guide_bot

guide_bot is a collection of MATLAB scripts and functions with the purpose of making it trivial to optimize neutron guide geometries using McStas and iFit. The basic principle is simple, the user tells guide_bot the most basic informations about the wanted beam on the sample and the overall layout of the guide to be optimized. The output is then given as McStas instrument files, iFit MATLAB scripts and bash scripts in a neat folder. This folder is then uploaded to a cluster (DMSC ESSS or mcc PSI), which will do the optimizations. When the work is done, an output folder can be downloaded which will contain the results and iFit MATLAB scripts to interpret them.

1 Installation

guide_bot is meant to run on Unix like systems, but does run on windows with a few additional steps. As guide_bot relies on other programs, these must be installed in order to have full functionality.

1.1 Dependencies

MATLAB is a necessity and needs a license which is generally available on larger research centers. The MATLAB add-on iFit is needed to do optimizations locally and visualize the results. The add-on is free and can be obtained from `http://ifit.mccode.org`, the most appropriate installation form is to download the source package and add the path in MATLAB. McStas is not essential if the user has an account on a supported cluster, but if it is needed to run optimizations locally it should be installed. Follow the directions from `http://mcstas.org` if needed. When running locally, unix is a necessity as iFit does not support MPI on Windows.

1.2 Obtaining guide_bot

The latest version of guide_bot can be downloaded from the guide_bot github repository, `https://github.com/mads-bertelsen/guide_bot`.

2 Basic use

2.1 Running the MATLAB script

Using the program is simple, open MATLAB and navigate to the folder named "guide_bot" which was downloaded. This folder contains the scripts and functions which make up guide_bot, but the user should only edit the "guide_bot_user_template.m" file. The name of this file is irrelevant, and the idea is that it should be copied and renamed for each new project. It is however a good starting example that have comments which explain each parameter and the necessary input. After copying the file, try to run the script in MATLAB, which should create a new folder named after the project name specified. All that is needed to follow the next section is such a project folder, but if you want to run a project more relevant to your problems, skip to section 3 and learn to to edit the input file before following 2.2.

2.2 Running the optimizations

The optimizations can in principle be run on a home computer, but as they are computationally heavy it is recommended to do the work on a cluster. Running locally is explained later in the section. To move the project folder to the cluster, simply use scp as follows:

```
scp -r project_folder username@login.esss.dk:/users/username/.
```

This will need your password for the cluster. To start the optimization it is necessary to ssh to the cluster as normal:

```
ssh -X username@login.esss.dk
```

We then navigate to the project folder.

```
cd /path_to/project_folder
```

Now we need to compile the McStas files, and for that some clusters need to load a module. For DMSC at ESSS this is done as follows:

```
module load mcstas/2.3
```

While for the mcc at PSI the syntax is:

```
module load mcstas/mcstas2.3
```

Now we can easily compile all the generated McStas instruments with the command:

```
./compile_all.sh
```

If the instrument files fail to compile, the mpi default might not be correct for McStas. The command mpi-selector is used to select the default, currently "openmpi_intel_qlc-1.4.3" is preferred. It can be chosen with the command below, but is only necessary if compilation fails.

```
mpi-selector --set openmpi_intel_qlc-1.4.3
```

Now run the compile script again, and if there are still errors contact the developer at mads.bertelsen@gmail.com.

When the instrument files have successfully compiled, all that remains is the following command:

```
./launch_all.sh
```

This will launch all the simulations to the selected queue on the cluster. The squeue command can be used to monitor the progress. When all the jobs have finished, the results can be downloaded to your own computer for visualization. Not all the data is necessary however, only the output folder is needed. On your home computer the folder can be downloaded with this command:

```
scp -r username@login.esss.dk:/users/username/project_folder/output .
```

If the simulations should be run on a local computer one needs to enter each folder for the different guides and run the ifit file(s), remember to also run the brilliance reference ifit file located in a separate folder.

