ATLAS), you might see which trade-off will result in the maximum speedup, since the library might create multiple threads by its own.

2.6 Using variables in calculations

Sometimes, it is interesting to define variables and to use them to parametrise some of the parameters of the simulation. Variables can be defined in an argument, for example with the following command: `print nomevar={value|expression}` and then can be used when needed without restriction in all parameters of all commands. Using the `print` command, you will see the result of the calculation. Expressions and variables should not contain spaces or should be included in double quotes. Let's see some examples:

```
# Here some new variables are defined and their value is printed
print radius=10e-6
print pi=3.141592654

# If you do not want the result to be printed, you should not use the
# print command, but the let command, as follows:

let radius=1e-6

# We use the variables in a calculation for a parameter of a
# command. In this case, we calculate the length of a section
# for a 90 degrees angle. If we change the variable radius,
# the length will be automatically modified.

section pi/2*radius

# ... some stuff. Now we use again the variable

bend radius

# Remember that you should not put spaces in the expressions...

section pi / 2*radius

# The previous line will give an error, since it will be interpreted
# as having  three different parameters instead of the the single
# one needed by the section command. This is not interpreted at all:

print "pi / 2*radius"
```

The recognized mathematical operators are shown in table 2.2.
You can also define arrays with the following syntax:

```
let i(0)=1
let i(5)=2
let i(8)=31
let i(10)=2.23421
```

The size of the array is defined by the highest index used in the definitions. You can access it via the `size` operator. Remember that the indices of the `i` array range between 0 and `size(i)-1`, in this case 10. In the following example, we show everything contained in `i`

Operator or function	Description	Priority
`+`	Sum	1
`-`	Substraction	1
`*`	Product	2
`/`	Division	2
`^`	Rise to the power	3
`(...)`	Parenthesis	1
`lg`	Base 10 logarithm	4
`ln`	Base e logarithm	4
`sin`	Sine	4
`cos`	Cosine	4
`tan`	Tangent	4
`sqr`	Square root	4
`abs`	Absolute value	4
`arctan`	Arcotangent	4
`arcsin`	Arcosine	4
`arccos`	Arcocosine	4
`sinh`	Hyperbolic sine	4
`cosh`	Hyperbolic cosine	4
`tanh`	Hyperbolic tangent	4
`=`	Assignment	0
`==`	Comparison (is equal)	1
`!=`	Is different	1
`>`	Greater than	1
`<`	Less than	1
`||`	Logical OR	1
`&&`	Logical AND	1
`!`	Logical NOT	4
`size`	Get the array size	4

Table 2.2: List of mathematical operators and functions recognised by afmm.

```
for :t 0 size(i) 1
    print i(t)
next :t
```

which with the previous definitions gives the following output:

```
1
0
0
0
0
2
0
0
31
0
2.23421
```

The index might be non integer, but will always be truncated. If some commands use text labels which may give rise to confusion with variables, enclosing labels between " will prevent them to be interpreted as variables:

```
let i=3

# The following command will not be interpreted correctly!
# In fact, i will be substituted with 3 and the command will not
# recognize it as a valid type.
inpstruct i 101 101 test
# -> Unrecognized output type.

# The correct way to specify a label if there is a variable
# defined with the same name is to enclose it between "

inpstruct "i" 101 101 test

# OK
```

2.7 Example: slab waveguide mode solver, choosing the best matrix development strategy

Which strategy choose for `matdev la` *in 2D calculations?*

Let's say one wants to calculate propagation modes of a 1D slab waveguide. The substrate index is $n_0 = 1.44$, the core index is $n_g = 3.5$, the thickness is $d = 500\,\text{nm}$. In this example, we orient the waveguide so that the confinement is done in the x direction. An index discontinuity is thus present and is parallel to the y-axis and perpendicular to the x-axis. The following input file can be used:

```
# We begin by defining the size of the calculation window
# Every size and distance is specified in meters.
# Here we choose a 2 microns by 200 nanometers window

size 2e-6 200e-9

# We then have to set the number of harmonics to be used.
# Since we just have one harmonic in the y axis, we can
# adopt a pretty high number of harmonics in the x
# direction

harmonics 201 1
```

```
# We set now the substrate refractive index

substrate 1.44 0

# And we define a rectangular waveguide.
# The centre of the waveguide corresponds to the centre
# of the calculation window.

rectangle 3.5 0 500e-9 200e-9 0 0

# We define the wavelength

wavelength 1550e-9

# We introduce PML's.
# Note how the size of the y absorber is zero.
# Thanks to the periodicity, we are indeed representing
# a 1D slab waveguide.

# One of those two following commands describes the PML's
pml 5 5 50e-9 0
# pml_transf 50e-9 0 .5 -.5

# Define the matrix development strategy

matdev nd
# matdev la 0.0
# matdev la 1.0 # Best strategy in this case.

# And then, we launch the calculation!!!
# solve will consider as "interesting" propagation modes
# having the real part of their effective index comprised
# between the real part of the substrate's (here 1.44)
# and the one specified in the rectangle command (3.5).

solve

# At this point, outgmodes may be used to write on a file
# the shapes of the modes found.
```

Since a 1D problem is being considered here, we are using only 1 spatial harmonic on the y axis and all features have a y size which is identical to the width of the calculation window. Of course, the thickness w_{py} of the y layers of type 1 PML's (see section 1.3) is zero. The output of afmm is as follows:

```
 ********************************************************************************
 *      Aperiodic Fourier Modal Method full vectorial 3D propagation            *
 *                           version 1.3.3                                      *
 *                                                                              *
 *     Davide Bucci, IMEP-LAHC march 2008 - april 2011                          *
 *     MINATEC-Grenoble INP, 3, parvis Luis Neel                                *
 *     38016, Grenoble CEDEX, France                                            *
 *                                                                              *
 *     bucci@minatec.grenoble-inp.fr                                            *
 *                                                                              *
 ********************************************************************************
Reading file: guide_plan.fmm
Calculation window size set to: 2e-06 m x 2e-07 m.
Number of harmonics set to: 201 x 1
Substrate refractive index set to: (1.44,0)
Generated rectangular section waveguide with core refractive index: (3.5,0)
```

Mode	*Analytical*	`matdev nd`	`matdev la 0.0`	`matdev la 1.0`
TE_0	3.2961297	$3.29613 + j1.56 \times 10^{-9}$	$3.29385 + j4.89 \times 10^{-7}$	$3.29613 + j1.56 \times 10^{-9}$
TE_1	2.6349060	$2.63490 + j1.53 \times 10^{-7}$	$2.62481 + j2.00 \times 10^{-7}$	$2.63490 + j1.53 \times 10^{-7}$
TE_2	1.4627967	$1.44467 - j1.46 \times 10^{-3}$	—	$1.44467 - j1.46 \times 10^{-3}$
TM_0	3.1791296	$3.17993 - j1.18 \times 10^{-7}$	$3.17993 - j1.18 \times 10^{-7}$	$3.17913 - j3.83 \times 10^{-8}$
TM_1	2.0969300	$2.10149 + j3.62 \times 10^{-6}$	$2.10149 + j3.62 \times 10^{-6}$	$2.09688 - j8.65 \times 10^{-7}$
TM_2	1.4408402	—	—	—

Table 2.3: Effective indices of the guided mode at $\lambda = 1.55\,\mu$m for the structure defined in section 2.7 using anisotropic PML's defined with: `pml 5 5 50e-9 0`.

Mode	*Analytical*	`matdev nd`	`matdev la 0.0`	`matdev la 1.0`
TE_0	3.2961297	$3.29613 - j1.00 \times 10^{-9}$	$3.29385 + j1.74 \times 10^{-9}$	$3.29613 - j1.00 \times 10^{-9}$
TE_1	2.6349060	$2.63490 - j5.74 \times 10^{-9}$	$2.62481 - j1.30 \times 10^{-7}$	$2.63490 - j5.74 \times 10^{-9}$
TE_2	1.4627967	$1.44467 + j1.89 \times 10^{-3}$	$1.46227 + j1.45 \times 10^{-3}$	$1.47145 + j1.89 \times 10^{-3}$
TM_0	3.1791296	$3.17993 + j1.42 \times 10^{-8}$	$3.17993 + j1.42 \times 10^{-8}$	$3.17913 + j1.22 \times 10^{-8}$
TM_1	2.0969300	$2.10153 - j1.18 \times 10^{-5}$	$2.10153 - j1.18 \times 10^{-5}$	$2.09691 - j1.10 \times 10^{-5}$
TM_2	1.4408402	$1.44608 + j6.50 \times 10^{-4}$	$1.44608 + j6.50 \times 10^{-4}$	$1.44482 + j5.06 \times 10^{-4}$

