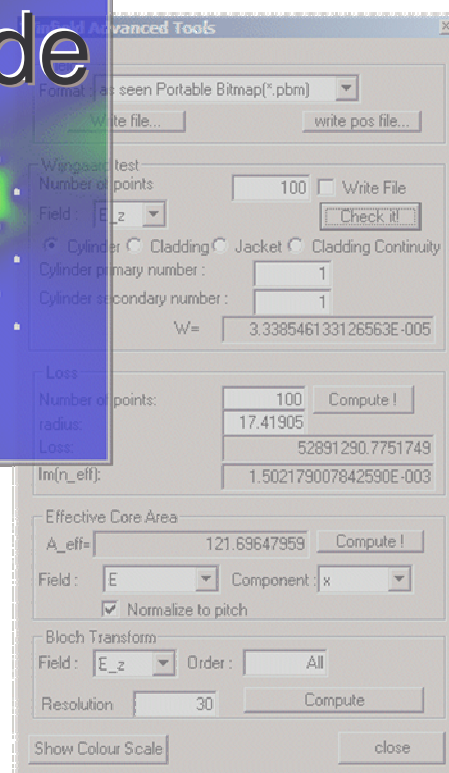
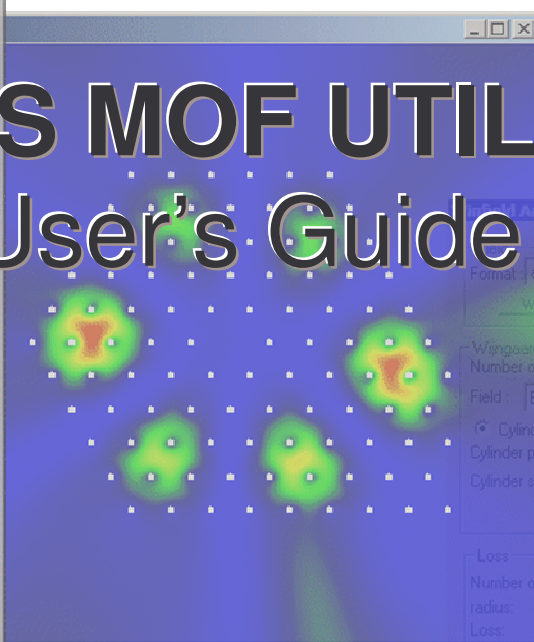


CUDOS MOF UTILITIES

User's Guide



Boris Kuhlmei, CUDOS/School of Physics A-28, University of Sydney NSW 2006 Australia
P: +612 9036 9430 | F: +612 9351 7726 | E: borisk@physics.usyd.edu.au | W: www.cudos.org.au

Contents

1	Introduction	3
2	Using FIBRE	4
2.1	Introduction	4
2.2	Installing and uninstalling FIBRE	4
2.3	Getting started: a first simulation	4
2.3.1	The input and output files	5
2.3.2	A closer look at the parameter file	6
2.4	Syntax and structure of the parameter file	8
2.4.1	Syntax	9
2.4.2	Structure	10
2.5	Parameter file keyword reference	11
2.5.1	Miscellaneous keywords	11
2.5.2	Fibre structure compiler keywords	11
2.5.3	Threshold keywords	14
2.5.4	Keywords for mode seeking	17
2.5.5	Keywords for computing dispersion curves	20
2.5.6	File name keywords	23
2.5.7	Keywords relating to the algorithm	23
2.6	Structure files	26
3	Using WINFIELD	28
3.1	Main dialog controls	28
3.2	Advanced tools dialog controls	29
4	Notes, tips, trouble shooting, known bugs:	31
4.1	FIBRE	31
4.2	WINFIELD	31
5	Acknowledgments	31

1 Introduction

The CUDOS MOF Utilities are a software package for simulating microstructured optical fibres (or MOFs, also known as photonic crystal fibres or holey fibres) using the mutipole method [1, 2], in a Microsoft Windows environment. The software has been designed to be easy to use and can be downloaded for free from the internet [3]. Usage of the CUDOS MOF Utilities is restricted to non-commercial research and teaching purposes, as detailed in the user-licence.

The package comprises two executables, `fibre.exe` (which we will refer to as `FIBRE`) and `winfield.exe` (`WINFIELD`).

1. `FIBRE` is a console based application for finding MOF modes and computing their dispersion curves, using simple macro-language based parameter files. It can deal with MOFs with circular inclusions in a circular matrix which may be surrounded by a jacket or a cladding and a jacket. Its use is described in section 2.
2. `WINFIELD` is a window based software for computing and visualising modal fields and derived quantities once the modes have been found with `FIBRE` . Its use is described in section 3.

2 Using FIBRE

2.1 Introduction

FIBRE is based on the multipole formulation for microstructured optical fibers described in [1, 2]. The algorithms used are also detailed in the above references, so that we will here concentrate on the practical use. Nevertheless, some parameters used in the files needed by FIBRE are intimately connected to the algorithms used, so that the references above might be necessary to completely understand their full meaning and impact. FIBRE makes an extensive use of the symmetry properties of the simulated structure. It can also deal with non-symmetrical structures. FIBRE supports automatic adjustment of parameters for finding the fundamental mode (and its dispersion properties) of a solid core MOF with hexagonal symmetry. However, for more complex structures the choice of the 'correct' parameters to find modes or compute their dispersion properties is not straightforward and one often has to change the parameters a few times before the wanted result is found.

2.2 Installing and uninstalling FIBRE

To install the CUDOS MOF Utilities, unzip `cudosmof.zip` to a temporary folder, and double click `Install.exe`. This will install the fibre software in a "CUDOS" subfolder of your "Program Files" folder, add a "My CUDOS MOF Files" folder in "My Documents", add links into the start menu and associate the Fourier Bessel Coefficient Files with the WinFIELD software.

The "My CUDOS MOF Files" folder will contain a `samples.zip` file containing several commented examples of parameter files you may want to have a look at to get an idea on the structure and keywords of parameter files.

To uninstall the software package, simply run the uninstall link in the CUDOS FIBRE folder of the Start menu, or alternatively use the windows Add/Remove Software tool in the control panel.

2.3 Getting started: a first simulation

In what follows we will assume that you have installed the FIBRE software package successfully according to the above paragraph, and that you have unzipped the `samples.zip` file.

In the windows Start menu, find the CUDOS MOF Utilities directory and click on Fibre Console Prompt: This opens a command line window with current directory set to your 'My CUDOS MOF Files' folder.

Change the directory to the directory containing the first sample parameter file by typing

```
cd sample\simple hexagon <Enter>
```

Now type in

```
fibre <Enter>
```

This runs the simulation, using the parameters described in the `parameter.txt` file of the current directory. After a few seconds a message will inform you that the simulation terminated successfully.

The simulation you just run consisted in finding the fundamental mode of a simple structure with a single ring of six holes in silica.

If you look at the “My CUDOS MOF Files\samples\simple hexagon” directory with *eg* Windows Explorer you will see that 12 new files were created through the simulation, among which two `.bcf` files, and two `.fbb` files. These contain the Fourier-Bessel coefficients of the modes found by the simulation. Right click on either of those files and select “high res open” in the contextual menu: The `WINFIELD` software opens the corresponding file, and you will now see the structure for which the fundamental mode was computed, along with the distribution of the axial component of the Poynting vector. At this stage you should play around with the different controls of `WINFIELD` to become familiar with the different settings.

2.3.1 The input and output files

Now let’s have a look at the other files in the folder: the first two files always have the same names, regardless of the simulation:

`parameters.txt`:

This text file, the parameter file, is the input file which was initially in the folder, and which was used by `FIBRE` to build the fibre structure, define the various parameters (*eg* wavelength) and run the simulation.

`errors.txt`:

This text file contains comments, warnings and errors which might have occurred while reading the parameter file, building the fibre structure or running the simulation. This file is useful when debugging a parameter file.

The following file will usually be called `progress.txt`, unless you explicitly rename it in the parameter file:

`progress.txt`:

This text file contains information on the progress of the simulation. It is useful when running a long simulation to check what the program is currently doing.

The remaining files will have different names at each simulation. They contain the results of the simulation.

`symhex.txt`:

This text file contains the definition of the fiber’s structure. In this example, the structure was compiled from basic fibre parameters (pitch, hole diameter, number of rings of holes, refractive indices etc.) contained in the parameter file, and the resulting structure was written to `symhex.txt`. Alternatively the structure to be simulated can also be defined directly in a structure file, in which case the structure file becomes an input rather than an output file.

`symhex_results.txt`

This text file contains an exhaustive list of all actual parameters used for the simulation. These include the parameters defined in the parameter file, but also the default parameters which don't need to be defined in the parameter file, or which have been automatically adjusted by the program. It also contains information on all modes found, and on various events occurring during a simulation. We will call this file the result file, it is an important file to keep track of the details of what was done during the simulation.

```
symhex_C03_det.bin
symhex_C03_det.log
```

These two files contain the initial determinant map computed by the program to find the modes. The `.log` file is a text file and can be read with a text editor or easily imported to third party plotting software. The `.bin` file is a binary file containing the determinant map in full precision and is used internally by the `FIBRE` software to avoid computing the determinant map twice when it can be avoided.

```
symhex_mode_table.txt
```

This text file summarizes information on all modes found during the simulation. Each line corresponds to one mode, and shows the mode's symmetry class, ordinal number, complex effective index, as well as the magnitude of the smallest and second smallest eigenvalues from the eigenvalue decomposition which has given the mode.

```
symhex_skipped_minima.txt
```

This file contains the values of the effective index at which a secondary minimum of the determinant map was found but not refined during the simulation. This file is often empty and is rarely used, we will see when and how it can be useful in a next version of this documentation.

```
symhex_L0000C03M001.fbb
symhex_L0000C04M001.fbb
symhex_L0000C03M001.bcf
symhex_L0000C04M001.bcf
```

These files contain the Fourier Bessel coefficients of each mode found. The `.fbb` files are binary files which can only be read by the `WINFIELD` software, whereas the `.bcf` files are text files. You can have a look at the latter to see the magnitude of all Fourier Bessel coefficient files. `WINFIELD` can read the `.bcf` files as well, but only if they are in the same folder as the original structure file, whereas `.fbb` files are stand-alone files, they contain the full set of parameters related to the mode, including the structure of the file.