Note that the folders for each instrument (not downloaded) have data in the normal McStas format, which can be viewed with mcplot, and that there is a parameter file ending in .par which can be used to visualize the guide geometry by using mcdisplay. The McStas files can also be edited as the Trace section is readable, but if changes to the initialize section are needed, add these at the end of the current initialize section to avoid problems.

2.3 Plotting the results

Inside the output folder, there is a folder called analysis. Open MATLAB and make sure iFit is loaded in the path. Navigate to the analysis folder and then each optimized guide will have a script for visualizing the results. Just run the analyse_all.m script to run all the scripts. The plots will be saved as png files in the same folder. It is possible to run the scan_plotter.m file instead (located in the root of the installation directory), which will make graphs describing the scan, or compare the simulated guides if a scan was not made.

3 The input file

The input file often named "guide_bot_user_something.m" is used to control what beam the optimizer will try to achieve and which guide geometries will be investigated. This is done by setting a list of demands and requirements and then write a series of geometries. The demands and requirements are in table 1 and 2 respectively. These are all necessary and can not be left blank. The demands specify the wanted beam properties at the sample and basic information on the overall layout. The requirements consist of more external factors which is more or less constant for each neutron facility.

The requirement.source is used to select between different moderator descriptions which is described in 3.1. Further details on the options can be found in section 3.4.

A last small set of options is simply called options, and contains controls which does not fit with demands or requirements. These are listed in table 3. Only the ones with the required keyword needs to be specified in the file, while the others are optional. The options.absolute_intensity_run is currently bugged, as an error will occur if it is set to 0.

Variable name	Explanation	Unit
demands.Hdiv	Horizontal divergence (half width)	Degrees
demands.Vdiv	Vertical divergence (half width)	Degrees
demands.Hsize	Horizontal sample size	cm
demands.Vsize	Vertical sample size	cm
demands.WaveLmin	Minimum wavelength to be accepted	Å
demands.WaveLmax	Maximum wavelength to be accepted	Å
demands.Dist	Distance between guide end and sample	m
demands.Mod_sample	Distance between moderator and sample	m

Table 1: The input used to specify the desired beam properties at the sample position. In the code, these are called demands. Notice the divergence is half width, so 0.5 deg will be interpreted as +/- 0.5 deg divergence figure of merit.

Variable name	Description	Unit
requirements.closest_element	Shortest possible distance between moderator and guide start	m
requirements.latest_start	Longest possible distance between moderator and guide start	m
requirements.moderator_size_x	Horizontal moderator size	m
requirements.moderator_size_y	Vertical moderator size	m
requirements.source	ESS source for absolute intensities ('cold','thermal','bispectral)	

Table 2: Options used to specify external requirements imposed on the guide from the facility. In the code, these are called requirements.

Variable name	Description		
options.absolute_intensity_run	Enables detailed absolute intensity run	required	[1]
options.minimalist	Enable the minimalist principle	required	[1,0,-1]
options.focusing	Assumed focusing guide	optional	1(yes),0(no)
options.phase_space_requirement	Phase-space volume used	optional	'total','homogen'
options.minimalist_factor	Ratio between necessary and received PS	optional	number
options.minimalist_direction	Controls if 'both', 'horizontal' or 'vertical'	optional	string
options.gravity	Enable or disable gravity in simulations	optional	[1,0]

Table 3: Options that did not fit in the requirement or demand category. These are detailed further in section 3.4

3.1 The source descriptions

guide_bot will always optimize a guide on a simple rectangular source use such for calculating brilliance transfer, but can use more complex sources to give a set of output in absolute flux. In table 4 the different possibilities for the requirements.source input are listed. The bispectral pancake source is not truly a bispectral guide solution, but is just the maximum of the cold and thermal spectrum for every lambda, which resembles a perfect bispectral switch.

Some of the sources have a few additional options, these are listed in table 5. The implementation of the Virtual_in source is basic and may be improved in the future. For now the input filename is currently hardcoded to be guide_bot_virtual_in.txt and should be placed in the components folder before a run. All files in the component folder is copied to every guide folder, and it is thus important to remove this file after runs with the virtual source have been made, as these large files will otherwise be copied to all future guide folders.