Table 2.4: Effective indices of the guided mode at $\lambda = 1.55\,\mu$m for the structure defined in section 2.7 using anisotropic PML's defined with the command: `pml_transf 50e-9 0 .5 -.5`. The best solution is evidenced in blue (`matdev la 1.0`).

```
Size 5e-07 m x 2e-07 m at location 0 m x 0 m
Wavelength set to: 1.55e-06 m.
Added PML layers alpha: (5,5), x thickness: 5e-08 m, y thickness: 0 m.
Matrix developments: nd; compact matrix development.
Now considering section 1 of 1.
Creating the electric field propagation matrix
Created X2... X4... X1... X3... Y2... Y1... Y4... Y3... DD, EE, FF, GG
Calculating eigenvalues and eigenvectors
LAPACK ZGEEVX executed, info = 0
Searching for interesting modes: 1.44 < Re{n_eff} < 3.5
                                 -0.1 < Im{n_eff} < 0.1
Interesting mode found 0, effective index: (3.17993,-1.18001e-07)
Interesting mode found 1, effective index: (2.10149,3.61802e-06)
Interesting mode found 2, effective index: (3.29613,1.56262e-09)
Interesting mode found 3, effective index: (2.6349,1.53038e-07)
Interesting mode found 4, effective index: (1.44467,-0.00146347)
```

At first, afmm shows a banner. The parameters used in the calculation as well as a few hints about the calculation advancement are then provided. This may appear verbose, but every command in some way gives important debug information about the way it is interpreted and executed. At the end, the effective indices of guided modes are prompted. In this example, the commands `lowindex` and `highindex` have not been used. The definition of the interval in which the real part of the effective indices of the modes are considered "interesting" is automatically defined by the `substrate` and `rectangle` commands. In a waveguide, a mode is considered guided if the real part of its effective index is comprised between the substrate refractive index and the refractive index of the core of the waveguide. The interval automatically defined by `substrate` and `rectangle` should be coincident with this property in simple cases such as the one we are considering. The value of the imaginary part must be less than 0.1.

Solving the TE and TM dispersion equation of this waveguide allows to obtain the effective indices of the guided mode, as shown in table 2.3. We may compare the results obtained with different matrix development strategies. For TE modes, we observe that for `matdev nd` a very good agreement (6 digits) is

obtained for the first two modes, which are well confined in the waveguide core. The TE_2 has a very low effective index and it is thus weakly confined. The difference between the result found by the analytical calculations and afmm may be explained by the fact that the PML's interact strongly with this mode, thus raising the real part as well as the imaginary part of its effective index. On the other hand, it may be apparent that a less good agreement is obtained for the determination of TM modes and, even worse, the TM_2 mode is not found at all by afmm.

A better agreement can be achieved with a proper use of the `matdev la` matrix development. It appears from table 2.3 that the configuration `matdev la 1.0` is a very good choice for this particular geometry, and allows to obtain a very good agreement for TE as well that for TM modes. In fact, in this configuration, the E_x field is discontinuous for the TM modes whereas E_y is continuos for the TE modes. The configuration `matdev la 0.0` gives very bad results here, but the situation would be the opposite one if the whole structure is rotated by 90° and thus the confinement is done on the y axis.

It can be noticed that the TM_2 mode is still not found by afmm, probably because of the anisotropic PML's. Substituting the command `pml 5 5 50e-9 0` with the command `pml_transf 50e-9 0 .5 -.5` yields to much better results, resumed in table 2.4. We notice that this time all guided modes have been found by afmm thanks to the more robust coordinate transform PML's. The imaginary part of each mode is almost always lower than for the case of the simple anisotropic PML's considered above. The use of coordinate transform PML's is quite often the best choice available in afmm.

2.8 Example: rectangular waveguide

We want now to calculate guided modes of a 2D rectangular waveguide. The substrate index is $n_0 = 1.44$, the core index is $n_g = 3.5$, the waveguide width is $d = 500\,\text{nm}$ and the height is $h = 200\,\text{nm}$. We want to perform calculations at a wavelength $\lambda = 1550\,\text{nm}$. The following input file can be used:

```
# We begin by defining the size of the calculation window
# Every size and distance is specified in meters.
# Here we choose a 1.5 microns by 1.5 microns window

size 1.5e-6 1.5e-6

# We then have to set the number of harmonics to be used
# We need to be careful with this choice: matrices become painfully
# memory consuming and the misuse of the virtual memory can slow
# down calculations.

harmonics 25 25

# We set now the substrate refractive index

substrate 1.44 0

# We define a rectangular waveguide
# Note that all sizes are given in meters

rectangle 3.5 0 200e-9 500e-9 0 0
```

```
# We define the wavelength

wavelength 1550e-9

# We introduce PML's

pml_transf 50e-9 50e-9 .5 -.5

# The longest side of the waveguide core is oriented vertically.
# Probably, this may be a wise choice, but an optimised alpha
# parameter might be chosen.

matdev la 1.0

# And then, we launch the calculation!!!

solve
```

And here is the program output:

```
 ********************************************************************************
 *      Aperiodic Fourier Modal Method full vectorial 3D propagation            *
 *                           version 1.3.3                                      *
 *                                                                              *
 *     Davide Bucci, IMEP-LAHC march 2008 - april 2011                          *
 *     MINATEC-Grenoble INP, 3, parvis Luis Neel                                *
 *     38016, Grenoble CEDEX, France                                            *
 *                                                                              *
 *     bucci@minatec.grenoble-inp.fr                                            *
 *                                                                              *
 ********************************************************************************
Reading file: rectangular.fmm
Calculation window size set to: 1.5e-06 m x 1.5e-06 m.
Number of harmonics set to: 25 x 25
Substrate refractive index set to: (1.44,0)
Generated rectangular section waveguide with core refractive index: (3.5,0)
Size 2e-07 m x 5e-07 m at location 0 m x 0 m
Wavelength set to: 1.55e-06 m.
Size of the coordinates transform region: 5e-08 m x 5e-08 m, complex factor:
(0.5,-0.5)
Matrix developments: la; Lalanne-type with alpha = 1.
Now considering section 1 of 1.
Creating the electric field propagation matrix
Created X2... X4... X1... X3... Y2... Y1... Y4... Y3... DD, EE, FF, GG
Calculating eigenvalues and eigenvectors
LAPACK ZGEEVX executed, info = 0
Searching for interesting modes: 1.44 < Re{n_eff} < 3.5
                            -0.1 < Im{n_eff} < 0.1
Interesting mode found 0, effective index: (2.39545,4.84724e-06)
Interesting mode found 1, effective index: (1.65379,0.000361698)
Interesting mode found 2, effective index: (1.44772,-0.000818871)
```

We note the presence of a relatively high imaginary part in the effective indices of the guided mode. This is probably due to the choice of a small calculation window, in which the evanescent part of the guided field is in interaction with the lossy PML.

2.9 Example: load a refractive index distribution from a file

The following script reads a refractive index distribution from a file, calculates the guided modes and writes the modal eigenfields on a file. This technique requires that the range of the effective indices to be retained is explicitly set, by using the commands `lowindex` and `highindex`. The file format of the input structure is the Optiwave rid format described in section D.2.1.

```
# Define the number of harmonics to be used as well as the
# operating wavelength
harmonics 21 21
wavelength 1550e-9

# Read the input refractive field profile
indfile 1e-6 profile.rid

# Use coordinate transform PML's at the border of the calculation
# window
pml_transf 2e-6 2e-6 .5 -.5

# Describe the structure used for calculation in a file
inpstruct i 200 200 structure.txt

# Define the minimum and maximum value of the effective index for
# a mode to be retained by the solve command

lowindex 1.50 -.1
highindex 1.60 .1

# Find eigenmodes
solve

# Write the modal eigenfield on files
outgmodes Ex m 200 200 mode
outgmodes Ey m 200 200 mode
```

2.10 Example: guided modes of a 1D bent waveguide

This example shows the calculation of the guided propagation modes of a slab waveguide having an internal bending radius of 14 μm. The low index contrast as well as the small radius make sort that the guided fundamental mode leaks and has serious bending losses. This configuration is particularly sensitive to the ability of the PML to absorb the field escaped from the waveguide. For this reason, only the coordinate mapping PML should be used. Please note the range of the real part of effective indices to be explored, which is considerably higher than the core index and includes a lot of unguided modes. A TE guided mode is characterised by a E_y field component which still has a peak inside the core of the waveguide.

```
size 25e-6 3e-6

harmonics 501 1
```

```
substrate 1.515 0

rectangle 1.615 0 4e-6 3e-6 -2e-6 0

wavelength 1550e-9

lowindex 1.5 -1
highindex 2  1

matdev la 1.0

pml_transf 16e-6 0 1 -1

bend 14e-6

solve

inpstruct i 1200 1 bend_14.structure

outgmodes Ey m 1200 1 bend_14
outgmodes Ex m 1200 1 bend_14

# effective indices given by conformal-AFMM (TE)
# 1.981291485856762 - 0.013844630825915i
# 1.852504876480825 - 0.029116654438393i
# 1.746972235496869 - 0.031405571400227i
```

Depending on the number of harmonics taken, the results of the effective indices given by afmm agree within 5 to 6 decimals for the real part and within 3 decimals for the imaginary part with the results given by conformal-AFMM, as described in [31].

