Simulation of microwave beams with PROFUSION (2022 Edition)

Burkhard Plaum

July 12, 2022

Contents

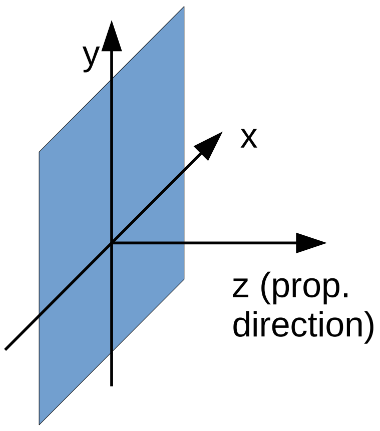

Figure 1: Geometric definitions

1 Introduction

Quasi-optical microwave beams are used in several applications, most notably in electron cyclotron heating systems of thermonuclear fusion experiments. Even in systems, where the actual transmission is done with oversized HE_{11} waveguides, there are free space beams before and after the transmission line. At the input, the beam coming out of the gyrotron window is coupled into the waveguide via a set of beam shaping mirrors and possibly polarisers. After the transmission line, a launcher is used to inject the microwave into the plasma. These launchers are also quasi-optical components.

In the simplest case, optical transmission systems can be designed using the well established Gaussian optics formalisms. In realistic scenarios, however, the higher-order modes need to be taken into account as well. They can cause high heat loads and/or arcing at possibly unexpected locations or disturb the beams, which are launched into the plasma. Especially with the ever increasing power and pulse lengths of current and future fusion experiments accurate calculation tools are mandatory.

The PROFUSION code package contains a large number of tools, which allow the calculation of different phenomena involving microwave beams. The tools are available as commandline programs for the Linux operating system, while the parameters are passed as commandline arguments. The basic data structure is the *field matrix*, which contains the sampled transversal electric field at one position on the propagation axis. Many of the tools require exactly one field matrix as input and produce another field matrix as output. Those tools can read the field matrix from the standard input and write the resulting field matrix to the standard output. This allows to connect multiple operations using Unix pipes without the need to store intermediate results in files. The implementation as commandline programs follows the old Unix approach (write one program for one task) and has numerous advantages over, e.g. a graphical user interface:

- The tools are independent executables, which are very simple since they do just one operation
- The data transfer via pipes avoids the need of temporary files, and is a well established and robust mechanism provided by the operating system
- Since all parameters can usually be passed as commandline arguments, arbitrary multidimensional parameter sweeps can be performed using the loop constructs provided by the shell
- All calculations are inherently scriptable using the Unix shell
- In addition to the Unix shell, PROFUSION commands can be used in all other environments, which allow the execution of external programs (e.g. Python scripts or GUI applications)

PROFUSION has been used in numerous projects and is considered a mature software package.

2 Field matrix structure

A field matrix describes a field on a limited rectangular area perpendicular to the beam axis. Fig. 1 shows the coordinate system. The z-axis always corresponds to the main propagation direction, the transversal directions are x and y. Typically, if we have a linear polarised field, we set the x-Component to zero. This way, the polarisation matches the usual definitions of the waveguide modes (TE10 for rectangular, TE11 for circular waveguides) found in literature, where the dominant polarisation is vertical.

For the full description of the field in the cross section of a microwave beam, we need the following parameters:

- The x and/or y components of the sampled field as complex matrices

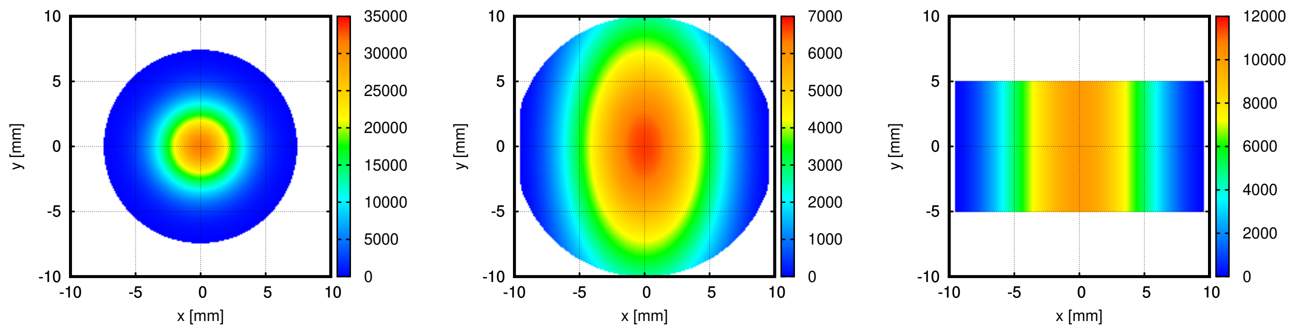

Figure 2: Absolute value of E_y for a Gaussian beam (left), a TE$_{11}$ mode in a circular waveguide (centre) and a TE$_{10}$ mode in a rectangular waveguide.

- The horizontal and vertical dimensions of the matrix
- The x and y positions of the boundaries
- The frequency

In the linear polarised case the matrix for the E_x or E_y component is missing rather than filled with zeros. This speeds up most operations for linear polarisation. All mentioned parameters are stored in a binary format, which is optimised for fast reading and writing. Internally all quantities are stored in SI-units, in particular the frequency is given in Hertz and lengths are given in Meters. If output files or plots contain different units, they are converted just before they are written. Also, complex numbers are *always* stored as real and imaginary parts. Only the output- and plot routines convert them to amplitude and phase.

3 Field generators

Field matrices can be generated from numerical (e.g. measured) data or from analytical functions. In the former case, we just need a reprices for converting the data into the field matrix format. For analytical field descriptions, there is the tool `fm_gen`. For generating e.g. a Gaussian beam at 140 GHz with a waist radius of 4.5 mm sampled at 200×200 points over a cross section of 20×20mm, one can call

```
fm_gen -xy 200,-0.01,0.01 -freq 140e9 -f gauss,4.5e-3 -o gauss.fm
```

In this case, the output is written to the file `gauss.fm`.

In addition to the simple Gaussian generator, there are numerous other field generators:

- Gaussian beam with general astigmatism
- Mixtures of Gauss-Hermite modes
- Mixtures of Gauss-Laguerre modes
- J$_0$-pattern for generating simplified HE$_{11}$ modes of corrugated cylindrical waveguides
- LP$_{mn}$-pattern for generating simplified mode spectra of corrugated cylindrical waveguides
- Radiating waveguide apertures (rectangular, cylindrical smooth, cylindrical corrugated, square corrugated) fed by arbitrary mode mixtures

Some examples for generated fields are shown in Fig. 2. Usually fields are generated with a power of 1 W. A normalised power is important for many analysis methods described in the following sections.

A special generator is the fieldgrid (`fg`), which can be generated from any fieldmatrix. It is generated once by propagating an initial field matrix to different distances up to z_{max}. It allows then a fast interpolation of the actual field values at different z-positions.

The z-resolution is given in terms of λ. When storing the field values in the interpolation grid, the field values are divided by the plane-wave phase of $e^{-jk(z-z_0)}$, which increases the effective wavelength and allows z-resolutions of 1.0 or more.

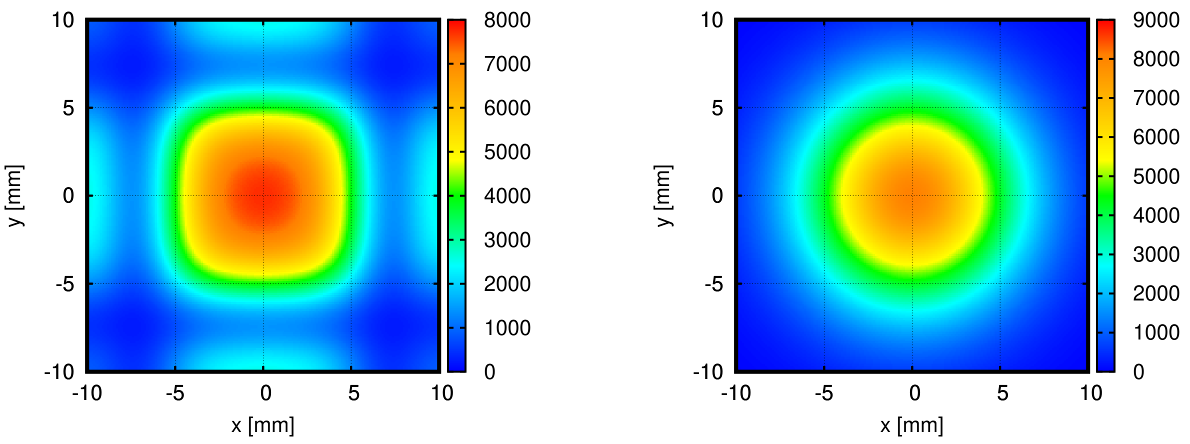

Figure 3: Gaussian beam from Section 3 propagated by 50 mm. Left: Without padding, right: With 50 mm padding.