3.2 The input string(s)

Next the input strings defining the geometry for each guide is written. If more than one is specified, they are all optimized. Just use this syntax to keep it simple. In MATLAB terms, the input variable is a cell array of strings.

```
input{1}='string1'
input{end+1}='string2'
```

Requirements.source	Description	Component
TDR_cold	ESS TDR moderator, aimed at cold part	ESS_moderator_long
TDR_thermal	ESS TDR moderator, aimed at thermal part	ESS_moderator_long
TDR_bispectral	ESS TDR moderator, aimed at cold/thermal boundery	ESS_moderator_long
cold_pancake	ESS cold pancake moderator spectrum, rectangular	Sourge_gen
thermal_pancake	ESS thermal pancake moderator spectrum, rectangular	Sourge_gen
bispectral_pancake	ESS bispectral pancake moderator spectrum, rectangular	Sourge_gen
Butterfly	ESS butterfly, both thermal and cold part	ESS_butterfly
Butterfly_only_cold	ESS butterfly, only cold part	ESS_butterfly
Butterfly_only_thermal	ESS butterfly, only thermal part	ESS_butterfly
Virtual_in	Source for using virtual input file	Virtual_input
Source_gen	Source for using custom spectrum	Source_gen

Table 4: List of possibilities for the requirements.source parameter with description of their effect and which McStas component is used.

Option name	Description	Possible values
	For TDR soruces	
beamport	sets the angle between perpendicular to moderator and beamport	-30:30 deg
	For butterfly sources	
butterfly_face	Moderator face viewed as compas directions	"N" "E" "W" "S"
beam_line	Selected beamport number	1-10 / 1-11
	For Source_gen	
source_gen_file	filename for custom spectrum	filename
	For Virtual_in	
virtual_in_repeat_optimize	Number of times virtual in file repeated in optimization	number, default 1
virtual_in_repeat_analyze	Number of times virtual in file repeated in analysis	number, default 10

Table 5: Additional options for a few sources.

A string needs to contain a series of modules with a space between each. The possible letters are listed in table 6. The following code will optimize two guides:

```
input{1}='S C S'
input{end+1}='P S C S'
```

The first, S C S, will generate a folder named SCS and consist of a straight guide followed by a curved guide and then a straight guide. The second will likewise generate a folder called P S C S and make a more complicated guide with a parabolic section into a straight guide followed by a curved section and finally a straight section. Note that it is required that all strings start and end with one of the guide modules, S, P or E.

Each module can be controlled further by some options which are inserted in parenthesis after the letter. These options are outlined in table 7. Each module can have multiple options, just separate with ",". We could

Module name	Description	Special options
S	Straight guide	reflectivity
E	Elliptic guide	reflectivity
P	Parabolic guide	reflectivity
G	Gap	none
K	Kink	kink angle, kink direction
C	Curved guide	reflectivity, angle, direction
Selene	Selene guide	reflectivity, slit system
Slit	Slit	none
M	Monochromator	dedicated manual

Table 6: The available modules of which an input string describing a guide can be assembled. The special options are possible options in addition to the general options listed in table 7.

adjust the previous example with some options.

```
input{1}='S(length=5) C S'
input{end+1}='P S(maxStartWidth=0.03) C S'
```

This will keep the length of the first straight section to 5 meters for the first guide, and keep the entrance width below 3 cm for the first straight section in the second guide. If the straight guide should not start before 7 m after the moderator, the options could be written as

```
input{end+1}='P S(maxStartWidth=0.03,minstart=7) C S'
```

It is not possible to use the minStart and maxStart parameters for the first module in the input string, since it is effectively controlled by closest_start and latest_start. It is however possible to just use start to set a specific starting point for the guide. It is a known bug that setting the length of the first module can fail to register, in that case set the start of the second module which is equivalent, and will work.

