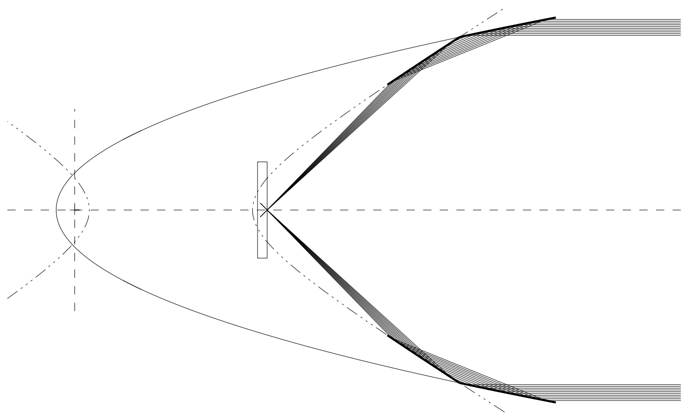


# MT_RAYOR

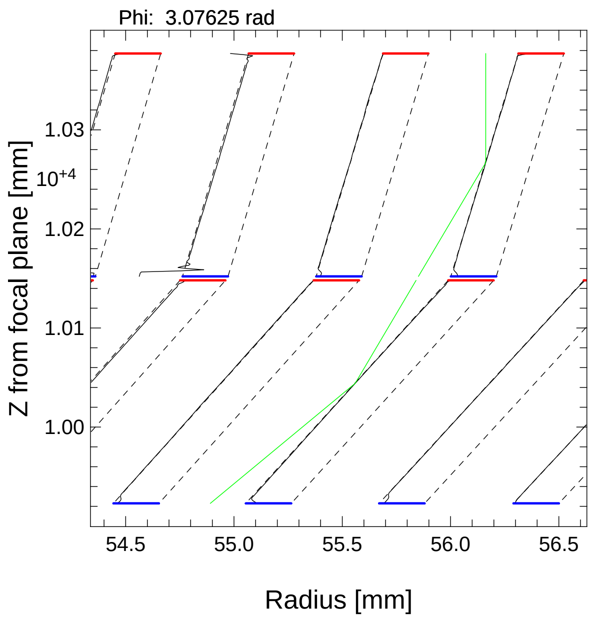
A raytracing system for X–ray telescopes

Version 4.6.5

Niels J. Westergaard

National Space Institute, Technical University of Denmark

March 5, 2020



maxwell.space.dtu.dk:/r4/njw/Htex/mt_rayor_man

Upper front page figure: Illustration of the Wolter-1 optical system.

Lower front page figure: Illustration of the path of a ray in a conical approximation telescope.

# Contents

**1 MT_RAYOR** **8**

**2 Basic concepts** **9**

**3 What can be learned about a telescope?** **10**

**4 First experience with MT_RAYOR** **11**

**5 Designing a telescope** **14**

5.1 Necessary parameters . . . . . 15

5.1.1 Focal length . . . . . 17

5.1.2 Type of optic . . . . . 18

5.1.3 Reference planes . . . . . 18

5.1.4 Mirror lengths . . . . . 19

5.1.5 Mirror thickness . . . . . 19

5.1.6 Outer and inner radii . . . . . 19

**6 MT_RAYOR functions** **20**

6.1 Overview of the raytrace . . . . . 20

6.2 Function descriptions . . . . . 20

6.2.1 mt_add_ysaves (U) . . . . . 20

6.2.2 mt_analysis (U) . . . . . 21

6.2.3 mt_aperture_stop (U) . . . . . 21

6.2.4 mt_bg_run_eff_area (U) . . . . . 22

6.2.5 mt_bg_run (U) . . . . . 22

6.2.6 mt_build_cat (U) . . . . . 23

6.2.7 mt_catalog2skydef . . . . . 23

6.2.8 mt_clear . . . . . 23

6.2.9 mt_create_om_<flavour> (D) . . . . . 24

6.2.10 mt_def_photons . . . . . 25

6.2.11 mt_det_add_bkg (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

6.2.12 mt_det_add_dxb_bkg (U) . . . . . . . . . . . . . . . . . . . . . . . . . 26

6.2.13 mt_det_add_instr_bkg (U) . . . . . . . . . . . . . . . . . . . . . . . . 26

6.2.14 mt_detector (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

6.2.15 mt_det_image (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

6.2.16 mt_drayplot (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

6.2.17 mt_dxb2skydef (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

6.2.18 mt_eff_area_photons (U) . . . . . . . . . . . . . . . . . . . . . . . . . 30

6.2.19 mt_eff_area_quick (U) . . . . . . . . . . . . . . . . . . . . . . . . . . 30

6.2.20 mt_fake_scatter_data (U) . . . . . . . . . . . . . . . . . . . . . . . . 31

6.2.21 mt_funcs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

6.2.22 mt_get_mirror_eff_factors (U) . . . . . . . . . . . . . . . . . . . . . . 32

6.2.23 mt_get_rcoef . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

6.2.24 mt_load (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

6.2.25 mt_mirdiag (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

6.2.26 mt_mirplot (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

6.2.27 mt_mk_mdeform_file (U) . . . . . . . . . . . . . . . . . . . . . . . . . 34

6.2.28 mt_photpr (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

6.2.29 mt_pre_def_photons . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

6.2.30 mt_propagate (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

6.2.31 mt_qimage . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

6.2.32 mt_rayplot (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

6.2.33 mt_raytrace_module . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

6.2.34 mt_reflplot (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

6.2.35 mt_run (U) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

6.2.36 mt_save . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

6.2.37 mt_scatter_data_file . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

6.2.38 mt_sel_scatter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

6.2.39 mt_setup_system . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

6.2.40 mt_spoke_blocking . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.2.41 mt_spoke_read . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.2.42 mt_skyima2skyspec . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.2.43 mt_skyspec2skydef . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.2.44 mt_upd_om_coating . . . . . . . . . . . . . . . . . . . . . . . . . . 43

**7 Data organization** **44**

7.1 Content of memory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
7.2 Telescope system definition files . . . . . . . . . . . . . . . . . . . . . . 44
7.3 Optical module description files . . . . . . . . . . . . . . . . . . . . . . 45
7.4 Mirror thickness file . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
7.5 Coating tables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
7.6 Scatter data files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
7.7 Spoke definition files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
7.8 Deformation description files . . . . . . . . . . . . . . . . . . . . . . . . 49
7.9 Photon save files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
7.10 Photon image files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
7.11 Photon flux files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
7.12 Source list files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
7.13 Sky definition files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

**8 Mirror geometry** **51**

**9 Cookbook** **52**

9.1 Download and installation . . . . . . . . . . . . . . . . . . . . . . . . . . 52
9.2 The GRI system with a single optical module . . . . . . . . . . . . . . . 52
9.2.1 Set up the system definition file . . . . . . . . . . . . . . . . . . . . . 52
9.2.2 Create the optical module file . . . . . . . . . . . . . . . . . . . . . . 53
9.2.3 Running a case . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
9.2.4 Display the result . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

9.2.5 Introducing scattering . . . . . . . . . . 55

9.3 Wolter I raytracing . . . . . . . . . . 56

9.4 Conical approximation raytracing . . . . . . . . . . 57

**10 Point source simulation 58**

**11 Simulation with source list as input 58**

**12 Simulation with sky image as input 58**

**13 Observation simulation 59**

13.1 Adding background . . . . . . . . . . 59

13.2 The detector . . . . . . . . . . 60

**14 A simulation example 61**

14.1 Event list and analysis . . . . . . . . . . 62

**15 Testing the optics in the lab 63**

**16 Appendix: Geometrical formulae 63**

16.1 Parabolic Mirrors . . . . . . . . . . 63

16.2 Wolter I optics . . . . . . . . . . 64

16.3 Conical approximations . . . . . . . . . . 64

16.4 Surface perturbations . . . . . . . . . . 65

**17 Appendix: Expressions for raytracing 65**

17.1 The surface normal . . . . . . . . . . 66

**18 Appendix: Word list 67**

**19 Appendix: External variables 67**

**20 Appendix: Installing Yorick 70**

20.1 Linux installation . . . . . . . . . . 70

20.2 Windows installation . . . . . . . . . . 70

**21 Appendix: Installing MT_RAYOR** **70**

**22 Appendix: The .scm format** **71**

**23 Appendix: History of updates** **72**

# 1 MT_RAYOR

This raytracing system for the analysis of axially symmetric grazing incidence X–ray telescopes has been developed at the Technical University of Denmark, National Space Institute. It is based on the Yorick interpreted language[1] and the FITS data storage format[2].

The X–ray telescopes that can be analyzed are of the nested mirror type based on ideal geometry but including realistic scattering and surface perturbations. The mirror surface reflecting properties are allowed to change from mirror to mirror.

The user interface is commandline based using the Yorick that allows an extended use of include files with command sequences for the execution of the various tasks.

Functions are also supplied for the design of a particular telescope by defining focal length, largest and smallest mirror radii, etc. Additional structural components such as baffles and mirror support spokes (aka spacers) may be included.

Both single reflection and double reflection (most common type) telescopes are supported. Axial symmetry is assumed apart from mirror surface perturbations and the support structure (spokes).

The raytracing is done by defining a number of photons of a given energy at the aperture of the telescope. Both celestial sources at infinity and laboratory sources at a finite distance can be simulated. Then each photon is followed through the optical system while tagging with point of mirror impact, grazing angle, outgoing direction (including scattering), and reflection coefficient. Finally the photons are propagated to an interception plane, usually the focal plane.

Keeping track of the fate of each photon allows the user to make in-depth analyses of the effects of the perturbations and the origin of single reflection photons that make it through the optics to mention a couple of examples. Similarly a reanalysis of the response can be done if the reflectivity is dependent on the point of reflection on the mirror surface, a situation that is not currently built into the system.

Better investigations of X-ray sources can be obtained by supplying photon flux table files (as a function of energy) where the energy for each photon is sampled from given flux table. Functions for using a sky image, a source catalog, or a laboratory source model as input are also included in the package. This is useful for modelling a field of sources with individual intensities and spectra and the modelling of diffuse or extended sources such as supernova remnants or clusters of galaxies.

For a complete observation simulation the detector properties can be taken into account: pixellation, redistribution matrix, and quantum efficiency. The sky coordinate system is reproduced in the resulting detector images which makes the comparison with the original observations and other simulations easier.

The version described here is the one that comes with the distribution: `mt_rayor-4.5.5.tar.gz`.

[1]David H. Munro, University of California, Lawrence Livermore National Laboratory, 1994
[2]Wells *et al.*, 1981, A&A Suppl. **44**, 363, and Harten *et al.*, 1988, A&A Suppl. **73**, 365

# 2 Basic concepts

A mirror section of the telescope where the photons are meant to undergo a single reflection is here called an optical module[3]. It is defined in a FITS table listing all the mirror radii and lengths (rotational symmetry is assumed) as well as other information. In the present framework a telescope can be composed of one or two optical modules. In the former case there will only be a single reflection concentrating the X-ray beam. In the latter case there will be two reflections required for getting true imaging.

The goal has been to define a system that is able to handle a variety of X-ray telescopes of thin foil design. Such a telescope can consist of a number of optical modules each defined in its own coordinate system (as long as the z-axis coincides with the optical axis) with a reference plane placed at $z = 0$. In the telescope description each reference plane will have a (positive) z-value and the photons will be moving in the negative z-direction.

It has not been possible to build a genuine 'Lego'-system where modules can be installed in a simple way because of the demand that the exit slits of the first module must match the entrance slits of the second module for maximal telescope throughput. Hence the first optical module is called the master module and the second one, the slave module.

With version 4.6.5 these types of systems can be handled: A parabolic or conical approximation single reflection concentrator such a the one proposed for the GRI (Gamma Ray Imager) of the ESA 'Cosmic Vision' scheme, and a Wolter I conical approximation system e.g. like the one used for NuSTAR, a NASA SMEX mission, and for the SXT on the Indian Astrosat mission. The proposed ESA mission Athena (formerly IXO) on the other hand foresees the use of a true Wolter I system.

A telescope system is defined by parameters given in an ASCII text file. All basic parameters are given there as well as the names of the files that describe the optical modules in detail.

The surface properties with respect to reflection and scattering are taken care of in the form of tables. Parameters for a given kind of surface and coating is stored in a FITS binary table file that holds both the coefficent of reflection and an array with the scattering distribution for an appropriate number of energies and incoming grazing angle values.

In the physical world no mirror has the ideal shape. The deviations, if known, can be included in the raytracing by deformation description files (see section 7.8) with a deformation map for each mirror in azimuth and z.

Once the Optical Module Description files have been produced as well as the scattering distributions the actual raytracing can start.

[3]This definition is different from the one used in Athena

## 3 What can be learned about a telescope?

When a raytracing run has come to and end a number of external variables exist in the memory. One of the most important ones is the 'Phs' that is an struct-array (see description in section 7.1, page 44). It holds data for every photon that was launched.

The photons can be converted to counts once the detector has been defined with an appropriate RMF (Redistribution Matrix File) and a quantum efficiency table.

The session as such can be saved, but individual results like the photon list, the event list, or the focal plane image can be saved for later study or documentation.

Functions in MT_RAYOR are designed to assist in the analysis of the telescope properties, other functions from the more general library of Yorick functions (comes with the distribution of the MT_RAYOR package).

Among the topics that can be studied are:

1. Point spread function.
2. Telescope effective area.
3. Photon reflection angle distribution.
4. Detector image.
5. Observation planning.

## 4 First experience with MT_RAYOR

Go to the `/myhome/yorick` directory and create a `toy1` subdirectory. Copy the files `system_toy1.scm` and `mthick_toy1.scm` (found in the distribution) to this new directory and change directory to it.

```
toy1$ yorick


 Reading custom.i in /r9/njw/yorick/yorick-2.1.06/yorick-2.1/i ...
 Function 'randomize' has been called from 'mfits.i'
 You are using jemx.i-2.5
GISTDIR: /r9/njw/yorick/yorick-2.1.06/yorick-2.1/g/
 Copyright (c) 2005.  The Regents of the University of California.
 All rights reserved.  Yorick 2.1.06 ready.  For help type 'help'


> #include "mt_rayor-4.5.0.i"
Loading MT_RAYOR Version 4.5.0
> mt_setup_system, "system_toy1.scm"



Number of modules in system :  1
Optical module file :  om_toy1.fits
                not found!
You may want to create it by 'mt_create_om_par1,filename="om_toy1.fits"'
Mirror thickness file: mthick_toy1.scm
No mirror deformation to be used for module 1
No spoke definition is to be used for module 1
No spoke definition is to be used for module 1
No detector has been loaded
Warning level 100
 Note that the use of scattering is excluded.
 An optical module file is missing!
```

The report on the warning level should be 0 (zero) for successful setting up. So take the advice:

```
> mt_create_om_par1,filename="om_toy1.fits"


WARNING freeing builtin function is_scalar
WARNING freeing builtin function is_vector
The parabolic system optical module file: om_toy1.fits has just been created
It has 108 mirrors
and it might need a coating update: mt_upd_om_coating,...
```

The two warnings here with capital letters is a message from Yorick that the functions 'is_scalar' and 'is_vector' are destroyed. But with the functions included in the distribution they are just replaced by similar functions so the messages can be ignored.
Try:

```
> ls



mthick_toy1.scm  om_toy1.fits     system_toy1.scm
```

The 'ls' command works like the Unix/Linux 'ls' command except for options.

Since the setup of MT_RAYOR failed it must be repeated:

```
> mt_setup_system, "system_toy1.scm"
```

```
Number of modules in system :  1
Optical module file :  om_toy1.fits
Mirror thickness file: mthick_toy1.scm
No mirror deformation to be used for module 1
No spoke definition is to be used for module 1
No spoke definition is to be used for module 1
No detector has been loaded
Warning level 0
 Note that the use of scattering is excluded.
 System OK!
```

All settings have now been accepted. The newly created file `om_toy1.fits` can now be inspected. Each row describes a mirror shell, where **R1** and **Z1** is the radius and z value (z in the local OM (optical module) coordinate system)) of the reflecting surface where photons enter – here called upper end. **R2** and **Z2** refer to the other, lower, end of the mirror. You'll notice that the length of the mirror shells is not constant. The applied function, `mt_create_om_par1`, defines the geometry such that the mirror spacing is constant, which implies that the inner mirrors are longer than the outer mirrors. This may be illustrated by

```
> mt_mirdiag,"om_toy1.fits"
```

or

```
> mt_mirdiag,"om_toy1.fits",rr=[54,90]
```

See figures 1 and 2 for the result of these.