4 Propagation

Propagation is the most common operation on field matrices. The tool `fm_prop` uses an FFT transform to decompose the field into a number of plane waves, which are tilted with respect to the beam axis [1]. These plane waves can be propagated trivially by any distance. The propagated field is calculated using an inverse FFT.

One issue with this method is the implicit assumption of the Fourier transform, that the field is infinitely periodic in x and y directions. This means, that the propagation is done for the beam plus an infinite number of virtual neighbour beams. Depending on the extent of the field matrix, the beam size and the beam divergence, we can observe an interference with these virtual neighbour beams in the propagated field.

To suppress these artefacts, we are padding the field matrix with a border containing a zero field before propagating. This border is removed from the propagated field before it is saved, so the size of the initial and propagated field matrices is identical. The border width is adjustable (with the `-pad` option). Unfortunately it is not possible to determine the optimum border width programmatically. The artefacts are, however, clearly visible and the border width can be increased experimentally until the artefacts disappear. Especially when doing parameter sweeps, the padding parameter should not be much larger than necessary because it increases the FFT size and thus the calculation time.

Fig. 3 shows the Gaussian beam of Fig 2 (left) propagated by 50 mm. In the left image we clearly see the neighbouring beams. In the right image, where a padding of 50 mm was applied, these artefacts are suppressed. As mentioned before, multiple commands can be chained together by pipes. Therefore the fieldmatrix in Fig. 3 (left) can be generated by:

```
  fm_gen -xy 200,-0.01,0.01 -freq 140e9 -f gauss,4.5e-3 | \
fm_prop -d 0.05 -pad 0.05 -o propagated.fm
```

In this case, the output is written to the file `propagated.fm`.

5 Focusing elements

Focusing elements are typically realised as metallic mirrors with a concave surface, especially in high power applications. To simulate these in PROFUSION, there is a tool `fm_lens`. It applies a thin lens with given focal lengths f_x and f_y in x and y-directions. A phase shift $\Delta\phi(x,y)$ is applied to all field points with:

$$\Delta\phi(x,y) = \frac{2\pi}{\lambda}\left(\frac{x^2}{2f_x} + \frac{y^2}{2f_y}\right) \tag{1}$$

where x and y are the locations on the field matrix. Special commandline options (`-dx`, `-dy`) exist for cases, where the centre of the lens is not in the origin of the coordinate system.

Fig. 4 shows the Gaussian beam from Section 3 propagated by 100 mm and sent through a lens with $f_x = f_y = 50\ mm$. As predicted from theory [3], the second focal point comes 86.95 mm after the lens. The beam profiles before and after the lens were plotted with the tool `fm_propprof`, which directly generates PNG files. Special commandline options ensure, that identical colour scales are used for both pictures.

6 Apertures

Mirrors have limited sizes, which can truncate a part of the Gaussian beam. This results in diffraction effects due to the field discontinuity at the mirror boundary. Furthermore, the power fraction situated outside of the mirror typically

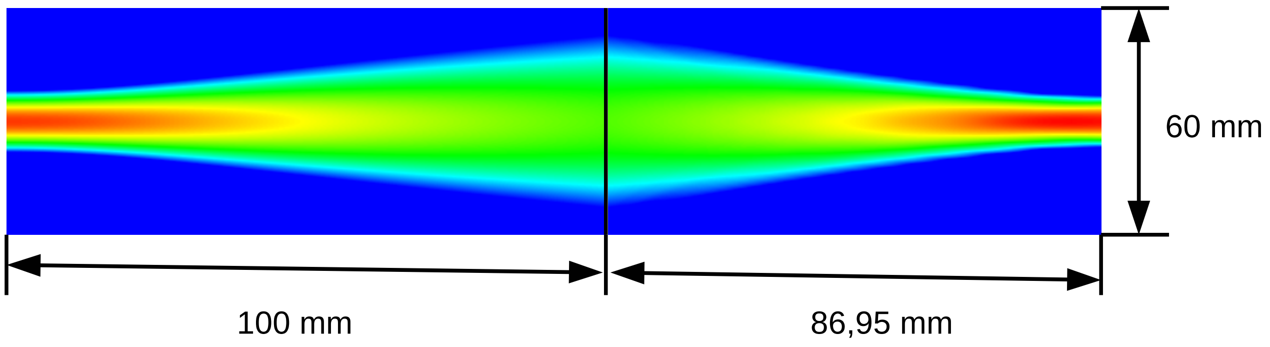

Figure 4: Gaussian beam from Section 3 propagated by 100 mm and sent through a lens with focal length 50 mm. The second focal point comes 86.95 mm after the lens

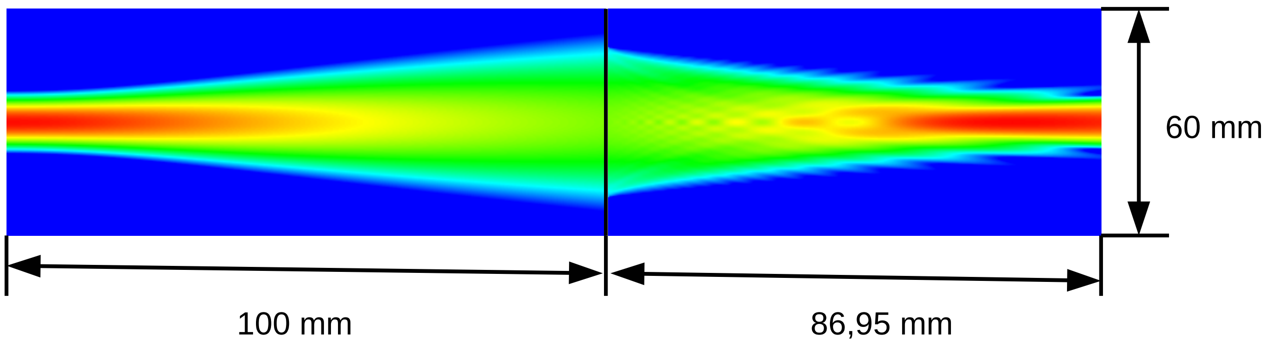

Figure 5: Gaussian beam from Fig. 4 but with a limited lens size of 40 mm

contributes to the stray radiation in the system.

In PROFUSION, there is a tool `fm_ap`, which applies a rectangular or elliptical aperture to a field matrix. This is done by simply setting the field values outside the aperture to zero. Fig 5 shows the example from Fig. 4 but with a limited lens diameter of 40 mm. One can see the truncation effect and the diffraction due to the field discontinuity.

To quantify the power loss due to the beam truncation, the naive approach would be to subtract the powers before and after the aperture. In the case of very small losses the result is the difference of two almost equal numbers. This means, that small errors in the powers lead to large errors in the difference. The correct way is to integrate the power, which lies outside the aperture. The tool `fm_truncloss` does the accurate calculation and supports the same aperture shapes as `fm_ap`. An example is given in Section 15.1.

7 Diagnostic tools

For obtaining relevant parameters from the calculated fields, a large number of diagnostic tools is available.

7.1 Power calculation

One of the most important diagnostic tool is the integration of the total power of a field distribution. Details of the calculation, however, are also applicable to other diagnostic methods outlined below.

The simplest method is to calculate the sum of the power densities over all pixels. A more precise method, however, is to use a higher-order interpolation scheme to emulate a smooth function, and an integration method with adaptive step-size and error estimation. The most accurate results are obtained with a Catmull-Rom interpolation. The integration is a 2D Simpson method as suggested in [2]. Another important detail is that the interpolation is applied to the *power density* instead of the field strength itself. Interpolating the E-field itself can distort the result because of a slight violation of the energy conservation. Table 7.1 shows the power of the Gaussian beam from Fig. 2 (left) for different sample densities and for least and most accurate algorithms. Note that even for 20×20 samples the interpolation method is sufficient in calculating the power.