Option	Description
length	The length of the module
start	Start point of the module (measured from the moderator)
StartWidth	Width of the start of the module
StartHeigth	Height of the start of the module
EndWidth	Width of the end of the module
EndHeight	Height of the end of the module

Table 7: The general options which are available for every module, which can all be modified to be limits for the optimization by adding max or min in front of the name for a maximum or minimum limit. If any are in conflict with what can be calculated from the minimalist concept, they are ignored, unless the minimalist concept is disabled.

3.3 Line of sight

It is possible to create guides which are out of line of sight. To do this, simply include a curved guide (C) or a kink (K) in the input string. It will automatically be calculated how much these needs to bend in order to escape line of sight, unless an angle is manually selected by using the rot option (explained further in the 4). In order to control the calculation of line of sight, there are a few specific options for this showed in table 8. Without using these options, line of sight is understood as sight through the entire guide, but using these options, this can be changed. These options can be used on the S E P and C modules, for example as shown below.

```
input{1}='S(los_start=0.5) C S(los_end=0.5)'
```

This would start the line of sight section halfway through the first straight guide segment, and end it halfway through the last straight segment. This would cause the curved guide to bend more than if the line of sight was simply from the start to the end. The option needs to be applied to a specific position in the guide, here 0.5 indicates halfway through the guide. Table 9 shows the three different ways to select this position. The three different ways to do this gives great freedom in setting up the line of sight section. It is even possible to have more than one line of sight section, as shown in this next example.

```
input{1}='S(los_start=5m_s) C S(los_divide=0.5) K  S(los_end=2m_e)'
```

Option	Description
los_start	Will start a los section
los_end	Will end a los section
los_divide	Will end the current los section and start a new

Table 8: The options used to specify intervals in the guide which should be used to escape line of sight.

Option	Description
=x	x times the length of the guide from the start [0:1]
=xm_s	x meters from the start of the guide
=xm_e	x meters from the end of the guide

Table 9: The ways to specify the los position within a module.

In this example there is two line of sight sections, one from 5 meters within the first straight guide (S) to half way through the second straight guide. In this line of sight section, there is a curved guide module (C) which will take care of eliminating line of sight between these two positions. The second line of sight sections is from the half way point of the second straight guide where line of sight was first escaped and to 2 meters before the end of the last straight guide. In this section, there is a kink (K) which will eliminate line of sight. Notice that there needs to be a line of sight breaker in each line of sight section. guide_bot will help to check this in the output to the MATLAB window. For the last example, it will look like this:

```
S C S K S
---
    -----
```

This tries to show the two line of sight sections. Also notice, that in cases such as the example below, it is assumed that the line of sight end will be the end of the guide as it is not actually specified, meaning that the two strings below will give the same guide_bot run. This also works for the start of the guide, so if nothing is specified it goes back to the standard of escaping line of sight from start to finish for the guide.

```
input{1}='S(los_start=0.5) C S'
input{end+1}='S(los_start=0.5) C S(los_end=1)'
```

It is possible to have several line of sight breakers in one line of sight section, they will simply help each other break line of sight. The example below is thus valid.

```
input{1}='S(los_start=0.5) C S C S (los_end=2m_e)'
```

The two curved guides need to curve in the same plane and in the same direction for each line of sight section, which is controled by the rots and rotd options which are detailed in the 4 section under the appropriate module. One guide can however have more line of sight sections in different directions, as in this example:

```
input{1}='S C(los_divide=1) C(rots=-1) S'
```

This will give an "s-shaped" guide, which is two times out of line of sight. It is possible to have more than one los option in one module, and at that point the order is important. The example below should demonstrate this.

```
input{1}='S C(los_start=5m_s,los_end=5m_e) S'
input{end+1}='S C(los_end=5m_s,los_start=5m_e) S'
```

The first guide have one line of sight section, where the curved guide will curve so much that there is not line of sight from 5 meters from the start to 5 meters from the end. In the second example, there are two unclosed line of sight sections, so a los_start=0 and los_end=1 will be inserted to the first and last guide respectively. That means the curved guide will have to curve so much that there is no line of sight from the start of the guide to 5 meters into the curved guide and no line of sight from 5 meters before the end to the end of the guide.