2.3.2 A closer look at the parameter file

Double click on `parameters.txt` to open the parameter file with your default text editor. The first thing you will see are a few lines of comments, explaining the purpose of the file. In a parameter file, comments start with an exclamation mark character and end by a new line. After the comments, the parameter file defines the structure: the first line which doesn't start with an exclamation mark and which hence is not a comment, starts with

```
pitch=6.75d0
```

followed by an exclamation mark introducing further comments. The above example shows the format of a class of instructions used in a parameter file, the *definition statements*:

```
keyword=value
```

The keyword is separated from its arguments through an equal character. Depending on the keyword, *value* has to be real, complex, integer, logical or a character string. The list of keywords and acceptable values is given in section 2.5.

In the above example, the keyword `pitch` designates the center to center distance between the inclusions of the fibre. Note that the value, `6.75d0`, is real; the `d` between `6.75` and `0` is equivalent to the usual 'e', or 'times ten to the' in scientific notation. You may use 'e' or 'd' indifferently, or omit the exponent. Further, no unit of length is given, this is because of the scale invariance of the problem. However, when material dispersion is taken into account all dimensions are assumed to be expressed in micrometers, so the safest is to assume that all dimensions are always expressed in micrometers.

The next line shows an example of a keyword with a complex argument

```
cylinder index=(1.d0,0d0)
```

This statement assigns the complex value $1.0 + i \times 0$ to the keyword `cylinder index`, which sets the refractive index of all inclusions. As you can see, a complex value is expressed in the standard notation (*real part,imaginary part*). It is, however, not acceptable to use a real notation when a complex value is needed, even if the imaginary part is naught: the statement

```
cylinder index=1.d0
```

is *not* acceptable, it will not cause an error, but it will add a random imaginary part to the real value.

The following two lines are self-explanatory (for details see the keyword list), we skip our comments until the lines

```
Nr=1
MNr=1
```

These statements define the number of rings of inclusions the fiber should have around the core, and the size of the core, we will explain them in more detail later. However, note that they are examples of keyword taking integer arguments.

The next two lines,

```
no cladding
no jacket
```

are examples of definition keywords not taking any argument. They tell the program that the fibre should not be surrounded by a cladding or a jacket, so that the inclusions are in bulk material extending to infinity.

The next statement,

```
build fibre
```

is somewhat different from the ones we have seen so far: instead of giving the program information, it asks the program to use the previously defined parameters to build the fibre, *ie* to compile the inclusion sizes, refractive indices and distances into a complete structure. We will call such keywords *action keywords*. This statement ends the structure definition block.

The next block defines the remaining physical parameters. In the example we're considering the only remaining parameter is the wavelength, defined by the line

```
lambda=1.45
```

The next two statements are action keywords asking the program to adjust remaining parameters automatically:

```
suggest n_eff range=fundamental
suggest order=close cylinders
```

The first line asks the program to estimate the range of effective indices where the fundamental mode should be. The action keyword used takes an argument, separated by an equal sign (=), indicating which mode we are interested in. When FIBRE will look for modes, it will compute a map of the determinant over the estimated region and refine all minima found in that region to find the modes. The second line asks FIBRE to estimate the order of truncation of Fourier Bessel series according to wavelength, structure and the previously estimated range of effective indices. The action keyword used here takes an optional argument, here 'close cylinders', indicating that the inclusions are relatively close to each other.

The next line consists of an action keyword telling FIBRE to save the compiled structure into a file.

```
save fibre=symhex.txt
```

The file name is given as a character string argument, separated from the keyword by an equal sign. When compiling a fibre structure from geometric parameters indicated in the parameter file, one should always save the structure using 'save fibre'; failing to do so will cause FIBRE to save the structure in an arbitrary file named 'INTERNAL' without any file extension.

We now reach the line

```
search modes
```

This action keyword tells FIBRE to start the simulation, using all previously defined parameters.

Finally, the last line of the parameter file

```
end
```

marks the end of the parameter file, and causes the program to exit. Any lines after an end statement are ignored. Now that you have seen a basic example of parameter file, you might want to change the parameter file and run for example simulations for a different structure.

2.4 Syntax and structure of the parameter file

The above example should have given you a general impression on how parameter files are structured. In this section we will see the more general syntax and structure rules of a parameter file.

2.4.1 Syntax

General: No line should be longer than 128 characters. Any succession of white spaces and/or tabulation characters (ASCII code 9) is condensed into one single white space while interpreting the file. Leading and trailing blanks are removed, from keywords as well as from arguments, *ie* you may insert any number of spaces and/or tabulations between the beginning of the line, the keywords, the equal sign, the arguments and the comments. Blank lines are ignored. All input text files should be in Windows ANSI or DOS ASCII encoding, and should not contain any non-printable characters except for new line characters, tabulations and whitespaces.

Comments start with an exclamation mark and end at the end of the line. You may add comments after keywords and arguments on the same line. Any characters between an exclamation mark and an end of line character are ignored.

Keywords and arguments are separated by an equal sign (=) and optionally any whitespace and tabulations. A keyword and its argument must be on a same line. The parts of a keyword consisting of more than one word must be separated by at least a white space or a tabulation, or any combination of white spaces and tabulations. **Keywords and arguments are case sensitive.**

Real arguments are to be given in the common scientific notation, 'e', 'E', 'd' or 'D' marking the exponent. For example the arguments
1.5e-3, 1.5d-03, 0.15D-2, 0.0015
are equivalent.

Complex arguments are to be given in the form

(real part,imaginary part)

with *real part* and *imaginary part* following the syntax of real arguments. Assigning a real value (without brackets and imaginary part) to a keyword requiring a complex argument is not valid and assigns a random value to the imaginary part of the complex value.

Integer arguments are given as usual integers.

Logical arguments take the values `.true.` or `.false.` (as in Fortran).

Character string arguments may contain white spaces and tabulations, but any sequence of white space or tabulation will be replaced by a single white space. A string argument starts with the first non-blank character following the equal sign and ends with the last non-blank character preceding an exclamation mark or an end of line.

end: the `end` statement ends a parameter file: lines following the line of the `end` statement are ignored. If a parameter file ends before an `end` statement is encountered, execution stops and a warning message is written to the error file.

2.4.2 Structure

The general structure of a parameter file is as follows

1. Definition of a structure,
2. Definition of other physical parameters *eg* wavelength,
3. Definition of FIBRE specific parameters, *eg* parameters relating to the algorithm used by FIBRE , such as number of points in the determinant map, thresholds, etc. Most of these parameters have default values and don't need to be defined unless their default value is not suited to the specific simulation.
4. Definition of file names or file name suffixes,
5. Action keywords,
6. end statement.

A parameter file may contain different simulations: any of the steps 1 to 4 can appear after an action keyword, redefining one or more parameters, before a new action keyword. An example of a parameter file running two successive simulations with different parameters is given in "My CUDOS MOF Files\samples\dispersion 2\parameters.txt". However, when doing so, one should make sure to redefine the file name suffixes in order to avoid that one simulation overwrites results from the preceding simulation. (See also the 'overwrite' keyword).

The order of definition statements between action statements is - up to a few exceptions (see section 2.5) - not important. The order of appearance of action keywords in the parameter file does, however, matter, since action keywords use information previously defined by information statements. Further, some action keywords define or modify previously defined parameters. For example you may not ask to save a structure with the 'save fibre' action keyword before a structure has been defined, *eg* using the 'build fibre' action keyword.

If a same information keyword is used more than once with different arguments, the argument will take the value defined in the last occurrence of the information keyword before encountering the action keyword. For example, if a parameter file contains

```
lambda=3.0
lambda=2.0
search modes
[...]
lambda=1.0
search modes
```

the first simulation will use `lambda=2.0`, the second simulation will use `lambda=1.0`.

If an action keyword occurs before all parameters required for the execution of the action have been defined, the program will stop and all required parameters which remained undefined will be listed in the error file.

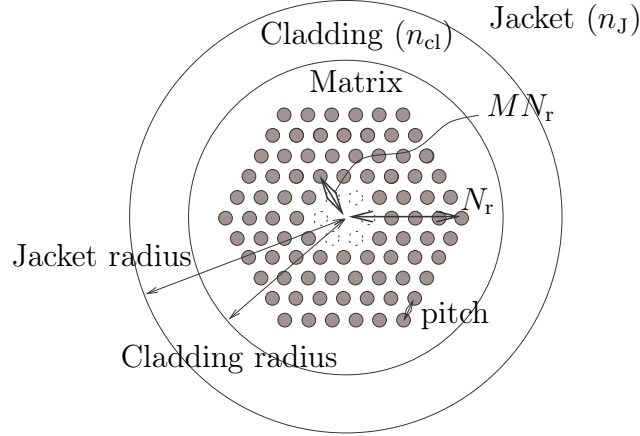


Figure 1: Example of a structure generated by the internal structure compiler. Here $MN_r = 2$ and $N_r = 5$.

2.5 Parameter file keyword reference

2.5.1 Miscellaneous keywords

- **end** *no arguments*: the **end** statement ends a parameter file: lines following the line of the **end** statement are ignored. If a parameter file ends before an **end** statement is encountered, execution stops and a warning message is written to the error file. (*Synonyms*: **END**)
- **verbose** = *.true.* [LOGICAL] Default value is *.false.*. If set to *.true.*, a lot more information will be written to the default output file. This proves useful when a mode is difficult to find. (*Synonyms*: **VERBOSE**, **Verbose**)
- **!** starts a comment. Comments end at the end of the line. You may add comments after keywords and arguments on the same line. Any characters between an exclamation mark and an end of line character are ignored.