2.11 Example: field propagation in a 2D structure

This example shows the propagation of the field as calculated by afmm in a structure composed by two straight waveguides and two bent waveguides, excited by the second TE mode at one end. Notice how the sign of the radius determines the direction of the bending. The results of the calculation are shown in figure 2.1

```
# Definition of the calculation window size
size 3e-6 .2e-6
harmonics 51 1
wavelength 1.55e-6

# Definition of all sections composing the structure

# Section 1
section 2e-6

    substrate 1.45 0

    rectangle 2.8002496 0 500e-9 200e-9 0 0

    lowindex 1.45 -0.01
    highindex 5  0.01
    pml_transf 2e-6 0e-9 .5 -.5
```

```
# Section 2
section 3.14159265e-6

    substrate 1.45 0

    rectangle 2.8002496 0 500e-9 200e-9 0 0

    lowindex 1.45 -0.01
    highindex 5 0.01
    pml_transf 2e-6 0e-9 .5 -.5

    bend 2e-6

# Section 3
section 3.14159265e-6

    substrate 1.45 0

    rectangle 2.8002496 0 500e-9 200e-9 0 0

    lowindex 1.45 -0.01
    highindex 5  0.01
    pml_transf 2e-6 0e-9 .5 -.5

    bend -2e-6

# Section 4
section 2e-6

    substrate 1.45 0

    rectangle 2.8002496 0 500e-9 200e-9 0 0

    lowindex 1.45 -0.01
    highindex 5  0.01
    pml_transf 2e-6 0e-9 .5 -.5

# Declare that the user wants to calculate the propagation
# of the field in the structure. All calculated results useful for
# that will thus be retained in memory and not discarded
wants propagation

# Calculate all eigenmodes for all sections
solve

# Write the modal fields containing the modes in the Optiwave
# file format

outgmodes Ex o 401 1 mode
outgmodes Ey o 401 1 mode

# Assemble the structure to calculate the propagation of the field.
assemble

# Use a forward excitation, read to a file in the Optiwave format
# and interpret it as a Ey excitation. The excitation coefficient
# for the mode which will be propagated is 1+0*j

excitation f fy 1 0 1e-6 mode_Ey_o_3.f3d

# Perform the propagation of the field and write down the results
# in a file.
```

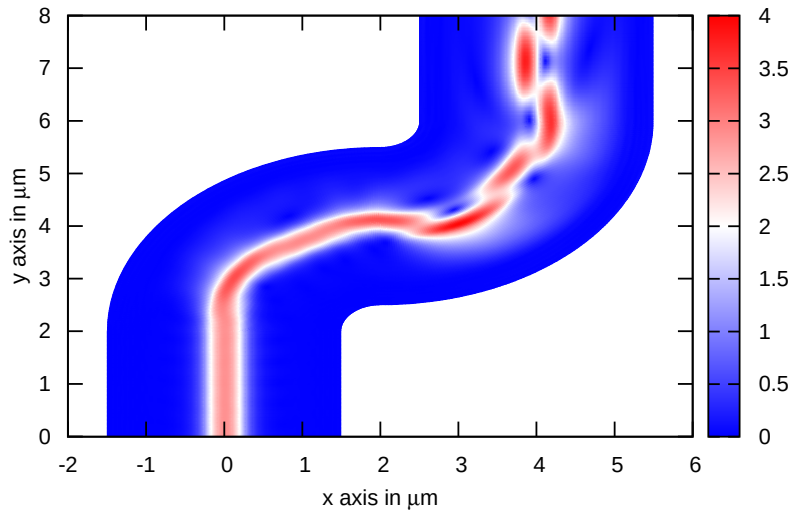

Figure 2.1: The magnitude of the E_y field propagation in the structure described in section 2.11.

```
propagation Ey m 25e-9 201 1 propagation.txt
```

2.12 Example: Bloch-mode calculation for guided waves in a Bragg grating

Bloch modes constitutes a powerful analysis tool for periodic structures. They can be calculated by afmm, by supposing that a periodisation exists in the propagation axis. In this case, the described structure is supposed to be repeated for an infinite number of times. The technique followed for the calculation of Bloch-modes is the same as described in [13] and is related to a generalised eigenvalue calculation once the global S matrix of the overall structure is obtained. The following script allows to define the same test structure described in [13] (studied also in [14]). Note that the structure is not the same for the TE and TM calculations. Authors chose $t_f = \lambda/\pi$ for TE and $t_f = \lambda/2$ for TM. The following script is in the TE configuration.

```
# Study of the Bloch modes of the structure described in
# Q. Cao, P. Lalanne, and J.-P. Hugonin
# "Stable and efficient Bloch-mode computational method for
# one-dimensional grating waveguides", JOSA A, Vol. 19, Issue
# 2, pp. 335-338 (2002)

# TE case (tf=lambda/pi)

harmonics 1 201
```

```
print pi=3.14159265359

print lambda=1e-6
print totx=1
print toty=3*lambda

print "nsup=" nsup=1
print "ng=" ng=sqr(3)
print "nsub=" nsub=sqr(2.3)

print "tg=" tg=lambda
print "tf=" tf=lambda/pi

print "y3=" y3=(tf+tg)/2
print "y4=" y4=-y3
print "y1=" y1=y3-tg/2
print "y5=" y5=y3-tg
print "y2=" y2=(y5+y4)/2
print "y6=" y6=-toty/2
print "th=" th=y4-y6
print "y7=" y7=(y4+y6)/2

# PML's definitions
print gammar=0.5
print gammai=-0.5
print py=lambda/8

# Lengths of the different sections.
# try to change epsilon and see that the
# output n_eff does not change.
print epsilon=lambda/8
print l3=lambda/4-epsilon
print l2=lambda/4
print l1=epsilon

print "alpha=" alpha=0.0

size totx toty
wavelength lambda
carpet

section l1
    substrate nsup 0
    rectangle ng 0 totx tf 0 y2
    rectangle nsub 0 totx th 0 y7

    matdev la alpha
    pml_transf 0 2*py gammar gammai

    inpstruct i 1 301 sect_l1.txt

section l2
    substrate nsup 0
    rectangle ng 0 totx tg 0 y1
    rectangle ng 0 totx tf 0 y2
    rectangle nsub 0 totx th 0 y7

    matdev la alpha
    pml_transf 0 2*py gammar gammai

    inpstruct i 1 301 sect_l2.txt
```

```
section l3
    substrate nsup 0
    rectangle ng 0 totx tf 0 y2
    rectangle nsub 0 totx th 0 y7

    matdev la alpha
    pml_transf 0 2*py gammar gammai

    inpstruct i 1 301 sect_l3.txt

wants propagation

solve

assemble

# Bloch-mode calculation
bloch

# Now we write all found modes having an abs(imaginary part) less
# than 1e-3
outbloch 1 il 1e-3

outbloch 1 fi spectrum.bloch_TE

# Uses the Bloch mode having n_eff the closest to 1.582 as an
# excitation for the propagation. Note that in this case both
# forward and backward  excitations are automatically provided.
excitation f bl 1 0 1 1.582

# Output of the field.
propagation Ex2 m 10e-9 1 201 propagEx.txt
```

Note the use of the Lalanne-type matrix developments, with the choice of α appropriate for achieving a good convergence rate. The result for the provided script is that there is a TE Bloch-mode with $n_{\text{eff,TE}} = 1.58200 - \text{j}2.25523 \times 10^{-3}$. Adapting the script for the TM mode yields $n_{\text{eff,TM}} = 1.60902 - \text{j}7.13916 \times 10^{-4}$. Values in ref. [13] are respectively $n_{\text{eff,TE}} = 1.583 - \text{j}2.24 \times 10^{-3}$ and $n_{\text{eff,TM}} = 1.609 - \text{j}7.4 \times 10^{-4}$.

2.13 Conclusion

In this chapter, we have seen the general use of the afmm program, as well as the description of the commands which can be employed. We begun with a few hints about the installation and the general configuration and we then described in detail each available command. We finally commented a few examples showing what afmm can do and how the commands should be combined in order to calculate guided propagation modes of a waveguide as well as propagating electromagnetic field in a structure.

Appendix A

Matrix constructions

In this chapter, we will see a few examples of modified block-Toeplitz matrices, as well as the vector representation.

A.1 The block-Toeplitz matrix

Let's consider the term:

$$T = \sum_{c_x} \sum_{c_y} M_{i-c_x, j-c_y} C_{c_x, c_y} \tag{A.1}$$

which is a convolution between the space harmonics representing μ_x and those representing H_y. We notice that this term comes directly from the product $\mu_x H_y$, and this is a direct consequence of the well known convolution theorem for Fourier transforms.