One last limitation of this system is that it can not include the Selene guide in a line of sight section, but as line of sight is blocked by Selene anyway that is not very relevant. In addition, it is not possible to make overlapping line of sight sections, which should be obvious from the way the input system works.

3.4 Details on options

This section have some further details about the options, and how they affect the guide design. The minimalist option controls a phase space transport approximation, which can be used to decrease the number of optimized parameters by calculating some of them from these equations. It basicly assumes that even more complicated

guides like the ellipse behaves as a straight guide by reshaping the phase-space volume in the simplest way imaginable. It also limits the ingoing phase-space volume to the phase-space-volume needed at the sample plus what is lost by gaps and the distance between the end of the guide and the sample. The result of each setting is described in 10. Regardless of the value set, the Minimalist Concept is still used to calculate the width of the end of the guide, this can however be changed using an option in the geometry string called Optimize_end. It is also possible to enable and disable the minimalist principle separately for the horizontal and vertical direction using the minimalist_direction option, which can be set to 'both', 'horizontal' or 'vertical'.

options.minimalist	effect
1	Calculates as much as possible, limits phase-space intake of the guide.
0	Calculates everything except the extraction, does not limit the phase-space intake.
-1	Only calculates the exit dimensions of the last guide element and the kink module.

Table 10: The effect of different options.minimalist values.

The options.absolute_intensity_run will as mentioned add figures which describes the guides performance on the ESS source, and currently has to be set to 1. This uses the requirements.source string.

The standard is the ESS TDR moderator. In the case of cold, the guide will be pointed to the center of the cold moderator. In case of thermal, the guide is pointed to the center of the thermal wing. If bispectral is selected, the guide will simply be pointed right at the boarder between the thermal and cold moderator. There is a options.beamport which can control what beamport is used, default is zero which is the center beamport. Can be maximum 30, and minimum -30 degrees from center. The beamport option does not affect other sources.

The focusing and phase_space_requirement options enable the user to control the assumptions used when using the minimalist principle. The possible options are 'total' and 'homogen', where 'total' is recommended. The minimalist_factor controls overillumination of the guide entrance when using the minimalist principle, 1 is default. This parameter can also be scanned.

It is possible to disable gravity, but this option can be a little buggy, check in mcdisplay if it works for all modules or not.

In addition, there are some optional options that controls the number of neutrons simulated and sizes of the monitors used. They are all relative to the normally used values, so one sets a multiplier where a value of 1 would be normal (or figure of merit for monitors). These are described in table 11.

options.minimalist	effect
options.ncount_multiplier_optimize	Adjusts the ncount used for optimization of the guide
options.ncount_multiplier_analyze	Adjusts the ncount for both analysis of the guide
options.ncount_multiplier	Adjusts the ncount for both analysis and optimization
options.psd2d_multiplier	Adjust the size of the 2d psd monitor
options.psd1d_multiplier	Adjust the size of the 1d psd monitors
options.div2d_multiplier	Adjust the divergency range 2d divergence monitors
options.div1d_multiplier	Adjust the divergency range 1d divergence monitors
options.apsd_multiplier	Adjust the size acceptance diagram monitors
options.adiv_multiplier	Adjust the divergence range of acceptance diagram monitors

Table 11: The effect of different optional multipler options.

3.5 Computing options

There are also a couple of options which controls more technical aspects and does not affect the guides to be optimized in any way. These are shown in table 12. The cluster option is used to select which cluster guide_bot prepares batch files for, currently the only working option is ESSS, which resembles the ESSS DMSC cluster. guide_bot is being setup at the MCC cluster at PSI, but there are still some problems. The queue option simply selects which queue to use. The enable_nested_compile option can enable the nested functions McStas flag. It should not be needed with the components currently in use, so default have it off. It can be re-enabled with the options.enable_nested_compile=1.

name	possible values
options.cluster	'ESSS','PSI'
options.queue (at ESSS)	'express','long','verylong'
options.queue (at PSI)	'test','short','medium','long'
options.mpi	number of cores on local machine
options.enable_nested_compile	0,1

Table 12: Computing options.