2.5.2 Fibre structure compiler keywords

Definition keywords

- **central cylinder** = *radius* [REAL] *epsilon* [COMPLEX]: This keyword takes two arguments. It adds a central cylinder with radius *radius* and permittivity *epsilon* to the structure. (*Synonyms*: **add central cylinder**)
- **central cylinder radius** = *radius* [REAL]: Adds a cylinder with radius *radius* centered on the origin to the structure. If the permittivity or refractive index of this cylinder is not defined by **central cylinder epsilon** or **central cylinder index**, the central cylinder's refractive index will be the same as the refractive index of all other cylinders.

- **central cylinder epsilon** = ϵ [COMPLEX]: Sets the central cylinder's permittivity to ϵ . If the radius of the central cylinder is not defined elsewhere, no central cylinder will be added to the structure. (*Synonyms*: central cylinder permittivity)
- **central cylinder index** = n [COMPLEX]: Sets the central cylinder's refractive index to n . If the radius of the central cylinder is not defined elsewhere, no central cylinder will be added to the structure. (*Synonyms*: central cylinder refractive index)
- **pitch** = l [REAL]: Sets the pitch (center to center distance) between cylinders to l . (*Synonyms*: PITCH, Lambda)
- **Nr** = N_r [INTEGER]: Number of rings. Sets the distance from the origin to the center of the last hole on the x -axis to $N_r \times \text{pitch}$. See Fig. 1. (*Synonyms*: nr, number of rings, NUMBER OF RINGS)
- **MNr** = MN_r [INTEGER]: Number of missing rings. Sets the distance from the origin to the center of the first hole on the x -axis to $MN_r \times \text{pitch}$. See Fig. 1. (*Synonyms*: mnr, missing rings, MISSING RINGS, number of core rings)
- **cladding radius** = R_{Cl} [REAL]: Defines the cladding radius (see Fig.1). (*Synonyms*: cladding_radius, CLADDING_RADIUS, Cladding radius, Cladding_radius, Cladding Radius)
- **jacket radius** = R_J [REAL]: Defines the jacket radius (see Fig.1). (*Synonyms*: jacket_radius, JACKET_RADIUS, Jacket radius, Jacket_radius, Jacket Radius)
- **No Cladding** (*no arguments*): This keywords informs the internal structure compiler that the structure does not have a cladding. The cladding refractive index will be set to the refractive index of the matrix, and the cladding radius will be set to an arbitrary value between the furthest point of all cylinders and the jacket radius. Note that when the matrix and the cladding have same index, FIBRE automatically replaces the reflection matrix of the matrix/cladding interface by the null-matrix and the transmission matrix between the matrix and the cladding by the identity matrix. The No Cladding keyword supersedes definitions of cladding index and radius: if the No Cladding keyword has been used anywhere before a `build fibre` statement, the structure will have no cladding, regardless of whether the cladding's refractive index and radius have been defined elsewhere. (*Synonyms*: no_cladding, No_cladding, NO CLADDING, no cladding, NO_CLADDING)
- **cylinder radius** = R [REAL] Sets the radius of all cylinders to R . Note that the radius of the central cylinder can be defined separately before or after this keyword using `central cylinder radius` or `central cylinder`. (*Synonyms*: cylinder_radius, CYLINDER_RADIUS, Cylinder radius, Cylinder_radius, Cylinder Radius)
- **diameter on pitch ratio** = q [REAL] Sets the radius of all cylinders to $0.5 * q \times \text{pitch}$. Note that the radius of the central cylinder can be defined separately before or after this keyword using `central cylinder radius` or `central cylinder`. (*Synonyms*: d/Lambda, d/pitch, diameter/pitch, diameter on pitch, diameter_on_pitch)

- **No Jacket** (*no arguments*): This keyword informs the internal structure compiler that the structure does not have a jacket. This keyword is only valid in combination with the **No Cladding** keyword, which must be stated before the occurrence of **No Jacket**. The jacket refractive index will be set to the refractive index of the cladding (and hence of the matrix), and the cladding and jacket radii will be set to arbitrary values so that the interfaces are beyond the cylinder furthest from the origin. Note that when the cladding and the jacket have same refractive index, **FIBRE** automatically replaces the reflection matrix of the cladding/jacket interface by the null-matrix and the transmission matrix of that interface by the identity matrix. The **No Jacket** keyword supersedes definitions of jacket index and radius: if the **No Jacket** keyword has been used anywhere before a **build fibre** statement, the structure will have no jacket, regardless of whether the cladding's refractive index and radius have been defined elsewhere. Using **No Jacket** before **No Cladding** will cause an error message in the error file and **FIBRE** will stop. (*Synonyms*: no_jacket, No_jacket, NO_JACKET, no jacket, NO_JACKET)
- **jacket index = n_J** [COMPLEX]: Sets the jacket's refractive index to n_J . (*Synonyms*: nj, nJ, n_J, n_j)
- **cladding index = n_{Cl}** [COMPLEX]: Sets the cladding's refractive index to n_{Cl} . (*Synonyms*: nc, nC, n_C, n_c)
- **cylinder index = n_i** [COMPLEX]: Sets the refractive index of all cylinders to n_i . Note that the refractive index of the central cylinder can be defined separately before or after this keyword using **central cylinder index**, **central cylinder epsilon** or **central cylinder**. (*Synonyms*: ni, nI, n_I, n_i)
- **matrix index = n_M** [COMPLEX]: Sets the refractive index of the matrix to n_M . (*Synonyms*: nm, nM, n_M, n_m)

Note that for keywords defining a refractive index or a permittivity, you may also use the character strings 'silica' (or any other text string including a capital or lowercase 's') to use the Sellmeier expansion of silica to define the refractive index.

Action keywords :

- **save structure = structure file name** [CHARACTER STRING]: Writes the current structure to *structure file name*. The part of *structure file name* before the last dot will be used as the radix on which all output file names are build, unless a later use of the **file name radix** redefines this radix. If no structure is loaded in memory (be it through **build fibre**, **load pos file** or **load bcf file**) a warning is written to the error file and the statement is ignored. (*Synonyms*: write structure, write structure file, save fibre, save fiber)
- **Build Fibre** *no arguments*: This keyword runs the internal structure compiler. All current structure parameters are compiled into a structure (definition, position, radius, refractive index of all cylinders of the structure, of the cladding and the jacket). Before running any simulation, a structure must be defined, either through the internal structure compiler or through loading a position file (with **load position file** or **load bcf file**) (*Synonyms*: Build Fiber, Build Structure, build fibre, build fiber,

build structure, make fibre, Make Fibre, make fiber, Make Fiber, BUILD FIBER, BUILD FIBRE, BUILD STRUCTURE, compile fibre, compile fiber)

- **load structure file** = *structure file name* [CHARACTER STRING]: Defines a structure by loading the structure file *structure file name*. See section 2.6 for the syntax of structure files. (*Synonyms*: structure file, structure, STRUCTURE FILE, structure_file, STRUCTURE_FILE, load fibre, load structure)

Other keywords affecting the fibre structure:

- **load bcf file** = *bessel coefficient file name* [CHARACTER STRING] See section 2.5.5 (*Synonyms*: load mode)

2.5.3 Threshold keywords

All keywords in this list are information keywords, and define thresholds used by the mode finding algorithm. All threshold have default values and are hence optional. A good understanding of these threshold requires a reasonable understanding of the multipole method and the root-searching algorithm described in Refs. [1, 2].

- **det_mode_threshold** = *determinant mode threshold* [REAL]: Defines the threshold below which the determinant must be before an eigenvalue decomposition is computed. The eigenvalue decomposition is numerically more expensive than the computation of the determinant, so the eigenvalue decomposition should only be done when there is a good chance to find a small eigenvalue. The best value of this parameter depends on the wavelength to pitch ratio, the size of the matrix, *ie.* on the number of cylinders, on the Bessel expansion truncation parameters and also (but this is negligible) on the class of symmetry. For a structure with one or two cylinders in the irreducible sector and with the Bessel expansion going from -5 to 5, 1e-10 is a reasonable value. For more complex structures or higher orders in the Bessel expansion, it might be necessary to go up to 1e-6, and in some cases (bandgap guidance near a band gap edge or extreme values of the wavelength to pitch ratio) values up to 1e10 are needed.

The default value of this threshold is set to 1e-10. However, when a minimum of the determinant is refined by the Broyden/zooming algorithm until the precision thresholds set for the real and imaginary parts are reached and the value of the determinant, although being a local minimum, is not below *determinant mode threshold*, FIBRE computes an eigenvalue decomposition for this minimum anyway. If an eigenvalue with magnitude less than *eigen-value threshold* (see below) is found, FIBRE automatically adjusts the determinant mode threshold to 100× the magnitude of the determinant for which the mode was found. This avoids time to be spent in excessive refinement for the next modes to be searched. It is therefore rare to have to adjust this threshold, since FIBRE adjusts it on its own. If you want to avoid FIBRE to adjust this threshold automatically, you can use the optimization=paranoid option. (*Synonyms*: det mode threshold, determinant threshold)

- **eigen_value_threshold** = *eigen-value threshold* [REAL]: This sets the threshold below which the modulus of an eigenvalue is considered to be zero, *ie* the value the

modulus of an eigenvalue has to reach before the associated eigenvector is considered to be a mode. The default value of this threshold is $1e-10$, and is generally sufficient for most simulations. For more complex structures (large number of rings or large diameter on pitch ratios) you might have to increase the threshold up to $1e-8$ or $1e-6$. You will know when you have to change this threshold from looking at the result file. If for a given minimum the result files mention that refining the root failed, but the smallest eigenvalue has a relatively small magnitude compared to the second smallest eigenvalue, you should re-run the same simulation using an eigenvalue threshold slightly above the magnitude of the smallest eigenvalue found. Note that if you rerun the same simulation just changing the eigen-value threshold, the determinant map will not have to be re-evaluated, but will be loaded from the determinant file, so that the second simulation should be much faster than the first. If you need to raise this threshold, you will have to check more thoroughly that the modes you get were sufficiently refined, through checking *eg* the Wijngaard test. (*Synonyms*: eigenvalue threshold, eigen-value threshold)

