McStas Union project v0.91

– Mads Bertelsen

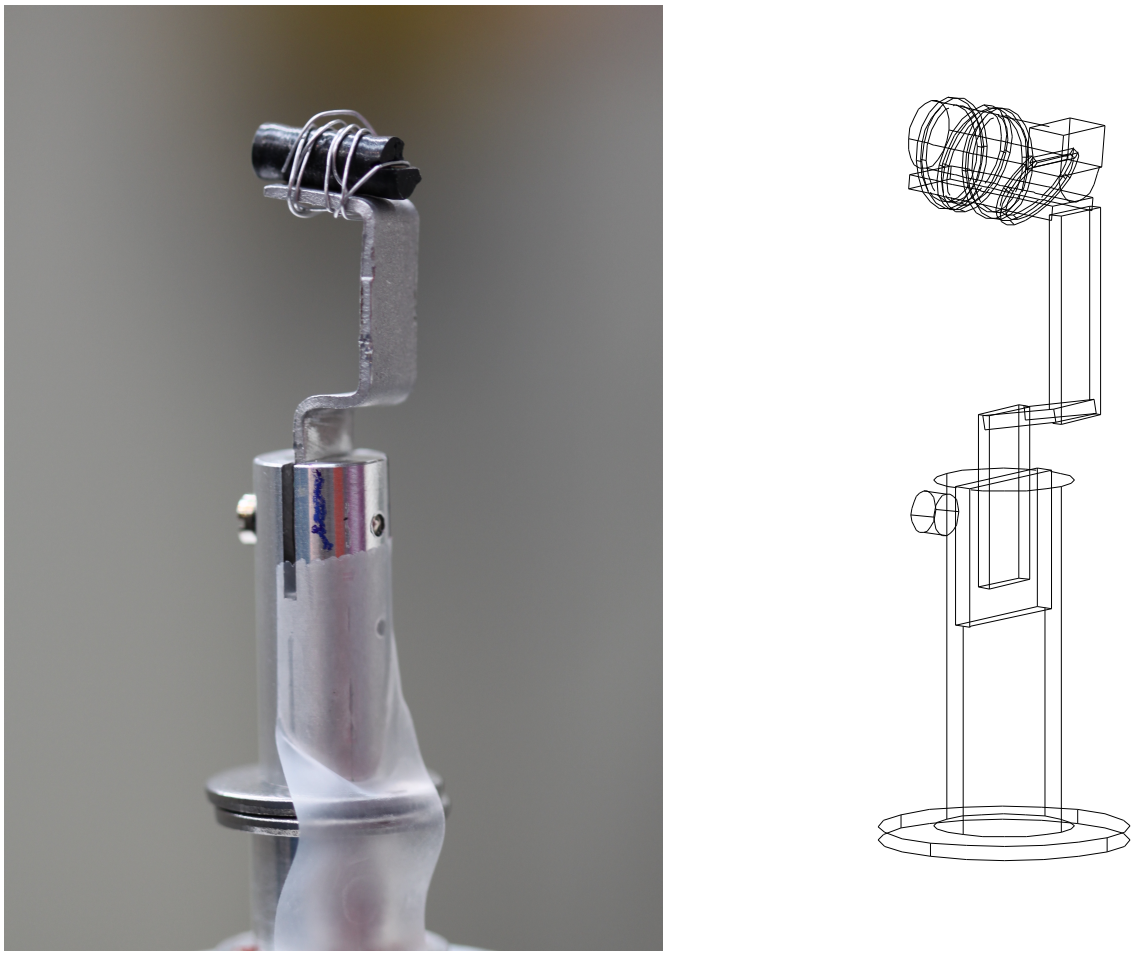


Figure 1: Left: Picture of LSCO sample taken by Pia Jensen Ray. Right: Recreation of sample and sample holder using Union components, output by mcdisplay.

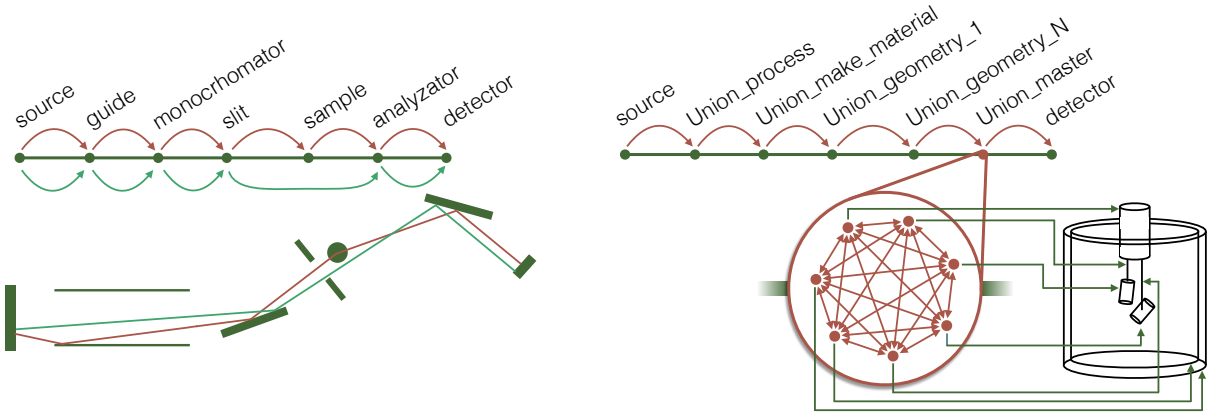
Contents

1 The Union McStas project

The McStas ray tracing package was made to simulate a linear succession of modular components separated in space. This allows contributions from different users to be assembled into one instrument, as long as each part is isolated in space, and only rays travelling through these components in the specified order is to be simulated. This approach is excellent for simulating the intended beampath for neutron scattering instrumentation, but ignores the unintended paths. Furthermore some of the most complex code is located in the sample simulation, and as it is isolated spatially it needs to be a single large McStas component.

The McStas Union components is a set of components that is intended to fundamentally change how McStas works by allowing multiple scattering independent of the order in which these components are placed into a McStas instrument file, and thus simulating not just the intended beampath, but also all others. In addition it separates physics and geometry with the aim of making it quicker to add new physics or geometry into the project. The scattering physics is further divided into physical processes, and writing such a process is kept as easy as possible, only requiring a description of the cross section as a function of wavevector and a description of a single scattering event.

A schematic view of a McStas instrument file is shown in the left part of figure 2 where the linear succession of components can be seen. The only possible change from the specified order would be if a component is skipped because the ray didn't intersect with a certain part of the instrument. The approach used by the Union project is seen in the right part of the figure, where all the ray tracing happens in a single master component that works around McStas in order to achieve native multiple scattering between a number of components. There is however no issue in using both regular McStas components and Union components in one McStas file, as long as the order is such that the Union master component is placed appropriately.



Traditional McStas instrument

McStas instrument file using Union components

Figure 2: The linear succession of components in a McStas instrument file, and how the Union components circumvent this restriction by collecting all simulation to a single component.

The resulting simulation is built from so called "volumes" which is a geometry placed in the simulation with some properties including a priority, an absorption cross section and a material definition that can include any number of physical processes.

The native multiple scattering of the Union components makes it possible to place volumes in parallel to create complex sample environments, detector tanks and the like. Because of the priority property it is even allowed to overlap different volumes to carve out entrance windows, hollow parts or simply layers of different materials, as the volume with highest priority will occupy the space where they overlap.

For a user familiar with McStas these new features are available with a minimum of learning, as the input is done using small components in the traditional McStas style, the only addition being that the user needs to link components together with some new parameters.

This report shows the current status of the project and will serve as an early version of the manual for the component set.

2 Demonstration

Here the flexibility of the Union McStas components is demonstrated by showing examples of what can be created.

2.1 Replica of sample holder

The flexibility of the geometrical part of the Union project is demonstrated nicely by figure 1 on the front page where a sample holder with sample is replicated. Even the cut in the sample is replicated by using a box to cut a shelf in the sample, and a small piece of aluminium runs in this gap as on the picture. In figure 3 an absorption picture of the geometry can be seen, where the aluminium absorption cross section is exaggerated. This could have been done using the any-geometry in McStas, but by using the Union project the data input is easier, all the different parts can have different properties, and the geometry need not be closed or non-convex.