The first problem we have to face is that up to now, the Fourier expansion has been done by using an infinite number of harmonics. We thus need to truncate the series to a finite number of terms. We will call S_x and S_y the number of positive and negative coefficients considered in x and y. The infinite sum of the convolution (A.1) thus becomes finite:

$$T'_{i,j} = \sum_{c_x=-(S_x-1)}^{(S_x-1)} \sum_{c_y=-(S_y-1)}^{(S_y-1)} M_{i-c_x, j-c_y} C_{c_x, c_y} \tag{A.2}$$

We now want to write it as a matrix-vector product:

$$T' = M\bar{C} \tag{A.3}$$

To obtain this, we can unroll the x and y components of the C_{c_x,c_y} space har-

monics in order to write the vector $\bar{C}$ as follows:

$$\bar{C} = \begin{pmatrix} C_{-(S_x-1),-(Sy-1)} \\ C_{-(S_x-2),-(S_y-1)} \\ C_{-(S_x-3),-(S_y-1)} \\ \cdots \\ C_{S_x-1,-(S_y-1)} \\ C_{-(S_x-1),-(Sy-2)} \\ \cdots \\ C_{-(S_x-1),0} \\ C_{-(S_x-2),0} \\ \cdots \\ C_{-1,0} \\ C_{0,0} \\ C_{1,0} \\ \cdots \\ C_{(S_x-1),(Sy-1)} \end{pmatrix} \tag{A.4}$$

This vector can be assembled by rearranging the coefficients of the $(2S_y - 1) \times (2S_x - 1)$ Fourier matrix:

$$C = \begin{pmatrix} C_{0,0} & C_{1,0} & \cdots & C_{S_x-1,0} & \cdots & C_{-1,0} \\ C_{0,1} & C_{1,1} & & & & \\ C_{0,2} & C_{1,2} & & & & \\ \cdots & \cdots & & & & \\ C_{0,S_y-1} & C_{1,S_y-1} & & & & \\ C_{0,-(S_y-1)} & C_{1,-(S_y-1)} & & & & \\ C_{0,-(S_y-2)} & C_{1,-(S_y-2)} & & & & \\ \cdots & \cdots & & & & \\ C_{0,-1} & C_{1,-1} & & \cdots & & C_{-1,-1} \cdots \end{pmatrix} \tag{A.5}$$

Which is the usual form in which the results of a 2D FFT algorithm are stored.[1] The matrix M_{T} can be constructed from $M_{i,j}$ as a block-Toeplitz matrix. We will use the notation M_{T} to stress that this is a block-Toeplitz matrix built from the $M_{i,j}$ Fourier coefficients:

$$M_{\mathrm{T}} = \begin{pmatrix} T_0 & T_{-1} & T_{-2} & \cdots \\ T_1 & T_0 & T_{-1} & \cdots \\ T_2 & T_1 & T_0 & \cdots \\ \cdots & & & \\ & & & \end{pmatrix} \tag{A.6}$$

in which each block T_j can be written as follows:

$$T_j = \begin{pmatrix} M_{0,j} & M_{-1,j} & M_{-2,j} & \cdots \\ M_{1,j} & M_{0,j} & M_{-1,j} & \cdots \\ M_{2,j} & M_{1,j} & M_{0,j} & \cdots \\ \cdots & & & \\ & & & \end{pmatrix} \tag{A.7}$$

[1] Note that in usual matrix notation $A_{i,j}$ means the i-row and j-column, while we want to maintain a strong relationship with space coordinates.

An example of construction of such a matrix is given in section A.4. The size of the (square) M_{T} matrix[2] is thus $(S_x S_y) \times (S_x S_y)$.

A.2 Representing derivatives: the modified Toeplitz notation

In equations describing field propagation, we can find complex convolutions, like the following one:

$$T_{i,j} = \sum_{d_x} \sum_{d_y} R_{i-d_x, j-d_y}(\mathrm{j})(j - d_y)\nu_y D_{d_x,d_y}(\mathrm{j}d_x\nu_x) \tag{A.8}$$

which in the Fourier base represents the term:

$$\frac{\partial}{\partial y}\frac{1}{\mathrm{j}\omega\epsilon_z}\frac{\partial H_y}{\partial x} \tag{A.9}$$

A difference from the convolution (A.1) is that there is a multiplication of each term D_{d_x,d_y} by the constant $(\mathrm{j}d_x\nu_x)$. This is given by the x derivative of H_y. We have also $R_{i-d_x,j-d_y}$ multiplied by the term $(\mathrm{j})(j-d_y)\nu_y$, which comes from the y derivative of ϵ_z^{-1}, as we have seen in section A.1. We want to write this convolution in a compact matrix by vector product:

$$T'' = R_{\mathrm{T}y}^{(x)} \bar{D} \tag{A.10}$$

where the $\bar{D}$ vector is built as usual from the $D_{i,j}$ Fourier components. The shorthand notation $R_{\mathrm{T}y}^{(x)}$ means that we have taken into account in the construction of the matrix some products coming from the derivatives. This can be obtained by using the Hadamard product of the block-Toeplitz matrix by two matrices $\mathcal{C}_y$ and $\mathcal{C}^{(x)}$, which contain the multiplication terms coming from derivatives:

$$R_{\mathrm{T}y}^{(x)} = R_{\mathrm{T}} \bullet \mathcal{C}_y \bullet \mathcal{C}^{(x)} \tag{A.11}$$

The first matrix $\mathcal{C}_y$ is the block-Toeplitz matrix built with space harmonics:

$$\mathcal{C}_y = \begin{pmatrix} T_0 & T_{-1} & T_{-2} & \cdots \\ T_1 & T_0 & T_1 & \cdots \\ T_2 & T_1 & T_0 & \cdots \\ \cdots & & & \end{pmatrix} \tag{A.12}$$

in which each block T_j is a constant matrix:

$$T_j = \begin{pmatrix} j\nu_y & j\nu_y & j\nu_y & \cdots \\ j\nu_y & j\nu_y & j\nu_y & \cdots \\ j\nu_y & j\nu_y & j\nu_y & \cdots \\ \cdots & & & \end{pmatrix} \tag{A.13}$$

The matrix $\mathcal{C}^{(x)}$ is made by $(S_x S_y)$ identical columns containing multiples of the fundamental space frequency arranged in the same order used during

[2] If we take S_x space harmonics, this gives us $2S_x + 1$ Fourier coefficient. We can compute only S_x Fourier terms of the result of the convolution, thus giving a matrix size of $(S_x S_y) \times (S_x S_y)$.

the construction of the block-Toeplitz matrix M_{T} seen above. The section A.5 presents a few examples of construction of small size $\mathcal{C}$ matrices.

We will call the matrix $R^{(x)}_{\mathrm{T}y}$ a modified block-Toeplitz matrix. This type of notation is completely general and will allow us to write matrix equations directly and quickly from equations in the original space. To represent in the finite size Fourier base products and derivatives, one has simply to write the corresponding modified block-Toeplitz matrix.[3]

A.3 Unrolling vectors

The first difficulty encountered in treating the 2D problem is that we need to store in a vector way the 2D set of space harmonics. The usual representation given for example by a standard 2D FFT stores the harmonics in a matrix. In our examples, we will consider a $5 \times 5 = (2S_y - 1) \times (2S_x - 1)$ matrix size in which the real part of each element gives the the x harmonic order, while the imaginary part gives the y harmonic order. In the real case, this will be used for the calculation of the Fourier transform by means of a FFT or an analytically calculated Fourier transform, but this will allow us to develop the examples for the following more complex constructions. Let's see an example:

$$A = \begin{pmatrix} 0 & 1 & 2 & -2 & -1 \\ \mathrm{j} & 1+\mathrm{j} & 2+\mathrm{j} & -2+\mathrm{j} & -1+\mathrm{j} \\ 2\mathrm{j} & 1+2\mathrm{j} & 2+2\mathrm{j} & -2+2\mathrm{j} & -1+2\mathrm{j} \\ -2\mathrm{j} & 1-2\mathrm{j} & 2-2\mathrm{j} & -2-2\mathrm{j} & -1-2\mathrm{j} \\ -1\mathrm{j} & 1-1\mathrm{j} & 2-1\mathrm{j} & -2-1\mathrm{j} & -1-1\mathrm{j} \end{pmatrix} \tag{A.14}$$

In our case, we obtain $S_x = 3$ and $S_y = 3$, which means that we are considering three space harmonics, including the zero-order (continuous) term. Unrolling the A matrix gives us the $\bar{A}$ vector:

$$\begin{aligned} \bar{A} = (&-2-2\mathrm{j}, -1-2\mathrm{j}, -2\mathrm{j}, 1-2\mathrm{j}, 2-2\mathrm{j}, \\ &-2-\mathrm{j}, -1-\mathrm{j}, 0-\mathrm{j}, 1-\mathrm{j}, 2-\mathrm{j}, \\ &-2, -1, 0, 1, 2, -2+\mathrm{j}, -1+\mathrm{j}, \mathrm{j}, 1+\mathrm{j}, 2+\mathrm{j}, \\ &-2+2\mathrm{j}, -1+2\mathrm{j}, 2\mathrm{j}, 1+2\mathrm{j}, 2+2\mathrm{j})^{\mathrm{T}} \end{aligned} \tag{A.15}$$

where T means the transpose operation to obtain a column vector.