- **real_precision_threshold** = *real precision threshold* [REAL]: Defines the limit precision for the real part of the effective index. The software stops refining the current minimum of the determinant if the *absolute* precision on the real part of n_{eff} is less than **real_precision_threshold** and no mode has been found. The default value of this threshold is $1e-13$. Lowering this threshold does not give higher precision on the real part (the main criterion for stopping the refinement of a mode is the magnitude of the eigenvalue), but may cause the refinement to slow down if the determinant mode threshold is inadequate. On the contrary, increasing this threshold may lead to inaccurate results if the eigenvalue threshold is inadequate. It is rare one has to change this value and it is recommended not to change it. (*Synonyms*: real precision threshold, real precision)
- **imag_precision_threshold** = *imaginary precision threshold* [REAL]: Defines the limit precision for the imaginary part of the effective index. The software stops refining the current minimum of the determinant if the *relative* precision on the imaginary part of n_{eff} is less than **imag_precision_threshold** and no mode has been found. The default value of this threshold is $1e-6$. Lowering this threshold does not give higher precision on the imaginary part (the main criterion for stopping the refinement of a mode is the magnitude of the eigenvalue), but may cause the refinement to slow down if the determinant mode threshold is inadequate. It is rare one has to change this value and it is recommended not to change it. (*Synonyms*: imag precision threshold, imaginary precision threshold, imag precision, imaginary precision)
- **degeneracy_threshold** = *degeneracy threshold* [REAL]: When symmetries are taken into account, classes of symmetry are treated separately and eigenvalues are (except for the rare cases of accidental degeneracies) non-degenerate. The software checks that the ratio of the modulus of the two smallest eigenvalues is less than this threshold. If this is not the case, the software does not consider that a mode has been found and continues the quest. When symmetries are not taken into account, two-fold degeneracies occur. The threshold is then used to correctly deal with degenerate modes: if after an eigenvalue decomposition, more than one eigenvalues have magnitude smaller than the eigenvalue threshold and their ratio is smaller than the condition

threshold, they are considered degenerate modes. The modes then get the same ordinal number and are distinguished by the degeneracy number (the file names of the Fourier-Bessel coefficient files then end with d1, d2 etc., corresponding to the degeneracy number). The default value for this parameter is 1e-3, and cases where this value is inadequate have never occurred to the present day. (*Synonyms*: condition threshold, condition_threshold)

- **minima_threshold** = *minima threshold* [REAL]: Sets the value below which a local minimum of the initial determinant map will be refined. The default value of this parameter is 1d30, which in practice means that all minima will be refined. In some very rare cases numerical instabilities appear associated with a diverging determinant (this can be seen by visualizing the determinant map: the determinant then takes very large values [of the order of 1d30 or more] and seems to fluctuate chaotically in that region), leading to false minima with large values of the determinant. In these cases it can save computational time to lower the minima threshold down to about 1e3. (*Synonyms*: minima threshold, minimum threshold)
- **upper_real_threshold** = *upper real threshold* [REAL]: This threshold limits the acceptable values of the real part of the effective index for the zooming algorithm to values such that $|\Re(n_{eff}) - n_{max}| > \text{upper real threshold}$, with n_{max} being the largest value of the real part of the refractive indices encountered in the structure. This is to avoid numerical errors occurring when the effective index is too close to n_{max} , and to avoid the zooming algorithm to enter an infinite loop. The default value of this threshold is 1e-4. You may want to decrease this value if you are looking for modes close to n_{max} , but avoid using values smaller than 1e-6 since this can lead the zooming algorithm to loose an unacceptable amount of time trying to follow a valley crossing n_{max} . (*Synonyms*: upper real threshold)
- **lower_real_threshold** = *lower real threshold* [REAL]: This threshold limits the acceptable values of the real part of the effective index for the zooming algorithm to values such that $|\Re(n_{eff}) - n_{min}| > \text{lower real threshold}$, with n_{min} being the smallest value of the real part of the refractive indices encountered in the structure. This is to avoid numerical errors occurring when the effective index is too close to n_{min} , and to avoid the zooming algorithm to enter an infinite loop. The default value of this threshold is 1e-4. You may want to decrease this value if you are looking for modes close to n_{min} , but avoid using values smaller than 1e-6 since this can lead the zooming algorithm to loose an unacceptable amount of time trying to follow a valley crossing n_{min} . (*Synonyms*: lower real threshold)
- **lower_imag_threshold** = *lower imaginary threshold* [REAL]: This threshold limits the smallest acceptable value of the imaginary part of the effective index for the zooming algorithm. This is to avoid numerical errors occurring when the effective index is too close to the real axis, and to avoid the zooming algorithm to enter an infinite loop. The default value of this threshold is 1e-15. It is rare to have to change this threshold, but if you observe numeric instabilities or divergences in the determinant map close to the real axis (this might occur for lossy structures) you might want to increase this threshold. On the other hand, if you know that the imaginary part of the effective index is going to be smaller than 1e-15, you should set the imaginary part to zero artificially

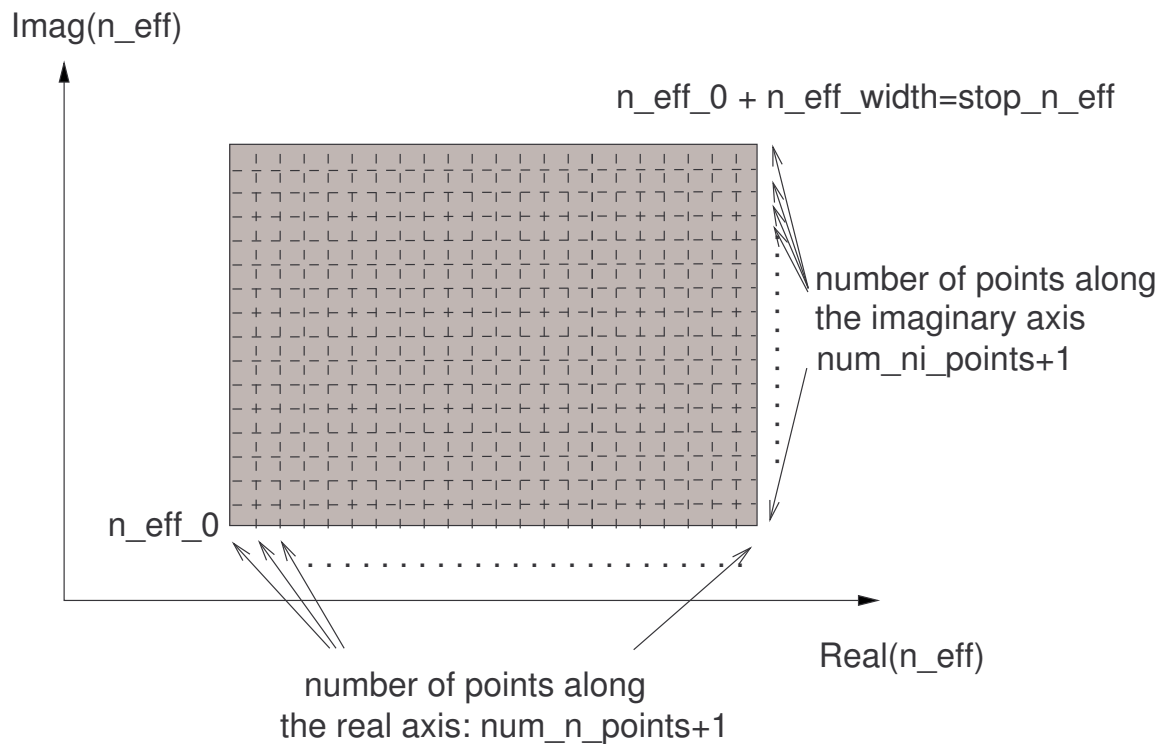


Figure 2: Illustration of parameters defining the n_{eff} region in complex plane in which modes are sought.

and search for modes on the real axis only. (See `stop_n_eff` and `n_width` on how to do this.) (*Synonyms*: lower imag threshold, lower imaginary threshold)

- `upper_imag_threshold` = *upper imaginary threshold* [REAL]: This threshold limits the largest acceptable value of the imaginary part of the effective index. This is to avoid numerical errors (especially divergences) occurring when the imaginary part of the effective index is too large. The default value of this threshold is 1e-1. It is extremely rare to have to change this threshold. (*Synonyms*: upper imag threshold, upper imaginary threshold)

2.5.4 Keywords for mode seeking

- `wavelength` = λ [REAL]: Wavelength at which the simulation should be run. When material dispersion is not taken into account, all lengths are relative and the wavelength is relative to the lengths defined by the structure. When material dispersion is taken into account, all lengths, including the wavelength, are assumed to be expressed in micrometers.

Other keywords affecting this value: `start_lambda`.

(*Synonyms*: WAVELENGTH, wl, WL, lambda)

- **n_eff_0** = n_{eff0} [COMPLEX]: Defines the lower left corner of the region of effective index (in the complex plane) in which modes will be searched. By lower left corner we mean the corner of the rectangle of the complex plane with smallest real part and smallest imaginary part (see Fig. 2). (*Synonyms*: start_neff, start_index, lower left n_eff corner)
- **stop_n_eff** = $stop_n_eff$ [COMPLEX]: Defines the upper right corner of the region of effective index (in the complex plane) in which modes will be searched. By upper right corner we mean the corner of the rectangle of the complex plane with largest real part and largest imaginary part (see Fig. 2). *Important: start_n_eff has to be used before stop_n_eff is used.* (*Synonyms*: upper right n_eff corner, final n_eff, stop n_eff, stop neff)
- **n_eff_width** = n_eff_width [COMPLEX]: Defines the region of effective indices in the complex plane in which to search modes by its width: The scanning region will go from n_eff_0 to $n_eff_0 + n_eff_width$ (see Fig. 2). (*Synonyms*: N_eff_width, N-EFF-WIDTH, n_width, N-WIDTH)

IMPORTANT NOTE: *the real part of stop_n_eff and start_n_eff should always differ from values taken by the structure's refractive indices, or severe numerical errors could stop the program's execution. Indeed if n_{eff} is exactly equal to the refractive index of the matrix (or inclusions etc.), the value zero will appear as the argument of Hankel functions, which diverge in zero. Example: if you're interested in finding the fundamental mode of a solid core MOF with matrix refractive index (1.45,0), and you expect the fundamental mode to have an effective index with real part close to 1.45, do not set the real part of stop_n_eff to 1.45, but to eg 1.449999.*