For noisy (i.e. measured) field data, however, the sum method is advantageous because single noise peaks in conjunction with interpolation distort the results and have a poor convergence in the integration routine. In the case of measured data, the error bars are typically dominated by the equipment rather than the analysis algorithm. In PROFUSION the tool `fm_getpow` can calculate the total power with various algorithms and `fm_norm` can normalise a field matrix to unity power.

Linear size	Sum	Interpolation
20	0.90250	0.99999
40	0.95062	0.99998
60	0.96693	0.99999
80	0.97514	0.99997
100	0.98009	0.99996
120	0.98339	0.99998
140	0.98575	0.99998
160	0.98752	0.99998
180	0.98890	0.99998
200	0.99001	0.99998

Table 1: Results of the power integration for a Gaussian beam (See Fig. 2 left) using the sum and Catmull-Rom interpolation.

7.2 Overlap integral

Overlap integrals are used to characterise the purity of a measured or calculated field with respect to some reference field distribution. Physically, the overlap integral can be interpreted as the complex power integral, where the E-component is taken from one field distribution, the H-field is taken from the other field:

$$C_{12} = \frac{\iint\limits_S \mathbf{E}_1 \times \mathbf{H}_2^* dS}{\sqrt{\left|\iint\limits_S \mathbf{E}_1 \times \mathbf{H}_1^* dS\right|}\sqrt{\left|\iint\limits_S \mathbf{E}_2 \times \mathbf{H}_2^* dS\right|}} \tag{2}$$

where the terms in the denominator ensure that both field components have a unity power. Although the calculation corresponds to a power calculation, the result is equivalent to a complex amplitude. The phase angle can be interpreted as an overall phase shift between the two fields and is not relevant in most cases. The purity, which is a power quantity, is therefore the squared absolute value ($|C_{12}|^2$) of the overlap integral. Overlap integrals are calculated with `fm_overlap`, which has similar options as `fm_getpow`.

Note that the power normalisation (as in the denominator of Eq. 2) is *not* done by `fm_overlap`. If necessary, the input matrices need to be normalised with `fm_norm` before.

7.3 Fitting of Gaussian beam parameters

The overlap integral as described in section 7.2 is a good measure for a beam purity in cases where a fixed set of beam parameters is defined as reference. This is, however, the most pessimistic characterisation, because no distinction is made between a variation of the beam parameters and the generation of higher-order modes. This is due to the fact, that in Gaussian Optics, each set of beam parameters (waist sizes and -locations) defines an individual orthogonal system of modes.

When designing optical transmission systems, however, the parameters of the fundamental TEM$_{00}$ mode as well as the higher-order modes are of interest and need to be available as separate quantities. The beam parameters can be used to design mirror surfaces. The higher order modes should be suppressed as much as possible because - due to their larger divergence angles - they are likely to contribute to the overall stray radiation, especially for long transmission lines.

The established method of characterising (e.g. measured) Gaussian-like beams consists of a fit procedure, which determines the beam parameters of in order to maximise the overlap integral (according to Eq. 2) for the fundamental TEM$_{00}$ mode.

The fractional power P_{HOM} of higher-order modes then becomes:

$$P_{HOM} = 1 - |C_{\mathrm{TEM}_{00}}|^2 \tag{3}$$

where $C_{\mathrm{TEM}_{00}}$ is calculated from (2) with the fitted parameters. In PROFUSION, there is a tool `fm_fit`, which performs the described procedure. The initial values are obtained from the field by statistical evaluation. When run in full mode, the following parameters are obtained:

- The complex beam parameter in horizontal and vertical directions
- The position (x, y) of the beam axis in the field matrix
- Tilt angles in $x-$ and $y-$directions

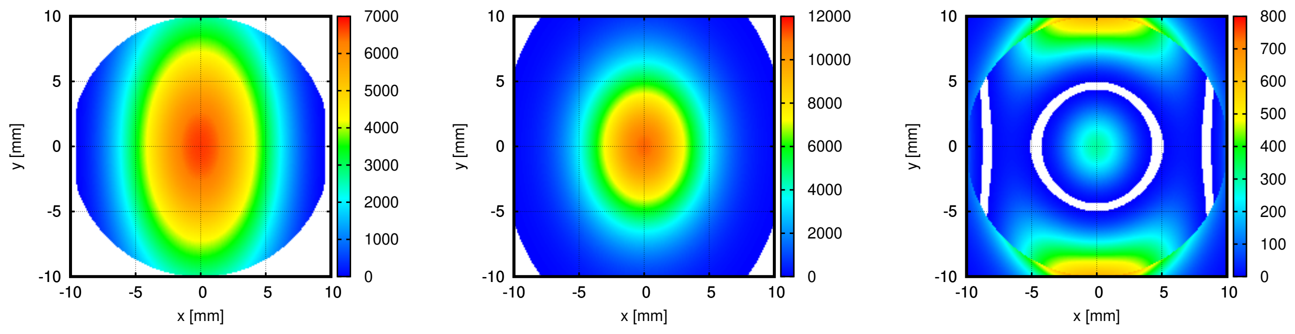

Figure 6: Absolute value of E_y for a TE_{11} field in a cylindrical waveguide aperture (left), the fitted Gaussian beam (centre) and the residual field (right).

Furthermore there are commandline options, which limit the number of fit parameters. When a field distribution is e.g. known to be centred with a beam axis perpendicular to the field matrix (e.g. because it is synthetically generated), the position and tilt angles can be omitted for the fit (`-nopos`, `-notilt`). Another option (`-circ`) can be used for field patterns, which are known to be rotational symmetric. In this case, the beam parameters in horizontal and vertical directions are assumed to be identical. Omitting fit parameters reduces the dimensions of the parameter space and improves convergence.

As an example, Fig. 6 shows the TE_{11}-field of an open-ended circular waveguide from Fig. 2. The fit procedure obtained an elliptical Gaussian beam which contains 88.86 % of the total power (centre). The right image shows the residual field, which can be interpreted as the sum of spurious modes.

Calculating the residual field allows to investigate the behaviour (e.g. divergence angles, dominant directions) of the spurious modes.

From equation 3 we can see, that it is essential, that the field passed to `fm_fit` has a unity power, especially if the value for the higher-order modes is important. This needs to be ensured before by calling `fm_norm`.

8 Further utilities

A large number of simple but useful tools for operating on field matrices has been developed over the years. They all support the commandline option `-help` to show the supported options.

- `fm_add`, `fm_diff` Adds/subtracts two field matrices
- `fm_xpol` Calculates the power in both polarizations
- `fm_peak` Gets the maximum field value
- `fm_tilt` Compensates small phase errors due to misalignments from measured data
- `fm_profile` Calculates the horizontal and vertical profiles from a field matrix
- `fm_dump` Prints information about the fieldmatrix
- `fm_cat` Converts a fieldmatrix to ASCII format
- `fm_extract` Extract single field values at given coordinates
- `fm_antennadiagram` Calculate an antenna diagram from an aperture field
- `fm_norm` Normalise the field matrix for unity power or other criteria
- `fm_plot` Plot the field components in gnuplot or png formats
- `fm_tweak` Change some internal parameters (e.g. frequency) of a field matrix

9 Interface with waveguides

In transmission systems for megawatt power at millimetre wave frequencies, oversized HE_{11}-waveguides are typically used. Since the wave impedance in these waveguide is very similar to the free space impedance, transitions are very simple because no impedance matching is necessary. At the input of a transmission line, the beam from the Gyrotron is fed via beam shaping mirrors into the waveguide. The oversized waveguides are, however, sensitive to misalignments of the coupling optics.

The calculation of the mode spectra, which are excited by the free space field at the waveguide entrance, is done with eq. 2, where one field is the free space field, the other is the transversal field of the waveguide mode. PROFUSION has tools for doing mode analyses for all supported waveguide types: `cyl_analysis` and `rect_analysis` are for smooth wall waveguides with circular or rectangular cross section. The commands `cyl_corr_analysis` and `sqc_analysis` are for corrugated waveguides with circular and square cross-section respectively. The modes, which should be taken into account, are given in the waveguide description (see section A.1). The field distribution is the same as for the `fm_gen` command (see section 3).