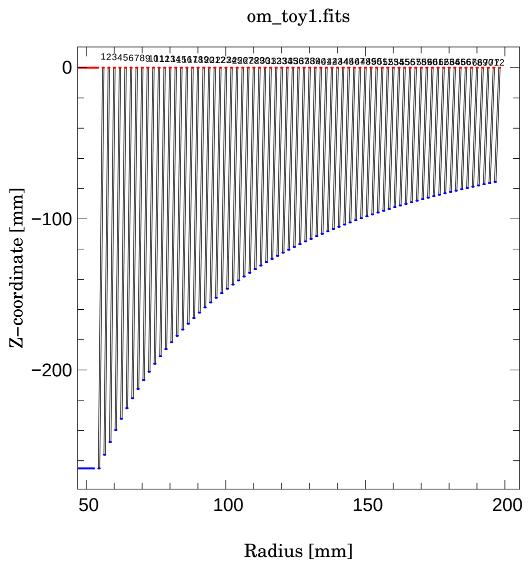


Figure 1: *Mirror configuration of all mirrors in 'toy1'*

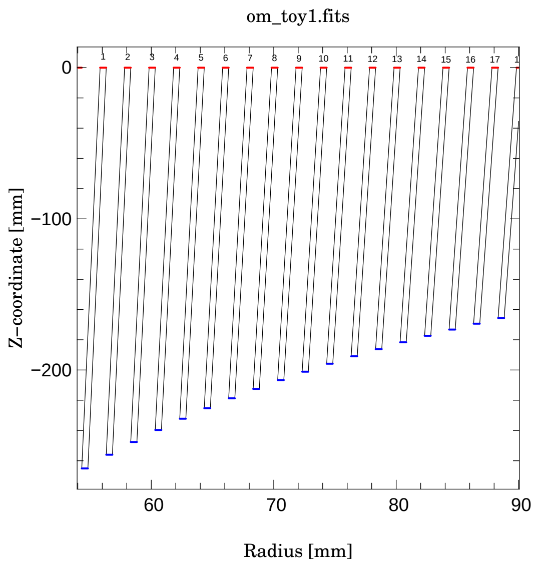


Figure 2: *Mirror configuration of the innermost mirrors in 'toy1'*

Now the raytracing can begin. The simplest command is:

```
> mt_run,1
```

where the argument defines the energy of the photons in $keV$.

```
CPU time for mt_run was:  16.355 s
```

To see the size of `Phs`, the array with photon information:

```
> info,Phs          ! this is a general Yorick function
array(s_Ray,113643) ! tells that Phs is an array of struct s_Ray
```

To get a plot of the distribution in the focal plane:

```
> mt_qimage
```

which produces the plot in figure 3. Note the distance scale, about a nanometer, which gives an impression of the precision of the calculations.

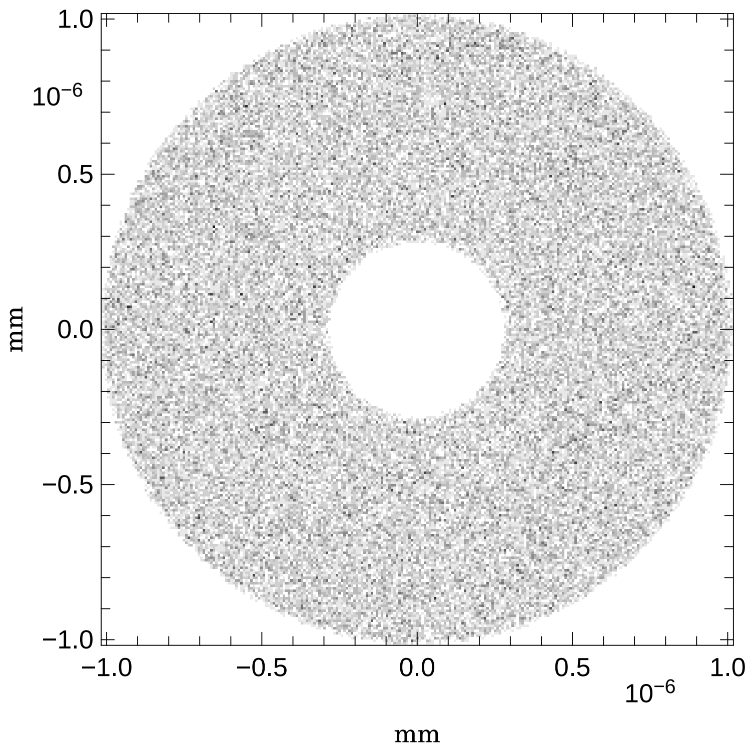


Figure 3: The focal plane image of the simple 'toy1' telescope for an on-axis source.

## 5 Designing a telescope

This section explains how to set up the basic files that define the telescope system. Users that already have such files available may skip this section and proceed immediately with section 9.2.3.

All measures of length are in *mm*. The coordinate system used is shown in figures 4 and 5.

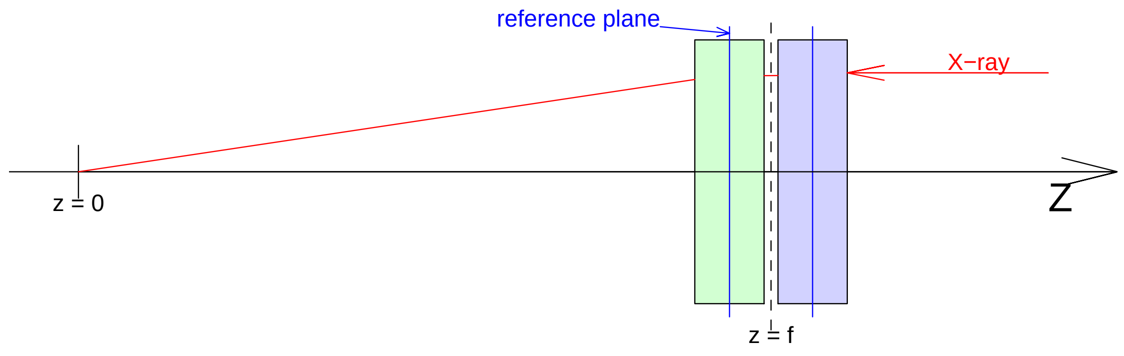


Figure 4: The case of an optic with a gap i.e. there is a small distance between the mirrors in the two modules. The first (master) optical module is shown in light blue, the second one (slave) is shown in light green. The focal plane is at 'z = 0'.

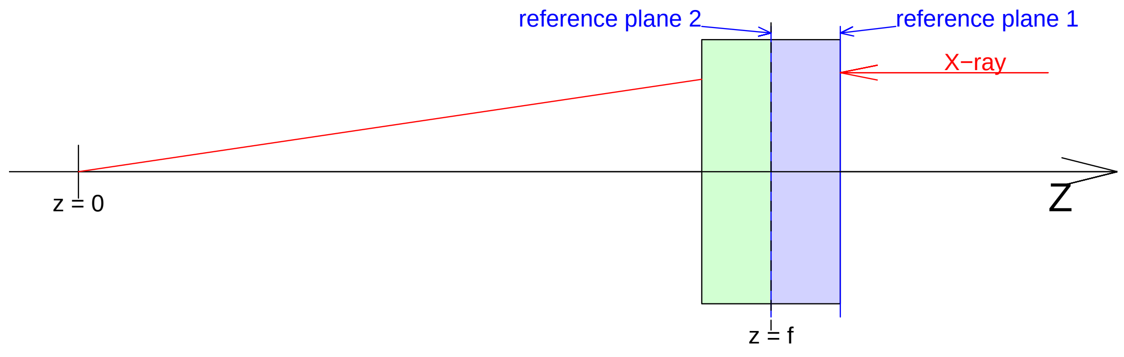


Figure 5: The case of an optic with no gap i.e. there is no distance between the mirrors in the two modules. The first (master) optical module is shown in light blue, the second one (slave) is shown in light green. The focal plane is at 'z = 0'.

Within the optical modules a 'local' coordinate system is used with an origin in the reference plane. The z coordinate has the same direction as in the general coordinate system and the x and y coordinates are unchanged between the systems with origin at the optical axis.

The design tools supplied in MT_RAYOR all work with a axial symmetry around the telescope optical axis except for the mirror deformations. It implies that telescopes of e.g. Kirkpatrick-Bayez types are not supported.

Three cases are shown in the following sections, a parabolic single reflection 'beam concentrator' (named toy1), a ideal Wolter-1 telescope (named toy2), and a conical approximation to a Wolter-1 system including a gap (named toy3).

## 5.1 Necessary parameters

The entire setup is defined by a single 'system' text file written in the .scm format (see section 22 page 71). Unique keywords defined as

```
// focal_length = 5000.0
```

anywhere in the file. For keywords that appear more than once the order matters as f.i. in the case of `z_reference` when two optical modules are defined.

### Toy1

The telescope system defition file is called `system_toy1.scm`.
It looks like this:

```
//   system_toy1     2015-09-04/NJW
//
//  A single reflection parabolic telescope
//
//  Focal plane is per definition z = 0
//
// num_modules = 1
//
// om_type = parabolic
// z_reference =   1200.0   ; mm
// Zfocus      =  -1200.0   ; mm
// mirror_length =   100.0  ; mm
//
// r_outer     =     85.0   ; mm
// r_inner     =     24.0   ; mm
// om_function = mt_create_om_par2
// om_parameter = 1.00      ; mm  Constant mirror spacing
// om_file = om_toy1.fits
// mirror_thickness_file = mthick_toy1.scm
//
// focal_length =  1200.0  ; mm
//
//  scat_file = ../sxt/scat_au_type2.fits
//
//  Pseudo Focal plane pixel settings (only used by mt_save):
//
//  Dim_focp = 441; Number of pixels of focal plane detector
//  Pix_focp = 0.1; [mm] Pixel size
//
```

### Toy2

The telescope system definition file is called `system_toy2.scm`.
It looks like this:

```
//   Wolter 1 system_toy2    2013-05-26/NJW
//
//    A parabolic module followed by a hyperbolic-like module
//
//  Focal plane is per definition z = 0
//
// num_modules = 2
//
// om_type = parabolic
// z_reference =  12102.004 ; mm
// Zfocus      = -24000.0   ; mm
// mirror_length =  102.004 ; mm
// r_outer     =    313.098 ; mm
// r_inner     =    259.198 ; mm
// om_function = mt_create_om_par2
// om_parameter = 1.0;  Dense packing
// om_file = om_toy2_par.fits
// mirror_thickness_file = mthick_toy2.scm
//
// om_type = hyperbolic
// z_reference =  12000.0   ; mm
// Zfocus      = -12000.0   ; mm
// mirror_length =  102.004 ; mm
// om_function = mt_create_om_hyp2
// om_parameter = -999.0; Dummy parameter
// om_file = om_toy2_hyp.fits
// mirror_thickness_file = mthick_toy2.scm
//
// focal_length =  12000.0  ; mm
//
//  Pseudo Focal plane pixel settings (only used by mt_save):
//
//  Dim_focp = 441; Number of pixels of focal plane detector
//  Pix_focp = 0.1; [mm] Pixel size
//
```

## Toy3

The telescope system definition file is called `system_toy3.scm`.
It looks like this:

```
//
// Product of mt_tel_design from the MT_RAYOR package
// Initiated 2014-09-18T11:56:35
//
// The system setup file for a Wolter 1 conical approximation
// telescope with a gap between the mirror sets (optical modules)
//
```

```
// num_modules = 2
// om_type = conical
// z_reference = 5102.000 ; mm
// Zfocus = -10204.000 ; mm
// mirror_length = 200.000 ; mm
// mirror_thickness_file = mthick_toy3a.scm
// r_outer = 200.000 ; mm
// r_inner = 50.000 ; mm
// z1_setup = 100.000 ; mm
// z2_setup = -100.000 ; mm
// om_function = mt_create_om_con4
// om_parameter = 1.000; packing density
// om_file = om_toy3a_1a.fits
// mirror_deform_file = mdeform_00.fits
// spoke_define_file = none
// spoke_define_file = none
//
// om_type = conical
// z_reference = 4898.000 ; mm
// Zfocus = -4898.000 ; mm
// mirror_length = 200.000 ; mm
// mirror_thickness_file = mthick_toy3a.scm
// z1_setup = 100.000 ; mm
// z2_setup = -100.000 ; mm
// om_function = mt_create_om_con5
// om_parameter = -999.; no effect for slave module
// om_file = om_toy3a_3a.fits
// mirror_deform_file = mdeform_00.fits
// spoke_define_file = none
// spoke_define_file = none
//
// focal_length = 5000.000 ; mm
//
// scat_file = scat_au_type2.fits
//
// detector_descr_file = none
// Dim_focp = 441 ; For use by 'mt_save' alone
// Pix_focp = 0.100 ; mm For use by 'mt_save' alone
// telescop = toy3a
// instrume = toy3a
```

The lines are explained in the following sections.

### 5.1.1 Focal length

For a single reflection telescope the focal length is measured from the aperture plane (perpendicular to the optical axis).

For truly imaging telescopes with two reflections the focal length is measured from the midpoint between the two optical modules on the optical axis, see figures 4 and 5.

```
// focal_length = 5000.0; mm
```

is a required parameter to be used e.g. in the conversion between length in the focal plane and angles.

#### 5.1.2 Type of optic

With MT_RAYOR version 4.0 and higher the following telescope systems are supported:

**GRI type mirrors.** This is a single reflection concentrator where the distance between the mirror shells is constant and the length of the mirrors decreases with radius in order to avoid leakage through the concentrator (function mt_create_om_par1).

**Parabolic concentrator.** This is a single reflection concentrator with constant mirror length and the mirror distance increases with radius (function mt_create_om_par2).

**Ideal Wolter 1.** A double reflection telescope with constant length mirrors in both optical modules (functions mt_create_om_par2 and mt_create_om_hyp2).

**Wolter 1 conical approximation.** A double reflection telescope with constant length mirrors in both optical modules (functions mt_create_om_con2 and mt_create_om_con3).

**Wolter 1 conical approximation with gap.** A double reflection telescope with constant length mirrors in both optical modules (functions mt_create_om_con4 and mt_create_om_con5).

#### 5.1.3 Reference planes

In the following equations $f$ is the focal length, $z_{ref}$ is the z-coordinate of the reference plane, $l_m$ is the mirror length, and $g$ is the gap width.

## Toy1

The way to calculate `z_reference` is

$$z_{ref} = f \tag{1}$$

## Toy2

The way to calculate `z_reference` for the modules is

$$z_{ref} = f + l_m \tag{2}$$
$$z_{ref} = f \tag{3}$$

## Toy3

The way to calculate `z_reference` for the modules is

$$z_{ref} = f + \frac{l_m + g}{2} \tag{4}$$
$$z_{ref} = f - \frac{l_m + g}{2} \tag{5}$$

### 5.1.4 Mirror lengths

The mirror lengths are defined in the telescope definition phase and are saved into the Optical Module Description file.

### 5.1.5 Mirror thickness

In the design of a telescope the thickness of the mirror shells plays a role, so it must be defined prior to creating the Optical Module Description file. An ASCII table in the .scm format, a socalled mirror thickness file, is expected as input.

### 5.1.6 Outer and inner radii

The outer and inner radii for the telescope are often determined by considerations about what size telescope fits into the available space, the production of the inner mirrors: can the radius of curvature be produced?. Also the effective area of the inner mirrors can be quite small, on the other hand the high energy response is expected to come from these mirrors.

In the current design the limiting mirror radii might not be met precisely, since the mirrors are placed one by one.

# 6 MT_RAYOR functions

## 6.1 Overview of the raytrace

The basic parameters of the telescope system are defined in a special text file that is used for setting up.

The raytracing process starts with a characterization of all the photons that impinge on the telescope entrance aperture with a given energy and direction but a random position uniformly distributed. The photon array is then sorted according to radius because the scattering properties most likely change with mirror number.

The definition of the first optical module is then loaded into memory and the photons are propagated through that module one by one. For each photon the appropriate mirror number is found and the connected scatter data file is loaded into memory if it does not already reside there based on the coating type. All photons that experience a single reflection are tagged with the actual coefficient of reflection that depends both on energy and grazing angle. The photons that make it through the exit aperture get assigned a status value of zero, the others get a (positive) status value according to where they miss the proper path.

When all photons are dealt with, the second optical module is loaded if it has been included in the telescope setup. All photons with zero status value are then propagated through the module in an analoguous fashion to the first module.

Finally all succesful photons are propagated to the focal plane.

At this point the raytracing results can be derived such as PSF and effective area. In order to make a full investigation over a range of energies and off-axis angle a raytracing run must be done for each combination although there are shortcuts if the scattering is energy independent.

A pixellated detector can be defined with quantum efficiency and redistribution matrix for the simulation of an observation. The photons that are accepted in the detector are saved as events with proper tagging (energy and pixel) for subsequent analysis.

Saving the entire session, writing the focal plane image to a FITS file, or saving all photon information to a FITS photon file can also be done.

## 6.2 Function descriptions

Here follows a description of each of the MT_RAYOR functions. In Yorick the command: *help, function_name* will cause a display of basic information about the function, including arguments and keywords.

Only a subset of the functions will be called directly by the user in normal raytracing applications. These are marked with a '(U)' in the title. Others may be called by the user for defining a telescope, these are marked with '(D)' in the title.

### 6.2.1 mt_add_ysaves (U)

---

Calling:

```
mt_add_ysaves
```

Keywords:

*file*: TBW
*list*: TBW
*reset*: TBW

Does what it is meant to do.

#### 6.2.2 mt_analysis (U)

---

Calling:

```
mt_analysis[,phs]
hpd = mt_analysis([phs])
```

Arguments:

*phs*: (optional) Array of struct s_Ray

Keywords:

*photon_file*: Name of input photon file to use
*geom*: (bool) If set all reflection coefficients will be set to 1.0
*frac*: Fraction of weighted counts inside diameter (defaults to 0.5)
*allbounce*: (bol) Include all bounce values
*silent*: (bool) Suppress terminal output

Reads the photon file or works on the photons stored in memory and produces a value for the HPD (Half Power Diameter). It finds the average position while excluding photons placed too far away and calculates the HPD around that point. For single reflection telescope this has only meaning for an on-axis observation. If the *photon_file* is not given, the contents of the memory ('`Phs`') is used.