- **num_n_points** = $number_n_points$ [INTEGER]: Defines the number of points parallel to the real axis on which the determinant is computed for the initial determinant map. The determinant will be computed for num_n_points+1 points on each line parallel to the real axis (see Fig. 2). This parameter must be greater or equal 4. Its default value is 100. The ideal value of this parameter depends on the expected density of modes and the width of the scanning region. The density of modes in the n_{eff} space depends on the complexity of the structure and the wavelength (it increases with decreasing wavelength). If one approximately knows where a mode is (for example if a similar structure with a smaller number of rings has been studied before) a value of 20 or even 10 for num_n_points can be enough. For a large scan or for complex structures values of 100 or 200 are more appropriate. When looking for a bandgap guided mode, values of 500 for a range of effective indices of 1e-2 are not excessive. If this parameter is too small, some modes might not be found, if it is too large, computing the map will be unnecessarily slow.
- **num_ni_points** = num_ni_points [INTEGER] Defines the number of points parallel to the imaginary axis on which the determinant is computed for the initial determinant map (see Fig. 2). The map will be computed for num_ni_points+1 points for each value of the real part of n_{eff} . This parameter must be either 0 (if $\Im(n_{width}) = 0$) or greater or equal to 4. Its default value is 4 if $\Im(n_{width}) \neq 0$ and 0 otherwise. As the scan along the imaginary axis is exponential, a value of 4 (8 for some bandgap guided modes) is

generally sufficient. If the structure has a very good confinement (ie. if the imaginary part of the effective index of the computed modes is expected to be less than 10^{-13}), setting this parameter to zero is not only useful but necessary.

- `start_mode = first mode` [INTEGER]: Sets the first class of symmetry [4] for which modes are searched. When modes belonging to a twofold degenerate symmetry class are sought, at least the first of the two degenerate symmetry classes must be included in the range of symmetry classes: If, say, classes 3 and 4 are degenerate, `start_mode` can not be set to 4, but must be set to 3 to find the modes belonging to symmetry classes 3 (and 4). (*Synonyms*: `first mode`, `start mode`)
- `stop_mode = last mode` [INTEGER]: Sets the last class of symmetry [4] for which modes are searched. `FIBRE` will search for modes in all classes of symmetry between `start_mode` and `stop_modes`. (*Synonyms*: `last mode`, `stop mode`)

Degenerate modes are computed simultaneously, so that it does not make a difference on computational time to ask for only one or both of two degenerate mode classes. A determinant map is computed for each non-degenerate mode class or pair of degenerate mode classes: computing the determinant map is the slow part of the code, so restricting the number of symmetry classes is a good way of saving time when a specific mode (*eg* the fundamental mode) is sought. When no symmetries are used, the only valid class of symmetry is 1, so that `start_mode` and `stop_mode` must be set to 1.

Action keywords:

- `suggest n_eff range = mode` [CHARACTER STRING]: This keyword adjusts the range of effective indices to find a specific mode specified as the argument. This has been implemented only for solid core MOFs with C_{6v} symmetry, no jacket or cladding, matrix refractive index close to the refractive index of silica and low index inclusions with refractive index around 1. The range of classes of symmetry is adjusted to the class of symmetry of the considered mode. The argument should be one of the following values:
 - `fundamental` Adjusts the range of effective indices and the class of symmetry to find the fundamental mode of the structure. (*Synonyms*: `Fundamental`, `FUNDAMENTAL`, `FUND`, `fund`, `fundamental mode`, 1)
 - `second` Adjusts the range of effective indices to find the second mode of the structure. The automatic adjustment of the range of effective indices for the second mode is experimental and should not be relied upon. (*Synonyms*: 2, `second mode`)

Before `suggest n_eff range` is used, a structure (see Sec. 2.5.2) must have been defined.

- `search modes` (no arguments) Runs the simulation to find the modes. Before this keyword is used, a structure (see Sec. 2.5.2), the order of truncature of Fourier-Bessel expansion (see `order`, `gorder` and `suggest order`, Sec. 2.5.7), the wavelength (see `wavelength`, above), a range of effective indices (see `n_eff_0`, `stop_n_eff`, `n_eff_width` and `suggest n_eff range`) and the classes of symmetry to search for (see `start_mode`,

`stop_mode` and `suggest n_eff range`) must have been defined. (*Synonyms*: `modesearch`, `SEARCH MODES`, `MODESEARCH`, `mode search`)

2.5.5 Keywords for computing dispersion curves

- **number of points = steps** [INTEGER]: Defines the number of wavelengths *steps* for which to compute the mode between the first and last wavelengths. Note that when `optimization` is set to `fast`, `FIBRE` only minimizes the determinant for each wavelength step, without computing the whole eigenvalue decomposition. The entire eigenvalue decomposition is then only computed when the modes are logged to a file (see `number of points between log` keyword). The algorithm to compute dispersion curves uses quadratic interpolation between the given and previously computed points or linear extrapolation of the previously computed points to estimate the effective index for the next wavelength step. The estimated effective index should be sufficiently close to the actual value of the effective index for the Broyden part of the root finding algorithm to converge, without the help of the much slower zooming algorithm. If you see (through a comment in the error file or through the progress output file) that the zooming algorithm is used while computing the dispersion, increasing (say doubling) the number of wavelength steps will most certainly *decrease* the computation time. Values for this parameter generally range between 50 (for small wavelength intervals or simple structures) to several hundreds. Note that the maximum value for this parameter is 4000. If several dispersion simulations are to be done within the same parameter file (unless they are separated by the `delete n_eff table` keyword), the sum of all number of steps, added to the number of wavelengths used to initialize the interpolation of the effective index (through `load dispersion file`, `load bcf file`, or `n_eff(lambda)` statements) must remain under 4000. (*Synonyms*: `steps`, `number_of_points`, `numpoints`)
- **number of points between log = logsteps** [INTEGER]: Defines the number of wavelength steps between two mode logs. Every *logsteps*, the mode will be saved to a file. When `optimization` is set to `fast`, this also defines the number of wavelength steps between two complete eigenvalue decompositions. It is generally not necessary to have the complete set of Fourier-Bessel coefficients for every single wavelength step, but it is good to be able to check that the mode tracked with varying wavelength remains the right one (to avoid the consequences of mode crossing for example). (*Synonyms*: `number of log points`, `numlogpoints`, `number of points between logs`, `number_of_points_between_logs`)
- **start n_width = start n_width** [COMPLEX]: When starting the computation of a dispersion curve with only one (or zero) data points (*ie* the effective index of the mode for one wavelength as defined through a single `load bcf file` or `n_eff(lambda)` instruction, or when starting a dispersion computation by defining a range of effective indices where the mode for the first wavelength is to be found), defines the region in the complex plane in which to look for the mode at the second initial wavelength step. The width of the region is defined in an absolute way for the real part and in a relative way for the imaginary part: the region in which the mode for the second initial wavelength is computed is centered on the effective index for the first wavelength and is of width (*real part of start n_width*, [*imaginary part of start n_width*]×[*imaginary part of the effective*

index for the first wavelength)). If the resulting range of effective indices has negative imaginary parts, it is readjusted to include positive values of the imaginary part only, but remains of the same width.

Example: You want to compute the dispersion curve for a given mode, which you have found for a wavelength of say 700nm. The mode has an effective index of (1.45,1e-7) at that wavelength. You initialize the dispersion computation by using the `n_eff(lambda)` statement:

```
n_eff(lambda)=0.7 (1.45,1e-7)
```

You then define the width of the region in which you expect the effective index of the mode to be at the next wavelength:

```
start n_width=(1e-2,1000)
```

The region in which the mode for the second wavelength will be searched will be centered on (1.45,1e-7), and have a width of (1e-2,1000*1e-7)=(1e-2,1e-4). The region would hence cover complex values in the region inside the rectangle defined by the two opposite corners $(1.45 - (1e-2)/2, 1e-7 - (1e-4)/2)$ and $(1.45 + (1e-2)/2, 1e-7 + (1e-4)/2)$, which includes negative values of the imaginary part. The region is hence rectified to the rectangle between $(1.45 - (1e-2)/2, 0)$ and $(1.45 + (1e-2)/2, 1e-4)$, which has the same width as the previous rectangle.

Unless the effective index has a large imaginary part (say 1e-2 or higher) it is best to set a large imaginary part for `start n_width` (10 to 1e5).

The default value of this parameter is (1e-2,10).

(*Synonyms*: `start nwidth`, `default nwidth`, `default n_width`, `start_width`)

- `start_lambda = λ_0` [REAL] Sets the initial wavelength from which FIBRE will compute the dispersion curve. If a single point of the dispersion curve has been given as a starting point, λ_0 should be the wavelength of that point.

Note when using dispersion and mode searching in a same parameter file: this keyword also sets the current wavelength (see `wavelength`).

Other keywords affecting this value: `load dispersion file`. (*Synonyms*: `start lambda`, `first wavelength`, `first lambda`, `start wavelength`)

- `stop_lambda = λ_f` [REAL]: Sets the final wavelength of the wavelength range in which the dispersion has to be computed. Note that λ_f can be less or greater than λ_0 .

(*Synonyms*: `stop lambda`, `last wavelength`, `last lambda`, `stop wavelength`)

- `n_eff(lambda) = λ` [REAL] `n_eff` [COMPLEX]: This keyword takes two arguments: a real number indicating a wavelength and a complex number being the effective index at that wavelength. The keyword adds an entry to the dispersion table used to extrapolate or interpolate values of the effective index as a function of wavelength. Other keywords used to add entries to the table of effective indices are `load bcf file` and `load dispersion file` (see below). Note that limitations on the number of entries in the table of effective indices apply (see `number of points`).