A.4 Creating bloc-Toeplitz matrix

We can write the corresponding Bloc-Toeplitz matrix for the vector $\bar{A}$. The multiplication of that matrix for an unrolled vector $\bar{B}$ will result in the convolution in the Fourier domain between the space harmonics of the two vectors. This typically represent a simple multiplication in the original space. The A matrix will have a $S_xS_y \times S_xS_y$ size, which in our example is 9×9.

[3] Please note that the multiplication times the j imaginary unit coming from the derivatives is not taken into account in the matrix construction and should be done explicitly.

The bloc-Toeplitz matrix in our case is the following one:

$$A_{\mathrm{T}} = \begin{pmatrix} 0+0\mathrm{j} & -1+0\mathrm{j} & -2+0\mathrm{j} & 0-1\mathrm{j} & -1-1\mathrm{j} & -2-1\mathrm{j} & 0-2\mathrm{j} & -1-2\mathrm{j} & -2-2\mathrm{j} \\ 1+0\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} & 1-1\mathrm{j} & 0-1\mathrm{j} & -1-1\mathrm{j} & 1-2\mathrm{j} & 0-2\mathrm{j} & -1-2\mathrm{j} \\ 2+0\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} & 2-1\mathrm{j} & 1-1\mathrm{j} & 0-1\mathrm{j} & 2-2\mathrm{j} & 1-2\mathrm{j} & 0-2\mathrm{j} \\ 0+1\mathrm{j} & -1+1\mathrm{j} & -2+1\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} & -2+0\mathrm{j} & 0-1\mathrm{j} & -1-1\mathrm{j} & -2-1\mathrm{j} \\ 1+1\mathrm{j} & 0+1\mathrm{j} & -1+1\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} & 1-1\mathrm{j} & 0-1\mathrm{j} & -1-1\mathrm{j} \\ 2+1\mathrm{j} & 1+1\mathrm{j} & 0+1\mathrm{j} & 2+0\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} & 2-1\mathrm{j} & 1-1\mathrm{j} & 0-1\mathrm{j} \\ 0+2\mathrm{j} & -1+2\mathrm{j} & -2+2\mathrm{j} & 0+1\mathrm{j} & -1+1\mathrm{j} & -2+1\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} & -2+0\mathrm{j} \\ 1+2\mathrm{j} & 0+2\mathrm{j} & -1+2\mathrm{j} & 1+1\mathrm{j} & 0+1\mathrm{j} & -1+1\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} \\ 2+2\mathrm{j} & 1+2\mathrm{j} & 0+2\mathrm{j} & 2+1\mathrm{j} & 1+1\mathrm{j} & 0+1\mathrm{j} & 2+0\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} \end{pmatrix} \tag{A.16}$$

The modification matrices $\mathcal{C}_x$ and $\mathcal{C}_y$ are the following ones :

$$\mathcal{C}_x = \nu_x \begin{pmatrix} 0 & -1 & -2 & 0 & -1 & -2 & 0 & -1 & -2 \\ 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 \\ 0 & -1 & -2 & 0 & -1 & -2 & 0 & -1 & -2 \\ 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 \\ 0 & -1 & -2 & 0 & -1 & -2 & 0 & -1 & -2 \\ 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 \end{pmatrix} \tag{A.17}$$

$$\mathcal{C}_y = \nu_y \begin{pmatrix} 0 & 0 & 0 & -1 & -1 & -1 & -2 & -2 & -2 \\ 0 & 0 & 0 & -1 & -1 & -1 & -2 & -2 & -2 \\ 0 & 0 & 0 & -1 & -1 & -1 & -2 & -2 & -2 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 2 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{A.18}$$

These terms represent in the Fourier space the x and y derivative of the first term of the product in the original space.

The modification matrices $\mathcal{C}^{(x)}$ and $\mathcal{C}^{(y)}$ are the following ones (for $\nu_x = \nu_y = 1$):

$$\mathcal{C}^{(x)} = \nu_x \begin{pmatrix} -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \end{pmatrix} \tag{A.19}$$

$$\mathcal{C}^{(y)} = \nu_y \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{A.20}$$

These terms represent in the Fourier space the x and y derivative of the first term of the product in the original space.

A.5 Modified matrices

Let's see an example of modified bloc-Toeplitz matrices. This represent in the Fourier space a product between two terms in which there are derivatives. The modified bloc-Toeplitz matrix can be seen as the Hadamard product between the usual bloc-Toeplitz matrix and the modification matrices. Here are a few examples (for $\nu_x = \nu_y = 1$):

$$A_{\mathrm{Tx}} = \begin{pmatrix} 0+0j & 1+0j & 4+0j & 0+0j & 1+1j & 4+2j & 0+0j & 1+2j & 4+4j \\ 1+0j & 0+0j & 1+0j & 1-1j & 0+0j & 1+1j & 1-2j & 0+0j & 1+2j \\ 4+0j & 1+0j & 0+0j & 4-2j & 1-1j & 0+0j & 4-4j & 1-2j & 0+0j \\ 0+0j & 1-1j & 4-2j & 0+0j & 1+0j & 4+0j & 0+0j & 1+1j & 4+2j \\ 1+1j & 0+0j & 1-1j & 1+0j & 0+0j & 1+0j & 1-1j & 0+0j & 1+1j \\ 4+2j & 1+1j & 0+0j & 4+0j & 1+0j & 0+0j & 4-2j & 1-1j & 0+0j \\ 0+0j & 1-2j & 4-4j & 0+0j & 1-1j & 4-2j & 0+0j & 1+0j & 4+0j \\ 1+2j & 0+0j & 1-2j & 1+1j & 0+0j & 1-1j & 1+0j & 0+0j & 1+0j \\ 4+4j & 1+2j & 0+0j & 4+2j & 1+1j & 0+0j & 4+0j & 1+0j & 0+0j \end{pmatrix} \tag{A.21}$$

$$A_{\mathrm{Ty}} = \begin{pmatrix} 0+0j & 0+0j & 0+0j & 0+1j & 1+1j & 2+1j & 0+4j & 2+4j & 4+4j \\ 0+0j & 0+0j & 0+0j & -1+1j & 0+1j & 1+1j & -2+4j & 0+4j & 2+4j \\ 0+0j & 0+0j & 0+0j & -2+1j & -1+1j & 0+1j & -4+4j & -2+4j & 0+4j \\ 0+1j & -1+1j & -2+1j & 0+0j & 0+0j & 0+0j & 0+1j & 1+1j & 2+1j \\ 1+1j & 0+1j & -1+1j & 0+0j & 0+0j & 0+0j & -1+1j & 0+1j & 1+1j \\ 2+1j & 1+1j & 0+1j & 0+0j & 0+0j & 0+0j & -2+1j & -1+1j & 0+1j \\ 0+4j & -2+4j & -4+4j & 0+1j & -1+1j & -2+1j & 0+0j & 0+0j & 0+0j \\ 2+4j & 0+4j & -2+4j & 1+1j & 0+1j & -1+1j & 0+0j & 0+0j & 0+0j \\ 4+4j & 2+4j & 0+4j & 2+1j & 1+1j & 0+1j & 0+0j & 0+0j & 0+0j \end{pmatrix} \tag{A.22}$$

$$A_{\mathrm{T}}^{(x)} = \begin{pmatrix} 0+0j & 0+0j & -2+0j & 0+1j & 0+0j & -2-1j & 0+2j & 0+0j & -2-2j \\ -1+0j & 0+0j & -1+0j & -1+1j & 0+0j & -1-1j & -1+2j & 0+0j & -1-2j \\ -2+0j & 0+0j & 0+0j & -2+1j & 0+0j & 0-1j & -2+2j & 0+0j & 0-2j \\ 0-1j & 0+0j & -2+1j & 0+0j & 0+0j & -2+0j & 0+1j & 0+0j & -2-1j \\ -1-1j & 0+0j & -1+1j & -1+0j & 0+0j & -1+0j & -1+1j & 0+0j & -1-1j \\ -2-1j & 0+0j & 0+1j & -2+0j & 0+0j & 0+0j & -2+1j & 0+0j & 0-1j \\ 0-2j & 0+0j & -2+2j & 0-1j & 0+0j & -2+1j & 0+0j & 0+0j & -2+0j \\ -1-2j & 0+0j & -1+2j & -1-1j & 0+0j & -1+1j & -1+0j & 0+0j & -1+0j \\ -2-2j & 0+0j & 0+2j & -2-1j & 0+0j & 0+1j & -2+0j & 0+0j & 0+0j \end{pmatrix} \tag{A.23}$$

$$A_{\mathrm{T}}^{(y)} = \begin{pmatrix} 0+0\mathrm{j} & 1+0\mathrm{j} & 2+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0-2\mathrm{j} & -1-2\mathrm{j} & -2-2\mathrm{j} \\ -1+0\mathrm{j} & 0+0\mathrm{j} & 10\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 1-2\mathrm{j} & 0-2\mathrm{j} & -1-2\mathrm{j} \\ -2+0\mathrm{j} & -1+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 2-2\mathrm{j} & 1-2\mathrm{j} & 0-2\mathrm{j} \\ 0-1\mathrm{j} & 1-1\mathrm{j} & 2-1\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0-1\mathrm{j} & -1-1\mathrm{j} & -2-1\mathrm{j} \\ -1-1\mathrm{j} & 0-1\mathrm{j} & 1-1\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 1-1\mathrm{j} & 0-1\mathrm{j} & -1-1\mathrm{j} \\ -2-1\mathrm{j} & -1-1\mathrm{j} & 0-1\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 2-1\mathrm{j} & 1-1\mathrm{j} & 0-1\mathrm{j} \\ 0-2\mathrm{j} & 1-2\mathrm{j} & 2-2\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} & -2+0\mathrm{j} \\ -1-2\mathrm{j} & 0-2\mathrm{j} & 1-2\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} & -1+0\mathrm{j} \\ -2-2\mathrm{j} & -1-2\mathrm{j} & 0-2\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 0+0\mathrm{j} & 2+0\mathrm{j} & 1+0\mathrm{j} & 0+0\mathrm{j} \end{pmatrix} \tag{A.24}$$

A.6 Conclusion

In this chapter, we have seen a few examples of implementation of the matrices useful for the 2D Fourier Modal Method mode solver. This can be useful for those needing to obtain a deep understanding of the inners of afmm, or wanting to work on the source code.