If called as a subroutine the results will be printed to the screen else the HPD value or for the chosen fraction in *mm* will be returned.

#### 6.2.3 mt_aperture_stop (U)

---

Calling:

```
mt_aperture_stop, z_position, open_radius
```

Arguments:

*z_position*: [mm] Position of the aperture stop in the Z-direction

*open_radius*: [mm] Radius of the circular opening of the sperture stop

Keywords:

*cen_dx*: TBW

*cen_dy*: TBW

*photfile*: TBW

*undo*: TBW

*chat*: TBW

Tags all photons that do not go through the aperture stop opening with error code XX.

#### 6.2.4 mt_bg_run_eff_area (U)

---

Calling:

```
mt_bg_run_eff_area, energy, offaxis, azimuth, system_file,
```

Arguments:

*energy*: Energy [keV] of infall photons

*offaxis*: Off-axis angle [arcmin] of infall photons

*azimuth*: azimuth angle [deg] of infall photons

*system_file*: Text file (.scm format) that gives the telescope parameters

Keywords:

*no_scatter*: (bool) Flag for eliminating the scatter

*no_mdeform*: (bool) Flag for excluding mirror deformations

This function has been designed for a HPC where several CPUs can be running simultaneously. It works by splitting up the process and keeping track of the status of the indiviual subprocesses. When all is done the results are combined and presented to the user.

#### 6.2.5 mt_bg_run (U)

---

Calling:

```
mt_bg_run, bla
```

Arguments:

*bla*: text

Keywords:

*bla*: text

This function works.

#### 6.2.6 mt_build_cat (U)

---

Calling:

```
mt_build_cat, catfile
```

Arguments:

*catfile*: text

Keywords:

*mode*: text

This function works.

#### 6.2.7 mt_catalog2skydef

---

Calling:

```
mt_catalog2skydef, catalog_file, skydef_file, ra_scx, dec_scx, posang
```

Arguments:

*catalog_file*: text

*skydef_file*: text

*ra_scx*: text

*dec_scx*: text

*posang*: text

Keywords:

*bla*: text

This function works.

#### 6.2.8 mt_clear

---

Calling:

```
mt_clear, bla
```

Arguments:

*bla*: text

Keywords:

*bla*: text

This function works.

### 6.2.9 mt_create_om_<flavour> (D)

---

These flavours of the optical module generator exist:

Subroutine: mt_create_om_par1, filename=

Subroutine: mt_create_om_par2, filename=

Subroutine: mt_create_om_hyp2, filename=

Subroutine: mt_create_om_con2, filename=

Subroutine: mt_create_om_con3, filename=

Subroutine: mt_create_om_con4, filename=

Subroutine: mt_create_om_con5, filename=

Calling:

```
mt_create_om_flavour
```

Keywords:

*filename*: Name of output file

Creates an optical module file with the FITS table for a mirror system. The *filename* keyword defines the name of the output file. If it is not given then the output will be placed in a file by name of 'om_abc_*nnn*.fits' where *nnn* is a serial number updated relative to the already existing files of that kind.

The input is defined by running *mt_setup_system* that consequently must be run before running *mt_create_om_abci*.

*mt_create_om_par1* creates a stand-alone parabolic optical module.

*mt_create_om_par2* followed by *mt_create_om_hyp2* creates optical modules for a Wolter I system.
No gap between the two modules is foreseen.

*mt_create_om_con2* followed by *mt_create_om_con3* creates optical modules for a conical approximation to a Wolter I system.
No gap between the two modules is foreseen and the reference plane is the same as the entrance aperture plane.

*mt_create_om_con4* followed by *mt_create_om_con5* creates optical modules for a conical approximation to a Wolter I system with a gap between the optical modules. The reference plane is not necessarily at the entrance aperture of the modules, the keywords 'z1_setup' and 'z2_setup' in the system definition file define the mirror position relative to the reference plane.

### 6.2.10 mt_def_photons

---

Calling:

```
mt_def_photons
```

Arguments:

*fraper*: Defines an entrance window, see below

*energy_or_file*: Photon energy in $keV$ or file with table

*R_or_lab*: Vector for photon direction of position of laboratory source

Keywords:

*dphot*: Surface density of photons (unit: $\mathrm{mm}^{-2}$, default value is 1.0)

*cont*: Flag for appending to existing `Phs` array

*flag*: Flag for debugging

*lab*: Flag for interpreting third argument as a lab source position

*eqillum*: Flag for 'equal mirror illumination – for special investigations

Defines the photons at the system aperture. The first argument 'fraper' (for Front Aperture) must be a two or four element array, where the two first elements are the inner and out radii. If a four element array is used then the two last elements are the limiting azimuth angles of the front aperture in degrees.

If the argument 'energy_or_file' is not given as a scalar number but as a string, it is assumed that it is a filename of a binary table FITS file where the first extension has (at least) three columns: ENERG_LO, ENERG_HI, and PHOTFLUX, where the units are expected as 'keV', 'keV', and 'ph $\mathrm{cm}^{-2}\mathrm{s}^{-1}\mathrm{keV}^{-1}$'. The name of the extension (keyword EXTNAME) must be PHOTON_FLUX.

An array (Phs) of the *struct* s_Ray is defined and initiated with values according to the function arguments unless the keyword 'cont' is set in which case Phs is expanded with the new photons. The photon positions are chosen at random between the radius limits with a uniform density. Afterwards they are sorted with increasing radii which reduces the need for updating scattering properties when starting a new photon.

See also:

mt_pre_def_photons

### 6.2.11 mt_det_add_bkg (U)

---

Calling:

```
mt_det_add_bkg
```

Keywords:

*instr*: Flag for including instrumental background

*dxb*: Flag for including diffuse X–ray background

*dxb_adjust*: Adjustment factor for the DXB

Can only be called after a call of 'mt_detector'.
Add instrumental and/or diffuse background to current event list (Evlist).

See also:

mt_det_add_dxb_bkg, mt_det_add_instr_bkg

#### 6.2.12 mt_det_add_dxb_bkg (U)

---

Calling:

```
mt_det_add_dxb_bkg
```

Arguments:

*filename*: Name of file with DXB pool

Keywords:

*adjust*: Adjustment factor for the DXB

Can only be called after a call of 'mt_detector'.
Add diffuse background to current event list (external variable: Evlist).

See also:

mt_det_add_bkg, mt_det_add_instr_bkg

#### 6.2.13 mt_det_add_instr_bkg (U)

---

Calling:

```
mt_det_add_instr_bkg
```

Arguments:

*filename*: Name of file with instrumental background countrate

Can only be called after a call of 'mt_detector'.
Add detector background to current event list (external variable: Evlist) as described in the .scm format file 'filename'.

See also:

mt_det_add_bkg, mt_det_add_dxb_bkg

### 6.2.14 mt_detector (U)

---

Calling:

```
mt_detector
```

Keywords:

*offset*: A two-element array with detector offsets in x and y

*cont*: A flag for continuing (adding to) the Evlist array

*bkglvl*: The level of additional background

*flag*: May be used to mark the latest raytraced photons

loads the detector description data from the 'detector_descr_file' and forms an event array 'Evlist' from the successful photons in 'Phs' i.e. those with status==0 and accepted by the random coefficient of reflection test.[4] The detector_descr_file is defined in the setup process by mt_setup_system.

Standard operation is resetting the event array, but setting the keyword 'cont' will cause the events to be appended to the already existing event array. If the keyword 'bkglvl' is set then extra events will be added with this number per detector pixel per keV. The keyword 'flag' is used to update the event tag 'flag' except for the background that allways gets a zero flag value.

### 6.2.15 mt_det_image (U)

---

Calling:

```
im = mt_det_image()
```

Keywords:

*emin*: Lower energy limit in keV

*emax*: Upper energy limit in keV

*outfile*: The image will be written to this FITS file

*bkglvl*: This level of background will be added to the image

returns the detector image in the energy range given by 'emin' and 'emax'. The default value for these is the minimum of E_MIN and maximum of E_MAX resp. as given in the detector description file. If keyword 'outname' is given then an image in FITS format is written to that name. Furthermore a background is added is keyword 'bkglvl' has a finite value (per pixel per keV). Note that a background can be defined in either of 'mt_detector' and 'mt_det_image'.

[4]Accepted if Rcoef is greater than a random number between zero and one drawn from a uniform distribution.

### 6.2.16 mt_drayplot (U)

---

Calling:

```
mt_drayplot
```

Arguments:

*iphot*: The photon indx number in 'Phs'.

Keywords:

*over*: Flag for overplotting in existing figure.

Plot the path of the chosen photon through both optical modules. See figure 6.

If the keyword 'over' has been set then the path is added to the current display (overplot).

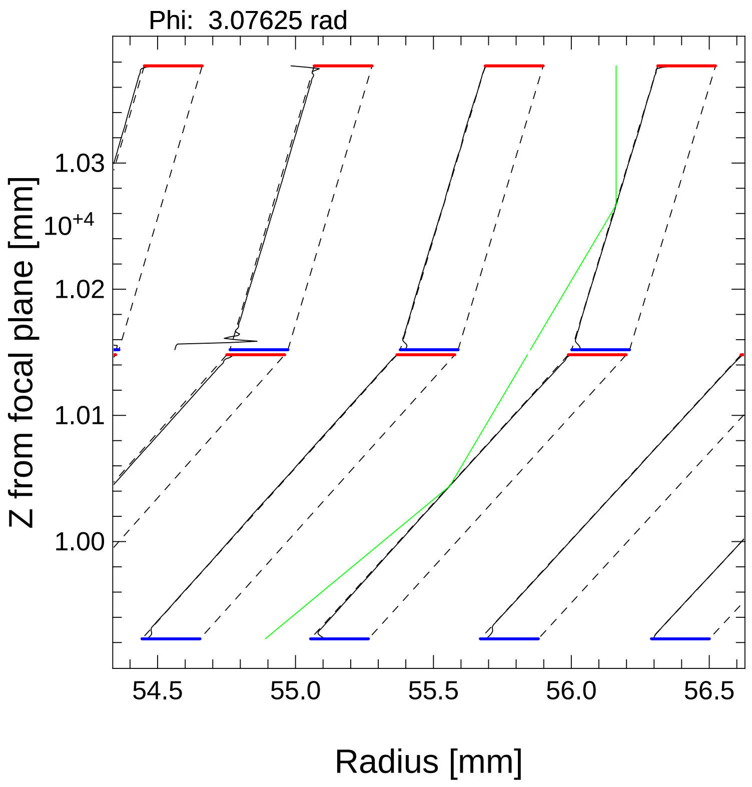


Figure 6: A result of mt_drayplot for a nominal passage (green lines) through a telescope. The dashed lines make the nominal mirror and the black curve shows the surface when deformations are taken into account (a rather extreme case is seen in one of the other mirrors). The red lines illustrate the upper baffles and the blue lines illustrate the lower baffles.

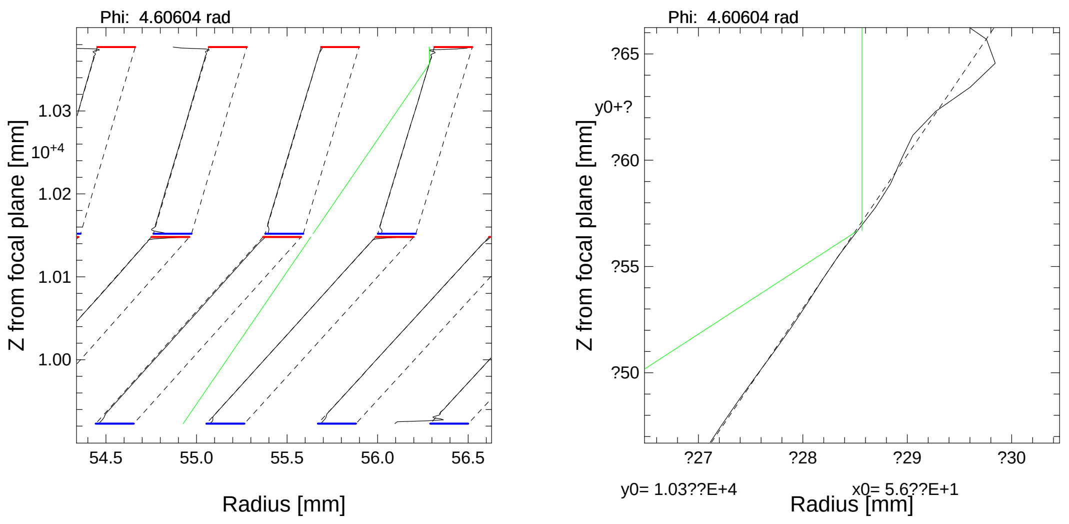


Figure 7: **Left panel:** The mirror deformation can give rise to a missed reflection in the second optical module even for on-axis rays. **Right panel:** A zoom of the reflection region detailing the mirror surface. (In Yorick you can easily zoom by left-clicking in the plot window and drag to the center.)

#### 6.2.17 mt_dxb2skydef (U)

---

Calling:

```
mt_dxb2skydef, skydefname, dol_dxbflux, wfov, n
```

Arguments:

*skydefname*: The name of the file for saving the Sky Definition File

*dol_dxbflux*: The name of the input photon flux file

*wfov*: The width of the FOV in deg

*n*: The number of sources from center to edge

Keywords:

*exposure*: The exposure time in s.

*mission*: The mission name.

*instrume*: The instrument name.

Produces a skydefinition file for DXB determination by arranging sources in a regular mesh around (RA,Dec) = (180,0).

Suggested sequence:

mt_run, skydefname;
mt_save,mode="e",outfile="DXB";

#### 6.2.18 mt_eff_area_photons (U)

---

Calling:

```
mt_eff_area_photons
eff_area = mt_eff_area_photons()
```

Keywords:

*earr*: Array of energies

*samp*: Sampling parameter

*outfile*: Name of output file

*silent*: Flag for suppressing screen output

returns the effective area (in $mm^2$) for a standard array of energies as given by the scatter files, or, if keyword 'earr' is supplied with an array of energies, for those energies. The photons in memory are used for this irrespective of their original energy. Thus this function is **not** valid if the scattering properties change with energy. In this case the computation must be split up into small energy intervals with a raytracing run followed a call of this function for each interval.

Only the photons with 'status' == 0 are used.

When the keyword 'outfile' has been set, an output FITS file with the effective area table will be written. If 'outfile' is a string it will become the name of the files else the result will be written to a file by name of *eff_area_nnnn.fits* where *nnnn* is a serial number.

Since this process it quite time consuming a sampling keyword 'samp' has been introduced. If used, one photon out of 'samp' will be used in the calculation.

#### 6.2.19 mt_eff_area_quick (U)

---

Calling:

```
mt_eff_area_quick
eff_area = mt_eff_area_quick()
```

Keywords:

*earr*: Array with energy values (default: energies from the scatter files

*outfile*: The result is written to this file

returns the on-axis effective area (in $mm^2$) for a standard array of energies, or, if keyword 'earr' is supplied with an array of energies, for those energies. The geometrical area of the master optical module is multiplied with the reflection coefficient for the average grazing angle for the reflection or for both reflections if two optical modules exist.

The effect of the obscuration by the spokes is included in the calculation if the function 'mt_get_mirror_eff_factors' has been run beforehand.

When the keyword 'outfile' has been set, an output FITS file with the effective area table will be written. If 'outfile' is a string it will become the name of the files else the result will be written to a file by name of *eff_area_nnnn.fits* where *nnnn* is a serial number.

### 6.2.20 mt_fake_scatter_data (U)

---

Calling:

```
mt_fake_scatter_data, filename
```

Arguments:

*filename*: The output file name

Keywords:

*fwhm*: Full Width Half Maximum (*rad*) of scatter distribution

*dist_angle_max*: Half range (*rad*) of scatter distribution

*n_angles*: Number of angle values in the scatter distribution

*angle_max*: Maximum grazing angle (*rad*) of incidence

*ener_min*: Minimum energy (*keV*) of energy range

*ener_max*: Maximum energy (*keV*) of energy range

*n_ener*: Number of energy values

*coat*: Coating number

Produces a scatter file as defined in 'filename' in the correct format with ad-hoc reflection coefficients and a gaussian scattering distribution the width of which can be chosen with the 'fwhm' keyword (unit: radian). Similarly the maximal angle (in radians) and energy (in keV) can be defined. The coating designation can be given by keyword 'coat' (defaults to 1 (one)).

### 6.2.21 mt_funcs

This is a number of functions to support the raytracing. Certain variables must be defined externally: Dcoef, Fcoef, Acoef, and Zfocus.

Function: rpar(z,phi)

returns the radius of the parabolic mirror surface including mirror deformation, if requested (assumes that Dcoef and Zfocus have been defined).

Function: rhyp(z,phi)

returns the radius of the hyperbolic mirror surface including mirror deformation, if requested (assumes that Fcoef, Acoef, and Zfocus have been defined).

Function: rcon(z,phi)

returns the radius of the conical mirror surface including mirror deformation, if requested (assumes that Mirror_angle and R1_mirror have been defined).

Function mdist(funcname, C)

returns the distance to the mirror (defined by 'funcname' and the related external variables) from the point in space defined by 'C'. Returns a positive value if inside mirror, negative otherwise.

Function impact( funcname, z1, z2, S, R )

calculates the interaction point of the ray defined by starting position S and direction vector R. The mirror is to be found between z1 and z2 ( z1 < z2 ). A vector of four elements: position plus final distance is returned.