(*Synonyms*: `n_eff_table`, `effective index`)

Action keywords:

- **delete n_eff table** (no arguments): Deletes and resets the table of effective indices. This keyword is useful when computing dispersion curves of different structures or different modes in a same parameter file. Unless this keyword is used, all previously entered or computed values of n_{eff} as a function of wavelength are taken into account to interpolate or extrapolate n_{eff} when computing a dispersion curve. (*Synonyms*: erase n_eff table)
- **load dispersion file** = *dispersion file name* [CHARACTER STRING]: Loads the file *dispersion file name* and adds its data to the table of effective indices. *dispersion file name* has to be a text file with three columns of real numbers. The first column indicates a wavelength and the two following columns represent the real and imaginary part of the effective index of the sought mode at the indicated wavelength. Note that dispersion output files have the correct format to be used as dispersion input files. After loading *dispersion file name*, `start_lambda` is set to the wavelength of the last line of the dispersion file. Note that limitations on the number of entries in the table of effective indices apply (see number of points).
- **load bcf file** = *bcf file name* [CHARACTER STRING]: Loads the Fourier Bessel text coefficient file *bcf file name*, loads the associated structure, and adds the wavelength and refractive index of the mode to the table of refractive indices. Note that the file must be a text file (ending with .bcf), not a binary file (.fbb are not valid). (*Synonyms*: load mode)
- **get dispersion curve** (no arguments): Runs the simulation to find the effective index of a mode as a function of wavelength. What FIBRE does depends on the content of the refractive index table (see :n_eff(lambda), load dispersion file, load bcf file)
 - If the table contains two or more entries, FIBRE directly enters the algorithm for tracking a mode with varying wavelength, using the table of refractive indices to extrapolate or interpolate first guesses of the refractive index for each new wavelength step. The table of refractive indices is completed after each new wavelength step.
 - If the table contains exactly one entry, FIBRE tries to find the mode for a wavelength close¹ to the only wavelength of the table of refractive indices (the latter should be equal to `start_lambda`), in a wavelength range defined as explained for the `start_width` keyword. If one mode is found, its effective index and wavelength are added to the table of effective indices and the algorithm for tracking a mode with varying wavelength is entered. If more than one mode is found, a warning message is written to the error file, the mode with largest real part of the effective index is added to the table of effective indices and the algorithm for tracking a mode with varying wavelength is entered. If no mode is found, execution is aborted.
 - If the table is empty, and the data needed to initiate a search for modes is defined, FIBRE search for the modes as explained for the `search modes` keyword, then

¹The wavelength for this initial step is defined by $start_lambda + 0.01 \times (stop_lambda - start_lambda) / number_of_points$

continues as above. If more than one wavelength is found during the initial search for modes, the mode with largest real part of the effective index is used for the further steps.

(*Synonyms*: compute dispersion curve, dispersion, lambda track, wavelength track)

2.5.6 File name keywords

- **verbose file** = *file name* [CHARACTER STRING]: Redefines the progress file name to *file name*. By default, the progress file is called *progress.txt*. Its content depends on the value of **verbose**. Each time this keyword is used, subsequent verbose output is directed to *file name*. You can redefine the progress file name at any time in a parameter file. You can hence have different progress files with different names for different simulations in a same parameter file. (*Synonyms*: progress file)
- **file suffix** = *suffix* [CHARACTER STRING]: The characters defined by this keyword are added to all output file names (except the error and progress files). A typical file name for an output file is *<file name radix><suffix><file type, mode and wavelength dependent suffix><file dependent extension>*. The default value for this parameter is the empty string. (*Synonyms*: file extension, file_extension, pers_name, file_name_extension, file name extension)
- **file name** = *file name radix* [CHARACTER STRING]: Defines the file name radix on which all output file names (except the error file and the verbose file) are based, as described for **file suffix**. By default the file name radix is the name of the last structure file (as saved or as loaded) without the .txt extension. If no structure file name has been defined, the default radix is INTERNAL. (*Synonyms*: file name radix, output file name radix)
- **overwrite** = *overwrite* [LOGICAL]: FIBRE automatically generates a number of output files with various names. By default, if a file with the same name as an output file exists prior to execution of FIBRE, it will be deleted and overwritten. This can lead to loss of data if one omits to change file name extensions or names between two simulations. If this flag is set to .false., the program will stop before overwriting any files. If set to .true. (default value), overwriting is allowed and will solely cause a warning in the error file. Note that the 'errors.txt', determinant map and temporary files are not affected by this option and will always be overwritten.

2.5.7 Keywords relating to the algorithm

- **optimization** = *option* [CHARACTER STRING], where *option* is one of the following:
 - **paranoid** Optimizes execution for maximum accuracy, with no compromises or self adjustment of parameters. This does not necessarily lead to safer execution, or more accurate results. This mode, intended for debugging and for modes which are extremely delicate to find, is generally not recommended. In this mode the determinant mode threshold is not changed from its initial value by the program, and when computing dispersion curves an eigenvalue decomposition is computed for each wavelength step.

- **normal** Normal execution mode. The eigenvalue threshold is updated automatically, and when computing dispersion curves an eigenvalue decomposition is computed for each wavelength step. This is the default value.
- **fast** Optimizes execution for speed: The eigenvalue threshold is updated automatically, and when computing dispersion curves an eigenvalue decomposition is only computed when a mode is saved to a file (see `number of points between log.`) This option is not supported for the search of modes (`search modes keyword`).

Note that the optimization option also affects the default optional argument of `suggest order`.

- **skip real borders** *val* [LOGICAL]: *val* is either `.true.` (default) or `.false.` . When true, minima on the edges parallel to the imaginary part of the effective index region over which the initial determinant map is computed are ignored. When false, all minima, including those on the latter edges, are refined. Minima on the edges of the determinant map are most likely not to be true local minima but points in valleys with minima far outside the region of effective index of interest. It is recommended to keep this parameter to `.true.`: when set to `.false.` a lot of time can be lost trying to track down the minimum of the mentioned valleys. (*Synonyms*: `skip_real_borders`)
- **order = order** [INTEGER]: Sets the order of truncature of the Fourier-Bessel expansions for all cylinders.
(*Synonyms*: `ORDER`, `Order`)
- **cladding order = order** [INTEGER]: Sets the order of truncature of the Fourier-Bessel expansions for the cladding and the jacket. (*Synonyms*: `gorder`, `GORDER`, `Gorder`, `Cladding Order`, `CLADDING ORDER`, `corder`, `Corder`, `CORDER`)

Action keywords:

- **suggest order = option** [CHARACTER STRING]: Asks FIBRE to evaluate automatically the order of truncature needed for the cylinders and for the jacket and cladding. In other words, using this keyword sets the parameters `order` and `cladding order` automatically. The choice of the order of truncature is based on [2] and our further experience. An important quantity for the choice of the order of truncature is the largest argument of Bessel and Hankel functions occurring during the simulation. The latter is proportional to the difference between effective indices of the modes and refractive indices of the structure, to the wavenumber in vacuum $2\pi/\lambda$ and to the longest distance occurring during the simulation. The longest distance to occur during the simulation is basically the longest distance between any two different centers of cylinders, however, this leads almost always to overestimates of the order of truncature. FIBRE lets the user choose which distance to take into account to evaluate the order of truncature, through the choice of *option* which is one of the following. Here we introduce λ , the wavelength as defined *prior to* the occurrence of the `suggest order` keyword, and Δn_{\max} , the maximum magnitude of the difference between all refractive indices in the structure and the effective index anywhere in the effective index range to be scanned:

- **fast** : (default value when optimization is set to *fast*) optimizes the choice of the order of truncature for speed: results are generally inaccurate, but can give a good idea of where to look for what mode in subsequent more precise simulations. With this setting, *order* is defined by $2\pi/\lambda R_{\max}\Delta n_{\max}$ where R_{\max} is the maximum radius of all cylinders. (*Synonyms*: FAST)
- **normal** : This is the default setting if optimization is set to *normal*. The resulting choice of the order of truncature, based on the radius of cylinders, gives good accuracy if the cylinders are not too close to each other. With this setting, *order* is defined by $1.5 \times 2\pi/\lambda R_{\max}\Delta n_{\max} + 1$ where R_{\max} is the maximum radius of all cylinders. (*Synonyms*: NORMAL)
- **close cylinders** : Same as normal, but adds 2 to the value of *order*. This is a safe choice if the cylinders are close to each other (*eg* diameter on pitch ratios larger than say 0.5 for C_{6V} structures).
- **second nearest neighbour** : Uses twice the minimal distance between cylinders to evaluate *order*. With this setting, $order = 1.5 \times 2\pi/\lambda 2 \times d_{\min}\Delta n_{\max} + 1$ where d_{\min} is the shortest distance between the centers of any two cylinders. (*Synonyms*: second neighbour)
- **core** : Uses the distance r_{core} between the origin and the center of the closest cylinder, or if a cylinder is centered on the origin, the radius of that central cylinder. With this setting, $order = 1.5 \times 2\pi/\lambda 2 \times r_{\text{core}}\Delta n_{\max} + 1$.
- **average** : Uses the average distance $\langle d \rangle$ between cylinders. With this setting, $order = 1.5 \times 2\pi/\lambda 2 \times \langle d \rangle \Delta n_{\max} + 1$.
- **rigorous** : Uses the actual longest distance d_{\max} occurring between any two interfaces (including the jacket) or center of cylinders. With this setting, $order = 1.5 \times 2\pi/\lambda 2 \times d_{\max}\Delta n_{\max} + 1$.
- **precise** : (default value when optimization is set to *paranoid*) Same as rigorous, but adds 3 to the obtained value of the order of truncature. (*Synonyms*: precision, PRECISE, PRECISION, accurate, accuracy, paranoid, PARANOID)

For cladding order, the distance d used is either the jacket radius (if the refractive indices of the cladding and the jacket are different) or the cladding inner radius (if the refractive indices of the jacket and cladding are the same but are different from the refractive index of the matrix), the only difference between the different options is the corrective factor. The order of truncature of the Fourier-Bessel expansion for the cladding and the jacket is

- *fast* option: $2\pi/\lambda\Delta n_{\max}d + 1$
- *paranoid* option: $1.5 \times 2\pi/\lambda\Delta n_{\max}d + 1$
- all other options: $2 \times 2\pi/\lambda\Delta n_{\max}d + 1$

Note that all options beyond *close cylinders* usually lead to orders of truncature much higher than what is needed for reasonable accuracy. Further, increasing the order of truncature does not necessarily lead to more accurate results: indeed, the numerical precision on Bessel functions decreases with their order, and the accuracy of the determinant diminishes with increasing matrix size.