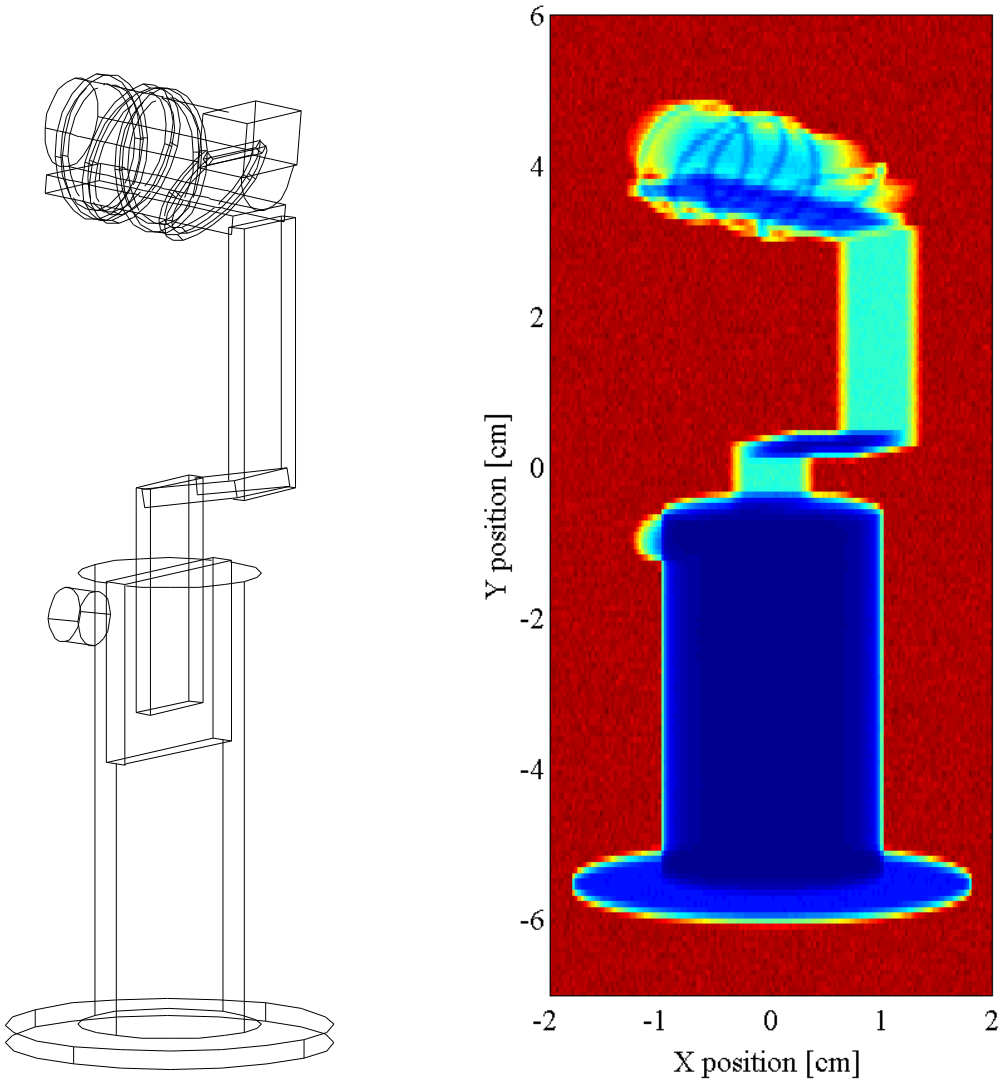


Figure 3: Mcdisplay output and absorption picture for the sample holder model shown on the front page. Aluminium cross section for absorption exaggerated to make details clear.

2.2 Cryostat with powder samples

It is feasible to create a simple model of an entire cryostat with several layers of aluminium, sample stick, sample holder and multiple sample containers while still being able to run the simulation on a laptop. Such an example is shown in figure 4, 5 and 6. To show the possibilities a weird sample holder was simulated, which includes a frame and 4 aluminium containers with different powders. One of the containers contain a mixture of two powders, and multiple scattering between the different powders in the same volume is simulated. On my personal laptop, simulating 1E6 rays with this geometry, that contains 38 volumes, takes under 2 seconds.

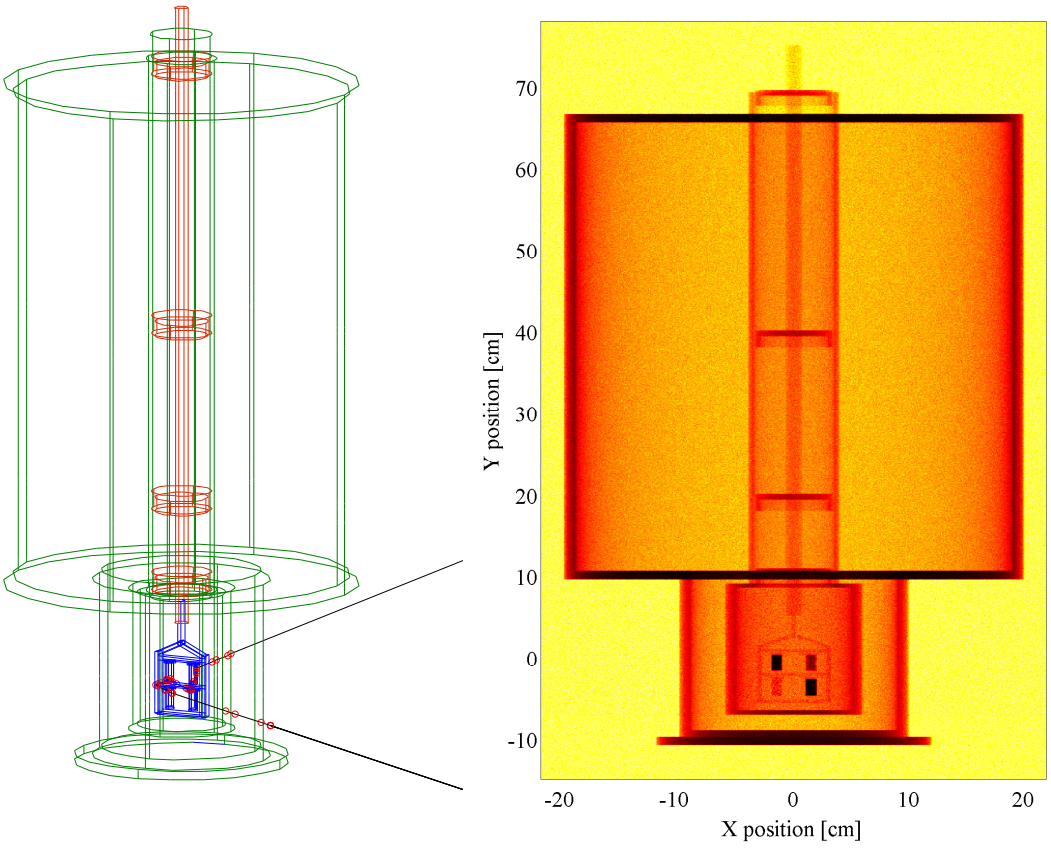


Figure 4: Left: Mcdisplay output for cryostat model created with Union components. The cryostat is colored green, the sample stick red, and the sample holder blue. Right: Absorption picture of the geometry.

2.3 Cryostat with co-aligned and twinned single crystals

A sample holder with 4 co-aligned and twined single crystals is placed in the same cryostat as in 2.2 and are all illuminated by the beam. The twinning is not simulated as two different spatial parts of each crystal, but as two separate processes, one taking 70% of the volume and the other 30%. The co alignment is done for the largest part of each crystal, and the smaller twin thus have a small random angular deviation, here in the order of 0.3°.

The samples mounted on a sample holder is shown in figure 7 with the resulting scattering pattern on a cold beam. A zoom in on two Bragg peaks can be seen in figure 8.

The features of the current version

The current version of the Union project features two geometries and three scattering processes as shown in table 1. These are described in detail later. It is intended that a user can contribute a physical process with significantly less work than contributing an entire component. It requires more work to contribute a new

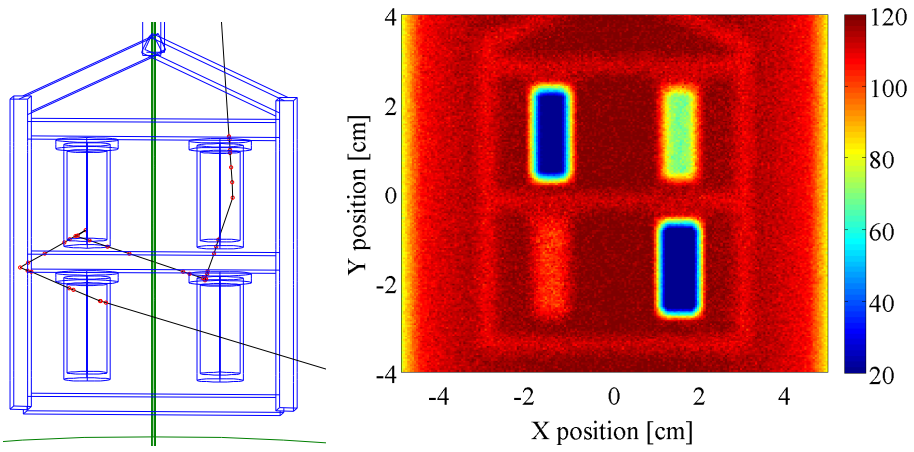


Figure 5: Left: Zoom in on mcdisplay output for cryostat model created with Union component, here showing the sample containers housed in the sample position of the cryostat. A ray which interacts with all 4 samples is shown. Right: Absorption picture of the geometry.