Appendix B

Plotting modes profiles with Gnuplot

Gnuplot[4] is a very flexible scientific graphical plotting system, even if it has a steep learn curve. One of its main advantages (apart the fact that is open source and free) is that it can be used through scripts which can perform complex operations. It runs on a variety of platforms, from classic and less classic Unices to Windows. In this appendix, we will give a few plotting scripts useful when using afmm, as some of its commands adopt a file format directly compatible with Gnuplot (see appendix D.1 for details).

B.1 Plotting a structure

The following script called `format_eps.gp` can be called from a Gnuplot interactive session by using the `call` command. The first parameter will be used as the name of the input file. The second parameter is the name of the output file, while the last parameter can be put equal to 1 if the output should be done also on the current terminal.

```
# format_eps.gp: plot a nice image from raw data file
# useful for afmm output
# The first parameter is the input file, formatted using the afmm
# conventions.
# The second parameter is the output file, which should be an eps
# file.
# The third parameter is 1 if the graph should be shown in the
# current terminal.
#
# usage in Gnuplot:
# call "format_eps.gp" "test.structure" "test.eps" 0
#
# Davide Bucci, february 17, 2009

# General settings: color map view
set style data lines
unset surface; set contour base
set view 0,0

# Put x and y axis labels. Note the direct use of Postcript code
# to obtain Greek characters
```

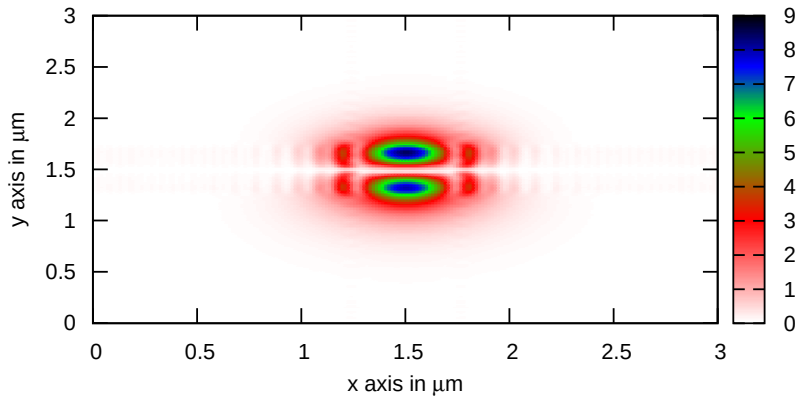

Figure B.1: The intensity of a guided mode (in arbitrary unit) calculated by afmm and plot by Gnuplot.

```
set xlabel 'x axis in {/Symbol m}m'
set ylabel 'y axis in {/Symbol m}m'

# Define the palette to be used
set key below
set pm3d
set palette color negative
set palette defined (0 0 0 0, 1 0 0 1, 2 0 1 0, 4 1 0 0, 6 1 1 1)
set pal maxcolor 256
set surface
unset contour
unset key
set view map

# Draw the color map of the file specified on the first parameter.
# Note the explicit conversion between meters and micrometers.

if($2==1) splot '$0' using ($$1*1e6):($$2*1e6):($$3) with pm3d

# Prepare an encapsulated Postscript file, ready for printing
set size 1.0, 0.4
set term push
set terminal postscript portrait color enhanced "Helvetica" 12
set output '$1';
splot '$0' using ($$1*1e6):($$2*1e6):($$3) with pm3d
set output

# Newer Gnuplot versions allow to use a stack for saving the
# settings of the terminal.

set terminal pop
set size 1,1
```

An example of a plot obtained with this script can be seen in figure B.1.

Appendix C

The slice utility

When working with big files obtained by calculating the field propagation in three dimensional structures, it is often useful to represent the behaviour of the fields by using some cut planes. The slice utility can be used for that and can provide files useful for representing fields with Gnuplot, as shown in appendix B.

C.1 Usage

The slice utility has the following syntax:

```
slice [-lx] [-ly] [-lz] {xy|xz|yz} quota tolerance in_file out_file
```

where:

- The `-lx`, `-lx` and `-lx` options indicate that slice should introduce a blank line in the output file each time there is an increment detected in the x, y or z coordinates. This behaviour is very useful for Gnuplot. In reality, the utility tries to understand the number of points (called a *scanline*) between each increment at the very beginning of the files and then uses the same increment for the rest of the file without checking. This behaviour is quite useful with curved sections where z ans x coordinates are changing at the same time in one scan line.
- `xy`, `xz` and `yz` indicates the plane which slice should be use for the representation of the fields.
- `quota` is the quota at which the field should be sliced. Remember that afmm always works in meters and so does slice.
- `tolerance` is the tolerance allowed for a field coordinate in the input file to be considered in the slice to be done. All points in the input file are reported in the output file only if their distance from the quota of the slicing plane is less than the given tolerance. Note that if the tolerance is too small or the choice of the quota is incorrect, no points will be present in the output file. On the other hand, if `tolerance` is too coarse, more than one set of points will be superposed in the output file.
- `in_file` is the file containing the 3D field to slice.
- `out_file` is the file which will contain the resulting slice.

C.2 Examples

A xz cut at quota $y = 0$ with tolerance $\Delta y = 10^{-12}$ m of the field contained in the `propagationEx.txt` file. The result will be written in `propagationEx_xz.txt` and will contain a blank line each time there is an increment in z, which is the propagation axis:

```
slice -lz xz 0 1e-12 propagationEx.txt propagationEx_xz.txt
```

Let's say now we want to represent a xy cut of the field at the end of the calculation window. First of all, we should have a peek at the very end of the `propagationEx.txt` file containing the results of the calculation. Very often, this file is quite big and can be a problem to open it with a text editor. The UNIX command `tail` works like a charm even on very big files and allows to read immediately the last lines of it:

```
[davidebucci@davide-bucci-portable]$ tail propagationEx.txt
9.022612e-06 1.496259e-06 1.199368e-05 3.132949e-02
9.027600e-06 1.496259e-06 1.199368e-05 3.123777e-02
9.032587e-06 1.496259e-06 1.199368e-05 3.103430e-02
9.037575e-06 1.496259e-06 1.199368e-05 3.072648e-02
9.042562e-06 1.496259e-06 1.199368e-05 3.032795e-02
9.047550e-06 1.496259e-06 1.199368e-05 2.985806e-02
9.052537e-06 1.496259e-06 1.199368e-05 2.934075e-02
9.057525e-06 1.496259e-06 1.199368e-05 2.880309e-02
9.062512e-06 1.496259e-06 1.199368e-05 2.827333e-02
9.067500e-06 1.496259e-06 1.199368e-05 2.777873e-02
[davidebucci@davide-bucci-portable]$
```

Data is represented here as x, y, z and magnitude of the field, so we see that the last z coordinate is $11.99368\,\mu$m Once we are sure that the quota $z = 11.99368\,\mu$m is indeed present in the `propagationEx.txt` file, we can fix a quite small tolerance. A xy cut at quota $z = 11.99368\,\mu$m with tolerance $\Delta z = 10^{-12}$ m of the field contained in the `propagationEx.txt` file is obtained with the following command. The result will be written in `propagationEx_xy_e.txt` and will contain a blank line each time there is an increment in y:

```
slice -ly xy 1.199368e-05 1e-12 propagationEx.txt endEx_xy.txt
```

C.3 Conclusion

We described the slice utility, which can be used to obtain slices of the field calculated by afmm during a full 3D field propagation simulation. We described the syntax of the command and some examples have been provided.

Appendix D

File formats used by afmm

D.1 2D and 3D formats, compatible with Gnuplot

The file format used is extremely simple and is organised in order to be directly compatible with Gnuplot way of life. It is a text file, each record being a line, terminated by the line ending style of the host operating system.

D.1.1 Header

No header is used, but afmm gives some kind of useful data as comments (the Gnuplot comment character used is # at the beginning of the line).