The transformation of a spectrum of waveguide modes to a free-space field is done by summing the modal fields (multiplied with the amplitudes). Here we also assume that the size of the waveguide is large enough such that the impedance mismatch can be neglected. An aperture field can be generated by using the the field generators `cyl`, `corr`, `rect` and `sqc` with the `fm_gen` command.

10 Gaussian optics calculations

PROFUSION contains tools, which do simple calculations with analytically given Gaussian beams. They can be used to do a preliminary design of an optical system with free space propagation and focusing elements. Since the tools work with simple analytical formulas, they are extremely fast and can be used to make multidimensional parameter studies.

The tools support circular as well as astigmatic Gaussian beams. Like the field matrices, they write the parameters to the standard output and read parameters from the standard input. The parameter format consists of 3 or 5 comma separated numbers: The frequency and the real and imaginary parts of the complex beam parameter (i.e. the distance from the waist and the Rayleigh length). For circular beams there are in total 3 parameters. For general astigmatic beams the complex beam parameter is given for the horizontal directions is followed by the complex beam parameter for the vertical direction resulting in 5 numbers for the beam description.

A Gaussian beam is generated with `gaussgen`. It always needs the frequency (`-freq`). The other parameters can are given in different combinations, depending on what is available. For example, one way is to give the waist radius w_0 (`-w0`) and the distance from the waist (`-z`). Another equivalent method is to give the beam radius $w(z)$ (`-wz`) and the curvature radius of the phase front (`-R`). These options generate circular beams. For astigmatic beams, the corresponding options are `-w0x`, `-w0y`, `-zx`, `-zy`, `-wzx`, `-wzy`, `-Rx` and `-Ry`.

One operation with a Gaussian beam is a propagation by a distance, which is done with the command `gaussprop`, which takes the distance as an argument and just changes the real part of the complex beam parameter. The other operation is the propagation through a thin lens with a focal length f, which is done by the command `gausslens`). The beam transformation is done via the ABCD-Matrix formalism.

Finally, all relevant parameters of the beam can be printed on the standard output with the command `gaussparams`.

With these tools, the following effects are neglected:

- Beam truncation from finite mirror sizes (use `fm_truncloss`)
- Non-perfect Gaussian beams (i.e. higher order modes, use field matrices)
- Mode conversion and generation of cross-polarised fields on focusing mirrors (use `fm_po_refl` from section 12)

The interfaces between the fieldmatrix code and the analytical Gaussian optics are built into the tools `fm_fit` and `fm_gen`. The option `-gauss` for the `fm_fit` command outputs the beam parameters as 3 or 5 numbers, which can further be processed by the analytical tools. For `fm_gen`, the option `-gauss` can be used to specify the Gaussian parameters. If a minus sign is passed to the `-gauss` option, the beam parameter is read from stdin making it compatible with the rest of the tools.

To fit a numerically given field distribution (in the file `field.fm`) to a Gaussian beam, propagate it by 0.1 m and convert it back to a field matrix, the following command can be used:

```
fm_fit -nopos -notilt -i field.fm -gauss | \
gaussprop -d 0.1 | \
fm_gen -gauss - -xy 200,-0.1,0.1 -o propagated.fm
```

The options `-nopos` and `-notilt` disable fitting of the transversal offsets and tilt angles.

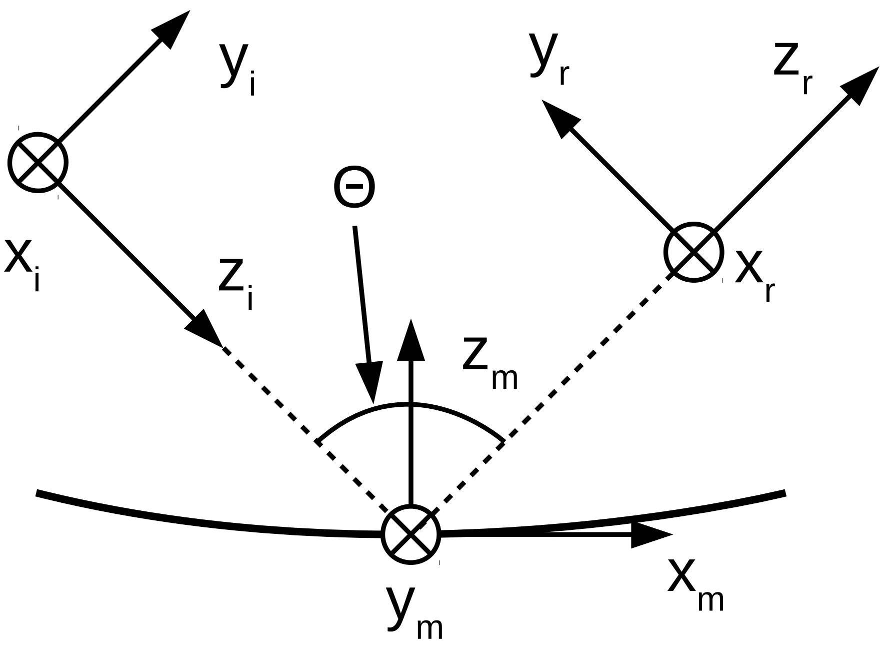

Figure 7: Coordinate systems for mirrors

11 Synthesis of mirrors

11.1 Focusing mirrors

For the synthesis of focusing mirrors, the method from [5] is implemented in the tool `gaussmirror`. It takes an astigmatic beam (defined by the waist radii and distances from the mirror origin) into account. The focal lengths can be given directly (`-f`, `-fx`, `-fy`) or in terms of the refractive power (`-r`, `-rx`, `-ry`). Another essential parameter is the angle Θ between the incident and reflected beams. Note that the synthesis method uses a higher order correction, which makes it, in the general case, non-symmetric with respect to the x-axis.

The resulting file is saved in a binary format (with preferred filename extension `.refl`). This file contains the size and sample counts of the mirror in x- and y-directions as well as the height data. To generate an ASCII-representation of the reflector, there is the command `refl_cat`.

11.2 Phase inverting mirrors

Another type of reflectors are phase-inverting mirrors. Mathematically they generate the conjugate complex of the incident beam. They can be synthesised from the phase-front curvature under consideration of the angle Θ (see Fig. 7).

Phase inverting (or phase reversal) mirrors are generated with the command `make_phaseinv`. It takes the field parameters of the incident field at the origin of the mirror coordinates. The file format of the resulting reflector surface the is same as for `gaussmirror` (See cap 11.1).

The program `make_phaseinv` assumes, that the (stretched) field pattern on the mirror is the same for all values of x in mirror coordinates (see Fig. 7) and that it is identical for then incident and reflected beams. In reality, this is not fulfilled because different x-positions on the mirrors correspond to different positions along the propagation axes of the incident and reflected beams. This mismatch leads to mode conversion.

11.3 Ellipsoids, cylinders and hyperboloids

Mirrors are often designed by other tools and specified in terms of surfaces such as ellipsoids, cylinders or hyperboloids. For the Physical optics tool (see section 12, page 13) we need the surface in terms of height profile in mirror local coordinates (see Fig. 8). The calculation of the z_m-coordinate for given values x_m, y_m is equivalent to the calculation of the intersection point of a line and the surface, which cannot to be solved analytically.

It is, however, possible to find an interval of two points, one above and one below the surface. An iterative algorithm then refines the interval until a specified accuracy is reached. This algorithm is available in the tool `gen_reflector`, which supports ellipsoids, cylinders and hyperboloids. It needs the type of the surface (`-t`), the centre point (`-center`), the directions of 1 or 2 axes (`-ax`, `-ay`, `-az`).