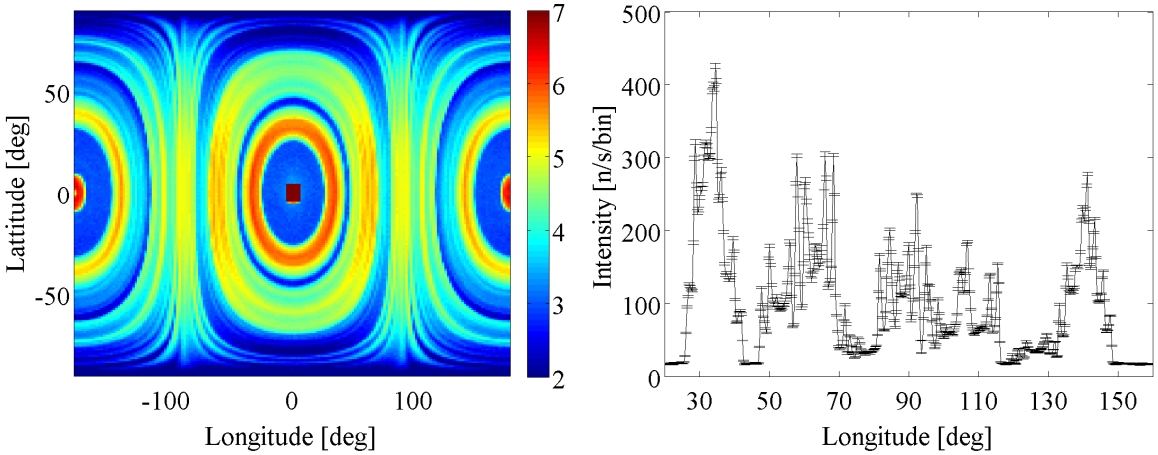


Figure 6: Left: Sphere detector arround the entire setup, the colorscale is for the logarithm of the intensity. Right: Banana detector showing the resulting mess of powder lines.

geometry, but should still be manageable. The core of the component adds many features so that they do not need to be added in processes or geometries.

Geometries	Scattering processes
Cylinder	Incoherent
Box	Powder
	Single crystal

Table 1: List of currently available geometries and scattering processes.

Absorption

Absorption is handled in the core of the component with a absorption cross section property for each defined volume.

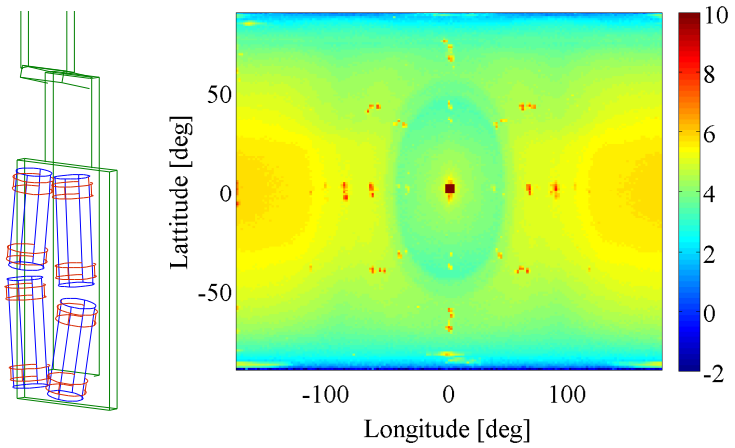


Figure 7: Left: mcdisplay output for the Union components simulating 4 twined and co aligned single crystals on a sample holder. The crystals are blue, the Aluminium rings holding them are red, and the sample holder green. Right: Scattering output from displayed geometry with all samples illuminated by a white beam.

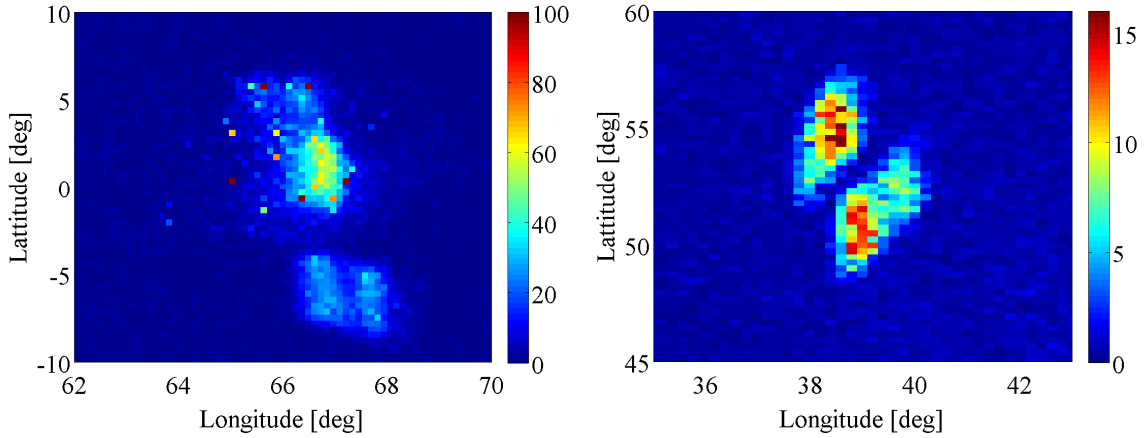


Figure 8: Detailed view of Bragg peaks from co aligned twined crystals.

Multiple scattering

Multiple scattering between all defined volumes and all processes automatically, meaning a contributor only needs to worry about a single scattering event.

Collecting many processes to a material

Each volume have a material definition, and this can contain any number of processes. In each step the cross section for all processes are calculated by functions defined in the physical process, but the core makes the Monte Carlo choice to select between the processes (and transmission) when the ray is propagated within volumes.

Overlapping of volumes

It is allowed to overlap volumes, which is handled by assigning each volume a priority. When two volumes overlap, the region of space they both occupy is taken by the one with the highest priority. In this way complicated geometries can be built, for example by multiple concentric cylinders to build up the walls of a cryostat. The mcdisplay only shows the part of the volume that is present, making it easy to check if mistakes were made when setting the priority of overlapping volumes.

Forced interact

Many McStas sample components have an option called p_interact that will set the probability to interact with the sample instead of the ray being transmitted through. In Union components the core handles this, and allows both to set the probability to interact with a certain volume, and to set the fraction of scattering events caused by each physical process assigned to the volume.

Focusing

In McStas focusing is used to direct the beam in a certain direction, and all the common options have been made available through the standard McStas functions, but it is up to each physical process component to use the created structure. If desired, physical processes can add additional focusing different from the standard.

Exit volumes

Normally the Union components will continue the simulation of a ray until it escapes the spatial region it covers. If one wants a regular McStas component within this space, it is possible to set an exit volume around such a component. Once the ray enters such a volume, the Union component stops and the next McStas component is executed, and the ray can thus not re-enter the Union component. This can for example be used to have monitors within a detector tank.

Tagging

A tagging system tracks the path through the volumes, and saves the order in which volumes and processes are visited. All these histories are collected and sorted after which provide the highest intensity. This gives the user a quick overview over which histories are relevant for the simulation, and gives valuable input in deciding how to set the interact parameters to better sample the histories considered important.

Gravity

The components does not support gravity, when gravity is used propagation still happens without gravity within Union components. The work needed to support gravity is to update the intersection functions.

Polarization

The component does not support polarization yet.

3 Algorithm

The most important part of the Union component is the trace algorithm which is outlined below in a pseudo code fashion. Apart from removing optimizations, error checks, some special cases that handles vacuum and customizations options like p_interact, this is an accurate description of the trace loop.

```
find current volume index from starting position
done = 0
while( done == 0 ) {
   calculate intersections with volumes on I list if not yet calculated
   find the lowest intersection time among these, t0
   if (A lowest intersection time is not found) {
      done = 1
   } else {
      calculate my (inverse penetration depth) for each process in current volume
      calculate my_sum = sum of the calculated my values
      if (rand01 > exp(-t0*v*my_sum) ) {
         select scattering process from weighted choice between my values
         scattering time = -log(1 - rand0max(1 - exp(-my*length_to_intersection)))/my
         propagate ray scattering time
         absorption correction exp(-my_a*(2200/v)*scattering_time*v)
         run scattering function for appropriate process -> new velocity vector
         clear table of calculated intersection times
      } else {
         propagate ray t0
         absorption correction exp(-my_a*(2200/v)*v*t0)
         if (the next intersection is with the current volume) {
            search D list for volume with highest priority that contain ray position
            new current volume index is set
         } else {
            new current volume index corresponds to the volume that was intersected
         }
      }
   }
}
```

Switch statements on selecting the right geometry and scattering process is avoided by using function pointers assigned in initialize, both to simplify the trace section and to avoid wasting time on if statements that gives the same answer for all rays. The mentioned I list and D list is dependent on the current volume, and stands for intersection check list and destination list. The intersection check list for a volume contains indexes of other volumes that needs to be checked for intersection when the ray is in that specific volume. These are generated in initialize and vastly reduce the computational time required by the component, as it avoids checking unnecessary intersections and destinations.