D.1.2 Data section of the file

There is a line for each point contained in the file. First of all the x and y coordinates are specified (in meters), then the contents are given. The exact content depends on the output of the afmm command being used.

D.1.3 Example

Let's consider a complete example, as issued by a command similar to the following (see section 2.4.8 for the details):

```
outgmodes Ex c 150 150 test
```

Here is what it may be obtained:

```
# neff = (3.233212, 2.1233242e-8)
# x         y         Ex.real       Ex.imag
0.000000e+00 0.000000e+00 -2.881303e-04 6.743971e-05
1.500000e-08 0.000000e+00 -2.261833e-04 5.277011e-05
3.000000e-08 0.000000e+00 -7.094755e-05 1.610860e-05
...
```

The first line contains a comment about the fact that this file is a propagation mode, giving the real and imaginary part of the effective refractive index. The

second line is again a comment, describing the association of each column. Since we requested to specify a complex representation of the E_x field, what we obtain is the x and y coordinates, followed by the real and imaginary value of the E_x field. The number of lines contained after the header is 150×150. The presence of an header (here a comment) is not compulsory.

D.2 OptiWave software compatible formats

D.2.1 Refractive index distribution: rid

The `rid` file format is described here and it is compatible with files provided by the OptiWave OptiBPM software. It is a text file, each record being a line, terminated by the Windows line ending (`CR` + `LF`).

Header

The first line of the file is ignored by afmm, but may contain `UPI3DRI 3.0`, in order to retain a compatibility with Optiwave. The second line contains the number of x and y space subdivisions, separated by a space. The third line of the header specifies the minimum and maximum ranges of the calculation window, for the x and y axis. Normally, those values are specified in micrometers, whereas afmm adopts the meter as the internal length unity. For this reason, in all commands on which this file format is used, a parameter called `mult` should be specified. The effect of this parameter is to multiply every length specified in the file for the given fixed quantity. For example, very often the user may need to convert from micrometers to meters, thus specifying `1e-6` as `mult`.

Data section of the file

The remaining part of the file after the header specifies a sampled index distribution, point by point, line by line. The refractive index is specified with its real part and imaginary part, separated by a comma.

Example

Here is an example of the header and the beginning of the data section:

```
UPI3DRI 3.0
240 159
0 12 0 7.96638655462185
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
1, 0.000000000000000
```

```
1, 0.000000000000000
1, 0.000000000000000
...
```

The file is composed by 240×159 points, with an x range of $12\,\mu$m and a y range of $7.966\,\mu$m, starting from the origin. This file may be loaded with the `mult` parameter set to 1×10^{-6}.

D.2.2 Electric fields: f3d

Optiwave's 3D field format is very close to the .rid format described in the previous paragraphs. The only notable difference is that the first line is

D.3 Conversion utility: o2g

The o2g utility is part of afmm and can be used to convert any of the Optiwave file format into a Gnuplot-compatible file. The usage is as follows:

```
[davidebucci@imep197 -231-6]$ ./o2g
Optiwave to Gnuplot converter.
Usage ./o2g file_in.optiwave file_out.gnuplot
[davidebucci@imep197 -231-6]$
```

Appendix E

Python interface

E.1 Introduction

Python is a highly powerful scripting language that has become vastly popular in the scientific community. It is sufficiently versatile to tackle a variety of problems, it comes with a comprehensive collection of libraries and can be learnt quickly even by unskilled users.

afmm is interfaced to Python and can be used as a module. The functions provided by that module are inspired by the commands described in chapter 2 and therefore a full description will not be given here.

E.2 Commands

In the following table, a summary of all afmm commands is given, with the correspondance in Python. Refractive indices are always complex values in the Python commands.

Commande afmm	**Python (use `import pyAFMM as afmm`)**	Section
angles	`afmm.angles(n0, thetax, thetay)`	2.3.13
assemble	`afmm.assemble()`	2.3.7
bend	`afmm.bend(radius)`	2.4.12
bloch	`afmm.bloch()`	2.3.12
carpet	`afmm.carpet()`	2.3.9
clear	`afmm.clear()`	2.5.17
coefficient		
eigenam		
eigenvc		
excitation		
harmonics	`afmm.harmonics(nx, ny)`	2.3.2
highindex	`afmm.highindex(complex)`	2.4.11
indfile	`afmm.indmatrix(list)`	2.4.6
inpstruct	`afmm.inpstruct()`	2.4.7
let		
load	`afmm.parsescript()`	2.5.7
lowindex	`afmm.lowindex(complex)`	2.4.10
matdev		
memocc		
monitor	`afmm.monitor(z,wx,wy,px,py)`	2.4.23
norm		
order	`afmm.order(min, max)`	2.4.14
outbloch		

outdata		
outgmodes	afmm.outgmodes(fieldtype, nx, ny)	2.4.8
parallel		
pml_transf	afmm.pml_transf(qx,qy,complex)	2.4.5
pml	afmm.pml(complex,wx,wy)	2.4.4
power		
powerz	afmm.powerz(z)	2.4.22
print		
propagation		
rectangle	afmm.rectangle(index, wx, wy, px, py)	2.4.3
section	afmm.section(length)	2.4.1
select	afmm.select(length)	2.4.16
selmodes		
size	afmm.size(sx, sy)	2.3.1
solve	afmm.solve()	2.3.4
spectrum	afmm.spectrum()	2.4.13
substrate	afmm.substrate(index)	2.4.2
symmetry		
wants	afmm.wants(type)	2.3.5
wavelength	afmm.wavelength(lambda)	2.3.3

E.3 Commands operating with tables

Some of the afmm commands read files or write on them. Working in Python allows a greater flexibility and therefore, instead of operating on files, the input and output of those commands is done with lists.

- `afmm.inpstruct()` returns a list of list of complex that represent the index cross section, i.e. a matrix as a list of lines.
- `afmm.outgmodes(f, sx,sy)` returns a list of list of list of complex that represent the interesting modes of the current section, i.e. (list of list of lines).
- `afmm.indmatrix(list)` takes a list of list of complex that contain the refractive index distribution to be loaded for the current section.
- `afmm.bloch()` returns a list of complex values that are the generalized eigenvalues found in the calculation.
- `afmm.spectrum()` returns a list of complex values that are the effective indices of the modes (both guided and a discretization of the radiated ones) that are supported by the section considered.

E.4 Python functions without an equivalent in the afmm script language

Some of the Python functions available do not have an equivalence in the afmm script language. Here they are:

E.5 Examples

The following Python script can be used to calculate the modes for the waveguide discussed in paragraph 2.7. Compare it with the afmm script discussed there.

```
# The following line imports the AFMM module that should have
# been correctly installed on your machine:
import pyAFMM as afmm

# Show the program AFMM banner and credits
afmm.banner()

# AFMM commands are mapped directly into Python functions:
afmm.size(2e-6,200e-9)
afmm.harmonics(201,1)
afmm.wavelength(1.55e-6)

# Each time in an AFMM script command there is a complex number
# to specify this is done by means of the real and imaginary part.
# In the Python access, this is handled directly by means of
# complex variables, as in the following command:
afmm.substrate(1.44+0j)

# Commands that are not yet accessible via Python can be accessed
# by means of 'parsescript'. You can even process a whole AFMM
# script contained in a Python string using this technique.
afmm.parsescript("matdev la 1.0")
afmm.rectangle(3.5+0j*0,500e-9,200e-9,0,0)

# Some commands give back a return value.
neff = afmm.solve()
print ("\nHere the effective indices in a Python list:\n")
print (neff)
```

Appendix F

Version history and change-log

F.1 Version 1.4.5

Released xx 2015, added command `modepos` and the position index selection in the `excitation` and `coefficient` command. Added commands `fscan`, `fcheck` and the read mode for `fopen`. Added the `do-while` construct. Corrected a bug in the `indfile` and in the `matdev` with the normal field commands which mirrored the read files.

F.2 Version 1.4.4

Released March 2015, added command `selmodes`.

F.3 Version 1.4.3

Released April 2014, added command `parallel`.

F.4 Version 1.4.2

Released July 2013, added commands `bloch`, `outbloch`. Bloch-mode excitation is available via the `bl` option of the `excitation` command. Added conditional test commands `if else endif` as well as several comparison operators `==`, `<`, `>`, `<=`, `>=`, `!=`, `||`, `&&`, `!`. Commands `wavelength`, `solve` and `bloch` now update the `ans` variable to provide an output. Indication of cycle variable becomes optional in `next` command. Some commands can provide a feedback via the `ans` variable. Write on a file is possible through `fopen`, `fprint`, `fclose`. Introduced `addspace` command for modifying way `print` and `fprint` work.

F.5 Version 1.4.1

Released 27 January 2013, great speedup of calculations via the implementation of symmetries. Added commands `power`, `powerz`, `monitor` and `symmetry`. More flexible comments are now allowed.

F.6 Version 1.4

Released July 2012, the main advancement has been the implementation of the normal vector field, as well as some bug fixes.