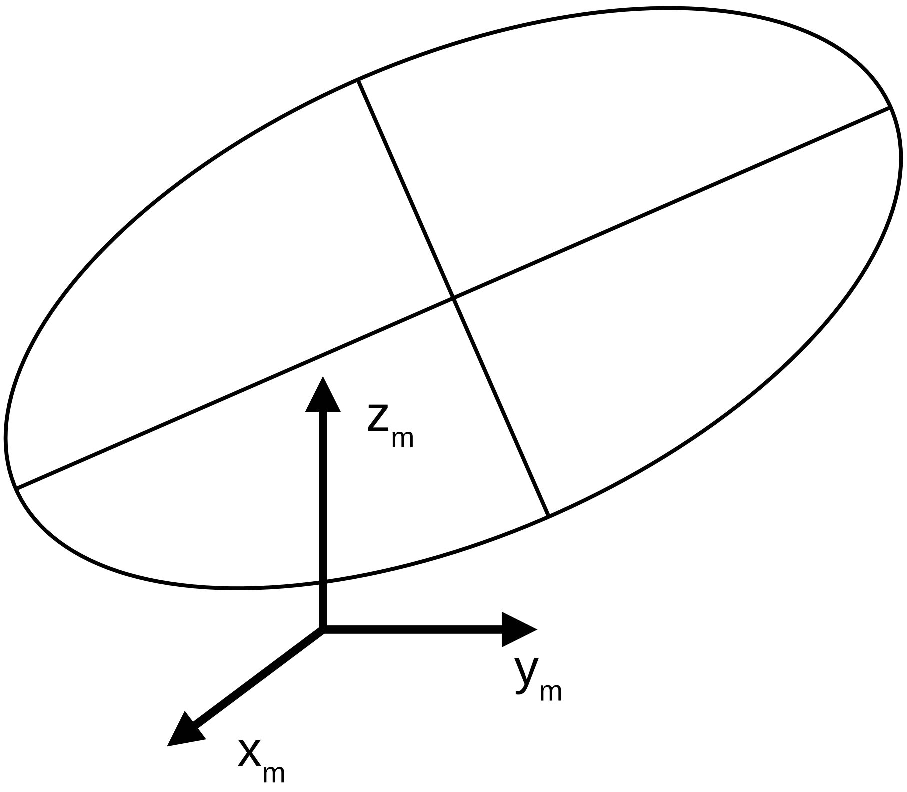

Figure 8: Example of an ellipsoid with a mirror coordinate system

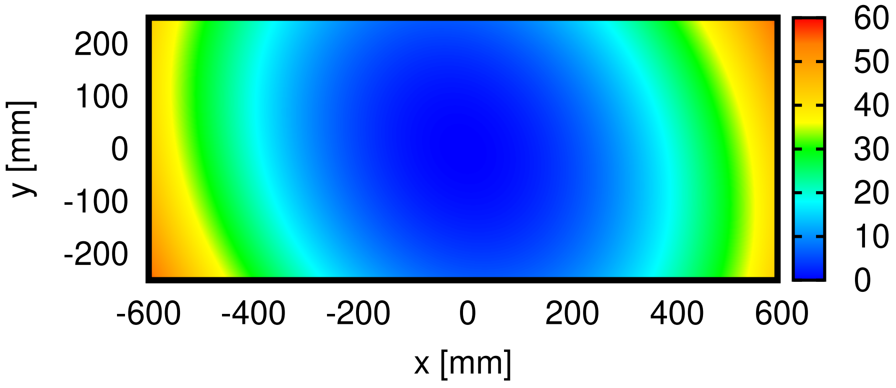

Figure 9: Example of a discretised ellipsoidal mirror surface

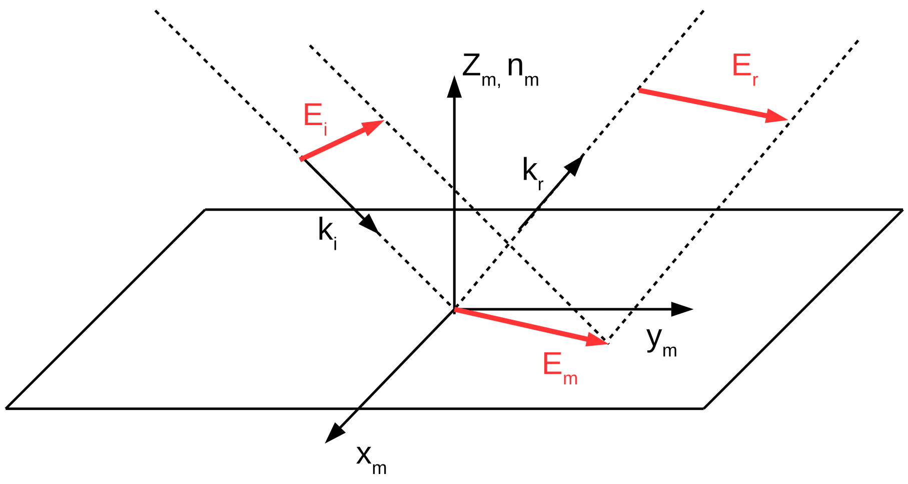

Figure 10: Polarisation vectors on a mirror

11.4 Polarisation tracking

In complicated optical transmission lines it is not always trivially possible to follow the polarisation vector through a sequence of mirrors. The involved vectors are shown in Fig. 10.

The rule for the polarisation transformation is, that the projection of the E-field along the k-vector onto the mirror surface results in a projected E-vector $\vec{E}_m$, which must be identical for the incident and reflected beam.

Mathematically, the direction of the E-field of the reflected beam can easily be calculated by introducing 2 planes P_i and P_r, which are defined by the k- and E-vectors of the incident and reflected beams respectively. A normal vector $\vec{n}_i$ of the incident plane P_i then becomes:

$$\vec{n}_i = \vec{k}_i \times \vec{E}_i \tag{4}$$

The direction of the projected E-field is then:

$$\vec{E}_m = \vec{n}_i \times \vec{n}_m \tag{5}$$

From this, we can obtain the normal vector $\vec{n}_r$ of the plane P_r:

$$\vec{n}_r = \vec{E}_m \times \vec{k}_r \tag{6}$$

The electric field $\vec{E}_r$ of the reflected beam is in the plane P_r and orthogonal to $\vec{k}_r$. It is thus given by:

$$\vec{E}_r = \vec{k}_r \times \vec{n}_r \tag{7}$$

Note that this derivation ignores the absolute values and dimensions. Furthermore, the polarisation angle is assumed to be in the range $[-\pi/2, \pi/2]$. This means that a polarisation angle of $\alpha + \pi$ is assumed to be equivalent to α.

For mirrors, the normal vector $\vec{n}_m$ can be calculated from the k-vectors of the incident and reflected fields using the reflection law:

$$\vec{n}_m = \frac{\vec{k}_r - \vec{k}_i}{|\vec{k}_r - \vec{k}_i|} \tag{8}$$

For gratings, the normal vector must be given separately. The tool, which does the transformation, is called `mirrorpol`. It allows to specify the field both in terms of a field vector in global coordinates as well as an angle with respect to beam-local coordinate systems.

12 Physical Optics tool

For the calculation of the reflection properties of metallic surfaces, there is the tool `fm_po_refl`. It uses a physical optics algorithm, which takes all effects (mode conversion, de-polarisation), which are neglected in the usual thin-lens approximations. It takes a reflector profile (as generated e.g. by `make_phaseinv` or `gaussmirror`).

It works in two steps: First the wall currents, which are induced by the incident beam, are calculated on the mirror surface:

$$\vec{J} = 2\vec{n} \times \vec{H} \tag{9}$$

where $\vec{H}$ is the magnetic field and $\vec{n}$ is the outward pointing normalised normal vector on the surface.

It is calculated for the surface elements with the step-sizes taken from the reflector files (i.e. no interpolation on the reflector surface is done). The radiated field is then obtained from the current elements by:

$$\vec{H} = \iint_{reflector} \left(-jk - \frac{1}{R}\right) \cdot \vec{J} \times \frac{\vec{R}}{R} \frac{e^{-jkR}}{4\pi R} dydx \tag{10}$$

Here, $\vec{R}$ is the vector from the mirror point to the point where the field is calculated, R is the distance.

For the input field we can use any field generator supplied by PROFUSION. It should, however, be noted, that not all field generators generate the accurate vectorial description at all field components (including the z-component) for arbitrary z-positions. Supported generators are the Gauss-generators (`gauss` and `gaussa`).

It is, however, always possible to generate a field matrix at the initial position and create a field grid (see cap 3), which was written for this purpose and allows fast interpolation of all field components in a given z-range using a pre-calculated grid.