A visualization of the trace algorithm in shown in figure 9. Here the concept of priority is shown, which is important when two volumes overlap in space, as the one with highest priority will occupy the space with overlap. In the first step the ray is in the vacuum around the component, and the intersection check list appropriate for this situation is used, which only contains the blue and green volume. Calculating intersections with the red and orange volume at this point would be a waste of time, as the ray can not go directly from the surrounding vacuum to either of these two volumes. When the ray enters the green volume in step 3, it becomes relevant to calculate the intersection with the red volume, and it is thus calculated. At the scattering in step 4, all previously calculated intersection times are cleared, and new are calculated with the green and red volume, but not the blue and orange, as the ray would need to travel through other volumes first. Only after propagating out of the green volume in step 5 is the intersection time with the blue volume calculated.

In step 5 the destination list for the green volume is used, as it is not clear which volume the ray will end in after intersecting the current volume. The destination list for the green volume contains the blue volume, and the surrounding vacuum, and it is thus checked if the position is inside the blue volume or not to determine which of these is the correct answer, here it was the blue volume. Similarly in step 6, the destination list for the blue volume is used, which contains the surrounding vacuum and the green volume, but as the ray is not inside the green box it is considered in the surrounding vacuum.

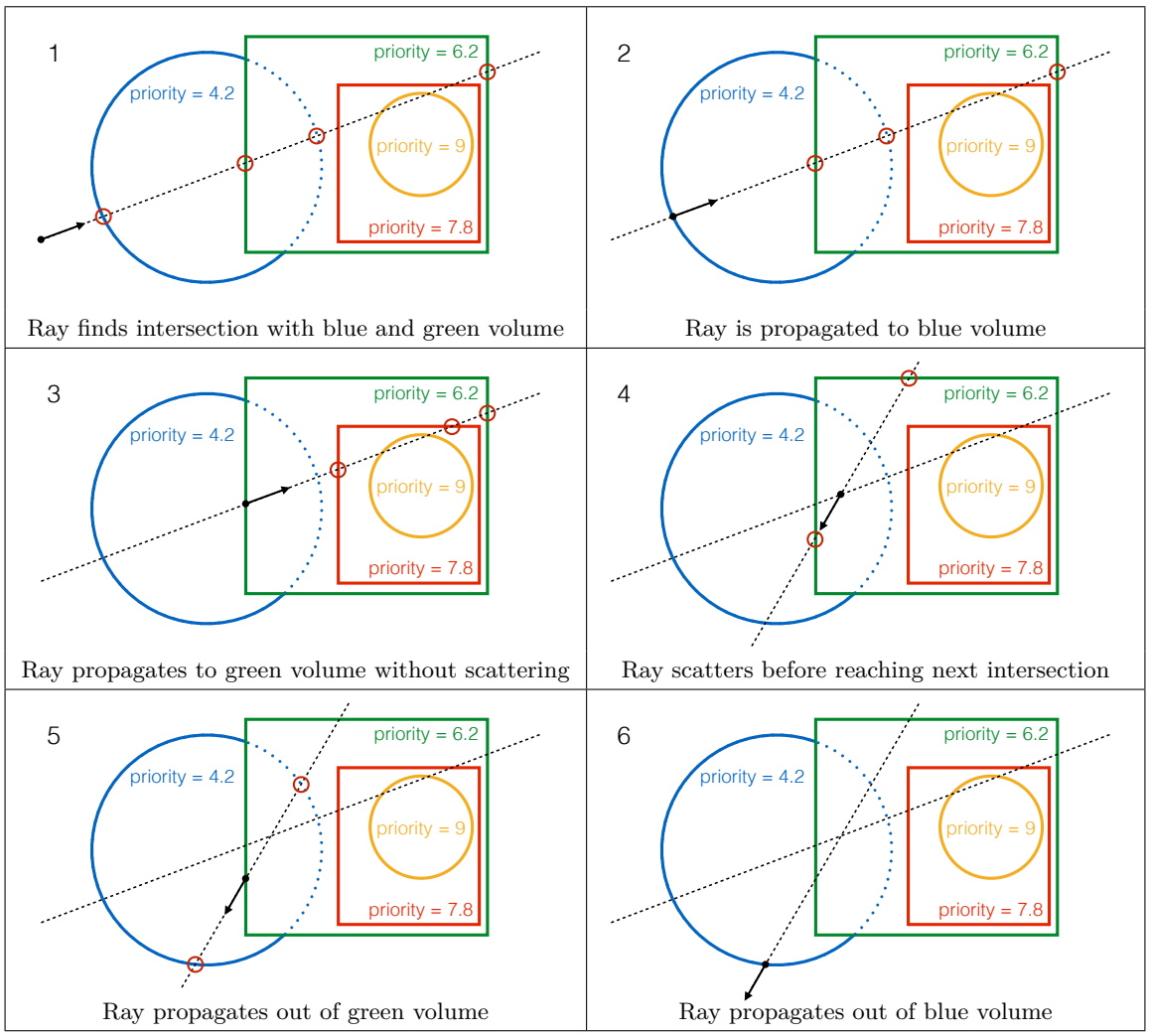


Figure 9: 2D visualization of the Union trace algorithm with a single scattering event, showing how unessecary intersection calculations are avoided.

4 Installation

Installation of the McStas Union components is as with all other McStas components, they can be placed in the folder of the instrument file or in the McStas path. There is no additional software needed.

5 Using the Union components

In this section each Union component and its options are described in turn. The reason for splitting the code over multiple components is to facilitate scalable, clear input of data. This approach is the major difference between the Union McStas components and previous components in terms of practical usage of the code.

The components are used in a specific order, as one first defines a number of physical processes using process components, then collects these into a material, and finally one places geometries in the simulation, each with a specific material. A geometry placed in the simulation with a defined material is referred to as a volume.

These components are all described in that order, and examples of each individual component is shown here, while an example showing how they work together is shown in section 6.

5.1 Scattering processes

A scattering process has no physical shape and only needs to describe the probability for scattering and what happens in a scattering event.

All processes support the interact_fraction options, which can be set to -1 to disable, and otherwise a number between 0 and 1 to describe the fraction of scattering events that select this particular process. When assigning multiple processes to one material, the sum of their interact fractions must be 1, and if a single interact_fraction is missing, it will be set so that the sum is 1.

5.1.1 Incoherent_process

The incoherent process is loosely based on the Incoherent McStas component. The process is isotropic and elastic. The setting parameters are shown in table 2. This component supports focusing, which is selected for each geometry component that chooses a material that contains this process, and is explained in section 5.3.

Parameter name	explanation	unit	default
sigma	Incoherent cross section per unit cell	[barns]	
unit_cell_volume	Volume of unit cell	[Å^3]	
packing_factor	Ratio between experimental powder density and theoretical	[0-1]	1
interact_fraction	Forces this fraction of processes to be incoherent	[0-1]	-1

Table 2: Setting parameters for Incoherent_process.comp

Below is an example of the Incoherent_process component being used to defined a process called "Vanadium_incoherent". The name of a process is very important, as it is needed later, however its position in space is irrelevant.

```
1 COMPONENT Vanadium_incoherent = Incoherent_process(sigma=6.08, packing_factor=1,
      unit_cell_volume=13.827, interact_fraction=0.5)
2 AT (0,0,0) ABSOLUTE
```

5.1.2 Powder_process

The powder process is a copy of the PowderN McStas component. Parameters names are the same, but absorption, incoherent scattering and geometry is removed while fraction_interact is added. An overview of the parameters is shown in table 3. All parameters after the dividing line is normally taken from the data file specified under reflections. This process does not support focusing defined in geometry inputs, but adds it's own through the d_phi parameter.