4 Details on modules

As table 6 showed there are some further options to some of the modules. These are all listed in the short summary of each module in the list below. It is important that each input string starts and end with one of the simple guide modules, straight (S), parabolic (P) or elliptic (E).

4.1 Guides: S P E

The three modules S P and E are the primary guides so far: straight, parabolic and elliptic. They have the exact same options, so they are collected in this entry. These modules are the only ones that can start or end an input string. All the general option applies. In addition the reflectivity model can be controlled and uses the standard McStas parameter names . The parameters are called: R0, Qc, alpha, W and m. Using alpha=0 and W=0 will enable Henrik Jacobsen's fit to Swiss Neutronic data so alpha, W and a second order term to the reflectivity will be calculated from m, but currently this is not recommended as there might be a bug for m<3.

For the elliptic guide there is an additional option that can limit the smallaxis of the elliptic guide. It is easy to make mistakes with this option, meaning it should be used by advanced users at this point. It is possible to set the maximum small axis for both simultaneously, or separately, as in the examples below. Here x refers to the horizontal direction, where y would be the vertical direction.

```
E(max_smallaxis=0.2) E(max_smallaxis_x=0.15)
```

4.2 Gap: G

The gap is simply a distance of vacuum between guides, usually inserted at a specific position because of a chopper. The Gap module can accept the general options, and has no specific options. Specifying the start width is equivalent to specifying the end width of the module before. The gap should be between two other modules, but not adjacent to a kink. Often used with the maxlength option to limit the length of the gap.

4.3 Kink: K

The kink module is used to break line of sight, and needs to be between two guides, meaning it can not start or end an input string. The kink includes a gap between the guides, so the general option length or maxlength is almost needed. Usually a maxlength=2 is used, which restricts the length without guide to 2 m. Normally the kink angle is calculated dynamically and thus depend on the specific geometry chosen by the optimizer. It is however possible to overwrite this and give the kink angle directly by using the option rot=X where X is the rotation in degrees (positive). It is also possible to give the direction of rotation using rotd and rots. rotd='h' makes the kink in the horizontal plane, and rotd='v' in the vertical. The other option rots determines the sign of the angle change, rots=1 being right where rots=-1 is left. If rotd='v' is used, rots=1 is down and rots=-1 is up. rotd and rots can be used even though rot is not used, and thus the angle is calculated so line of sight is avoided.

4.4 Curved guide: C

The curved guide module is special because the used McStas component can not have different entrance and exit dimensions. Trying to use the global options to get around this simply does not have an effect. Like the other guides, the reflectivity can be adjusted using reflectivity options, and like the kink module the angle change and direction thereof can be adjusted by rot, rotd and rots. As with the kink module, the change in angle is calculated for every optimizer attempt. Remember this module can not start or end an input string.

4.5 Selene guide: Selene

This module will insert the entire Selene concept guide, but can be in combination with other guides. The selene guide can not be between a los_start and los_end, but this is not very useful as it breaks line of sight on its own. It is important to notice that the m values of the Selene guide are calculated based on the length of the guide and the divergence it should deliver, so it can not be controlled by the m option. Furthermore, there is a limit to the m value it will use at m = 6, simply in order to avoid simulating unrealistic coating without the user knowing. If Selene is the first module there are some important options to be aware of. If no option is specified there will be inserted a slit 2 m from the moderator and the first focal point of the guide will be at this slit. This enables precise control over the size of the beam at the next focal point. It is possible to optimize the position of this first slit, simply call the module as below.

```
Selene(optimize_first_slit)
```

It is possible to discard the slit entirely, which is beneficial for smaller moderators. In that case call the first Selene module as below. Note however that a second Selene guide should be added after the first Selene in order to take advantage of the ability to control the size of the outgoing beam with a slit between the first and second Selene guide system.

```
Selene(moderator_focus)
```

If the Selene guide is used later in the guide, it does not have a slit as default since it is normally not needed. If however, the Selene guide follows a gap or kink, a slit is appropriate to use the background rejecting abilities of this concept, and can be inserted as below.

```
E G Selene(force_slit=1)
```