F.7 Version 1.3.5

Released April, 3, 2012, this release introduces commands `eigenam`, `eigenvc` as well as `memocc` have been added. Some bugs have been corrected, one affecting the phase of the magnetic field.

F.8 Version 1.3.4

Released around January 2012. The commands `goto`, `label`, `for` and `next` have been added.

F.9 Version 1.3.3

Released May, 1, 2011, several bugs about the field propagation have been corrected. The commands `norm`, `outdata`, `system` have been added.

F.10 Version 1.3.2

Released around February 2011, in this version the command `clear` has been added.

F.11 Version 1.3.1

Released around November 2010, in this version the commands `carpet` and `matdev` have been added.

F.12 Version 1.3

Released July 30, 2010, this is a major upgrade of afmm, introducing the possibility of calculating the propagation of the field. It introduces the `propagation`, `assemble`, `section`, `order`, `excitation`,`select`, `coefficient quit`, `help`, `load` and `print` commands. It introduces the interactive mode, as well as the possibility of defining variables and making simple calculations inside a script. Double quotes are used for preventing breaking up parameters containing spaces. The implementation of PMLs is slightly improved.

F.13 Version 1.2.3

Released November 23, 2009, it introduces the `spectrum` command.

F.14 Version 1.2.2

Released October 5, 2009, it allows the calculation of x and y components of the magnetic field. The `wants` command has been introduced.

F.15 Version 1.2.0

Released June 8, 2009, it corrects a bug in the implementation of coordinate transform PMLs. The `bend` command has been added, in order to calculate propagating modes of bent structures.

F.16 Version 1.1.0

Released April 28, 2009, several bugs in the FFT calculations have been corrected. The use of coordinate transform PMLs is possible, as well as reading an arbitrary refractive index distribution from a file. The `lowindex`, `highindex`, `pml_transf`, `indfile` commands have been added.

F.17 Version 1.0.2

Released February 18, 2009, it uses the FFTW3 library to calculate the input structure index distribution as well as the modal field associated to each guided mode. The `inpstruct` as well as the `outgmodes` have been added.

F.18 Version 1.0.1

Released February 11, 2009, a few bugs have been fixed as well as a complete support of the imaginary part of the refractive index used for rectangular structures in the calculation window. It can calculate only the effective index of the guided modes.

F.19 Version 1.0

It is the first working version, released January 11, 2009. It has several bugs in the calculation of permeability and permittivity vectors. It can calculate only the effective index of the guided modes.

Appendix G

List of publications related to afmm

G.0.1 Proceedings of international meetings and workshops

D. Bucci, B. Martin, A. Morand, "Study of propagation modes of bent waveguides and micro-ring resonator by means of the Aperiodic Fourier Modal Method," *Proceedings of SPIE*, Volume 7597 (2010)

G.0.2 International reviews

D. Bucci, B. Martin, A. Morand, "Application of the three-dimensional aperiodic Fourier modal method using arc elements in curvilinear coordinates," *JOSA A*, Vol. 29, Issue 3, pp. 367-373 (2012)

Bibliography

[1] "Atlas website." [Online]. Available: http://www.netlib.org/atlas/

[2] "Blas website." [Online]. Available: http://www.netlib.org/blas/

[3] "FFTW website." [Online]. Available: http://www.fftw.org/

[4] "Gnuplot website." [Online]. Available: http://www.gnuplot.info/

[5] "LAPACK website." [Online]. Available: http://www.netlib.org/lapack/

[6] "libf2c website." [Online]. Available: http://www.netlib.org/f2c/

[7] "OpenBLAS website." [Online]. Available: http://www.openblas.net/

[8] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[9] J. Bischoff, "Formulation of the normal vector rcwa for symmetric crossed gratings in symmetric mountings," *JOSA A*, vol. 27, no. 5, pp. 1024–1031, 2010.

[10] L. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, and Others, "An Updated Set of Basic Linear Algebra Subprograms (BLAS)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.

[11] D. Bucci, B. Martin, and A. Morand, "Study of propagation modes of bent waveguides and micro-ring resonators by means of the aperiodic Fourier modal method," in *Proceedings of SPIE*, vol. 7597, 2010, p. 75970U.

[12] ——, "Application of the three-dimensional aperiodic fourier modal method using arc elements in curvilinear coordinates," *JOSA A*, vol. 29, no. 3, pp. 367–373, 2012.

[13] Q. Cao, P. Lalanne, and J.-P. Hugonin, "Stable and efficient bloch-mode computational method for one-dimensional grating waveguides," *JOSA A*, vol. 19, no. 2, pp. 335–338, 2002.

[14] K. Chang, V. Shah, and T. Tamir, "Scattering and guiding of waves by dielectric gratings with arbitrary profiles," *JOSA*, vol. 70, no. 7, pp. 804–813, 1980.

[15] "Cygwin website." [Online]. Available: http://www.cygwin.com/

[16] E. W. Dijkstra, "Letters to the editor: go to statement considered harmful," *Communications of the ACM*, vol. 11, no. 3, pp. 147–148, 1968.

[17] O. El Daif, E. Drouard, G. Gomard, Y. Park, A. Kaminski, A. Fave, M. Lemiti, X. Letartre, P. Viktorovitch, S. Ahn, *et al.*, "Photonic band-engineering absorption enhancement of amorphous silicon for solar cells," in *Proc. SPIE*, vol. 7411, 2009, p. 74110O.

[18] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

[19] R. Gray, *Toeplitz and circulant matrices: A review*. Now Pub, 2006.

[20] J. Hench and Z. Strakoš, "The RCWA method-a case study with open questions and perspectives of algebraic computations," *Electronic Transactions on Numerical Analysis*, vol. 31, pp. 331–357, 2008.

[21] S. Hewlett and F. Ladouceur, "Fourier decomposition method applied to mapped infinite domains: scalar analysis of dielectric waveguides down to modal cutoff," *Lightwave Technology, Journal of*, vol. 13, no. 3, pp. 375–383, 1995.

[22] J. Hugonin and P. Lalanne, "Perfectly matched layers as nonlinear coordinate transforms: a generalized formalization," *J. Opt. Soc. Am. A*, vol. 22, no. 9, september 2005.

[23] J. Hugonin, P. Lalanne, I. Villar, and I. Matias, "Fourier modal methods for modeling optical dielectric waveguides," *Optical and quantum electronics*, vol. 37, no. 1, pp. 107–119, 2005.

[24] P. Lalanne, "Improved formulation of the coupled-wave method for two-dimensional gratings," *Journal of the Optical Society of America A*, vol. 14, no. 7, pp. 1592–1598, 1997.

[25] P. Lalanne and M. Jurek, "Computation of the near-field pattern with the coupled-wave method for transverse magnetic polarization," *Journal of Modern Optics*, vol. 45, no. 7, pp. 1357–1374, 1998.

[26] P. Lalanne and G. M. Morris, "Highly improved convergence of the coupled-wave method for TM polarization," *Journal of the Optical Society of America A*, vol. 13, no. 4, pp. 779–784, 1996.

[27] P. Lalanne and E. Silberstein, "Fourier-modal methods applied to waveguide computational problems," *Optics Letters*, vol. 25, no. 15, pp. 1092–1094, 2000.

[28] L. Li, "Formulation and comparison of two recursive matrix algorithms for modeling layered diffraction gratings," *Journal of the Optical Society of America A*, vol. 14, no. 5, pp. 1024–1035, 1996.

[29] ——, "Use of Fourier series in the analysis of discontinuous periodic structures," *Journal of the Optical Society of America A*, vol. 13, pp. 1870–1876, 1996.

[30] ——, "Recent advances and present limitations of the electromagnetic theory of diffraction gratings," *Diffractive Optics and Micro-Optics*, 2000.

[31] B. Martin, "Etude et réalisation d'un spectromètre compact en optique intégrée sur verre," Ph.D. dissertation, Grenoble INP, January 2009.

[32] J. Michallon, "Etude et optimisation de l'absorption optique et du transport électronique dans les cellules photovoltaïques à base de nanofils," Ph.D. dissertation, Grenoble INP, January 2015.

[33] "MinGW website." [Online]. Available: http://www.mingw.org/

[34] M. Moharam and T. Gaylord, "Rigorous coupled-wave analysis of planar-grating diffraction," *JOSA*, vol. 71, no. 7, pp. 811–818, 1981.

[35] "rlwrap website." [Online]. Available: http://utopia.knoware.nl/ hlub/uck-/rlwrap/

[36] T. Schuster, J. Ruoff, N. Kerwien, S. Rafler, and W. Osten, "Normal vector method for convergence improvement using the rcwa for crossed gratings," *JOSA A*, vol. 24, no. 9, pp. 2880–2890, 2007.

[37] E. Silberstein, P. Lalanne, J. Hugonin, and Q. Cao, "Use of grating theories in integrated optics," *Journal of the Optical Society of America A*, vol. 18, no. 11, pp. 2865–2875, 2001.

[38] R. Whaley, A. Petitet, and J. Dongarra, "Automated empirical optimizations of software and the ATLAS project," *Parallel Computing*, vol. 27, no. 1-2, pp. 3–35, 2001.