Parameter name	explanation	unit	default
interact_fraction	Forces this fraction of processes to be incoherent	[0-1]	-1
reflections	data file containing powder lines	string	"NULL"
format	Name of the format, or list of column indexes		
packing_factor	Ratio between experimental powder density and theoretical	unit less	1
d_phi	Vertical angular range to focus to, e.g. detector height	[deg,0-180]	0
	Remaining options used to overwrite parts from datafile		
Vc	Volume of unit cell	[Å^3]	
delta_d_d	Global relative delta_d_d/d broadening	[0-1]	0
DW	Global Debey-Waller factor when 'DW' column is not available	[0/1]	0
nb_atoms	Number of sub-unit per unit cell	[1]	1
density	Density of material. rho=density/weight/1e24*N_A.	[g/cm^3]	
weight	Atomic/molecular weight of material	[g/mol]	
barns	Flag to indicate if $\|F\|^2$ from 'reflections' is in barns or fm^2	0(laz)/1(lau)	1
Strain	Global relative delta_d_d/d shift	[ppm]	0

Table 3: Setting parameters for Powder_process.comp

Below is an example of a simple powder process called "Al_powder" that gets most information from the standard McStas Aluminium laz datafile, and uses focusing to only emit rays in a $\pm 10°$ interval (from the xz plane).

```
COMPONENT Al_Powder = Powder_process(reflections="Al.laz", d_phi=20)
AT (0,0,0) ABSOLUTE
```

5.1.3 Single_crystal_process

The single crystal process is a copy of the Single_crystal Mcstas component. Parameter names are kept the same, with the removal of references to absorption and incoherent and addition of interact_fraction and packing_factor. The parameters are shown in table 4. The powder/PG modes and the crystal curvature mode have not been ported yet, and is thus not yet supported.

Since the single crystal process is not isotropic, it will follow the orientation of the volume it is assigned to, but it is possible to add an additional rotation when defining the process by using the ROTATED keyword, it however needs to be relative to ABSOLUTE and not another component.

There is some issue in both the original Single_crystal component and the Union process where a ray can be trapped in a near infinite loop. In the Union process there is currently a limit of 10^4 iterations, after which the ray is discarded.

An example of the use of the Single_crystal_process component is shown below, where the orientation is rotated 1 degree around the McStas x-axis and 5 degrees around the McStas y-axis whenever this process is assigned to a geometry.

```
COMPONENT YBaCuO_single_crystal = Single_crystal_process(
          delta_d_d=delta_d_d, mosaic = mosaic,
          ax = 3.8186, ay = 0,      az = 0,
          bx = 0,      by = 3.8843, bz = 0,
          cx = 0,      cy = 0,      cz = 11.6777,
          reflections="YBaCuO.lau", barns=0, packing_factor=1)
AT (0,0,0) ABSOLUTE
ROTATED (1,5,0) ABSOLUTE
```

Parameter name	explanation	unit	default
reflections	data file containing powder lines	string	"NULL"
delta_d_d	Lattice spacing variance, gaussian RMS	[1]	1E-4
mosaic	Crystal mosaic (isotropic), gaussian RMS. Puts the crystal in the isotropic mosaic model state, thus disregarding other mosaicity parameters.	[arc minutes]	
mosaic_a	Horizontal (rotation around lattice vector a) mosaic (anisotropic), gaussian RMS. Put the crystal in the anisotropic crystal vector state. I.e. model mosaicity through rotation around the crystal lattice vectors. Has precedence over in-plane mosaic model.	[arc minutes]	
mosaic_b	Vertical (rotation around lattice vector b) mosaic (anisotropic), gaussian RMS.	[arc minutes]	
mosaic_c	Out-of-plane (rotation around lattice vector c) mosaic (anisotropic), gaussian RMS.	[arc minutes]	
mosaic_AB	In Plane mosaic rotation and plane vectors (anisotropic), mosaic_A, mosaic_B, A_h,A_k,A_l, B_h,B_k,B_l. Puts the crystal in the in-plane mosaic state. Vectors A and B define plane in which the crystal rotation is defined, and mosaic_A, mosaic_B, denotes the resp. mosaicities (gaussian RMS) with respect to the the two reflections chosen by A and B (Miller indices).	[arc_minutes, arc_minutes, 1, 1, 1, 1, 1, 1]	
recip_cell	Choice of direct/reciprocal (0/1) unit cell definition	[0/1]	0
ax	Coordinates of first (direct/recip) unit cell vector	[AA or $AA^{-}1$]	
ay	a on y axis	[AA or $AA^{-}1$]	
az	a on z axis	[AA or $AA^{-}1$]	
bx	Coordinates of second (direct/recip) unit cell vector	[AA or $AA^{-}1$]	0
by	b on y axis	[AA or $AA^{-}1$]	0
bz	b on z axis	[AA or $AA^{-}1$]	0
cx	Coordinates of third (direct/recip) unit cell vector	[AA or $AA^{-}1$]	0
cy	c on y axis	[AA or $AA^{-}1$]	0
cz	c on z axis	[AA or $AA^{-}1$]	0
interact_fraction	Forces this fraction of processes to be single crystal reflections	[0-1]	-1
packing_factor	Ratio between experimental powder density and theoretical	unit less	1
	Remaining options used to overwrite parts from datafile		
aa	Unit cell angles alpha, beta and gamma. Then uses norms of vectors a,b and c as lattice parameters	[deg]	
bb	Beta angle	[deg]	
cc	Gamma angle	[deg]	
barns	Flag to indicate if $\|F\|^2$ from 'reflections' is in barns or fm^2. barns=1 for laz and isotropic constant elastic scattering (reflections=NULL), barns=0 for lau type files	[0/1]	0

Table 4: Setting parameters for Single_crystal_process.comp

5.2 Union_make_material

The task of the Union_make_material component is to collect a number of physical processes into a material, and gives it a name that can be used to refer to this material later. The setting parameters for the component are shown in table 5. The process_string takes a comma separated list of process names that have already been defined in the instrument file, and collects the corresponding processes into a single material that can be

used when defining a volume using a geometry component. If the process_string is not set, the make_material component will collect the processes defined between itself and the previous make_material component. It is recommended for new users to manually enter the process_string to explicitly show which processes are used. If the absorber option is set, no processes will be collected regardless of the process_string, and the material will be an absorber.

It is not allowed to call a material "Vacuum" or "Exit", as these names are reserved to make Vacuum and Exit volumes. One can think of these as being defined per default.

Parameter name	explanation	unit	default
process_string	Comma separated names of defined processes	[string]	0
my_absorption	Inverse penetration depth at v = 2200 m/s	[1/m]	
absorber	If no processes wanted, set absorber to 1	[0/1]	0

Table 5: Setting parameters for make_material.comp

Below is an example of a material called "Weird_material" being made with the processes from the last three examples, "Vanadium_incoherent", "Al_powder" and "YBaCuO_single_crystal". The name "Weird_material" is to be used when assigning this material to a geometry to make a volume.

```
COMPONENT Weird_material =
    Union_make_material(process_string="Vanadium_incoherent,Al_powder,
        YBaCuO_single_crystal", my_absorption=100*1.2/66.4)
AT (0,0,0) ABSOLUTE
```

5.3 Geometry components

All geometry components have some common setting parameters describing necessary properties of a volume, which are described in table 6.

The material_string is used to select a user defined material made with make_material, "Vacuum" or "Exit". A Vacuum volume has no physical processes and a zero absorption cross section. An Exit volume lets the ray exit the multiple scattering loop when the ray is propagated into this volume, which is useful for adding detectors or other components inside a complicated geometry.

The priority of a volume needs to be unique, the component will exit if this is not the case. The priority is used to determine which volume occupy space that both geometries cover. When placing a volume inside another volume, the one on the inside needs the highest priority, otherwise it will not be used.

The visualize option can be used to avoid mcdisplay drawing the volume, which can be useful to avoid clutter in mcdisplay.

The p_interact option sets the probability for a ray to undergo a scattering event in the volume, regardless of the length travelled in the volume. In contrast to other McStas implementations of p_interact, this applies for all steps of multiple scattering as well, meaning setting a interaction probability of 50% would result in a 25% chance of two scattering events. It is hence not advised to set this parameter close to 1, as it results in high probabilities for higher order multiple scattering.

The focusing options are standard for McStas, leaving them all to the default value of 0 will disable focusing, allowing scattering into 4π. It is not guaranteed that all processes assigned to a material support focusing, but all that does will follow the focusing set in the geometry component.

5.3.1 Union_cylinder

The Union_cylinder component uses the standard McStas cylinder intersect function. The cylinder symmetry axis is along the McStas y axis as shown in figure 10. The McStas AT and ROTATED keywords are used as normal. No ray tracing is done in this component, it merely sets up for the master union component.