(*Synonyms*: eval order, auto order)

6	symmetry C_{6v}
2	number of cylinders in the irreducible sector
5	order of truncature for cylinder Bessel expansion
72	order of truncature for cladding Bessel expansion
(2.102,0.0d+0)	matrix epsilon
(1.0d0,0.0d0)	exterior epsilon
6.50d0 6.655172d0	cladding inner and outer radius
(1.315d0,0.0d+0)	cladding epsilon
0.0d0 0.0d0 0.2d0 (1.0d0,0.0d0) 0 0	cylinder 1: r,theta,radius,epsilon, symmcat,axis
1.0d0 0.314159265358d0 0.2d0 (1.0d0,0.0d0) 0 0	cylinder 2: r,theta,radius,epsilon, symmcat,axis

Table 1: Example of a structure file

2.6 Structure files

The structure file details the structure: position, size and refractive index of inclusions, cladding and jacket. The order of truncature of the Bessel expansions are also given in this file, but can be redefined in the parameter file. The structure file is a text file which can be edited with your usual text editor. Table 1 gives an example of the structure of the file (each line of the table should be a line of the file). The file has a header defining global parameters and the cladding structure:

1. Symmetry: symmetry class, 0 if no symmetries are used, 1 for C_{1v} , 2 for C_{2v} , 4 for C_{4v} , 6 for C_{6v} , 8 for C_{8v} , 10 for C_{10v} . Other symmetries are not yet implemented.
2. Number of cylinders in the irreducible sector (integer)(*ie* the smallest angular sector given for that symmetry in Ref. [4]) The irreducible sector is $(0, \pi/10)$ for C_{10v} , $(0, \pi/6)$ for C_{6v} , $(0, \pi/2)$ for C_{2v} , $(0, \pi)$ for C_{1v} , the entire plane if no symmetries are used.
3. Order of truncature for Fourier Bessel expansions around cylinders (integer): The Fourier-Bessel expansion around each cylinder will be truncated from -order to order
4. Order of truncature for Fourier-Bessel expansion in the cladding and jacket (also called *gorder*, integer): The Fourier-Bessel expansion on the cladding and the jacket will be truncated from -*gorder* to *gorder*
5. Matrix epsilon (complex) : permittivity of the matrix (or background)
6. Jacket epsilon (complex) : permittivity of the jacket.
7. Cladding inner (real) and outer radius (real) (the latter is also called the jacket radius). If no cladding is used, set the cladding and jacket permittivities to a same value, set the inner radius to be the jacket radius and give an arbitrary outer radius (say, for example, inner radius+1.0).
8. Cladding epsilon (real) : permittivity of the cladding. If no cladding is used, give the same permittivity as for the jacket. If the structure should have no jacket and no cladding, set the matrix, cladding and jacket permittivities to a same value.

9. After the header follows the list of cylinders, with following specifications:

- (a) r , θ (real): cylindrical coordinates of the center of the cylinder, θ is in rad
- (b) radius (real): radius of the cylinder
- (c) epsilon (complex): permittivity of the cylinder
- (d) `symm_cat` (integer):
 - -1 if no symmetries are used:
 - 0 if the cylinder is located at the origin
 - 1 if the cylinder is on the border of the irreducible sector.
 - 3 if the cylinder is not on the border of the irreducible sector.
- (e) `axis` (integer):
 - 0 : if the cylinder is at the center or not on the border
 - 1 if the cylinder is on the $\theta = 0$ axis
 - 2 if the cylinder is on the other border of the irreducible sector (for example $\theta = \pi/6$ for a C_{6v} structure)

Parameters on a same line are separated by blanks, tabulations and/or commas. Comments can be added at the end of each line. Additional blank lines between parameters are not permitted. Splitting the lines in the cylinder list is possible if no comments are added.

When the permittivities of the cladding and the matrix are equal, `FIBRE` replaces the reflection and transmission matrices by the null matrix and the identity matrix respectively. The same applies to the cladding/jacket interface. However, setting the cladding inner and outer radii to be equal is not a valid way of “suppressing” the cladding, and may lead to division by zero operations.

If the Sellmeier equation of Silica should be used to compute a permittivity according to the wavelength, one has to set the concerned permittivity(ies) to (-1.0d0,-1.0d0). Note that in this case all dimension become absolute: the dimensions given in the structure file and in the parameter file (for the wavelength) are then in μm .

3 Using WINFIELD

WINFIELD is a software to compute and analyse fields from their Fourier-Bessel expansions, using a dialog based user interface for Microsoft Windows.

3.1 Main dialog controls

- **open file...** button: Click here to open the standard Windows file open dialog and select a .bcf or .fbb file to open.
- **refresh** button: updates the field representation (and runs the field computation when needed)
- **resolution** text edit: number of points along each axis on which the field is computed.
- **view pos file...** button: Click here to open the standard Windows file open dialog and select a structure (or cylinder position) file (.txt extension) for visualization.
- **More options...** button: Opens the advanced tools dialog, including controls to compute the Bloch transform, the Wijngaard test, the alternate method to compute losses, and controls for exporting field data in various formats (various formats of raw data or bitmap)
- **field selector** : The three central drop down lists select the field (Poynting vector S , fields H or E), the component (Cartesian x , y or z , cylindrical r or θ , norm, Bloch transform, or structure), and the part of the component (real part, imaginary part, magnitude, phase or magnitude in log scale) to visualize.
- **zoom** group: Enter a zooming coefficient and click the **zoom now** button to zoom on the current coordinate.
- **current coordinates** group: sets the center for zooming. You can enter the coordinates by clicking on the field map. Clicking on the field map will also give you the z -coordinate, that is the value of the selected part / component / field at the point clicked, and show you which part of the structure you clicked on (in the status bar on the bottom).
- **color range** group: lets you chose whether you want to use the extremal values of the current part / component / field as the limits of the color scale, or if you prefer to set the limits of the color range manually (uncheck **use extremal values** and enter the minimum and maximum values). You can also inverse the color scale, or use grey scale representation. If you check the *Flux norm* box, WINFIELD will use a consistent flux based normalization, useful to compare field densities.
- **interpolate** uses bi-cubic interpolation to give a smoother representation of fields, and enables you, with the **contour** check box next to it, to draw filled contour plots.
- **no cladding** and **no jacket** check boxes: hide the field in the cladding and jacket, and adjust the color range accordingly.

- **ignore cladding** performs all field computations as if the cladding and jacket didn't exist (useful - and automatically set- when the cladding and jacket index are the same as the matrix index).
- **draw cylinders** and **fill cylinders** check box: enables you to draw the contour of the cylinders and optionally fill them.

3.2 Advanced tools dialog controls

The 'Advanced Tools' dialog appears when clicking on the 'more options...' button in the main dialog.

- **Field group:** Controls in this group enable you to write the computed fields to files in various formats. The file format can be selected in the **Format** drop down list. Once you have chosen the file format, click on the **Write file...** button to open a standard windows 'Save As' dialog. File formats are as follows:
 - **selected part text xyz (*.txt)** Writes the selected part (real, imaginary, abs, phase, log(abs)) of the selected component and field (as selected in the main dialog) to a text file in three columns separated by spaces. The first two columns are the x and y coordinates, the third column is the value of the selected part / component / field. Note that by default no file extension is added to the file name, so that you have to type in the desired file extension in the 'Save As' dialog.
 - **selected complex field text xyz (*.txt)** Writes the selected component and field (as selected in the main dialog) to a text file in four columns separated by spaces. The first two columns are the x and y coordinates, the third and fourth column are the real and imaginary part of the selected component and field. Note that by default no file extension is added to the file name, so that you have to type in the desired file extension in the 'Save As' dialog.
 - **Selected part text matrix (*.txt)** Writes the selected part (real, imaginary, abs, phase, log(abs)) of the selected component and field (as selected in the main dialog) to a text file in matrix format. The file contains a $\text{resolution} \times \text{resolution}$ matrix of numbers representing the value of the selected part/component/field for each pixel. You can write the x,y coordinates associated with each pixel in the same format using the **x,y coor matrix (*.txt)** option (see below). Note that by default no file extension is added to the file name, so that you have to type in the desired file extension in the 'Save As' dialog.
 - **selected complex field text matrix (*.txt)** Writes imaginary and real parts of the selected component and field (as selected in the main dialog) to a two text files in matrix format. The names of the matrix file containing the real and imaginary part are formed by adding '_r' and '_i', respectively, to the file name selected in the 'Save As' dialog. The files contains a $\text{resolution} \times \text{resolution}$ matrix of numbers representing the real/imaginary part of the selected component and field for each pixel. You can write the x,y coordinates associated with each pixel in the same format using the **x,y coor matrix (*.txt)** option (see below). Note that by default no file extension is added to the file name, so that you have to type in the desired file extension in the 'Save As' dialog. This format is particularly suitable for importing the complex fields in matrix manipulating software *eg* Matlab or Octave.

- x,y coor matrix (*.txt) Writes the x and y coordinates of each pixel in matrix form in two different files. The names of the matrix file containing the x and y coordinates are formed by adding '_x' and '_y', respectively, to the file name selected in the 'Save As' dialog. The files contains a `resolution` \times `resolution` matrix of numbers representing the x/y coordinates of each pixel. Note that by default no file extension is added to the file name, so that you have to type in the desired file extension in the 'Save As' dialog.
- as seen Windows Bitmap(*.bmp) Writes the exact content of the WINFIELD 's field window to a windows bitmap file you can open with any Windows image editor (*eg* Paint, Windows Picture and Fax viewer etc.). If no file extension is added in the 'Save As' dialog, the .bmp extension is automatically added.
- Wijngaard test group: Used to perform Wijngaard tests (see Eq. (4), Ref. [2]) around inclusions or at the cladding boundary. It can also check the continuity of fields at the cladding and jacket boundaries. To compute the Wijngaard integral W around a given inclusion, enter the number of points used to compute the integral (**number of points** edit box) click on the desired inclusion (in the field plot window), select the field (E_z or H_z) then click the **Check it!** button. If you check the **Write File** check box, you can also write the local and Wijngaard fields as a function of angle at the boundary of the inclusion to a file. The resulting file contains 4 columns of numbers, the first is the angle (radians), the second the local field, the third the value of the field resulting from the Wijngaard expansion, and the fourth is the magnitude of the difference between the second and third column.