Function deriv( funcname, x, phi )

returns the partial derivative of the function in x by setting dx = 1.

Function deriv2( funcname, x, phi )

returns the partial derivative of the function in phi by setting dphi = 0.01745 (corresponding to 1°)

Function normal_vector( funcname, x, phi )

returns a normalized normal vector the the (almost) axially symmetric surface described by 'funcname'.

Function spec_reflect( R, N, Q )

Calculates the specularly reflected ray as Q when the incoming ray is R and the surface normal is N. The returned value is the grazing angle in radians.

Function scat_reflect( R, N, scat_angle, Q )

Calculates the scattered reflected ray as Q when the incoming ray is R and the surface normal is N. The scattering angle scat_angle [rad] is measured from the specular reflection with negative values towards the reflecting surface and with positive values away from the reflecting surface. The returned value is the grazing angle in radians.

### 6.2.22 mt_get_mirror_eff_factors (U)

---

Calling:

```
mt_get_mirror_eff_factors
```

(takes no arguments) calculates the efficiency factors for each mirror shell by taking the reduction due to spokes into consideration. The external variable 'Mirror_eff_factors' (an array with as many elements as there are mirror shells) is evaluated from the spoke definition files. If present this array is used by the 'mt_eff_area_quick' function.

### 6.2.23 mt_get_rcoef

---

Calling:

```
coef = mt_get_rcoef( energy, angle)
```

Arguments:

*energy*: Energy of photon (keV)

*angle*: Grazing angle (rad)

interpolates in the currently loaded scatter table to return the coefficient of reflection (energy in keV and angle in radians).

### 6.2.24 mt_load (U)

---

Calling:

```
mt_load
```

Keywords:

*omfile*: Name of optical module file to load

*scatfile*: Name of scatter file to load

*mdeffile*: Name of mirror deformation file to load

*makelog*: ??

*chat*: Chatter level requested

Loads the necessary arrays for an optical module, a scatter file, and/or a mirror deformation file into memory as external variables.

If the keyword 'omfile' is a string with an optical module file name then the external variables for the module are filled.

If the keyword 'scatfile' is a string with a scatter file name then the external variables for the scatter information are filled.

If the keyword 'mdeffile' is a string with a mirror deformation description file name then the 'Mirror_deform_arr' variable is updated.

### 6.2.25 mt_mirdiag (U)

---

Calling:

```
mt_mirdiag
```

Keywords:

*rr*: 2 element array with radius limits in mm

*zr*: 2 element array with z limits in mm

*z_offset*: Offset in mm of the optical module in the z direction *over*: Boolean for overplotting on existing frame

*phi*: Azimuth angle in radians to show mirror surface

plots a diagram of the mirror configuration with nominal mirror surface and the actual surface if mirror deformations have been defined. Red and blue lines for the baffles are added as well.

#### 6.2.26 mt_mirplot (U)

---

Calling:

```
mt_mirplot, mirror_number, phi
```

Arguments:

*mirror_number*: Number of the requested mirror

*phi*: Azimuth angle in radians

Keywords:

*yr*: Set the plotting range in y (a two element array)

*nz*: Number of z-values to be used for the curve

plots the mirror surface as a function of z.

#### 6.2.27 mt_mk_mdeform_file (U)

---

Calling:

```
mt_mk_mdeform_file, filename, mode, param
```

Arguments:

*filename*: Output file name

*mode*: Type of mirror deformation, read specific help

*param*: A parameter to control the magnitude of the deformation, depends on 'mode'

Keywords:

*naz*: Number of azimuth values

*nz*: Number of z values

Produces a FITS file with mirror deformations for an example with a parabolic deformation in z (no azimuth dependence). This routine is quite limited and real mirror deformations must be handled outside this package.

#### 6.2.28 mt_photpr (U)

---

Calling:

```
mt_photpr, iphot
```

Arguments:

iphotIndex of photon in array 'Phs'

prints the contents of the external struct 'Phs' for photon number 'iphot' in a nice way on the terminal.

#### 6.2.29 mt_pre_def_photons

---

Calling:

```
mt_pre_def_photons, fraper, energy_or_file, src_offaxis, src_azim
```

Arguments:

*fraper*: Front aperture array (see text)

*energy_or_file*: Single energy value or a FITS file with specifications

*src_offaxis*: [arcmin] Off axis position of source

*src_azimuth*: [degrees] Azimuth angle of source position

Keywords:

*dphot*: [/mm$^2$] Density of infalling photons

*cont*: Boolean for ...

is called by 'mt_run' to define the photons to be raytraced. The first argument 'fraper' (for Front Aperture) must be a two or four element array, where the two first elements are the inner and out radii. If a our element array is used then the two last elements are the limiting azimuth angles of the front aperture in radians. The surface density of photons is given by the keyword 'dphot' (unit: mm$^{-2}$, default value is 1.0). The *energy* is to be given in *keV*, the angle for the source position away from on-axis, *src_offaxis* in *arcmin*, and the source azimuth angle, *src_azim*, in *degrees*.

If the third argument 'energy_or_file' is a scalar number or the name of a photon flux file then the control is immediately given to 'mt_def_photons' (see page 25 where the other arguments are explained).

'energy_or_file' can be the name of a sky definition file (SDF) that can define multiple celestial sources in the FOV. 'mt_def_photons' will be called for each of these sources to build up a complete array with input photons. See section 10, 11, and 12 for the required file format.

#### 6.2.30 mt_propagate (U)

---

Calling:

```
mt_propagate, new_z_value
```

Arguments:

*new_z_value*: [mm] New z-positions for the photons in 'Phs' places all photons in the external variable 'Phs' at the new z value (measured from the focal plane). Useful if e.g. the detector is not placed in the focal plane as when doing laboratory experiments.

### 6.2.31 mt_qimage

---

Calling:

```
mt_qimage, dz
image = mt_qimage(dz)
```

Arguments:

*dz*: Z-position of the image plane

Keywords:

*size*: [mm] Width and height of the image

*dim*: Image dimension (a square image is produced)

*cen*: [mm] A two-element array that defines the image center. May be given as a scalar zero, in which case the image center is (0,0)

*bsel*: Selection on the 'bounce' value

*lg*: A Boolean for logarithmic image representation

*win*: Index of plotting window

*pal*: Name of palette to use

If called as a subroutine it will show an image (i.e. the 2D distribution of the photons) in the plane perpendicular to the optical axis at the z-value '*dz*' (that defaults to zero, i.e. in the focal plane). If called as a function, the image will be returned. The input data are the *status*==0 photons in memory (external variable 'Phs'). Keyword *size* is a scalar with requested image size in mm, *dim* is an integer scalar with requested image size in pixels, and *cen* can be zero for centering on (0,0), a non-zero scalar to center on the average position, or a 2-element array with coordinates of the center of the requested image.

The keyword 'bsel' can be used for a selection on the 'bounce' flag. If 'rcoef' is set, then the photons will be weighted with the reflection coefficients for a better representation of the focal plane image.

When called as a subroutine the keywords 'lg', 'win', and 'pal' can tell if the image is wanted in a logarithmic colorscale, what window pane is requested, and, finally, what palette is to be used. The general purpose '*disp*' subroutine is used to present the image.

### 6.2.32 mt_rayplot (U)

---

Calling:

```
mt_rayplot, index_of_photon, module_number
```

Arguments:

*index_of_photon*: Index in 'Phs' array

*module_number*: Number of optical module

Keywords:

*over*: Boolean for overplotting on existing plot

*z_offset*: [mm] A shift of z-values e.g. to adapt to existing plot

*rr*: [mm] (2-element array) Set the range of radius values for the plot

*extend*: Boolean for extending ray lines beyond the optical module

Plot the path of the chosen photon through an optical module. Keyword 'over' will cause an overplotting, 'z_offset' is an additive coordinate change in Z, 'extend' will show more of the photon path at entrance (extend $> 0$) or at exit (extend $< 0$).

### 6.2.33 mt_raytrace_module

---

Calling:

`mt_raytrace_module` Keywords:

*no_scatter*: Boolean to exclude scattering

*no_mdeform*: Boolean to exclude the mirror deformations

*chat*: Level of chattiness

Follows each photon in the current 'Phs' array through the mirror module previously loaded with 'mt_load'. The photon position is updated to where the reflection (if any) happens. The struct member 'rcoef' is multiplied by the reflection coefficient for the energy and incidence grazing angle. The struct member 'status' is changed to a positive value if the photon misses the entrance slit, the exit slit, or the mirror.

The status value description:

| | |
|---|---|
| **1** | stopped by a spoke at entrance |
| **2** | stopped at entrance by baffle slot inner edge |
| **3** | stopped at entrance by baffle slot outer edge |
| **4** | hit neighboring mirror edge |
| **5** | hit mirror edge |
| **6** | into backside of neighboring mirror |
| **7** | second reflection on the same mirror |
| **8** | exits module "behind" the mirror |
| **9** | stopped at exit by baffle slot inner edge |
| **10** | stopped at exit by baffle slot outer edge |
| **11** | stopped by a spoke at exit |

When the blocking happens in the second optical module 100 will be added to the above values.

The struct member 'bounce' contains a binary code for occurrences of bounces (or reflections) as:

| | |
|---|---|
| **0** | no reflection happend |
| **1** | reflection only in first module |
| **2** | reflection only in second module |

**3** reflection in both modules

It updates the 'Phs' array that contains the information described in section 7.1 (page 44).

#### 6.2.34 mt_reflplot (U)

---

Calling:

```
mt_reflplot
```

Makes a plot of the currently loaded reflection coefficients. Two examples from preliminary NuSTAR scatter files are shown in figure 8 as a function of energy for all angles present in the scatter file.

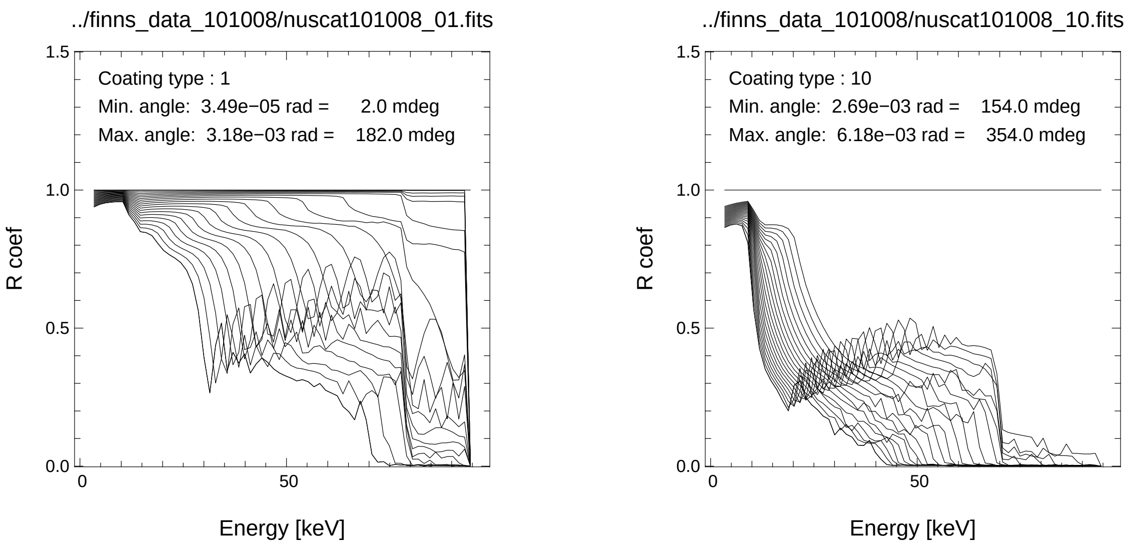


Figure 8: **Left panel:** Reflection coefficients for the coating of the innermost NuSTAR (preliminary) mirrors. **Right panel:** Same but for outermost mirrors.

#### 6.2.35 mt_run (U)

---

Calling:

```
mt_run, energy_or_file, src_offaxis, src_azimuth
```

Arguments:

*energy_or_file*: A scalar photon energy [keV] or a FITS file

*src_offaxis*: [arcmin] Off axis angle of the X-ray source

*src_azimuth*: [degrees] Azimuth angle of the X-ray source

Keywords:

*dphot*: [/mm$^2$] Photon density on the aperture plane

*samp*: QQQ

*no_scatter*: Boolean to exclude scattering

*no_mdeform*: Boolean to exclude the mirror deformations

*chat*: level of chattiness

*fraper*: Front aperture array

*flag*: Boolean for stopping raytracing after first optical module

*labxoff*: Offset in X-direction of laboratory source

*labyoff*: Offset in Y-direction of laboratory source

runs a raytracing session for a defined energy and source position ('src_offaxis' (arcmin) and 'src_azim'(degrees)).

The first argument is either a scalar number that defines the energy (in keV) of the photons or a scalar string giving the filename of a photon flux file (section 7.11), a source list file (section 7.12), a sky definition file (section 7.13), or a laboratory source defining file.

The photon density at the telescope i.e. photons per mm$^2$ can be defined by the keyword 'dphot' (default is 1.0). Only used if a specific energy is given ('energy_or_file is a scalar number).

Scattering can de unselected by setting the keyword 'no_scatter' and similarly mirror deformation can be avoided by setting the keyword 'no_mdeform'.

Setting the keyword 'chat' to a value larger than 0 will cause extra output on the screen with running information. The keyword 'flag' will cause the raytracing process to stop after the first optical module (mostly used for diagnostic purposes).

The keyword 'fraper' (front aperture) is a two or four element array of numbers:

[r_inner, r_outer] or [r_inner, r_outer, azimuth1, azimuth2]

In the two-element case there is no limitation in azimuth. Figure 9 shows an example of such a front aperture slit.

The position of a laboratory source (at a finite distance) can be off-set by the keywords 'labxoff' and 'labyoff'.

When this function has been executed the array 'Phs' will contain photons propagated to the focal plane.

#### 6.2.36 mt_save

Subroutine: mt_save, mode=, samp=

Saves various parts of the external variables to a FITS files and to a Yorick 'save' file. The keyword 'mode' controls the selection. It must be a string with one or more of the letters "f", "p", "e", or "y". If 'mode' is omitted then all four will be assumed.

**f**: the focal plane image weighted with the reflection coefficient is written to a FITS image file 'focal_plane_*nnnn*.fits', see section 7.10.

**p**: all photon data (or sampled with 'samp') are written to 'photons_*nnnn*.fits'.

**e**: all event data (or samled with 'samp') are written to 'events_*nnnn*.fits'.

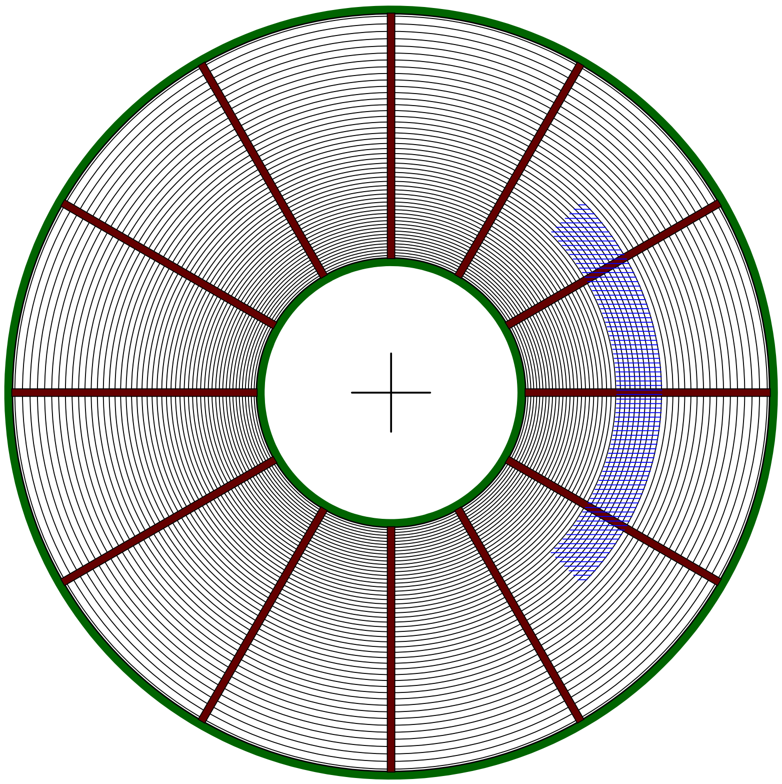


Figure 9: A schematic illustration of the front of an optical module with spokes. A limited front aperture is shown as the blue hatched area with the 'fraper' keyword set to [140,180,5.798,7.066] (note that the second azimuth value must be larger than the first one).

**y**: most external variables are saved to 'ysession_*nnnn*.ysav'.

Here *nnnn* is a serial number which is written to the file 'mt_serial.txt' in the current working directory.

A sampling for writing the photons to the photon file can be defined by keyword 'samp' (default is 1) in order to save disk space. It means that 1 out of 'samp' photons will be put in the photon file. Similarly for the event file.

#### 6.2.37 mt_scatter_data_file

Subroutine: mt_scatter_data_file, dir, template, coating, fwhm=, outfile= \
dist_angle_max=, unit=, graze_angle_max=, gunit=, skip=, attenuate=

This function produces a FITS scatter table in the format required by *mt_load* from text file input. A standard gaussian scattering distribution (the width of which is defined by keyword *fwhm*) is added to all table rows.