For the calculation, the command needs:

- The coordinate system of the input beam (`-inx`, `-inz`, `-inpos`)
- The coordinate system of the reflected beam (`-outx`, `-outz`, `-outpos`)
- The coordinate system of the reflector (`-mx`, `-my`, `-mc`)
- The reflector profile (`-refl`)
- The size of the field matrix of the reflected beam (`-x`, `-y` or `-xy`)

For the simplified case of coordinates as in Fig. 7, it is sufficient to pass just the angle between the incident and reflected beams (`-Theta`) and the distances from the beam coordinates to the mirror (`-ind`, `-outd`).

13 Generic parallelization front-end

In many cases, e.g. when optimising components or doing multidimensional parameter scans, a task can be parallelised by running the same executable multiple times with different parameters (or, in the case of optimizations, different random seeds). They are much simpler than the general case because they don't need to exchange intermediate results between the worker processes or -threads and no synchronisation mechanisms are necessary. Instead, each job runs independently and when it is finished, the next job is started. The total number of jobs running simultaneously is ideally the number of CPU-cores in a single-user system. The scheduling, i.e. the distribution of the tasks among the available CPU-cores, is left to the operating system.

PROFUSION contains a simple job queuing mechanism for this category of parallelization tasks. It consists of a *command file*, which contains the commands to be run. To prevent conflicts, each command should run in a separate directory. Furthermore, the standard outputs and error outputs should be redirected to files to prevent having all messages from all worker programs at one console. If the actual calculations are in a shell-script `calculate.sh` and take one floating point parameter as an argument, the command file can look like:

```
  cd param_0.1; ../calculate.sh 0.1 > out.txt 2> err.txt
  cd param_0.2; ../calculate.sh 0.2 > out.txt 2> err.txt
  cd param_0.3; ../calculate.sh 0.3 > out.txt 2> err.txt
```

This will change to the subdirectories (`param_0.1`, `param_0.2` etc.) and run the command script by using the relative path. The directories need to be created beforehand. To run these commands in parallel, change to the directory, where `calculate.sh` and the subdirectories are located and call:

```
parallelize -cmd commands.txt
```

Here we assume, that the commands are in the file `commands.txt`. If the same directory is available on multiple machines via NFS, you can start the same `parallelize`-command on multiple machines simultaneously.

The running and finished jobs are tracked in a subdirectory `stamps`. When a job is starting, a file in `stamps` is created with zero-size and with the line-number from the command file (in hexadecimal representation) as filename. When the job is finished, one byte is written to that file. When you start to run a parallelised job in a directory, you *must* manually delete the `stamps`-directory if it is left over from previous runs. When you cancel a parallelization on one or more machines (by typing Ctrl+C in the terminal, where the `parallelize`-command is running), the corresponding stamp files are deleted. However, you need to run the `parallelize`-command again on at least one machine after cancelling the last one to ensure, that all jobs are completed.

If you want a summary of the progress of the task, you can call:

```
parallelize -cmd commands.txt -stats
```

It also prints an estimate of the time, when the last job is finished. This estimation is, however, not very accurate because it assumes, that all jobs need the same time to complete. Furthermore it can only give an estimate if at least one job is finished.

To prevent race-conditions when running a parallelised task across multiple machines, the network file system must support the `open()`-systemcall with the flag `O_EXCL` as an atomic operation. This is the case e.g. for the NFS implementations in recent Linux kernels.

14 Evaluator of mathematical expressions

PROFUSION-tools are normally used on the commandline, e.g. on Linux systems with the bash-Shell. In this case, it is useful to have a method for evaluating mathematical expressions on the commandline. There are several standard tools (`bc`, `awk`), which can be used for this. The most powerful however, is Python. It provides the `eval()`-Function to evaluate Python expressions.

The tool `pf-eval` takes a single argument, which is a mathematical expression. It must result in a single number. The expression should be enclosed in double quotes to allow spaces and shell variables but avoid further interpretation of special characters (such as `*`), which have a special meaning in the shell.

Within the mathematical expressions, all functions of the Python `math`-module can be used. In addition, it supports the following trigonometric functions for angles given in degrees: `sind`, `cosd`, `tand`, `asind`, `acosd`, `atand`, `atan2d`.

Furthermore, the following constants are available: `PHYS_c`, `PHYS_mu`, `PHYS_epsilon`, `PHYS_Z`.

15 Examples

15.1 Losses of a waveguide gap

This example calculates the losses due to a gap in an oversized corrugated waveguide carrying an HE_{11}-Mode. The HE_{11}-Mode is known to have a small divergence when radiating from an open-ended waveguide, provided that the waveguide diameter is much larger than the wavelength, which is usually fulfilled in HE_{11}-Waveguides.

Gaps can be used as simple and efficient filters for higher-order modes or as a DC-Break to isolate the waveguide electrically. For the calculation of the losses, we distinguish 2 quantities: The first is the power, which is actually lost because it will not be inside the waveguide aperture due to the divergence of the free-space beam in the gap. The second part will be converted into some higher-order modes, like the HE_{12} or HE_{13}, which continue to propagate in the waveguide as spurious modes.

The following script does the whole calculation:

```
#!/bin/sh
# Force decimal dot even for non-English systems
export LC_NUMERIC=C

# Generate HE11 field
fm_gen -f corr,corr.wg,,31.75e-3 -xy 1000,-0.08,0.08 \
  -freq 170e9 -o gap_start.fm

# Loop over gap widths
for i in ‘seq 0.001 0.001 0.500‘; do

  # Propagate (gap_start.fm -> gap_end.fm)
  fm_prop -i gap_start.fm -d $i -pad 0.1 -o gap_end.fm

  # Calculate the truncation loss
  TRUNCLOSS=‘fm_truncloss -s 63.5e-3 -i gap_end.fm | \
    grep -v ’^#’ | awk ’{ print $3 }’‘

  # Calculate the excited HE11 mode after the gap
  cyl_corr_analysis -wg corr.wg -r 31.75e-3 -freq 170e9 \
    -o ampl.dat -f fm,gap_end.fm

  # Format the amplitude for Writing to the file
  HE11AMPL=‘dump_ampl -i ampl.dat | \
    grep -v ’^#’ | \
    head -n 1 | \
    awk ’{ print 1.0-($1/100) }’‘

  # Write output
  echo "$i $TRUNCLOSS $HE11AMPL"

# End of the loop
done
```

The waveguide diameter is 63.5 mm, the frequency is 170 GHz. The field in the waveguide cross-section is generated with the `fm_gen` command. We choose an area of 160×160 mm^2, which should be large enough to also contain the power, which leaves the waveguide through the gap. This is necessary because to obtain a high accuracy, the truncation losses are calculated by integrating the field *outside* of the aperture.

The loop over the gap is done using the `seq`-command, which is available on Linux systems and generates a sequence of equidistant numbers. Within the loop, the field is first propagated with `fm_prop`. Then, the truncation losses are calculated by integrating the power, which lies outside of the waveguide cross section after the propagation.

Finally, we call `cyl_corr_analysis`, which expands the free space field at the end of the gap into waveguide modes. We are only interested in the HE_{11}-Model, but the amplitudes of the higher-order modes could be calculated as well by adding them to the waveguide definition file.

The losses have been discussed in literature ([4], [6]), where analytical formulas were derived for typical waveguide modes. In the case of the HE_{11}-mode, the loss can be approximated with:

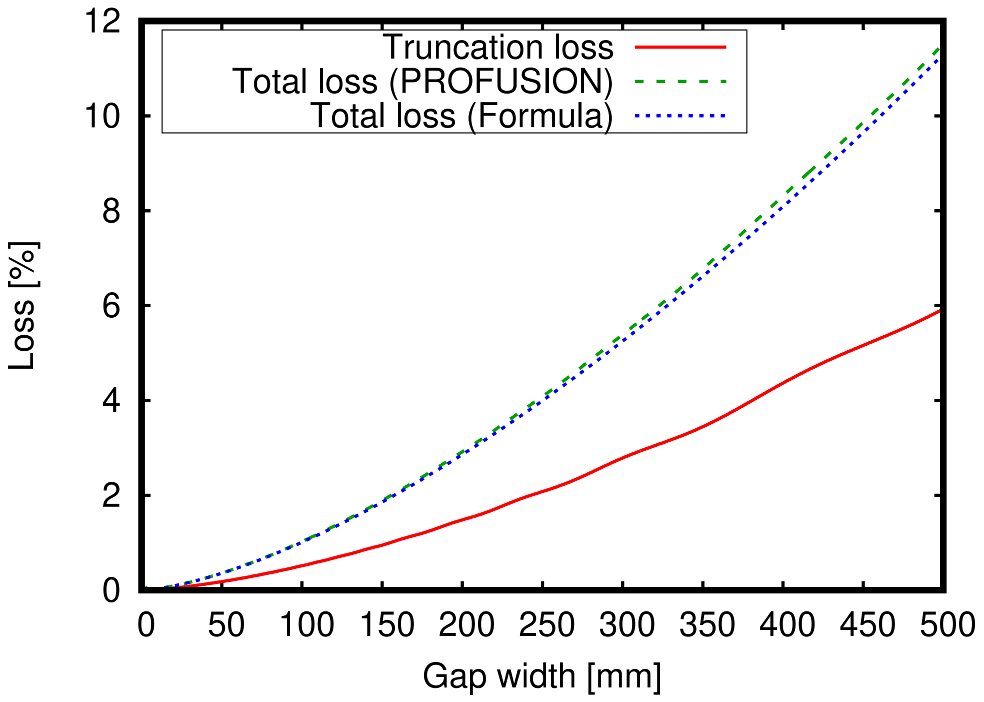

Figure 11: Losses of a gap in am HE_{11}-Waveguide as a function of the gap width

$$1 - |A_{\mathrm{HE}_{11}}|^2 = \frac{2}{\sqrt{\pi}} \frac{X_{\mathrm{HE}_{11}}^2}{3} \left(\frac{L}{k_0 a^2} \right)^{\frac{3}{2}} \tag{11}$$

Here, $X_{\mathrm{HE}_{11}}$ is the Eigenvalue (2.405), a is the waveguide radius, k_0 is the free space wavenumber and L is the width of the gap. Figure 11 shows the results. We can see, that the numerically calculated total loss agrees very well with the prediction from the formula.

Note that the power is not re-normalised before coupling into the waveguide. This means all losses are relative to the total power of the HE_{11}-mode *before* the gap.

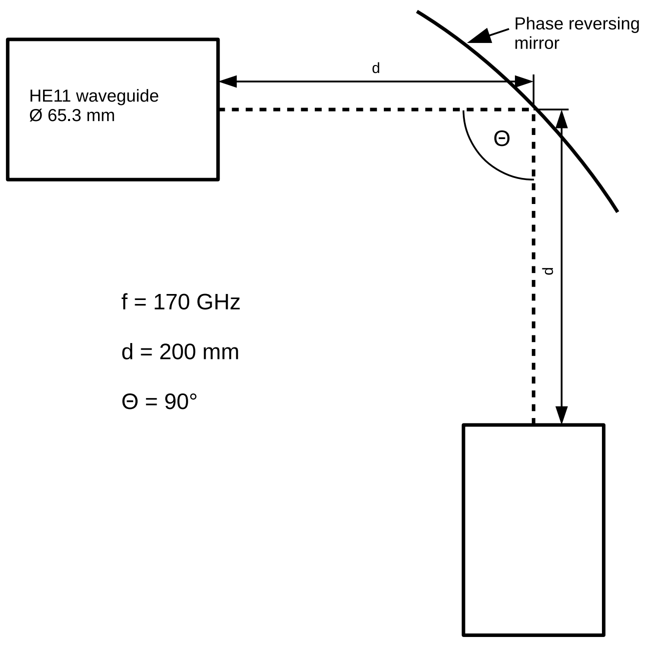

Figure 12: Setup for a Quasi-optical mitre-bend

15.2 Phase inverting mirror

A phase inverting mirror can be used e.g. in a quasi-optical mitrebend. This approach has the advantage, that the beam divergence due to the free-space can be compensated. Furthermore, is does not suffer from the partly wrong corrugation orientation in conventional mitre bends and can, in addition, act as a DC-break between input and output. An example setup is shown in Fig. 12.
For generating the mirror, we first need the field, which is incident on the mirror. This is done by first generating a the HE11-pattern at the waveguide output:

```
fm_gen -freq 170e9 -f j0,31.75e-3 -xy 500,-0.068,0.068 > HE11.fm
```

Next we need to propagate the field by 200 mm to the location of the mirror:

```
fm_prop -i HE11.fm -pad 0.1 -d 0.2 > HE11-prop.fm
```

The mirror surface is then generated with:

```
make_phaseinv -f fm,HE11-prop.fm -x 750,-0.085,0.085 -y 500,-0.06,0.06 -o reflector.refl
```

The profiles of the mirrors in *x*- and *y*-directions can be obtained with:

```
refl_profile -i reflector.refl -pre reflector
```

The resulting height profile is shown in Fig. 13, the cross-sections along the *x* and *y*-axes are shown in Fig. 14.
The reflected fields in a distance of 200 mm from the reflector are calculated with the physical-optics tool `fm_po_refl`. The command is:

```
fm_po_refl -ind 0.2 -outd 0.2 -f fg,HE11.fm,0.4,1.0,0.1 -xy 200,-0.065,0.065 \
  -o HE11-refl.fm -refl reflector.refl -Theta 90.0
```

The result is shown in Fig. 15. To obtain the mode purity the field can be re-normalised in order to neglect the radiation outside the mirror and outside the destination field matrix. A command, which calculates the mode purity under consideration of an ideal HE_{11}-distribution, but allows a tilt-angle and an transversal offset to be considered, is:

```
fm_norm -exact -i HE11-refl.fm | fm_fit -f j0,31.75e-3
```

The resulting offsets are below $1 \cdot 10^{-5}$ m, the fitted tilt angles are smaller than $0.0001°$. The mode purity is 99.89 %.

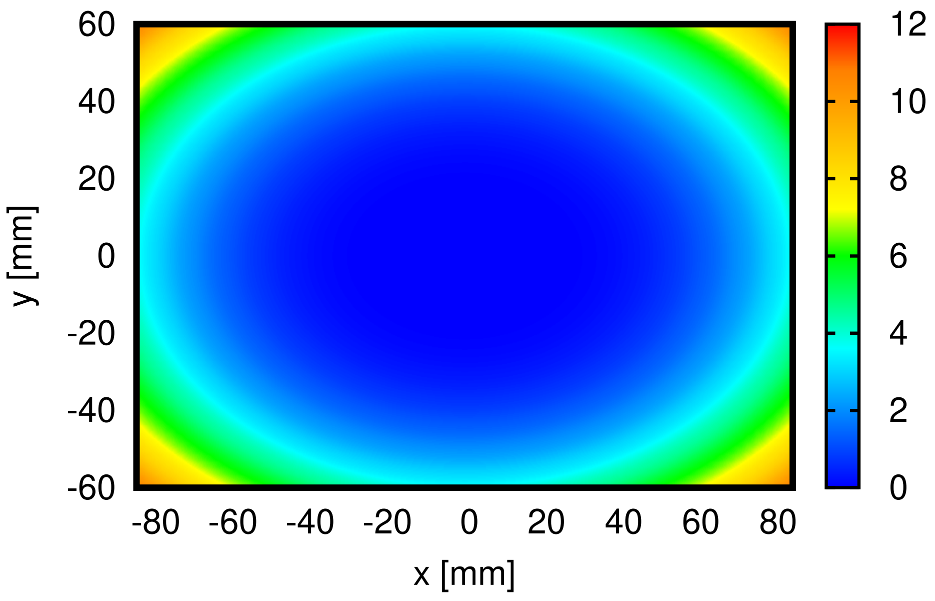

Figure 13: Height profile of the phase inverting mirror.

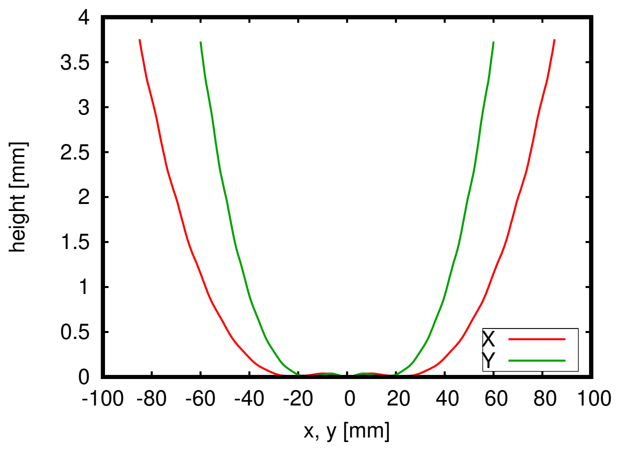

Figure 14: Cross-sections along the *x*- and *y*-Axes of the reflector.