4.6 Slit: Slit

The slit module is very rarely useful, because under normal circumstances each module attaches to the former and latter, meaning that inserting a slit between two guides will not have any effect. Inserting it between a gap and a guide will not have any effect either, but it is possible to insert it between two gaps and create a more realistic chopper restriction, which is the intention of this module. Additionally, it is only effective when using minimalist = -1 to disable the phase space transport code, as it would restrict the next guide opening to be very small if a small chopper opening is used. In the following example a gap of 40 cm is made with a 2x2cm^2 pinhole in the middle.

```
P G(start=6.5,length=0.2) Slit(StartWidth=0.02,StartHeight=0.02) G(length=0.2) E
```

4.7 Monochromator: M

The monochromator module was recently contributed by Leland Harriger, who provided a dedicated manual for the module. The manual can be found in the documentation folder of the software, and will be merged with this documentation at a later date. The monochromator module is capable of optimizing over a wavelength range with appropriate rotation for each small wavelength range. Focusing is available in both horizontal and vertical direction, which can be optimized together with the guide.

5 Scanning demands

It is possible to scan some of the parameters in one or two dimensions. This can be done by simply entering a vector in MATLAB where you would otherwise just specify a scalar. The parameters which allow scans are shown in table 13.
The following is part of a simple guide_bot user file.

```
demands.Hdiv=0.50;
demands.Vdiv=0.70;
demands.Hsize=1.2;
demands.Vsize=1.2;
```

demands.Hdiv
demands.Vdiv
demands.Hsize
demands.Vsize
requirements.moderator_size_x
requirements.moderator_size_y
options.minimalist_factor

Table 13: Parameters which can be scanned by entering a vector.

In order to scan the vertical size of the sample, just change the code to the following.

```
demands.Hdiv=0.50;
demands.Vdiv=0.70;
demands.Hsize=1.2;
demands.Vsize=[1.2 1.3 1.4];
```

Currently the scanning system is limited to a maximum of two dimensions, meaning only two vectors. Notice that the same value can be used more than once in the vector in order to do the same simulation several times to check how consistent the optimizer performs for a given geometry. There is an script called scan_plotter.m which can analyze and collect data from a large scan. Simply run the script while MATLAB/iFit have an output/analysis folder as the active folder. Further documentation is in the top of the scan_plotter.m file. It still needs some work, but is very useful when analyzing large runs because they can be broken up in smaller sections. If matlab crashes or the process is killed, scan_plotter.m will remember how far it made it and start from there when it is launched again.

When scanning it can often be useful to lock to parameters together, for example to test 5 different square sample sizes. This can be done using the options.locked command.

```
demands.Hsize = [1 1.5];
demands.Vsize = [2 3];
option.locked.Vsize = 'Hsize';
```

Without the locked command the four possible combinations of the sample size would be checked, but with the command, only two are simulated, 1x2 cm^2 and 1.5x3 cm^2.

6 Optimizer modes

The standard optimization approach is to optimize the guide on a simple rectangular source, and then use the same guide on the more realistic source for the absolute intensity characterisation. For the TDR and Butterfly moderator, it is however possible to do the optimization on the realistic source, and use the same guide for the brilliance transfer characterisation. When optimizing for a more realistic moderator, the angle between the direction of the beamport and the guide direction is important, as it determines what parts of the moderator are viewed. This angle is added as a free parameter when selecting to optimize on a realistic source. In addition, it is possible to optimize the guide on the normal rectangular source, but then optimize this angle when the guide is used on the realistic source, which is called combined mode.

The options used to select between the three modes are displayed in table 14

name	possible values
options.optimizer_mode = 'ideal_source'	Optimization on rectangular source
options.optimizer_mode = 'realistic_source'	Optimization on selected realistic source
options.optimizer_mode = 'combined'	Optimization on rectangular source, then angle realistic source

Table 14: The three supported optimizer modes for TDR and Butterfly moderators.