The text file names are expected in a given format: *TTTTnn_x.xxx* where *nn* is the coating identification number (two digits) and *x.xxx* is the grazing angle in degrees. *TTTT* is the string given as keyword: `template` There should be at least two data columns, the first with energy in keV and the second with the corresponding coefficient of reflection.

An example, a file by name of 'IXO_ml00_0.580' is shown here:

```
;
;  IMD Data
;
;  Saved on Fri Apr 15 14:01:20 2011
;
;  Structure:
;
;  B4C layer (1), z=80.00 A, sigma=4.00 A (err. fun.)
;  Ir layer (2), z=100.00 A, sigma=4.00 A (err. fun.)
;  SiO substrate, sigma=4.00 A (err. fun.)
;
;  delta(lambda)=0.0025 keV
;
;  Theta=0.5800 deg
;
;  E [keV]  R
;-------------
      0.10000000        0.0000000
      0.25050505       0.91731060
      0.40101010       0.92592556
      0.55151515       0.93643846
      0.70202020       0.94280731
      0.85252525       0.94734738
       1.0030303       0.95132724
       .
       .
       .
       13.795960     0.0079416739
       13.946465     0.0063930611
       14.096970     0.0049206135
       14.247475     0.0036044061
       14.397980     0.0025267925
       14.548485     0.0017089891
       14.698990     0.0011724647
       14.849495    0.00092233064
       15.000000        0.0000000
```

All files with identical coating number are expected to have identical energy table values since the output FITS file defines a regular grid of energy and grazing angle values.

*mt_scatter_data_file* will select files with the correct coating type based on the 'template' string and do the data reformatting. An artificial extension of the grazing angle range is defined in keyword *graze_angle_max* (with affilated keyword *gunit*) will replicate the table with the largest angle value in order to accommodate extraordinarily large grazing angle reflections.

In some cases the first values in the table give incorrect reflection coefficient values; they can be skipped by keyword *skip*. Also the number of energy values can be quite large and an attenuation might be appropriate to avoid too large files that will slow down the process. Hence the keyword *attenuate*.

It is very likely that this function will require some editing with a new text file format.

### 6.2.38 mt_sel_scatter

Function: distribution = mt_sel_scatter( energy, angle_in, >rcoef )

returns bi-linearly interpolated scatter distribution, normalized to sum = 1. The coefficient of reflection is returned in the last argument (here 'rcoef') and the corresponding angles can be found in the external 'Anglesarr'.

### 6.2.39 mt_setup_system

Subroutine: mt_setup_system, system_defining_file

Imports the system as defined in *system_xxx.scm* (standard but not mandatory name), where *xxx* is a suitable identifier string.

### 6.2.40 mt_spoke_blocking

Subroutine: flag = mt_spoke_blocking( xy_array )

Returns 1 (one) if the photon hits a spoke at position 'xy_array', a two-element array. Otherwise it returns 0 (zero).

### 6.2.41 mt_spoke_read

Subroutine: mt_spoke_read, filename, pos=

Reads the spoke definition file in .scm format (see section 7.7) and sets up the appropriate external variables (*Phi_spokes1, Width_spokes1, Reverse_spokes1* for entrance spokes and *Phi_spokes2, Width_spokes2, Reverse_spokes2* for exit spokes. The external variable *Spoke_define_files* holds the filenames. There are two spoke definition files per optical module (may be identical) or none.

The keyword 'reverse' in the spoke definition file will cause a reverse action of the spokes so that photons are allowed through the spokes but not elsewhere. This may be used for checking if the definition of the spokes works as expected.

### 6.2.42 mt_skyima2skyspec

```
Subroutine: mt_skyima2skyspec, dol_skyima, dol_arf, emin, emax, outfile, \
                                     sc=, nh=, p1=
```

Uses a skyimage from an X-ray observation together with its ARF. The image must be in countrates per pixel to produce a sky spectral definition file which is a FITS file with one or more extensions:

A map of normalization factors, a map of the spectral parameters, and a map of column densities. The last two may reduce to keywords in the first extension if a constant value is to be used. The ancillary response file (ARF) from the X-ray observing instrument is copied to the new file.

The output of this function only depends on the spectral models and the image producing instrument.

### 6.2.43 mt_skyspec2skydef

Subroutine: mt_skyspec2skydef, sky_spec_def_file, skydef_file, ra_scx, dec_scx, \
posang, lim=, exposure=, e1=, e2=, nchan=, \
radius=, fluxdir=, mission=, instrume=, telescop=

This function makes a sky definition file and corresponding photon files adequate for the instrument to be simulated.

### 6.2.44 mt_upd_om_coating

Subroutine: mt_upd_om_coating, coating_table, filename

Updates the column with coating information in an optical module description file (section 7.3, page 45) '*filename*'. The input is taken from a text file in '.scm' format, '*coating_table*', which is an list of coating numbers, see section 7.5 page 46.

# 7 Data organization

## 7.1 Content of memory

The information of the photons used in the raytracing is kept in a number of external variables as described in the appendix section 19, page 67.

Running *mt_run* creates an array by name of 'Phs' of the struct: 's_Ray' with members:

| | |
|---|---|
| **E** | photon position (array of 3) |
| **R** | photon direction (array of 3) |
| **angle_in1** | grazing angle of incoming ray of first reflection |
| **angle_in2** | grazing angle of incoming ray of second reflection |
| **angle_out1** | grazing angle of outgoing ray from first reflection |
| **angle_out2** | grazing angle of outgoing ray from second reflection |
| **rcoef** | reflection coefficient |
| **energy** | energy in keV |
| **mirror** | mirror number |
| **status** | photon status indicator |
| **bounce** | binary flag for photon reflections |
| **E1** | position at entrance of first module (array of 3) |
| **E2** | position at entrance of second module (array of 3) |
| **I1** | position at first reflection (array of 3) |
| **I2** | position at second reflection (array of 3) |
| **D1** | direction before first reflection (array of 3) |
| **D2** | direction before second reflection (array of 3) |

It means that the user can easily access this information upon completion of a raytracing run.

After a call of 'mt_detector' an event list is found in the array 'Evlist' that is an array of struct 's_MTEvent' with members:

| | |
|---|---|
| **rawx** | (int) Pixel number on the detector in x |
| **rawy** | (int) Pixel number on the detector in y |
| **detx** | (float) Position on the detector in x |
| **dety** | (float) Position on the detector in y |
| **pha** | (int) PHA (or PI) value |
| **energy** | (float) [keV] Energy assigned to event |
| **bounce** | (int) binary flag for photon reflections |
| **flag** | (int) Optional use (the background has a zero) |

## 7.2 Telescope system definition files

The complete telescope system is defined in a text file in .scm format. A number of keywords will give specific parameters or names of auxiliary files with additional information.

The following keywords give the information for the complete system and they must be found in the telescope system definition file:

**num_modules** Number of optical modules in the telescope system (1 or 2).

**detector__descr_file** Name of file with detector information.

The following keywords must be defined for each optical module in correct order with parabolic (or parabolic-like) first:

**om_type** Can be either `parabolic`, `hyperbolic`, or `conical`.

**z_reference** The $z$ coordinate of the reference plane in the telescope system.

**Zfocus** The $z$ coordinate of the focus in the coordinate system of the optical module itself.

**r_outer** The radius of the outer mirror shell at the aperture (only to be given for the master module).

**r_inner** The radius of the innermost mirror shell at the aperture (only to be given for the master module).

**om_file** The name of the optical module describing file.

**mirror_thickness_file** The name of the file with a table of mirror thickness as a function of radius. The values are found by linear interpolation.

**mirror_deform_file** The name of the deformation description file.

**spoke_define_file** The name of the spoke definition file. If this one is given the same spokes will be used for entrance and exit of the optical module.

**spoke_define_file_entry** The name of the spoke definition file for the entrance of the optical module.

**spoke_define_file_exit** The name of the spoke definition file for the exit of the optical module.

The reflection/scattering files are given by one or more of the 'scat_file' keywords. The one to use for a given mirror is identified by the coating type number which is to be found in a column of the 'om_file' and as a FITS keyword in the scattering file.

**scat_file** A scatter property describing file (see section 7.6) to be considered. The coating type is used to identify which one is appropriate for a given mirror.

## 7.3 Optical module description files

The description of an optical module is (usually) found in a file by name of 'om_abc_NNN.fits' where NNN is a serial number and 'abc' characterizes the geometrical type. Each row corresponds to a mirror shell. The table is found in FITS extension one as:

| Column name | Unit | Description |
|---|---|---|
| R1 | mm | Radius of reflecting surface at Z1 |
| R2 | mm | Radius of reflecting surface at Z2 |
| Z1 | mm | Z coordinate of mirror aperture |
| Z2 | mm | Z coordinate of mirror inner edge |
| DCOEF* | mm | Coeffient describing a parabolic surface |
| ACOEF* | mm | Coeffient describing a hyperbolic surface |
| MIRROR_ANGLE* | rad | Mirror angle for a conical mirrors |
| MLENGTH | mm | Mirror length |
| MTHICK | mm | Mirror thickness |
| COATING | | Number identifying the coating type |
| RB1I | mm | Inner radius of entrance slit |
| RB1O | mm | Outer radius of entrance slit |
| RB2I | mm | Inner radius of exit slit |
| RB2O | mm | Outer radius of exit slit |
| ZB1I | mm | Z value of inner radius of entrance slit |
| ZB1O | mm | Z value of outer radius of entrance slit |
| ZB2I | mm | Z value of inner radius of exit slit |
| ZB2O | mm | Z value of outer radius of exit slit |

* Only the appropriate column is included.

The column 'COATING' controls the scattering and reflection properties of the mirror shell. It can be changed by using the function 'mt_upd_om_coating' (section 6.2.44, page 43) with input from an ASCII coating table (see below).

## 7.4 Mirror thickness file

A text file in .scm format with two columns: radius and mirror thickness.

The keyword lines:

```
    // colname = radius
    // colname = mirror_thickness
```

**must** be present.

Both must be given in *mm*.

## 7.5 Coating tables

A text file in .scm format with two columns: mirror number and coating number. (Remember that mirror numbering starts from the outside).

Mirror numbers are assumed to be increasing and consequtive so missing numbers are filled in with the latest coating number.

The keyword lines:

```
    // colname = mirror
    // colname = coating
```

**must** be present.

An example where e.g. mirrors 16 thru 29 will be assigned coating 9, is shown below:

```
//    Coating table for Nustar   2011-01-04
//
//    10 recipes as in /home/njw/yorick/mraytrace/finns_data_101008
//
// colname = mirror
// colname = coating

   1   10
  16    9
  30    8
  45    7
  58    6
  72    5
  85    4
  98    3
 110    2
 122    1
```

## 7.6 Scatter data files

The data input for the scattering properties can in principle have many different formats and the user must be prepared to build some kind of interface to produce the scatter files as described below.

An example of such an interface is given by the function 'mt_scatter_data_file' (page 40).

The format has been chosen to make it possible to handle all kinds of scattering distributions. When making use of the data in the raytracing scheme an interpolation between values is performed. In order to ease this process it is required that the energy and grazing angle values are given in a regular grid.

The scattering properties depend (possibly) on the coating type and there must be a file for each coating.

The table must be stored as a binary table as the first FITS extension where the header contains the following keywords:

| **Keyword name** | **Value** | **Unit** | **Description** |
|---|---|---|---|
| EXTNAME | SCATTER_FILE | | The name of the extension |
| COATING | (appropriate integer) | | The coating identifier |

The table itself must include these columns:

| **Column name** | **Unit** | **Description** |
|---|---|---|
| ENERGY | keV | Energy of photons |
| ANGLE_IN | deg | Grazing angle |
| R_COEF | | Reflection coefficient |
| DISTRIBUTION | | (Vector column) Scattering distribution |
| DATA_ORIGIN | | String describing the origin of the data |

The table itself one has an exception in row one, where the column 'DISTRIBUTION' holds the scattering angles (in radians) corresponding to the distribution. The other information in the first row is therefore meaningless.

## 7.7 Spoke definition files

The keyword 'spoke_define_file' in the telescope system file is used to indicate the spoke definitions is to be found. If it is omitted then no spokes will be included. Since entrance spokes and exit spokes may not be identical then the 'spoke_define_file' keyword must appear twice for each optical module as entrance and exit resp. or not at all.

The spoke data are expected in .scm format with keywords 'angle_unit' and 'width_unit'. There are only two options for 'angle_unit', namely 'deg' or 'rad'. The 'width_unit' can be 'mm' or 'cm'.

There can be two or four table columns in the file depending on if radial information is required as well. The two columns with angle and width information must be identified by keywords 'colname' as 'angle' and 'width'. The radial information is in columns named 'rstart' and 'rstop' – if present.

In this version (4.6.5) of MT_RAYOR the spokes are assumed to have a constant width (independent of radius) and full extent from inner radius to outer radius if 'rstart' and 'rstop' are not given, else these are used to confine the spokes in the radial direction. This implies that several spokes can be defined with the same angle.

It is assumed that no overlapping spokes have been defined.

More elaborate models will be constructed when the need arises.

An example is shown here:

```
//
// Spoke definitions for Nustar  1-alpha-top
//
//  angle_unit = deg;   mandatory keyword
//  width_unit = mm;    mandatory keyword
//
//  reverse = 0
//
//  colname = angle;    mandatory keyword
//  colname = width;    mandatory keyword
//  colname = rstart;   mandatory keyword
//  colname = rstop;    mandatory keyword
//
//  angle     width     rstart     rstop
//
      0.0       7.2     50.000    55.805; inner three layers
      0.0       6.8     55.805   191.751
      7.5       2.2    104.784   191.751
     15.0       2.6     50.000    55.805
     15.0       2.2     55.805   191.751
     22.5       2.2    104.784   191.751
     30.0       2.6     50.000    55.805
     30.0       2.2     55.805   104.784
     30.0       6.8    104.784   191.751
     37.5       2.2    104.784   191.751
     45.0       2.6     50.000    55.805
     45.0       2.2     55.805   191.751
     52.5       2.2    104.784   191.751
     60.0       7.2     50.000    55.805; inner three layers
     60.0       6.8     55.805   191.751
     67.5       2.2    104.784   191.751
      .
 (continues the full circle)
      .
```

## 7.8 Deformation description files

The keyword 'mirror_deform_file' in the telescope system file is used to indicate where the deformation information is to be found. It can be omitted if no deformations are to be included.

The deformation data are expected in a FITS image extension (must be the first in the file apart from the primary header data unit) in a three dimensional image where the first dimension is azimuth, the second one is z, and the third is mirror number. In this way each slice is a map of deviations in radius from the ideal surface measured in *mm*.

The convention adopted for the sign is that

$$r_{true} = r_{ideal} - \delta r_{deform} \tag{6}$$

where $\delta r_{deform}$ is the value obtained by linear interpolation in the mirror 'map'.

The azimuth values (the first dimension) are taken to be distributed evenly between zero and $2\pi$.

Similarly the z values (the second dimension) are assumed to be equidistant between the lower and upper edge.

## 7.9 Photon save files

A raytracing section consists of sending a number of photons towards the opening of the telescope. The photons are held in the memory and by a call of 'mt_save' (page 39) they will be written to a binary table FITS file by name 'photons_NNNN.fits' where NNNN is a serial number. Keywords in the header explain what telescope defining file has been used.

Irrespective of a double or single reflection telescope the photons are described in the FITS extension one as:

| Column name | Unit | Description |
|---|---|---|
| DETX | mm | X coordinate of count on focal plane |
| DETY | mm | Y coordinate of count on focal plane |
| RAYX | | X direction of ray at focal plane |
| RAYY | | Y direction of ray at focal plane |
| RAYZ | | Z direction of ray at focal plane |
| ANGLE_IN1 | rad | Grazing angle in-coming |
| ANGLE_OUT1 | rad | Grazing angle out-going |
| ANGLE_IN2 | rad | Grazing angle in-coming |
| ANGLE_OUT2 | rad | Grazing angle out-going |
| MIRROR | | Mirror number |
| RCOEF | | Reflection coefficient |
| AZIMUTH | rad | Azimuth angle at entrance of photon |
| ENERGY | keV | Photon energy |
| BOUNCE | | Flag for what reflections occurred |
| I1Z | mm | Internal first module z-value where reflection happened |
| I2Z | mm | Internal second module z-value where reflection happened |
| STATUS | | Describes fate of photon (0 means OK) |

Calling 'mt_save' may also produce a focal plane image in 'focal_plane_NNNN.fits', an event file 'events_NNNN.fits', as well as saving the entire Yorick session in 'ysession_NNNN.ysav', where NNNN is the same as for the photon file.

## 7.10 Photon image files

The focal plane image is put into a FITS file by name 'focal_plane_NNNN.fits' where NNNN is a serial number by the *mt_save,mode="f"* command.

The image size is defined by keywords 'Dim_focp' and 'Pix_focp' defined in the system defining file (see later).

## 7.11 Photon flux files

A standard format for representing the source photon spectrum has been defined for realistic simulations.

The source photon flux is described by a FITS binary table in extension number 1. The header must have the following keyword included:

| Keyword name | Value | Description |
|---|---|---|
| EXTNAME | PHOTON_FLUX or DXB_PHOTON_FLUX | The name of the extension |

The FITS columns of the binary table must be:

| Column name | Unit | Description |
|---|---|---|
| ENERG_LO | keV | Lower boundaries of the energy channels |
| ENERG_HI | keV | Upper boundaries of the energy channels |
| PHOTFLUX | photons $\mathrm{cm}^{-2}\,\mathrm{s}^{-1}\,\mathrm{keV}^{-1}$ | Photon flux values |

Then the appropriate number of photons is calculated (the keyword 'exposure' (in seconds) must be supplied in the call of *mt_run*) and the energies are sampled from the distribution given by the flux.