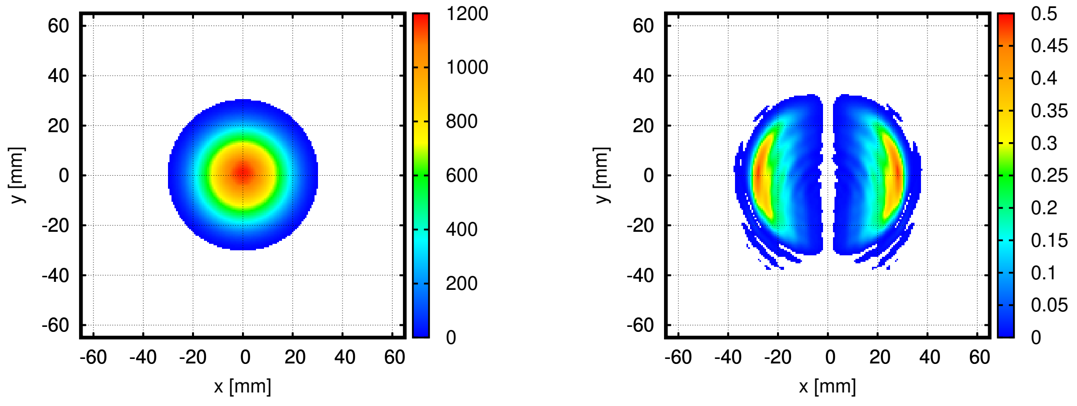

Figure 15: Fields of the reflected beam in Co- (left) and Cross-polarisation (right)

16 Conclusions

The PROFUSION package contains a versatile set of tools for the calculation of free space microwave beams. The realisation, which consists of a number of commandline programs, is easy to learn for users with experience on the Unix commandline. The strengths in comparison with GUI tools are the versatility because the programs can be combined in any way, and the inherent scriptability. PROFUSION is used in many ongoing projects and the code is under permanent maintenance. Also, new features are permanently added.

Numerous comparisons were made to compare the results with measurements, analytical formulas and other code packages, and the agreement was always very good.

This work is supported by the Max Planck Institute for Plasma Physics, Garching and Greifswald/Germany.

A File formats

Configuration files are in a simple ASCII format, which can be edited with any text editor. All file formats have a signature as the first line, which specifies the files type. It prevents accidental usage errors and confusions. Most formats contain a simple set of name/variable pairs. Sectioning or other hierarchical structures are not supported. Below is an example for a hypothetical format:

```
SIGNATURE
Var1: val1
Var2: 1.0
Var3: 1
Var4: 100,0.2,0.5
```

The variable names are followed by a colon, after the colon we can have an arbitrary amount of whitespace. The first non-space character up to the end of the line is the value. The hash-character ('#') starts a comment, which extends to the end of the line.

A.1 Waveguide descriptions

Waveguide description follow the format described above, except that the "Mode" variable can be used multiple times to add multiple waveguide modes. The first mode is considered the "main mode". For commands, which need a spectrum of mode amplitudes in a separate file (see section A.2), the default is to assume 100% of the power and zero phase for the main mode and zero power for all others.

A.1.1 Circular waveguide (smooth wall)

An example for a waveguide configuration is shown below. The file signature is "`WAVEGUIDE_CYL`". The supported variables are:

- `Radius`: Waveguide radius in meters. Can be overridden at the commandline for many programs
- `Material`: Wall material defined by 3 numbers separated by spaces: The specific resistance (in Ωm), the relative magnetic permeability μ_r and a factor R_c which takes the surface roughness into account. The default values (`0.0 1.0 1.0`) are for a PEC waveguide wall.
- `Frequency`: 3 Values (number of points, minimum, maximum). Alternatively, for single frequency setups, the Frequency can be given as a single number

Finally, a number of modes can be given with the "Mode" keyword. It contains a string specifying the mode type, followed by two indices for corresponding the transversal coordinates.

```
WAVEGUIDE_CYL
# Radius (in m)
Radius: 19.0e-03
# Material: rho mu_r R_c
Material: 0.000000000e+00 1.000000000e+00 1.000000000e+00
# Frequency: num_points min max
Frequency: 5 70.0e+9 110.0e9
Mode: TE 1 1
Mode: TE 1 2
Mode: TE 1 3
Mode: TM 1 1
Mode: TM 1 2
Mode: TM 1 3
```

A.1.2 Rectangular waveguide (smooth wall)

The format for rectangular waveguide is identical to the cylindrical waveguide with the exception, that the file signature is "`WAVEGUIDE_RECT`" and the "`Radius`" variable is replaced by "`Width`" and "`Height`", which specify the inner dimensions of the waveguide.

Furthermore, the variable "`Pairs`" specifies, that $\mathrm{TE}_{\mathrm{mn}}$ and $\mathrm{TM}_{\mathrm{mn}}$ (m, $m > 0$) modes are combined into pairs, which are linearly polarised. It is a Boolean value, which can be 0 or 1. These mode pairs can be specified by either TM or TE. One application for mode pairs are E-plane bends, which are fed by a TE_{10}-Mode.

A.1.3 Circular waveguide (corrugated wall)

The format for rectangular waveguide is identical to the cylindrical waveguide except that the file signature is "`WAVEGUIDE_CYLCORR`". An additional variable "`WP`" specifies the ratio of the groove width and the groove period of the corrugation. It can in most cases be set to 0.5.

A.1.4 Square waveguide (corrugated wall)

The square corrugated waveguide module handles only the simple case of linearly polarised balanced HE_{mn}-Modes (corrugation depth: $\lambda/4$) in the square waveguide. Therefore, the dimensions are given just by "`Width`". The file signature is "`WAVEGUIDE_SQC`".

A.2 Mode amplitudes

Mode amplitudes are saved in separate files. It has the signature "`AMPLITUDES`", followed by a single variable "`Num`", which specifies the number of mode amplitudes to follow. The remaining lines contain amplitudes in the complex format (real and imaginary part, separated by a space). Mode amplitudes are always defined such that an amplitude of 1 corresponds to unity power. For most applications, the power sum of all modes should be 1.

You can generate a PROFUSION compliant amplitude file from a simpler format, which contains no header and the mode powers (in percent) and the phase (in degrees) with the `make_ampl` command. For the example of a 95%/5% mode mixture, with a phase shift of 90°, one can generate a file:

```
95.0 0.0
5.0 90.0
```

and with `make_ampl` the resulting amplitude file will be:

```
AMPLITUDES
Num: 2
9.746794345e-01 0.000000000e+00
1.369196746e-17 2.236067977e-01
```

The real part of result for the second mode is nonzero due to some rounding errors of the square root and cosine and sine functions. To display the contents of the mode amplitudes, use the tool `dump_ampl`

References

[1] Goodman, J.W.: Introduction to Fourier Optics. Physical and Quantum Electronics Series. McGraw-Hill (1968)

[2] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C, 2 edn. Cambridge University Press (1992)

[3] Self, S.A.: Focusing of spherical gaussian beams. Appl. Opt. **22**(5), 658–661 (1983). DOI 10.1364/AO.22.000658. URL `http://ao.osa.org/abstract.cfm?URI=ao-22-5-658`

[4] Vaganov, R.B.: Diffraction of asymmetric waves by a wide slit in a circular waveguide. Radiophysics and Quantum Electronics **12**, 504–506 (1969). DOI 10.1007/BF01037026

[5] Vinogradov, D.: Mirror conversion of gaussian beams with simple astigmatism. International Journal of Infrared and Millimeter Waves **16**(11), 1945–1963 (1995). DOI 10.1007/BF02072550. URL `https://doi.org/10.1007/BF02072550`

[6] Wagner, D., Thumm, M., Kasparek, W., Müller, G.A., Braz, O.: Prediction of TE-, TM-, and hybridmode transmission losses in gaps of oversized waveguides using a scattering matrix code. International Journal of Infrared and Millimeter Waves **17**(6), 1071–1081 (1996). DOI 10.1007/BF02101439. URL `https://doi.org/10.1007/BF02101439`