Below are two examples of the use of the Union_cylinder component, the first using the already defined "Weird_material" to make a cryostat wall, and another instance of the Union_cylinder to hollow it out using

Parameter name	explanation	unit	default
material_string	String selecting a defined material	string	
priority_input	The priority of the volume	double	
visualize	Control if this volume should be shown in mcdisplay	[0/1]	1
p_interact	Probability for scattering event when propagating in this volume	[0-1]	0
Focusing position			
target_index	Index of component used as focusing target	integer	0
target_x	Focusing target coordinate if target_index not set	[m]	0
target_y	Focusing target coordinate if target_index not set	[m]	0
target_z	Focusing target coordinate if target_index not set	[m]	0
Angular focusing			
focus_aw	Angular focusing in width direction	[deg]	0
focus_ah	Angular focusing in height direction	[deg]	0
Rectangular focusing			
focus_xw	Spatial focusing rectangle width	[m]	0
focus_xh	Spatial focusing rectangle height	[m]	0
Circle focusing			
focus_r	Spatial focusing circle radius	[m]	0

Table 6: Setting parameters for make_material.comp

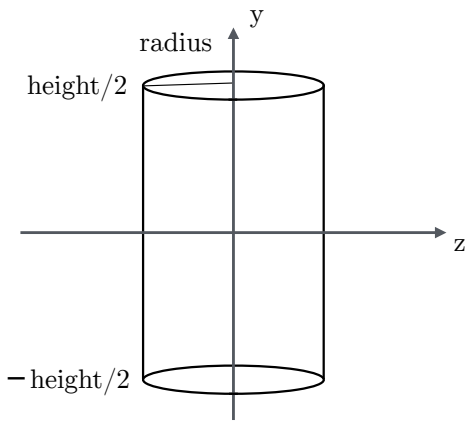


Figure 10: Basic cylinder geometry used for Union_cylinder.

Parameter name	explanation	unit	default
radius_input	Radius of the cylinder	[m]	
height_input	Height of the cylinder	[m]	

Table 7: Setting parameters for Union_cylinder.comp

the predefined ”Vacuum” material. By displacing the second 1 cm upwards, a 1 cm thick bottom is created while leaving the top open.

```
COMPONENT cryostat_wall = Union_cylinder(radius_input=0.1, height_input=0.2,
    priority_input=10, material_string="Weird_material", p_interact=0.2)
AT (0,0,0) RELATIVE sample_position
ROTATED (0,0,0) RELATIVE sample_position

COMPONENT cryostat_vacuum = Union_cylinder(radius_input=0.09, height_input=0.2,
    priority_input=11, material_string="Vacuum")
AT (0,0.01,0) RELATIVE sample_position
ROTATED (0,0,0) RELATIVE sample_position
```

5.3.2 Union_box

The box geometry can be a simple box, or by using the optional parameters the face at positive Z can have different height and width, as shown in figure 11. Unique setting parameters for the Union_box component are shown in table 8. The McStas AT and ROTATED keywords are used as normal. No ray tracing is done in this component, it passes all the information to the Master_union component.

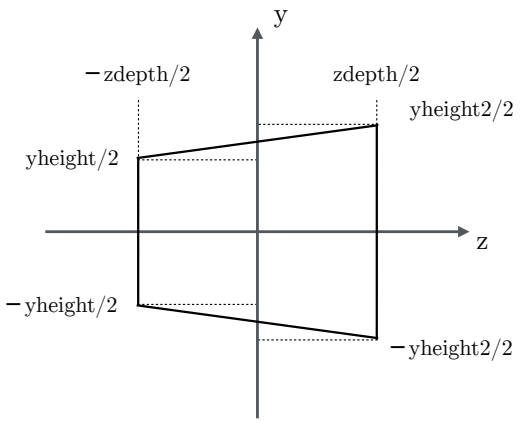


Figure 11: Basic box geometry used for Union_cylinder.

Parameter name	explanation	unit	default
xwidth	Width of the box	[m]	
yheight	Height of the box	[m]	
zdepth	Height of the box	[m]	
xwidth2	Optional width of the box (positive z face)	[m]	-1
yheight2	Optional height of the box (positive z face)	[m]	-1

Table 8: Setting parameters for Union_box.comp

Here the Union_box component is used to create an Exit volume that fits around a PSD_monitor.

```
COMPONENT detector_exit_volume = Union_box(xwidth=0.05, yheight=0.05, zdepth=0.0001,
    priority_input=15.2, material_string="Exit")
AT (0,0,0) RELATIVE detector_position
ROTATED (0,0,0) RELATIVE detector_position

COMPONENT detector = PSD_monitor(xwidth=0.05, yheight=0.05, nx=100, ny=100, filename="
    PSD.dat", restore_neutron=1)
AT (0,0,0) RELATIVE detector_position
ROTATED (0,0,0) RELATIVE detector_position
```

5.4 Union_master

The Union master component is simply called Union_master, and has no setting parameters yet. It is this component that executes the actual simulation, and thus places the Union into the linear succession of McStas components. In case of multiple Union_master components in one instrument file, It will include all volumes defined after the previous Union_master component. If any volume is defined after the last Union_master component, it will not be included in the simulation.

The McStas keyword SCATTER is used in a non-standard way in this component to show any point a ray propagates from one volume to another in mcdisplay. For this reason there is a variable called number_of_scattering_events that can be used in EXTEND to retrieve the information normaly available through the SCATTERED variable, and an array called scattered_flag that contains the number of scattering events in each volume. The example below uses these two variables to set scattering flags available in the instrument file.

```
COMPONENT sample = Union_master()
AT (0,0,0) RELATIVE sample_position
ROTATED (0,0,0) RELATIVE sample_position
EXTEND
%{
if (number_of_scattering_events > 0) scattering_flag = 1; else scattering_flag = 0;
if (scattered_flag[3] > 0) scattering_flag_V3 = 1; else scattering_flag_V3 = 0;
%}
```

6 A simple instrument

The trace section of a simple instrument file is shown to highlight the linking needed between Union components. The file is distributed with the early versions of the code.

```
COMPONENT Al_inc = Incoherent_process(sigma=4*0.0082,packing_factor=1,unit_cell_volume=66.4)
AT (0,0,0) ABSOLUTE

COMPONENT Al_powder = Powder_process(reflections="Al.laz",d_phi=20)
AT (0,0,0) ABSOLUTE

COMPONENT Al = Union_make_material(my_absorption=100*4*0.231/66.4,process_string="Al_inc,Al_powder
    ")
AT (0,0,0) ABSOLUTE

COMPONENT Fe_inc = Incoherent_process(sigma=2*0.4,packing_factor=1,unit_cell_volume=24.04)
AT (0,0,0) ABSOLUTE

COMPONENT Fe_powder = Powder_process(reflections="Fe.laz",d_phi=20)
AT (0,0,0) ABSOLUTE

COMPONENT Fe = Union_make_material(my_absorption=100*2*2.56/24.04,process_string="Fe_inc,Fe_powder
    ")
AT (0,0,0) ABSOLUTE

COMPONENT Progress = Progress_bar()
  AT (0,0,0) ABSOLUTE

COMPONENT Source = Source_simple(xwidth=0.05, yheight=0.05, dist=2, focus_xw=.05, focus_yh=.05, E0
    =14, dE=2)
AT (0,0,0) RELATIVE Progress

COMPONENT Guide = Guide_gravity(w1=0.05, h1=0.05, w2=0.05, h2=0.05, l=12, R0=0.99, Qc=0.0219, alpha
    =6.07, m=3.0, W=0.003)
AT (0,0,2) RELATIVE Source

COMPONENT sample_position = Arm()
AT (0,0,12.5) RELATIVE Guide

COMPONENT Hexagonal_container_1 = Union_box(xwidth=0.01, yheight=0.04, xwidth2=0.01+2*0.01*sin(30*
    DEG2RAD), zdepth=0.01*cos(30*DEG2RAD), priority_input=1, material_string="Al", p_interact=0.3)
AT (0,0,-0.5*0.01*cos(30*DEG2RAD)-0.00001) RELATIVE sample_position
ROTATED (0,0,0) RELATIVE sample_position

COMPONENT Hexagonal_container_2 = Union_box(xwidth=0.01+2*0.01*sin(30*DEG2RAD), yheight=0.04,
    xwidth2=0.01, zdepth=0.01*cos(30*DEG2RAD), priority_input=2, material_string="Al", p_interact
    =0.3)
AT (0,0,0.5*0.01*cos(30*DEG2RAD)+0.00001) RELATIVE sample_position
ROTATED (0,0,0) RELATIVE sample_position

COMPONENT Sample = Union_cylinder(radius_input=0.008, height_input=0.36, priority_input=3,
    material_string="Fe", p_interact=0.5)
AT (0,0,0) RELATIVE sample_position
ROTATED (0,0,0) RELATIVE sample_position

COMPONENT Master = Union_master()
AT(0,0,0) RELATIVE sample_position

COMPONENT Banana_monitor = Monitor_nD(radius=1, yheight=0.1, options="banana, theta limits
    =[20,170], bins=200",filename="banana.dat")
AT (0,0,0) RELATIVE sample_position
```