A function, *mk_photflux*, to produce such photon flux files for simple standard spectra such as power law, thermal bremsstrahlung, and blackbody radiation, can be found in the auxiliary package 'xray.i'.

### 7.12 Source list files

A number of sources can be defined in a FITS file with the following format:

The header of the first extension that must be a binary table must have the following keyword included:

| **Keyword name** | **Value** | **Unit** | **Description** |
|---|---|---|---|
| EXTNAME | SKY_DEFINITION | | The name of the extension |
| RA_SCX | | deg | Right Ascension of boresight |
| DEC_SCX | | deg | Declination of boresight |
| POSANG | | deg | Position angle of telescope (satellite) |
| EXPOSURE | | s | Exposure time for suggested observation |

The FITS columns of the binary table must be:

| **Column name** | **Unit** | **Description** |
|---|---|---|
| X_SKY | arcmin | Source position along x-axis |
| Y_SKY | arcmin | Source position along y-axis |
| DPHOT | mm$^{-2}$ | Photon density on the entrance aperture |
| RENORM | | Renormalization factor to apply |
| DOL | | DOL of photon flux definition FITS extension |

Each of the photon definition DOLs must have an extension name as indicated in section 7.11.

### 7.13 Sky definition files

FITS file with several extension describing the intensity map and maps of the spectral parameters.

# 8 Mirror geometry

The basic coordinate system for a telescope system has its origin in the focal plane of the telescope and the Z axis along the telescope axis of symmetry. The photons will have direction vectors (small, small, close to -1.0).

The unit of length is consistently *mm*.

In the ideal case the formula $r = f(z)$ is assumed to describe the mirror shell reflecting surface. With $df/dz = f'(z)$ then the normal vector in the $(r, z)$ system is $(-1, f'(z))$. When mirror deformations are considered as well then the radius is also a function of the azimuth angle: $r = f(z, phi)$ and the derivation of the normal vector is presented in section 17.1.

# 9 Cookbook

The user is guided through two example sessions step by step with additional explanations.

The used character font indicates the function in the dialogue:

`written by shell or program`. *To be entered by user.*
Comments and remarks

## 9.1 Download and installation

It is assumed that the user is in possession of the file: *mt_rayor-4.6.5.tar* that contains the MT_RAYOR software itself but also a number of auxiliary packages necessary for the execution of MT_RAYOR functions.

The downloading and installation of Yorick for Linux and Windows platforms is explained in the appendix section 20 (page 70) and the installation of the *mt_rayor-4.6.5* package in section 21 (page 70).

## 9.2 The GRI system with a single optical module

This subsection demonstrates the steps right from the beginning to produce some raytracing results for a single reflection telescope such as the one proposed for the GRI mission.

In the GRI geometry the spacing between the successive mirror shells is kept constant. Then the mirrors need to get longer and longer as the radius decreases in order to fill out the entrance opening (thereby avoiding a 'leaky' module). This is implemented in `mt_create_om_par1` (see later).

The first example walks through a case where scattering is disregarded and therefore no coating table is required.

### 9.2.1 Set up the system definition file

> *cd /basdir/yorick/mraytrace*

A sample system definition file `system_gri.scm` has been provided in the *mt_rayor-4.6.5.tar* package. It contains a number of parameters for the telescope system definition and can be created/modified with a text editor.

A '.scm' style has been used. It has some similarities with the FITS header format (see section 22 page 71).

The file is reproduced here:

```
//   system_gri      2008-04-20/NJW
//
//  Single parabolic optical module as the one proposed for
//  the GRI project for high-energy gamma-ray focussing.
//
//  Focal plane is per definition z = 0
//
```

```
// num_modules = 1
// om_type = parabolic
// z_reference = 100000.0; mm
// Zfocus = -100000.0; mm
//
// r_outer    = 300.0   ; mm
// r_inner    = 100.0   ; mm
//
// mirror_length = -999.0; Implying N/A
// mirror_thickness_file = gri_mirror_thickness.scm
//
// focal_length = 100000.0; mm
//
//   Based on constant spacing between mirrors so that the
//   mirrors get shorter and shorter with increasing radius
//
// om_function = mt_create_om_par1
// om_parameter = 0.775; [mm], 'spacing' parameter
// om_file = /home/njw/yorick/mraytrace/om_par_01.fits
// mirror_deform_file = none
//
// Here follows a list of scatter information files.
// They can appear in any order:
//
// scat_file = /home/njw/yorick/mraytrace/scat_fake_10.fits
// scat_file = /home/njw/yorick/mraytrace/scat_fake_20.fits
// scat_file = /home/njw/yorick/mraytrace/scat_fake_30.fits
//
//  Focal plane pixel settings (only used by mt_save):
//
//  Dim_focp = 251; Number of pixels of focal plane detector
//  Pix_focp = 0.5; [mm] Pixel size
//
```

See 7.2, page 44, for an explanation of the mandatory keywords. Note, for example, that the `mirror_deform_file` has been set to 'none' which implies that it is no to be used.

### 9.2.2 Create the optical module file

Now it is time to invoke the Yorick scripting language:

```
mraytrace> yorick
Copyright (c) 2005.  The Regents of the University of California.
All rights reserved.  Yorick 2.1.05 ready.  For help type 'help'
                  Load all the required functions:
> #include "mt_rayor-4.6.5.i"
                  Setup while checking what is missing
> mt_setup_system,"system_gri.scm"
Number of modules in system :  1
Optical module file :  om_par_01.fits
Opt module file:  om_par_01.fits does not exist
You may want to create it by 'mt_create_om_par1,filename="om_par_01.fits"'
Scatter file :  scat_fake_10.fits does not exist
Scatter file :  scat_fake_20.fits does not exist
Scatter file :  scat_fake_30.fits does not exist
Warning level 30100
 One or more scatter files are missing!
 An optical module file is missing!
```

If scattering is not requested the warning can be ignored.
First job is to create a parabolic optical module file.
You can cut and paste the command given above

> *mt_create_om_par1, filename="om_par_01.fits"*

```
The parabolic system optical module file:  om_par_01.fits has just been created
It has 260 mirrors
and it might need a coating update:  mt_upd_om_coating,...
```

Repeat the setup command to check integrity and set external variables

> *mt_setup_system,"system_gri.scm"*

```
Number of modules in system :  1
Optical module file :  om_par_01.fits
Scatter file :  scat_fake_10.fits does not exist
Scatter file :  scat_fake_20.fits does not exist
Scatter file :  scat_fake_30.fits does not exist
Warning level 30000
```

Implying that everything is in order except for scattering

### 9.2.3 Running a case

> *mt_run, 50, 1, 45, dphot=0.03, chat=2, no_scatter=1*

Running with E=50 keV, 1 arcmin offaxis, 45 deg azimuth
Setting the 'no_scatter' keyword is important

```
Using optical module /home/njw/yorick/mraytrace/om_par_11.fits
   with 259 mirrors and Zfocus = -100000 mm
    7540 photons in total
    5169 with status 0
    1187 with status 1
     583 with status 2
       0 with status 3
     601 with status 4
       0 with status 5
```

The non-zero status values indicate an absorption in the module.
Due to the randomization the actual numbers may deviate from these.

### 9.2.4 Display the result

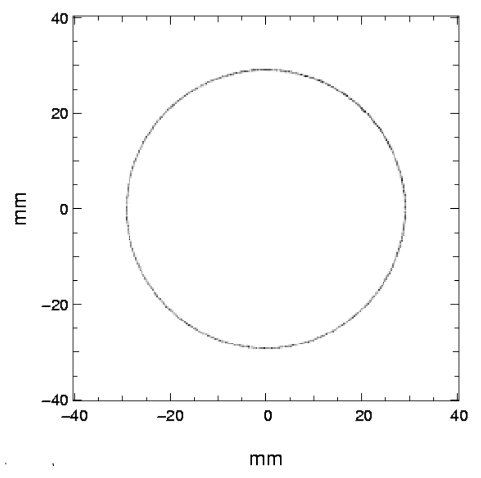
> *mt_qimage,size=80*

Making an image of the focal plane

An image similar to the left panel of Figure 10 here should be seen.

The appearance away from the focal plane can be inspected:

> *mt_qimage,5000,size=100*



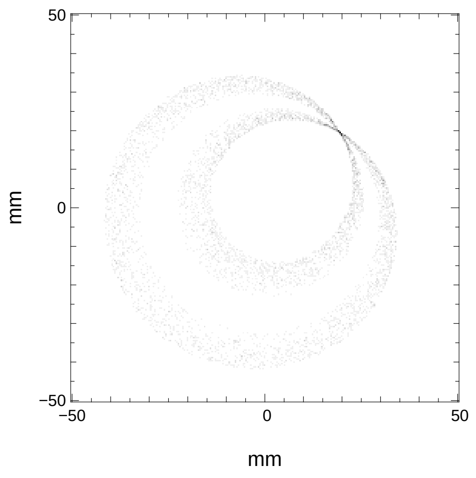


Figure 10: Left panel: Focal plane. Right panel: 5000 mm above focal plane

Making an image 5000 mm above the focal plane

### 9.2.5 Introducing scattering

It is assumed that the scattering properties depend on energy and the type of mirror surface coating. The scattering properties for a given coating are kept in a FITS file, see section 7.6, page 47. Such scatter data files may be created independently, but two functions are supplied for creating energy independent scatter data files.

One of them, *mt_fake_scatter_data*, introduces an ad hoc energy dependence of the reflection coefficient and a gaussian scatter profile. This is is useful for investigating the effect of various widths of the distribution on the loss of photons.

The other one, *mt_scatter_data_file*, reads a set of text files in .scm format and converts to FITS files in the proper format.

The following commands will produce the faked scatter files required for the example above:

> *mt_fake_scatter_data, "scat_fake_10.fits",fwhm=4.85e-5,coat=1*
The FWHM is given in radians (here equal to 10 arcsec)
> *mt_fake_scatter_data, "scat_fake_20.fits",fwhm=9.70e-5,coat=2*
The FWHM is given in radians (here equal to 20 arcsec)
> *mt_fake_scatter_data, "scat_fake_30.fits",fwhm=1.455e-4,coat=3*
The FWHM is given in radians (here equal to 30 arcsec)

The optical module file has a column for the proper coating to use for each mirror. It can be updated indepently but a function to do it from a simple text file has been supplied. The format of the text file is described in section 7.5, page 46. Two examples, 'mircoat_gri.scm' and 'mircoat_nustar.scm' come with this distribution.

> *mt_upd_om_coating, "mircoat_gri.scm","om_par_01.fits"*
Updates the COATING column in om_par_01.fits
> *mt_setup_system,"system_gri.scm"*

```
Number of modules in system :  1
```

```
Optical module file :  om_par_01.fits
Scatter file :  scat_fake_10.fits coating 1
Scatter file :  scat_fake_20.fits coating 2
Scatter file :  scat_fake_30.fits coating 3
Warning level 0
 System OK!
```

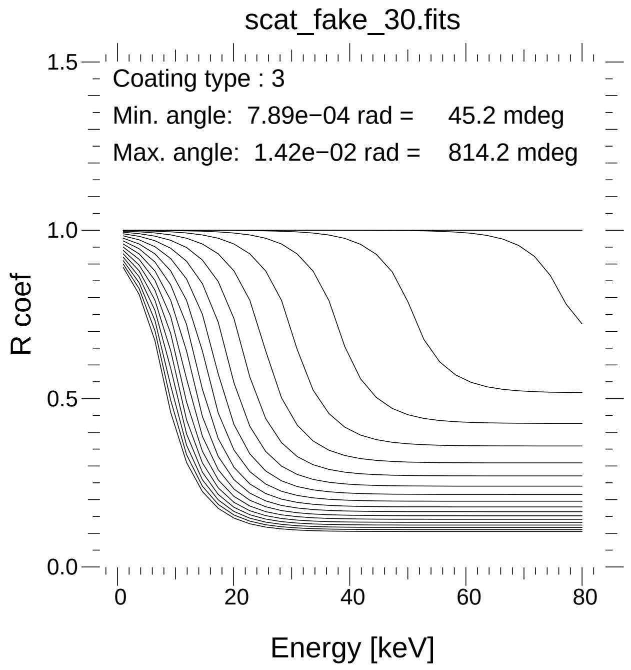
Should yield OK status

Then run the on-axis case

> *mt_run, 50, 0, 0, dphot=0.05*

The used coefficients of reflection can be shown:

> *mt_reflplot*



Evaluation of the HPD:

> *mt_analysis*

```
Analysis based on Phs with 6287 status==0 photons
   The focal length is 100.00 m
Center with all 10434 photons:  0.045 0.009
Improved center with 10245, 10251 photons:  0.033 -0.000
HPD : 4.129 mm <> 8.52 arcsec
```

## 9.3 Wolter I raytracing

> *mt_setup_system,"system_wolter1.scm"*

```
Number of modules in system :  2
Optical module file :  om_par_w1.fits
Opt module file:  om_par_w1.fits does not exist
You may want to create it by 'mt_create_om_par2,filename="om_par_w1.fits"'
```

```
Optical module file :  om_hyp_w1.fits
Opt module file:  om_hyp_w1.fits does not exist
You may want to create it by 'mt_create_om_hyp2,filename="om_hyp_w1.fits"'
Scatter file :  scat_fake_10.fits coating 1
Scatter file :  scat_fake_20.fits coating 2
Scatter file :  scat_fake_30.fits coating 3
Warning level 200
 An optical module file is missing!
```

First job is to create a parabolic optical module file

> *mt_create_om_par2, filename="om_par_w1.fits"*

The resulting optical module file name is reported

Next job is to create a hyperbolic optical module file

> *mt_create_om_hyp2, filename="om_hyp_w1.fits"*

The resulting optical module file name is reported

> *mt_upd_om_coating,"mircoat_wolter1.scm","om_par_w1.fits"*
> *mt_upd_om_coating,"mircoat_wolter1.scm","om_hyp_w1.fits"*

Run a case with an on-axis source

> *mt_run, 50, 0, 0, dphot=0.05, no_scatter=1"*

This will make a run launching 50 keV photons with a density of 0.05 $\mathrm{mm}^{-2}$.

## 9.4 Conical approximation raytracing

> *mt_setup_system,"system_conw1.scm"*

```
Number of modules in system :  2
Optical module file :  om_cpar_w1.fits
Opt module file:  om_cpar_w1.fits does not exist
You may want to create it by 'mt_create_om_con2,filename="om_cpar_w1.fits"'
Optical module file :  om_chyp_w1.fits
Opt module file:  om_chyp_w1.fits does not exist
You may want to create it by 'mt_create_om_con3,filename="om_chyp_w1.fits"'
Scatter file :  scat_fake_10.fits coating 1
Scatter file :  scat_fake_20.fits coating 2
Scatter file :  scat_fake_30.fits coating 3
Warning level 200
 An optical module file is missing!
```

First job is to create a conical optical module file

> *mt_create_om_con2, filename="om_cpar_w1.fits"*

The resulting optical module file name is reported

Next job is to create a second conical optical module file

> *mt_create_om_con3, filename="om_chyp_w1.fits"*

The resulting optical module file name is reported

> *mt_upd_om_coating,"mircoat_wolter1.scm","om_cpar_w1.fits"*
> *mt_upd_om_coating,"mircoat_wolter1.scm","om_chyp_w1.fits"*

Run a case with an on-axis source

> *mt_run, 50, 0, 0, dphot=0.05, no_scatter=1"*

This will make a run launching 50 keV photons with a density of 0.05 $\mathrm{mm}^{-2}$.

## 10 Point source simulation

In real astrophysical observations sources are not monochromatic. A representative source spectrum can be simulated by issuing a number of calls of *mt_run* with proper choices of the *dphot* keyword and energy values.

A facility to simplify this job has been built into *mt_run*. If the first argument is not given as a number but as a (scalar) string that must be the name of a photon flux file (see section 7.11) with intensities and energies are defined suitably.

The number of photons is derived from the given spectrum and their energies are sampled appropriately from the distribution.

## 11 Simulation with source list as input

For the simulation of a sky field with a number of sources in the Field-Of-View *mt_run* may be given as first argument the name of a source list file. It gives the positions of the sources in Right Ascension and Declination as well as their spectra, see section 7.12.

## 12 Simulation with sky image as input

The simulation of extended sources can be done by using an intensity image e.g. from a real observation. The idea is that each pixel is regarded as a separate source. In order to get a realistic image of a extended source the pixels must be sufficiently small i.e. smaller than the size of the FWHM of the PSF of the simulated observation. On the other hand, too small pixels may lead to long execution times and large requirements for intermediate file storage.

The simulation proceeds in two steps.

The first step is to define a spectral model for each source (pixel). A spectral model for each pixel (i.e. each source) in the image must be defined. The model is characterized by a normalizing factor, a parameter (photon index for a power law spectrum and temperature (kT) for a thermal bremsstrahlung or blackbody spectrum), and an absorbing column density. A map of the spectral parameter can be supplied in an array (FITS map) of the same dimensions as the sky image and similarly for the column density.

A sky spectral definition file is a FITS file with several extensions. The first extension is the normalization map.

**Case 1**: The pixel values in the input image is the countrate per pixel and per second for the instrument that has recorded the image.