When selecting the **Jacket** radio button, the local and Wijngaard fields are replaced by the fields just inside and just outside the jacket boundary respectively. When selecting the **Cladding continuity** radio button, the Wijngaard field is the Wijngaard field expansion at the matrix/cladding boundary, and the local field results from the field expansion in the cladding at the matrix/cladding boundary.
- Loss group: Used to compute the losses and imaginary part of the effective index with the perturbative formula from Eq. (10) in Ref.[2]. To compute the surface integral in that equation, WINFIELD uses the fields computed for the field representation. For accurate results, `resolution` should hence be at least 150, and the current view should include all inclusions. Note that even with a perfect numerical integration, the relative error obtained through this method is of the order of the largest W Wijngaard integral of all inclusions. The contour integral is computed using **Number of points** points on the circle centered on the origin and with radius specified by the user. If the radius is left to zero, $0.99 \times$ the cladding radius is used.
- Effective Core Area group: used to compute effective areas. The conventional effective area formula is used, but the user can select which field and component to use in the integrals. Checking the **normalize to pitch** check box displays the result divided by the pitch squared. (Note that the algorithm extracting the pitch works properly only with C_{6v} structures). Again, numerical integration relies on the fields previously computed for displaying fields, and hence the accuracy of the effective area will depend on `resolution` and the current viewport.

- Bloch Transform group: is used to compute the Bloch transform as defined in Ref [5]. The field used for the Bloch transform can be selected either in this group or with the field drop-down list in the main dialog. The resolution to be computed and displayed for the Bloch transform can be defined separately from the resolution for the fields. The Order edit box lets you select the order of the Bloch transform. If you enter a value greater than the truncation order of the Fourier-Bessel series, the total Bloch transform will be displayed.
- Shoz Colour Scale displays the colour scale currently in use.

4 Notes, tips, trouble shooting, known bugs:

4.1 FIBRE

- *Computer slows down and becomes unusable when FIBRE is used.* By default, FIBRE is run with normal priority. Since FIBRE is computationally demanding, this results in a slowdown for other applications. To avoid this, you can start the simulation using

```
start /low fibre
```

or

```
start /belownormal fibre .
```

This will run FIBRE with a lower priority. Alternatively, you can redefine the priority using the Windows Task Manager.

- *FIBRE causes a numerical error, but the effective index region does not overlap any of the structure's refractive indices.* Check that there are no 'fictitious' inclusions, *ie* inclusions with same refractive index as the surrounding matrix.

4.2 WINFIELD

- In a viewport symmetric around the origin, if the resolution is set to an odd number of points, the central pixel will correspond to the origin of the plane, for which some components of some fields may be undefined in the analytic internal representation of the fields used by WINFIELD. This results in a black pixel at the centre of the picture. To avoid this, set resolution to an even number, or slightly shift the viewport.
- If you try to load a .bcf associated with a structure file which defines a symmetry different from the symmetry used in the .bcf file, WINFIELD will stop.

5 Acknowledgments

The software as it is was written by Boris Kuhlmeier. The initial draft of software which has evolved into the CUDOS MOF Utilities was written during his PhD studies under the supervision of Gilles Renversez, Ross McPhedran, Daniel Maystre, and de facto also C. Martijn

de Sterke. Tom White wrote a similar piece of software almost simultaneously as part of his honours project, and although the current version of the software doesn't include code from Tom White, it certainly has benefited a lot from his pioneering work. After his PhD Boris Kuhlmeij joined CUDOS, and rewrote a substantial part of the code to make it more stable and more user-friendly with the aim of making it publicly available. Prof. Ross McPhedran and Prof. C. Martijn de Sterke are also with CUDOS, and Dr. Gilles Renversez and Dr. Daniel Maystre are with the Institut Fresnel, UMR 6133, Marseille, France. Tom White is currently finishing his PhD with CUDOS/the University of Sydney. Boris Kuhlmeij would also like to acknowledge the support of the system administrators of both the School of Physics/University of Sydney (Dr. Sebastian Juraszek, Dr. Tony Monger and George Shan) and the Institut Fresnel (Frederic Forestier).

This work was produced with the assistance of the Australian Research Council under the ARC Centres of Excellence Program, and has benefitted from travel support provided by the French and Australian governments under the PICS/IREX and cotutelle schemes.

References

- [1] T.P. White, B.T. Kuhlmeij, R.C. McPhedran, D. Maystre, G. Renversez, C.M. de Sterke, and L. C. Botten. Multipole method for microstructured optical fibers. I. Formulation. *J. Opt. Soc. Am. B*, 19(10):2322–2330, 2002.
- [2] B.T. Kuhlmeij, T.P. White, G. Renversez, D. Maystre, L. C. Botten, C.M. de Sterke, and R.C. McPhedran. Multipole method for microstructured optical fibers. II. Implementation and results. *J. Opt. Soc. Am. B*, 19(10):2331–2340, 2002.
- [3] <http://www.physics.usyd.edu.au/cudos/mofsoftware/>.
- [4] P. R. Mclsaac. Symmetry-induced modal characteristics of uniform waveguides-I: Summary of results. *IEEE Trans. Microwave Theory Tech.*, MTT-23:421–429, 1975.
- [5] B. T. Kuhlmeij, R. C. McPhedran, and C. M. de Sterke. Bloch method for the analysis of modes in microstructured optical fibers. *Optics Express*, 12(8):to be published, 2004.

Index

- MN_r , 12
- N_r , 12
- R , 12
- R_{Cl} , 12
- R_J , 12
- λ , 17, 21
- λ_0 , 21
- λ_f , 21
- n_{Cl} , 13
- n_J , 13
- n_M , 13
- n_i , 13
- q , 12
- n_{eff} , 21
- n_{eff0} , 18
- .bin file, *see* determinant files
- .log file, *see* determinant files
- .true., 11
- _results.txt, *see* result file
- action keyword, 8
- argument, 9
 - character string, 9
 - complex, 7, 9
 - integer, 9
 - logical, 9
 - real, 7, 9
 - real, used as complex argument, 7
- average, 25
- bcf file name, 22
- bessel coefficient file name, 14
- Bitmap, 30
- Bloch Transform, 31
- Bloch transform, 28
- build fibre, 8
- Build Fibre, 13
- central cylinder, 11
- central cylinder epsilon, 12
- central cylinder index, 12
- central cylinder radius, 11
- cladding index, 13
- cladding order, 24
- cladding radius, 12
- close cylinders, 25
- color range, 28
- comments, 6, 9
- core, 25
- current coordinates, 28
- cylinder index, 7, 13
- cylinder radius, 12
- definition statements, 7
- degeneracy threshold, 15
- delete n_eff table, 22
- det_mode_threshold, 14
- determinant files, 6
- determinant mode threshold, 14
- diameter on pitch ratio, 12
- dispersion file name, 22
- draw cylinders, 29
- eigen-value threshold, 14
- eigen_value_threshold, 14
- end, 8, 9, 11
- epsilon, 11, 12
- error file, 5
- errors.txt, *see* error file
- exponent, 7
- fast, 24, 25
- Field, 29
- field selector, 28
- file name, 23
- file name radix, 23
- file suffix, 23
- first mode, 19
- Fourier-Bessel coefficients files
 - binary, 6
 - text, 6
- fundamental, 19
- get dispersion curve, 22
- ignore cladding, 29
- imag_precision_threshold, 15
- imaginary precision threshold, 15
- Installing FIBRE , 4

interpolate, 28
 jacket index, 13
 jacket radius, 12
 keywords
 order, 10
 l, 12
 lambda, 8
 last mode, 19
 load bcf file, 14, 22
 load dispersion file, 22
 load structure file, 14
 logsteps, 20
 lower imaginary threshold, 16
 lower real threshold, 16
 lower_imag_threshold, 16
 lower_real_threshold, 16
 material dispersion, 7
 matrix index, 13
 minima threshold, 16
 minima_threshold, 16
 MNr, 7, 12
 mode, 19
 mode table file, 6
 More options..., 28
 n, 12
 n_eff(lambda), 21
 n_eff_0, 18
 n_eff_width, 18
 No Cladding, 7, 12
 no cladding, 28
 No Jacket, 7, 13
 normal, 24, 25
 Nr, 7, 12
 num_n_points, 18
 num_ni_points, 18
 number of points, 20
 number of points between log, 20
 number_n_points, 18
 open file..., 28
 optimization, 23
 option, 23, 24
 order, 24
 overwrite, 23
 parameter file, 5
 structure, 10
 syntax, 9
 parameter file(, 6
 parameters.txt, *see* parameter file
 paranoid, 23
 pitch, 7, 12
 precise, 25
 progress file, 5
 progress.txt, *see* progress file
 radius, 11
 real precision threshold, 15
 real_precision_threshold, 15
 refresh, 28
 resolution, 28
 result file, 6
 rigorous, 25
 save structure, 13
 scale invariance, 7
 search modes, 8, 19
 second, 19
 second nearest neighbour, 25
 skip real borders, 24
 skipped minima, 6
 start n_width, 20
 start_lambda, 21
 start_mode, 19
 steps, 20
 stop_lambda, 21
 stop_mode, 19
 stop_n_eff, 18
 structure file, 5, 8
 structure file name, 13, 14
 suffix, 23
 suggest n_eff range, 8, 19
 suggest order, 8, 24
 Uninstalling FIBRE , 4
 upper imaginary threshold, 17
 upper real threshold, 16
 upper_imag_threshold, 17
 upper_real_threshold, 16
 val, 24

verbose, 11

verbose file, 23

view pos file..., 28

wavelength, 17

Wijngaard test, 15, 28, 30

zoom, 28