7 Tagging output from Union component

The Union component writes a file called Union_histories.dat that contains all sampled histories. A history consist of a list of all volumes the ray entered in chronological order and all processes the ray undertook. So all rays that goes from volume 0, to volume 1, undergoes scattering process 2 of volume 1, then goes back to volume 0 is considered the same history. These histories are sorted after the total intensity that leaves the Union component with this history. A sample of the top 15 histories for a simple setup is displayed here.

```
1  History file written by the McStas component Union_master
2  When running with MPI, the results may be from just a single thread, meaning
       intensities are divided by number of threads
3  ------ Description of the used volumes ------------------------------------------------
4  V0: Surrounding vacuum
5  V1: powder_container  Material: Al   P0: Al_incoherent P1: Al_Powder
6  V2: powder_inside_container  Material: Cu_powder   P0: Cu_incoherent_process P1:
       Cu_powder_process
7  ------ Histories sorted after intensity ------------------------------------------------
8  12221626 N I=4.188166E-06    V0
9  1882051  N I=1.052731E-06    V0 -> V1 -> V2 -> V1 -> V0
10 1517013  N I=6.213315E-07    V0 -> V1 -> V0
11 188661   N I=8.043799E-08    V0 -> V1 -> V2 -> P0 -> V1 -> V0
12 752943   N I=3.823911E-08    V0 -> V1 -> V2 -> P1 -> V1 -> V0
13 771437   N I=2.176363E-08    V0 -> V1 -> P1 -> V0
14 181532   N I=1.101451E-08    V0 -> V1 -> P1 -> V2 -> V1 -> V0
15 286771   N I=7.628450E-09    V0 -> V1 -> V2 -> V1 -> P1 -> V0
16 18736    N I=1.948079E-09    V0 -> V1 -> V2 -> P0 -> P0 -> V1 -> V0
17 75736    N I=1.319653E-09    V0 -> V1 -> V2 -> P1 -> P0 -> V1 -> V0
18 33463    N I=1.100647E-09    V0 -> V1 -> V2 -> V1 -> P1 -> V2 -> V1 -> V0
19 74590    N I=1.016961E-09    V0 -> V1 -> V2 -> P0 -> P1 -> V1 -> V0
20 301006   N I=6.623860E-10    V0 -> V1 -> V2 -> P1 -> P1 -> V1 -> V0
21 18169    N I=4.341707E-10    V0 -> V1 -> P1 -> V2 -> P0 -> V1 -> V0
22 13310    N I=4.054218E-10    V0 -> V1 -> P0 -> V0
```

The file starts with a short description of each volume including what material it is made of and the scattering processes associated with that material.

The first number in a row of data is the number of rays with this history, the next is the intensity, and then the string which is the history itself. VX refers to volume number X, and PX refers to process number X within the current volume. When there are volumes with different materials, PX can refer to different processes depending on which volume the ray is currently in, meaning P0 in "V0 -> V1->P0" and "V0 -> V2 -> P0" refers to two different processes if volume 1 and volume 2 are different materials.

In this example, 2E7 rays were simulated, but only 2.5E4 unique histories were sampled, making the data file manageable at just 2.8MB. The size for more complicated cases can be significantly larger, and if it becomes problematic (needs to fit in RAM) the feature can be turned of with a simple control variable called "enable_tagging" in the declare section of the Union_master component.

The practical uses for this data file are many. If for example one wants to estimate the background originating from a certain part of a sample holder described by volume X, one simply adds the intensity of all histories that scattered in that volume by searching for "VX -> P". If only background at the detector is important, one can place an exit volume with number Y at the detector and add the intensities for all histories that contain the string "VX -> P" and ends in "VY". This have always been possible with McStas by tagging the neutrons manually using EXTEND, but that requires running the simulation again in order to investigate a new problem, here all histories are available after one simulation.

In the current version there are no easy ways to get the history each ray in the EXTEND section of Union_master, but this will be made available in later versions to make the traditional tagging method available.

8 Validation

Here results of early validation efforts are shown. It is intended to do a more comprehensive validation against both analytical results and other McStas components later.

8.1 Incoherent scattering

A Union component of a simple cylinder was compared to the same geometry using the Incoherent McStas component. The scattering and absorption cross section are for Vanadium. A comparison for two of the monitors are shown in figure 12. The results are as expected.

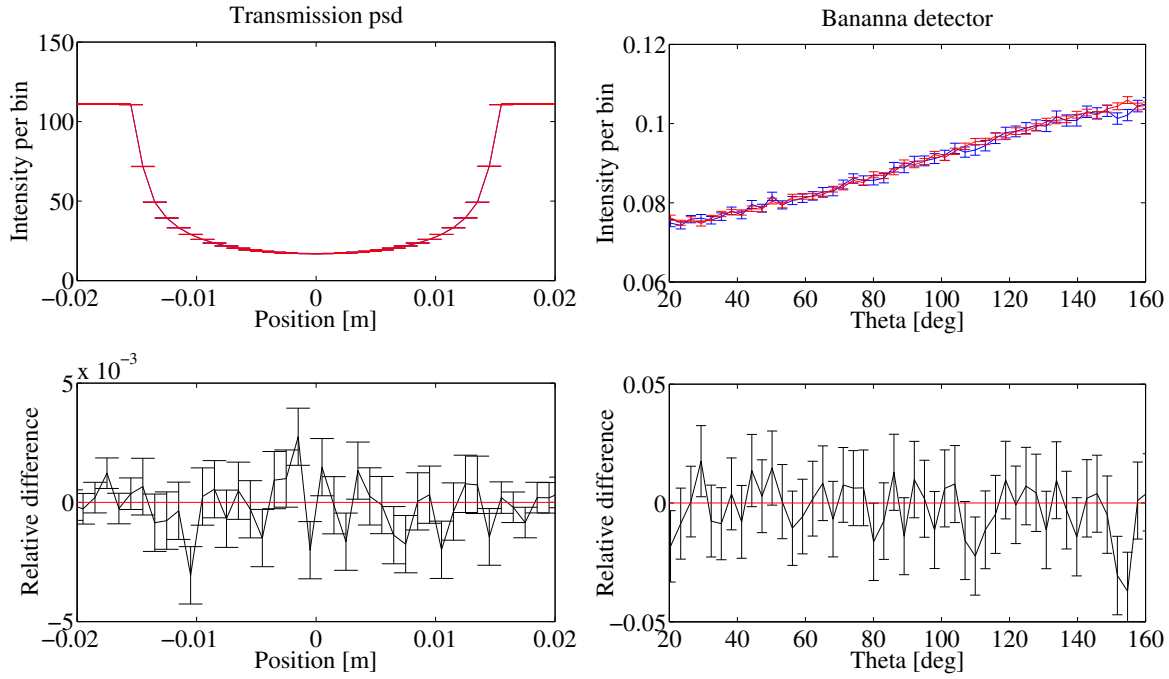


Figure 12: Left: Transmission psd for both with relative deviation below. Right: Banana detector for both with relative deviation below. The blue is for the Union component while the red is for the Incoherent component.

8.2 Powder scattering

Here the Union components process are validated against the PowderN component. The PowderN component have both powder and incoherent scattering, so these two Union processes are added to a single material. The powder process is based directly on the PowderN component. There is however one important difference, as the Union project adds multiple scattering, but this is taken into account by having separate detectors that ignore multiple scattering. This validation have been done for many compounds, and the results for Cu are shown in figure 13 and 14. For the banana detector where the full scattering is taken into account, a clear deviation can be seen in areas without powder peaks, as the multiple scattering from the powder peaks are comparable to the incoherent background. When comparing using monitors only taking single scattering into account, the results are much closer, but perhaps better statistics could reveal some differences.

The computation time for the Union component is approximately 50% longer than the PowderN component, which may be partially explained by excessive error checking during debug phase, but is hoped to be reduced in the future.