With $f_i(E)$ as the photon flux apart from the normalizing factor $k_i$ for pixel $i$ and $A_{eff}(E)$ as the effective area (assumed to be the same for all pixels) then the count rate becomes

$$c_i = \int_{E_1}^{E_2} k_i f_i(E) A_{eff}(E) dE \qquad (7)$$

where $E_1$ and $E_2$ are the energy limits for detection. With usual units the photon flux, $k_i f_i(E)$, is in photons $\mathrm{cm}^{-2}\,\mathrm{s}^{-1}\,\mathrm{keV}^{-1}$ and energy is measured in keV and $A_{eff}$ in $\mathrm{cm}^2$.

**Case 2**: The pixel values are relative and the normalization must be found

Suppose the energy flux ($S_X$) is known for a part of or the entire image between energies $\epsilon_1$ and $\epsilon_1$:

$$S_X = 1.6\,10^{-9} \sum_i \int_{\epsilon_1}^{\epsilon_2} E\, k_i f_i(E) dE \tag{8}$$

The summation is over the appropriate set of pixels.

The normalization is provided by the ratio of the input sky image with the calculated count rate with unit normalization.

The normalization of each pixel is saved to a sky spectral definition file in FITS format. The used ARF is saved to another extension in the same file. If the spectral parameter is given as a map, then that map is saved to the next extension; else it is written as a FITS keyword to the normalization extension. The same applies for the column density.

These actions are handled by 'mt_skyima2skyspec'.

The next step consists of producing a sky definition file i.e. by converting each pixel to a row in the extension. That is taken care of by function 'mt_skyspec2skydef'.

# 13 Observation simulation

When a raytracing process has finished the detection can be simulated by the 'mt_detector' function. It uses as input a detector description file, see section 13.2 below.

The detector is assumed to be pixellated with a center on the focal point (the boresight). The energy dependent detector quantum efficiency is also involved as well as the redistribution matrix (RDM) that describes the probability that a registrered photon of a certain energy ends up as an event in a particular energy bin (defined by the detector electronics).

The function 'mt_detector' will on the basis of a given quantum efficiency table, an RDM, detector position resolution or pixel size and number produce an event list from the photon list (the array 'Phs') to simulate an observation.

The event list is stored in the external variable 'Evlist' that is an array of the struct 's_MTEvent' (see section 7.1). The function 'mt_det_image' can be used to generate a detector image.

## 13.1 Adding background

In most detectors there are two contributions to the detector background

**Instrument background** This is caused by ionised high energy particles of solar or cosmic origin passing through the detector or gamma rays from Compton events from particle reactions in other parts of the satellite body.

**Diffuse X-ray background** This is X-radiation from all parts of the sky presumably caused by numerous distant sources but at low energies in particular other truly diffuse sources may be responsible.

In MT_RAYOR the instrument background is asssumed to be uniform across the detector and described in a table in .scm-format (see section 22) with two columns: the energy in keV and the count rate in cts keV$^{-1}$s$^{-1}$cm$^{-2}$.

The command *mt_det_add_instr_bkg* will add counts at random according to the given energy distribution.

The diffuse X-ray background (DXB, aka CXB) cannot be described in general but can be included in the following way: Define a large number of sources in the FOV of the telescope with a flux as given by the adopted model for the radiation. Then run all these sources through the telescope and turn the succesful photons into events by a call of *mt_detector*. An event file can then be generated by command *mt_save,mode="e"*.

The name of the thus produced file is given as keyword 'dxb_bkg_file' in the detector description file, see 13.2. Then the command *mt_det_add_dxb_bkg* will sample the appropriate number of events from this and add them to the general event list ('Evlist').

The command *mt_det_add_bkg* combines both types of backgrounds i.e. calls both *mt_det_add_instr_bkg* and *mt_det_add_dxb_bkg*.

## 13.2 The detector

The prerequisite for an observation is a detector with certain properties. The information on those are expected in an .scm format file such as:

```
//
//  Detector definition file
//
// n/a qeff_file = /home/njw/nustar/QEFF_test.fits
//    In extension 'QUANTUM EFFICIENCY': ENERGY, QUANTEFF
//
// rmf_file = /home/njw/nustar/info100204/nustar_sens_nosvn/outfile/rmf.fits
//    In extension 'MATRIX' : ENERG_LO, ENERG_HI, MATRIX
//                 'EBOUNDS': E_MIN, E_MAX
//
// num_pixels1 = 64
// num_pixels2 = 64
// pixel_size1 = 0.6; mm
// pixel_size2 = 0.6; mm
//
// instr_bkg_file = instrument_bkg_countrate.scm
// dxb_bkg_file = events_0000.fits
//
```

The mandatory keywords are:

**qeff_file** The table of the detector quantum efficiency as as function of energy. Only to be supplied if the information is not contained in the RMF. (In the actual example the string 'n/a' has been added to the keyword name).

**rmf_file** This must be the name of a FITS file in OGIP format[5] with at least two extensions: 'MATRIX' and 'EBOUNDS' that contains all the necessary detector energy range and resolution information.

**num_pixels1** Number of pixels in the first coordinate (x-direction).

**num_pixels2** Number of pixels in the second coordinate (y-direction).

**pixel_size1** Size of pixels [mm] in the first coordinate (x-direction).

**pixel_size2** Size of pixels [mm] in the second coordinate (y-direction).

This format allows for no spacing between the pixels and assumes that all pixels have the same size.

If the quantum efficiency file name is given then the information must reside in an extension by name of '`QUANTUM EFFICIENCY`' and with the columns: `ENERGY` and `QUANTEFF`.

## 14 A simulation example

Below is given a file to include to run an complete simulation of a NuStar observation of the Bullet cluster of galaxies from a count rate image obtained by Chandra:

```
#include "mt_rayor-2.5.i"
Sys = "system_nustar_con.scm";
mt_setup_system,Sys;

outfile = "skyspec_bullet.fits";

remove,"skydef_bullet.fits";
remove,"bullet_5-80.fits";
remove,"bullet_5-7.fits";
remove,"bullet_7-10.fits";
remove,"bullet_10-15.fits";
remove,"bullet_15-40.fits";
remove,"bullet_40-80.fits";
// "no background" images
remove,"bullet_5-80_nb.fits";
remove,"bullet_5-7_nb.fits";
remove,"bullet_7-10_nb.fits";
remove,"bullet_10-15_nb.fits";
remove,"bullet_15-40_nb.fits";
remove,"bullet_40-80_nb.fits";

dol_skyima = "08-55.73sm.fits+0";
dol_arf = "arf_chandra.fits[SPECRESP]";
emin = 0.8; // keV - applies to dol_skyima
emax = 5.5; // keV - applies to dol_skyima
mt_skyima2skyspec, dol_skyima, dol_arf, emin, emax, outfile, \
            sc="TB", p1="tmapc.73sm_updated.fits+0", nh=5.6e20;

e1 = 5.0; // keV - applies to NuStar simulation
e2 = 79.0; // keV - applies to NuStar simulation
```

[5]"Office of the Guest Observer Program" see e.g. http://www.adass.org/adass/proceedings/adass94/corcoranm.html

```
radius_lim = 0.1556; // deg - max angle in NuStar FOV
mt_skyspec2skydef, outfile, "skydef_bullet.fits",104.625,-55.941, \
       0.0, lim=0.05, exposure=5.e5, \
       e1=e1, e2=e2, nchan=50, radius=radius_lim, fluxdir="/scratch/njw/fluxfiles", \
       telescop="NuSTAR",instrume="MT_RAYOR-4.5 sim";

// Remove the 'Exposure' external variable
mt_clear;

// Start the raytracing with an exposure time as indicated in the
//         sky definition file
mt_run,"skydef_bullet.fits",chat=1;

// Produce the event list
mt_detector;

// first save images without background

im = mt_det_image(outname="bullet_5-80_nb.fits");
disp,im,pane=0;
write,format="%i counts in the detector (no bkg)\n", numberof(Evlist);
mt_det_image,emin=5.,emax=7.,outname="bullet_5-7_nb.fits";
mt_det_image,emin=7.,emax=10.,outname="bullet_7-10_nb.fits";
mt_det_image,emin=10.,emax=15.,outname="bullet_10-15_nb.fits";
mt_det_image,emin=15.,emax=40.,outname="bullet_15-40_nb.fits";
mt_det_image,emin=40.,emax=80.,outname="bullet_40-80_nb.fits";

// then add the background

mt_det_add_bkg;

// and save the simulated images

im = mt_det_image(outname="bullet_5-80.fits");
disp,im,pane=1;
write,format="%i counts in the detector, bkg included\n", numberof(Evlist);
mt_det_image,emin=5.,emax=7.,outname="bullet_5-7.fits";
mt_det_image,emin=7.,emax=10.,outname="bullet_7-10.fits";
mt_det_image,emin=10.,emax=15.,outname="bullet_10-15.fits";
mt_det_image,emin=15.,emax=40.,outname="bullet_15-40.fits";
mt_det_image,emin=40.,emax=80.,outname="bullet_40-80.fits";

// Write the events to a FITS binary table for further analysis

mt_save,mode="e";
```

## 14.1 Event list and analysis

Some general NJW supplied Yorick functions are available for making a data analysis once the raytracing has been done as well as the conversion to events:

1. 'specbinning' can produce a spectrum from the event list

2. 'spec2phaii' can produce a FITS file in the PHAII format that spectral analysis tools such as 'xspec' can read.

## 15 Testing the optics in the lab

A laboratory setup with an X-ray source at a finite distance can be simulated. The first argument of *mt_run* must be a FITS file with the source information as shown below:

The FITS header of the first extension of the 'Lab Source File' must have the following keywords included:

| **Keyword name** | **Value** | **Description** |
|---|---|---|
| EXTNAME | LABSOURCE_DEFINITION | The name of the extension |
| ZPOSIT | A real number | [mm] Z position of the laboratory source |

The FITS columns of the binary table must be:

| **Column name** | **Unit** | **Description** |
|---|---|---|
| X | mm | X position of this source element |
| Y | mm | Y position of this source element |
| STRENGTH | photons $\mathrm{mm}^{-2}$ | Photon fluence on telescope aperture |
| ENERGY | keV | Energy of photons |
| DOL | | DOL of photon flux table |

Any number of source elements can be defined to simulate a laboratory source with a finite extent and perhaps varying spectral properties across its surface. Each source element is defined by its position (X,Y) relative to the optical axis of the optics. The source position relative to the reference plane of the first optical module is given by the FITS keyword 'ZPOSIT'.

The spectrum emitted by each laboratory source element can be given in a photon flux file (indicated by its DOL) (with extension name 'PHOTON_FLUX'). If the DOL has zero string length or is given as "none" then a monochromatic source is assumed with energy from column "ENERGY".

The detector properties can be entered in precisely the same way as for the detector for celestial observations. The particular shape of the laboratory source can be modelled by supplying any number of source elements. Any absorption apart from what happens in the optics must be included in the spectral description.

When running *mt_detector* the default placement of the detector in the Z-direction is in the focal plane. If the detector is placed at another Z-position then the photons must be propagated to that place by running *mt_propagate* before calling *mt_detector*.

## 16 Appendix: Geometrical formulae

### 16.1 Parabolic Mirrors

With the focus at $(0, 0, z_f)$ the formula is:

$$r = \sqrt{2d(z - z_f) + d^2} \tag{9}$$

where $d$ defines a mirror shell. If the shell is requested to have the radius $r_m$ at $z = z_m$ then

$$d = \sqrt{(z_m - z_f)^2 + r_m^2} - z_m \tag{10}$$

In the coordinate system of the optical module $z_f$ will have a negative value.

## 16.2 Wolter I optics

For the Wolter I combination of cone sections the parallel rays are reflected by the paraboloide to converge to a focal point, F$_1$. But before they get there they intersect the hyperboloide which makes them converge to the other focal point, F$_2$. In this way F$_1$ is common for the paraboloide and the hyperboloide.

Using $f_f$ as half of the distance between the two foci and $f_p$ as the distance from F$_2$ to the reference plane of the optical module then the equation (9) is changed to

$$r = \sqrt{2d(z + 2f_f + f_p) + d^2} \tag{11}$$

and similarly equation (10) becomes

$$d = \sqrt{(z_m + 2f_f + f_p)^2 + r_m^2} - (z_m + 2f_f + f_p) \tag{12}$$

A second parameter is required to describe the hyperboloide. The half value of the major axis, $a$, has been chosen here. Again, $f_h$ is the distance from F$_2$ to the reference plane of the (hyperbolic) optical module.

Then

$$r = \left[ (z + f_f + f_h)^2 \frac{f_f^2 - a^2}{a^2} - (f_f^2 - a^2) \right]^{\frac{1}{2}} \tag{13}$$

If the shell is requested to have the radius $r_m$ at $z = z_m$ then

$$b^2 = (z_m + f_f + f_h)^2 + f_f^2 + r_m^2 \tag{14}$$

$$a = \left[ \frac{1}{2} \left( b^2 - \sqrt{b^4 - 4(z_m + f_f + f_h)^2 f_f^2} \right) \right]^{\frac{1}{2}} \tag{15}$$

In the solution for $a$ in (15) the choice of the appropriate branch of the hyperboloide has been made.

## 16.3 Conical approximations

Each mirror is characterized by a radius at its beginning and an angle with respect to the telescope axis. In the design included all the mirrors have the same lengths. For the master module a ray parallel to the telescope axis that hits the midpoint of a mirror is requested to be reflected to cross the telescope axis at a given focal point. For the slave module it is requested that a ray converging with given angle (namely two times the angle of the first mirror) cross the telescope axis at at given focal point.

This leads to a third order equation which is solved in an exact way.

### 16.4 Surface perturbations

There will always be small deviations of the actual mirrors from the ideal shape. Such deformations can be introduced by a two-dimensional array for each mirror that describes the deviation as a function of azimuth and z-coordinate. Thus the surface is given by

$$r = r_{ideal}(z) - r_{deform}(z, \phi) \tag{16}$$

where $r$ is the radius and $\phi$ the azimuth angle. The sign has been chosen by following a common convention.

## 17 Appendix: Expressions for raytracing

A ray with direction $\mathbf{R}$ encounters a reflecting surface with normalized normal $\mathbf{N}$ and the specularly reflected ray is then

$$\mathbf{Q} = \mathbf{R} - 2(\mathbf{N} \cdot \mathbf{R})\mathbf{N} \tag{17}$$

In order to get the direction after a scattering away from the specular direction the direction perpendicular to the specular direction but still in the plane defined by $\mathbf{N}$ and $\mathbf{Q}$

$$\mathbf{V} = \frac{\mathbf{N} + (\mathbf{N} \cdot \mathbf{R})\mathbf{R} - 2(\mathbf{N} \cdot \mathbf{R})^2\mathbf{N}}{\sqrt{1 - (\mathbf{N} \cdot \mathbf{R})^2}} \tag{18}$$

The scattering angle, $\alpha$, is the angle away from the specular direction

$$\mathbf{S} = \mathbf{Q}\cos\alpha + \mathbf{V}\sin\alpha \tag{19}$$

where $\mathbf{S}$ is the scattered direction. It is assumed here that there is no scatter out of the $\mathbf{N}, \mathbf{Q}$–plane.

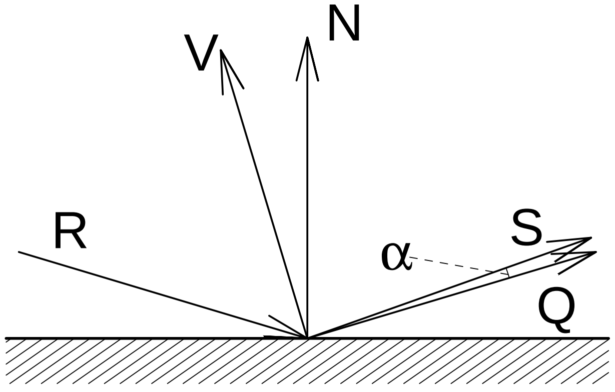


Figure 11: A diagram of the vectors involved in the reflection process and the scattering angle.

## 17.1 The surface normal

The mirror surface in parametrized representation (with $r = r(z, \phi)$ as in equation 16):

$$x = r(z,\phi)\cos\phi \tag{20}$$
$$y = r(z,\phi)\sin\phi \tag{21}$$
$$z = z \tag{22}$$

Then the surface normal is the cross product of the two vectors of the partial derivatives with respect to the two parameters, $\phi$ and $z$:

$$\frac{\partial x}{\partial \phi} = \frac{\partial r(z,\phi)}{\partial \phi}\cos\phi - r(z,\phi)\sin\phi \tag{23}$$
$$\frac{\partial y}{\partial \phi} = \frac{\partial r(z,\phi)}{\partial \phi}\sin\phi + r(z,\phi)\cos\phi \tag{24}$$
$$\frac{\partial z}{\partial \phi} = 0 \tag{25}$$

$$\frac{\partial x}{\partial z} = \frac{\partial r(z,\phi)}{\partial z}\cos\phi \tag{26}$$
$$\frac{\partial y}{\partial z} = \frac{\partial r(z,\phi)}{\partial z}\sin\phi \tag{27}$$
$$\frac{\partial z}{\partial z} = 1 \tag{28}$$

so the expression for $\mathbf{N}$ becomes

$$\mathbf{N} = \left(\frac{\partial x}{\partial z}, \frac{\partial y}{\partial z}, \frac{\partial z}{\partial z}\right) \times \left(\frac{\partial x}{\partial \phi}, \frac{\partial y}{\partial \phi}, \frac{\partial z}{\partial \phi}\right) = r\left(-\cos\phi - \frac{1}{r}\frac{\partial r}{\partial \phi}\sin\phi, -\sin\phi + \frac{1}{r}\frac{\partial r}{\partial \phi}\cos\phi, \frac{\partial r}{\partial z}\right) \tag{29}$$

where the order of the vector product has been chosen so that $\mathbf{N}$ points more or less to the telescope axis.

## 18 Appendix: Word list

**Coating number** Each type of mirror coating is identified by a number.

**Event file** A FITS binary table file with information for each event registrered in the detector or samples thereof.

**Photon file** A FITS binary table file with information for each raytraced photon or samples thereof.

**Scatter file** A FITS binary table file with a scattering description and reflection coefficients for a grid of energy and grazing angle values.

**Sky definition file (SDF)** A FITS binary table file where each row represents a source to be raytraced.

**Sky spectral definition file (SSDF)** A FITS file with several extensions:

1. Mandatory: A map of the normalization constant including a keyword (SC) for the spectral code ('PL', 'BB', or 'TB')
2. Optional: A binary table with the ARF information for the instrument that observed the input image
3. Optional: A map of the spectral parameter (only if it varies across the image; a constant value is given as a keyword (PARAM1) with the first extension)
4. Optional: A map of the column density (only if it varies across the image; a constant value is given as a keyword (NH) with the first extension)

**System definition file** A text file in the .scm format describing the details of the mirror modules and the detector to use.

## 19 Appendix: External variables

The used external variables are as follows:

| | |
|---|---|
| Acoef | Single A-coefficient value for hyperbolic shape |
| Acoefarr | Array of A-coefficient values for hyperbolic shape |
| Angle_inarr | Grazing angle values from the scatter data file |
| Angle_uniq | Array of unique grazing angle values from scatter data file |
| Anglesarr | Angles for the scattering distributions |
| Cen_dx_as | Shift of central aperture stop position [mm] in the x direction |
| Cen_dy_as | Shift of central aperture stop position [mm] in the y direction |
| Coat_list | (struct) List of available coating types and the scatter files |
| Coating_scat | Coating number for loaded scatter file |
| Dcoef | Single D-coefficient value for parabolic shape |
| Dcoefarr | Array of D-coefficient values for parabolic shape |
| Dead_pixel_map | Map of the dead detector pixels |
| Dec_scx | Declination of telescope pointing axis |
| Det_bkg_file | Name of detector background file (in .scm format) |
| Det_offset | Two element array to describe an offset of the detector wrt. the optical axis |
| Dim_focp | Dimension of focal plane array for 'mt_save' |
| Distributionarr | Scatter distribution data, 2D array Nangles x number_of_cases |
| Dphot | Photon density (in $mm^{-2}$) for incoming radiation |

| | |
|---|---|
| Dol_dead_pixel_map | DOL of data array describing the dead detector pixels |
| Dxb_bkg_file | Name of DXB background description file of type .scm |
| E_max | Array of upper energy bounds (in keV) for detector energy bins |
| E_min | Array of lower energy bounds (in keV) for detector energy bins |
| E_mnx | Array of center energies for detector energy bins |
| E_uniq | Array of energy values from scatter data file |
| Earr | Entire energy array from scatter data file |
| Energ_hi | Array of upper energy bounds (in keV) for ARF and RDM |
| Energ_lo | Array of lower energy bounds (in keV) for ARF and RDM |
| Energy | Energy as defined in latest run of *mt_run* |
| Evlist | Array of structs (s_MTEvent) with event (or counts-) information |
| Exposure | Exposure time [s] for photon flux file source definition |
| Fcoef | Single F-coefficient value for parabolic shape |
| Focal_length | Focal length of combined system [mm] |
| Fraper | The four-element array to describe the front aperture |
| Fraper_area | Area of current fraper used [mm$^2$] |
| Full_length_spokes1 | Full length of the spokes at entrance of optical module |
| Full_length_spokes2 | Full length of the spokes at exit of optical module |
| Instr_bkg_file | Name of the .scm file with a table of instrument background |
| Instrume | Content of OGIP FITS keyword 'INSTRUME' |
| Logfilename | Name of log file to receive various information |
| Logflag | Flag to tell if a log file is in use and the amount of output |
| Mirror_angle | Current mirror angle for conical approximation |
| Mirror_anglearr | Array of mirror angles for conical approximation |
| Mirror_coating | Array of defined mirror coatings for an optical module |
| Mirror_deform_arr | Mirror deformation array (data cube) |
| Mirror_deform_files | List of mirror deformation files to be used |
| Mirror_eff_factors | Array of efficiences relative to spoke blocking |
| Mirror_length | Length [mm] of current mirror |
| Mirror_lengths | Array of mirror lengths [mm] |
| Mirror_number | Number of current mirror |
| Mirror_thicknessarr | Array of mirror thicknesses [mm] |
| Mirror_thickness_files | Array of mirror thickness table files |
| Modtype | Type of current optical module (given as a string) |
| Module_num | Number of current optical module |
| N_mirrors | Number of mirrors (also highest mirror number) |
| Ne_mnx | Number of detector energy bins |
| Num_modules | Number of optical modules in the system |
| Num_pixels1 | Number of detector pixels (x direction) |
| Num_pixels2 | Number of detector pixels (y direction) |
| Num_warn | Number of warnings for out-of-limits angles or energies |
| Om_files | Array with optical module description files |
| Om_functions | List of function names for optical module creation |
| Om_parameters | Array with parameters for the optical module |
| Open_radius | Radius [mm] of the aperture stop |
| Opt_module_file | Name of FITS file with optical module information |
| Phi_spokes1 | Array of entrance spoke azimuth angles [rad] |
| Phi_spokes2 | Array of exit spoke azimuth angles [rad] |
| Phs | Array of struct (s_Ray) with current photon information |
| Pix_focp | Pixel size [mm] of focal plane array for 'mt_save' |
| Pixel_size1 | Pixel size [mm] of detector pixels in the first coordinate |
| Pixel_size2 | Pixel size [mm] of detector pixels in the second coordinate |

| | |
|---|---|
| Posang | Position angle [deg] of telescope (satellite)' |
| Q_ener | Array of energies [keV] for the quantum efficiences |
| Qeff | Array of quantum efficiences for the detector |
| R1_mirror | Radius [mm] of current mirror at entrance for conical approximation |
| R1arr | Array of mirror radii at entrance, determined by the refl. surface |
| R2arr | Array of mirror radii at exit, determined by the refl. surface |
| Ra_scx | Telescope axis pointing, Right Ascension [deg] |
| R_coefarr | Array of reflection coefficients from the scatter data file |
| R_inner_design | Radius of innermost mirror (refl. surface) at entrance |
| R_inner | Radius of innermost baffle at entrance |
| R_outer | Radius of outermost mirror (backside) at entrance |
| Rb1iarr | Array of inner radii of entrance slit |
| Rb1oarr | Array of outer radii of entrance slit |
| Rb2iarr | Array of inner radius of exit slit |
| Rb2oarr | Array of outer radius of exit slit |
| Rdm | Redistribution matrix for the detector |
| Reverse_spokes | A flag for reversing the spoke action |
| Rmf_file | Name of FITS file with RMF information |
| Roll_phot | The roll-angle [deg] to be applied to the photons |
| Rstart_spoke1 | Array of lower radius limits [mm] for entrance spokes |
| Rstart_spoke2 | Array of lower radius limits [mm] for exit spokes |
| Rstop_spoke1 | Array of lower radius limits [mm] for entrance spokes |
| Rstop_spoke2 | Array of lower radius limits [mm] for exit spokes |
| Scatter_files | Array with coating scattering properties description files |
| Scatter_file | FITS file with scatter data |
| Scatter_type | Flag to tell if scatter data are of type 1 or type 2 |
| Spoke_define_files | File names (in .scm format) of spoke definition files |
| Src_azimuth | Azimuth angle [deg] of source position |
| Src_offaxis | Offaxis angle [arcmin] of source position |
| System_filename | Name of the .scm file with parameters for the telescope |
| Telescop | Content of OGIP FITS keyword 'TELESCOP' |
| Use_mdeform | Boolean to tell if mirror deformations should be included |
| Use_scatter | Boolean to tell if scattering should be included |
| Version | A string variable that holds the MT_RAYOR version number |
| Width_spoke1 | Array of entrance spoke widths [mm] |
| Width_spoke2 | Array of exit spoke widths [mm] |
| Xpixlims | Two element array with detector edges [mm] in x direction |
| Xpixlo | Array with lower detector pixel edges [mm] in x direction |
| Ypixlims | Two element array with detector edges [mm] in y direction |
| Ypixlo | Array with lower detector pixel edges [mm] in y direction |
| Z1_mirror | Z1 value [mm] of current mirror |
| Z1_setups | Z1 values [mm] to define 'gap' telescopes |
| Z1arr | Array of mirror entrance z values |
| Z2_setups | Z2 values [mm] to define 'gap' telescopes |
| Z2arr | Array of mirror exit z values |
| Z_position_as | Z coordinate of the aperture stop position |
| Z_reference | Array with z coords. of the opt. module reference planes |
| Zb1iarr | Array of z of inner, upper entrance slit |
| Zb1oarr | Array of z of outer, upper entrance slit |
| Zb2iarr | Array of z of inner, lower entrance slit |
| Zb2oarr | Array of z of outer, lower entrance slit |
| Zfocus | z coords of current opt. module |

| | |
|---|---|
| Zfocusarr | Array of z coords of the foci of the optical modules |

## 20 Appendix: Installing Yorick

The main URL for the yorick homepage is:

http://yorick.sourceforge.net/index.php

A basic description and a user manual can be found there.

Downloads
Find version 2.1.05 (or later) for your operative system.

### 20.1 Linux installation

Create a directory called e.g. 'yorick': `/myhome/yorick`.

Place the downloaded file 'yorick-2.1.05.tgz' in it and cd to the directory. Then

```
yorick$ tar zxf yorick-2.1.05.tgz
yorick$ cd yorick-2.1
yorick$ make
```
Creates some output that should end with an OK message.
```
yorick$ alias yorick /myhome/yorick/yorick-2.1/yorick/yorick
```

Select the directory (`/myhome/yorick` would be very suitable) to place all standard code files (with name extension '`.i`' in this directory.

Locate the file `custom.i` and edit.

### 20.2 Windows installation

[To written.]

## 21 Appendix: Installing MT_RAYOR

The package comes in a file: mt_rayor-4.6.5.tar, that contains all the necessary source files (apart from those already included in the Yorick distribution, see the appendix section 20, page 70).

Assuming that Yorick has been installed under directory `/myhome/yorick` the unpacking of mt_rayor-4.6.5.tar should happen there. Then the specific mt_rayor functions are found in 'mt_rayor-4.6.5.i' which is created locally.

Furthermore a subdirectory `mraytrace` is created with a number of 'starter files' with template files for system definition etc., e.g. the optical module defining files, the scattering data file, the photon files, and the image files. This subdirectory is supposed to be the working directory.

The functions of MT_RAYOR are run directly under Yorick which implies that an extensive set of additional functions for further analysis and plotting of the raytracing results are available. Such general collections of functions are: 'basic.i', 'datafit.i', 'euler.i', 'idlx.i', 'image.i', 'kfits.i', 'mfits.i', 'mcomplex.i', 'plot.i', and 'scom.i'. The filename extension '.i' is commonly used for yorick files, any filename is accepted but this is practical.

Information about these collection can be obtained by looking into the files but also under yorick by the very useful 'help' command. Generally
> `help,funcname`
will provide help for function 'funcname'. For the above mentioned packages there is a similar functionality when you add the three letters: doc to the call e.g.:
> `help,imagedoc`
will give you an overview of the extra image display possibilities supplied in 'image.i'.

## 22 Appendix: The .scm format

A text file format has been used here for tabular data supplemented with comments and keywords.

Any line starting with either of .+−0123456789 is interpreted as a data line with blanks (spaces) as separators.

Keywords are given as:
`// keyword = value; comment`
where `keyword` contains no space-characters and the '=' sign must be present. The `value` can be an integer, a floating point number or a string. Everything following and including the ';' sign is ignored. Here a string may contain spaces since it is read from the first to the last non-space character. If several identical keywords are given then all their values will be read into an array.

Example:

```
// A valid text file in .scm format
// with a table of three columns and four rows

   11.22   3      55
  .3 17.001 -54.6
A line is a comment when it does not start with a numerical character
A keyword can be given as
// temperature = 17.34; degrees

  -12.34 23.45 77.9
  34 99 47
```

# 23 Appendix: History of updates

**Version 4.5.1 (2013-05-31):**

Added the XSPEC functionality to generate source photon flux as a function of energy.

**Version 4.0 (2011-12-01):**

Reversed the mirror counting from outside-in ot inside-out, i.e. the convention used for most telescopes.

**Version 3.5 (2011-08-24):**

The quick and approximate evaluation of effective area in
function *mt_eff_area_quick* was introduced with the accompanying
function *mt_get_mirror_eff_factors* to evaluate the blocking by the spokes.

**Version 3.3 (2011-04-26):**

The search for the photon entrance baffle slit now takes into account that the z-values of the baffles (and mirror edges) may vary. This is added to accommodate the IXO (or Athena) telescope.

The search for reflections now happens along the entire photon path when mirror deformations are included. A second reflection may be found in which case the photon is considered lost.

Function *mt_scatter_data_file* has been updated with more keywords for maximum grazing angle and data attenuation.

Function *mt_photpr* added.

Bug 003 fixed.

**Version 3.2 (2011-01-31):**

The innermost mirror is no longer a dummy mirror.

The keyword 'system_type' has changed to 'om_type' (contents are unaltered) in the system definition file.

Reflection coefficients are calculated even if scattering is unselected.

**Version 3.1 (2011-01-25):**

Introduced keywords 'labxoff' and 'labyoff' for 'mt_run' and 'mt_pre_def_photons' to offset the laboratory source position.

Bug 001 has been fixed so that radius information is no longer necessary in spoke defining files if the spokes extend all the way from the smallest to the largest radius.

Bug 002 has been fixed.

**Version 3.0 (2011-01-20):**

Introduced radial limits on spokes as well as the possibility to use different spoke patterns for module entry and exit.

Mirror thickness can now be dependent on the mirror radius. Described in a text table (.scm format) as a function of radius for linear interpolation.

A struct element 'bounce' has been added to the *s_Ray* struct to tell where reflections have occurred.

No reflection is no longer a reason for a value status $\neq 0$.

**Known bugs:**
001: Spoke defining files must have radius definitions (columns 'rstart' and 'rstop').
002: Photon flux file input creates a crash in 'mt_def_photons'.
003: Azimuth error in mt_drayplot.

**Version 2.6 (2010-10-20):**

Introduced spokes.

# Index

basic coordinate system, 51
bounce, 36, 37, 44

Coating table, 46

Deformation description file, 9, 49

Effective area, 20, 30, 58
Evlist, 27, 44, 59

fraper, 35
fraper (keyword), 39
Front Aperture, 25, 35

GRI, 9, 52

Half Power Diameter, 21
help, 20
HPD, 21

Lab Source File, 63

Mirror thickness file, 19, 46
mt_add_ysaves, 20
mt_analysis, 21
mt_aperture_stop, 21
mt_bg_run, 22
mt_bg_run_eff_area, 22
mt_build_cat, 23
mt_catalog2skydef, 23
mt_clear, 23
mt_create_om_con2, 24
mt_create_om_con3, 24
mt_create_om_con4, 24
mt_create_om_con5, 24
mt_create_om_hyp2, 24
mt_create_om_par1, 24
mt_create_om_par2, 24
mt_def_photons, 25
mt_det_add_bkg (U), 25
mt_det_add_dxb_bkg (U), 26
mt_det_add_instr_bkg (U), 26
mt_det_image (U), 27
mt_detector (U), 27
mt_drayplot (U), 28
mt_dxb2skydef (U), 29
mt_eff_area_photons (U), 30
mt_eff_area_quick (U), 30
mt_fake_scatter_data (U), 31
mt_funcs, 31
mt_get_mirror_eff_factors (U), 32
mt_get_rcoef, 32
mt_load (U), 33
mt_mirdiag (U), 33
mt_mirplot (U), 34
mt_mk_mdeform_file (U), 34
mt_photpr (U), 34
mt_pre_def_photons, 35
mt_propagate (U), 35
mt_qimage, 36
mt_rayplot, 36
mt_rayplot (U), 36
mt_raytrace_module, 37
mt_reflplot (U), 38
mt_run (U), 38

normal vector, 51
NuSTAR, 9

Optical module, 9
Optical Module Description File, 9, 19, 45
Optical Module File, 24

Photon flux file, 50
Photon image file, 50
Photon save file, 49
Phs, 10, 25, 27, 36–39, 44, 56, 59

Quantum efficiency, 10, 20, 59, 60

Reference plane, 18
RMF, 10, 20, 59, 60

s_MTEvent (struct), 44, 59
s_Ray (struct), 44
Scatter data file, 47
Sky Definition File (SDF), 51, 67
Sky Spectral Definition File (SSDF), 42, 58, 67
Source list file, 51
Spoke definition file, 48
Status value description, 37
System Definition File, 67

Telescope system definition file, 15, 16

unit of length, 51

Wolter I, 9, 24, 56, 64