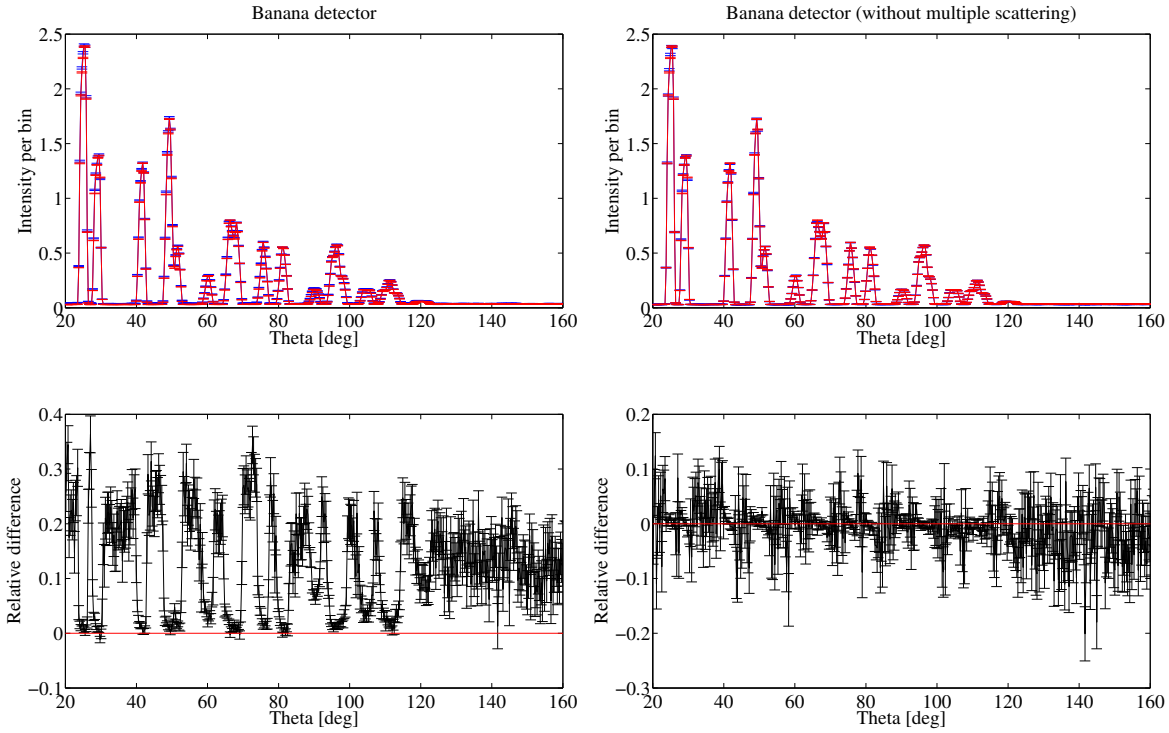


Figure 13: Banana detector results for Union components (blue) and PowderN (red). Left: All scattering. Right: Only single scattering.

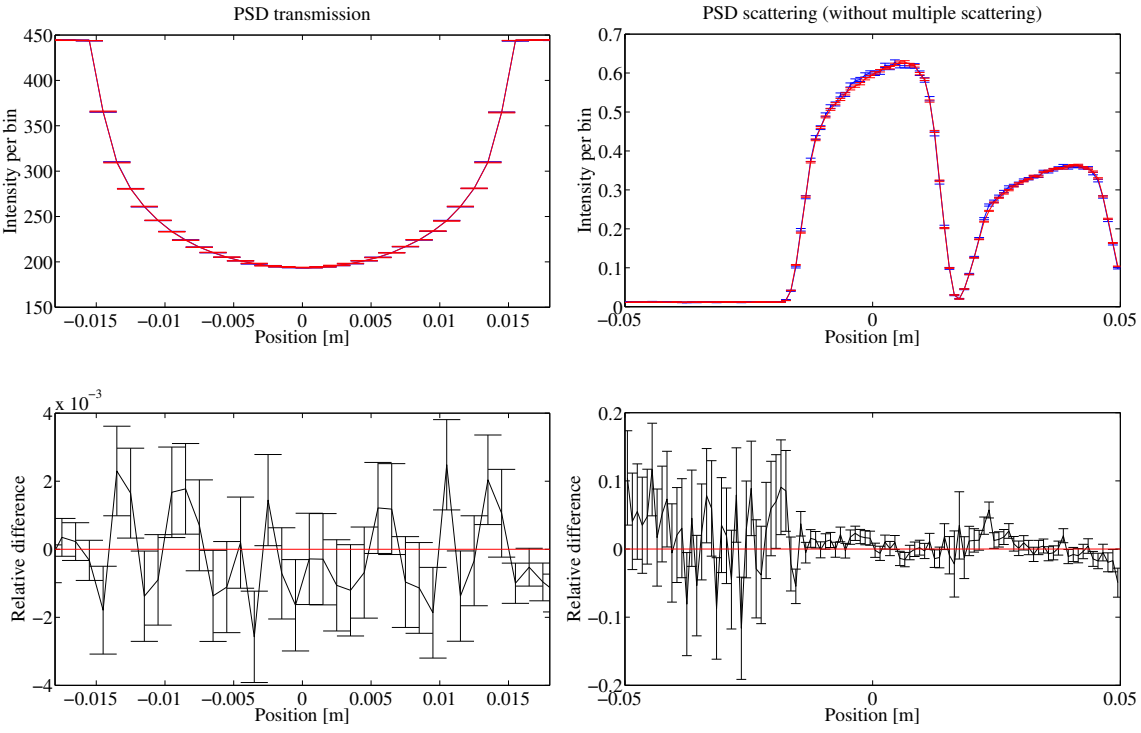


Figure 14: Validation results for Union components (blue) and PowderN (red). Left: Transmission position sensetive detector. Right: Position senestive detector that probes a few powder peaks and some incoherent background.

8.3 Single crystal

Validation results for comparison with the Single_crystal component are shown in figure 15 and 16. Both position sensitive detectors are placed directly behind the single crystal, the difference being one only detects

rays that have scattered and the other only detects rays that did not. For the detector measuring rays that did not scatter there is a clear difference between the two components, but it is in the order of 0.1%, and likely originates from slightly different ways of handling scattering position and absorption. Further work have to be done to confirm which of the two approaches are correct, if any.

Another difference is the mentioned problem in the single crystal algorithm where rays can end in infinite loops, they are terminated after 1E4 iterations in both codes, but in the Union project these are discarded, while in the Single_crystal component they are forced to escape the crystal with whichever wavevector it currently has. For this test case the escaping wavevector would have either a direction towards the detector or away from them, but the Union component is the one with greater intensity, meaning it is does not explain the deviation.

None of the other detectors show clear deviations.

With strongly scattering crystals it is not recommended to use the p_interact feature, as the resulting scattering pattern is heavily dependent on very high orders of multiple scattering which will not be probed when this feature is used.

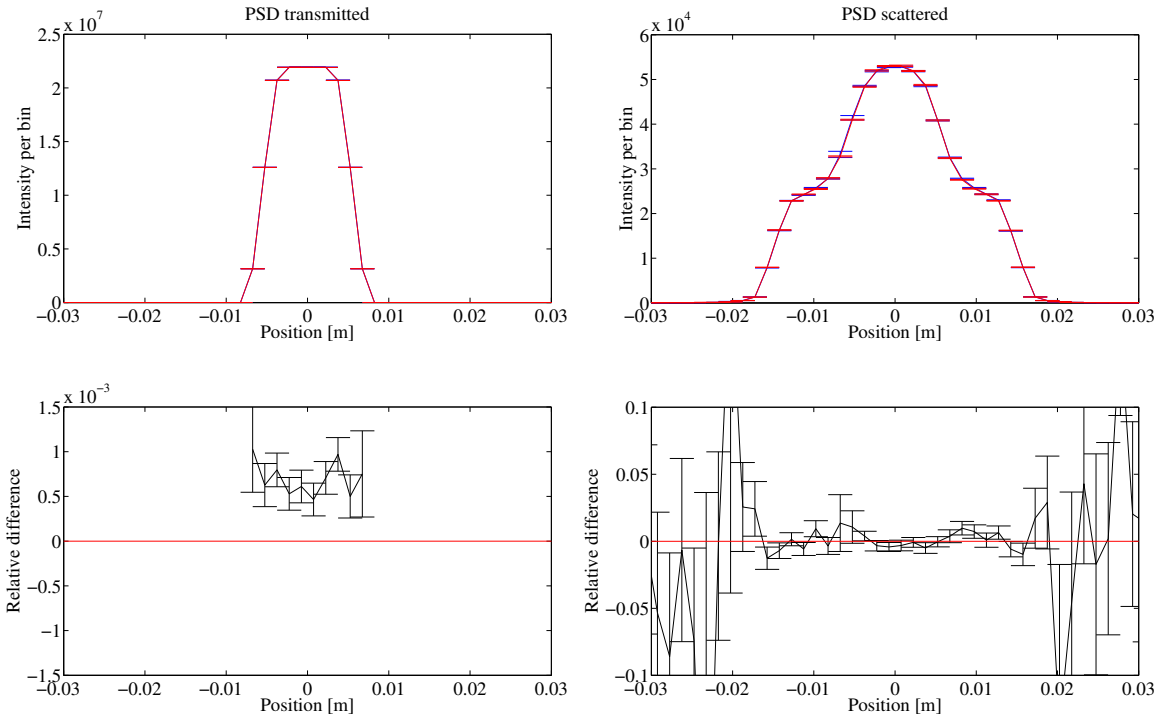


Figure 15: Validation results for Union components and Single_crystal. Left: Transmission position sensitive detector, where only neutrons transmitted are counted. Right: Transmission position sensitive detector where only scattered neutrons are counted.

8.4 Conclusion on validation

The quick validation done here is sufficient to show the scattering processes in the Union project are in reasonable agreement with the ones present in current McStas components. It also shows the ability of the Union components to balance two different scattering processes correctly, as using two processes together was in agreement with a single component hardcoded to balance these. The tests were repeated with forced interaction fraction for the powder and incoherent processes and geometries, and the results were similar, meaning these are also validated to a reasonable degree. In the case of the single crystal process, the results differs significantly when using the forced interaction fractions for processes and geometries, as high number of coherent multiple scattering events are needed to accurately describe the resulting beam.

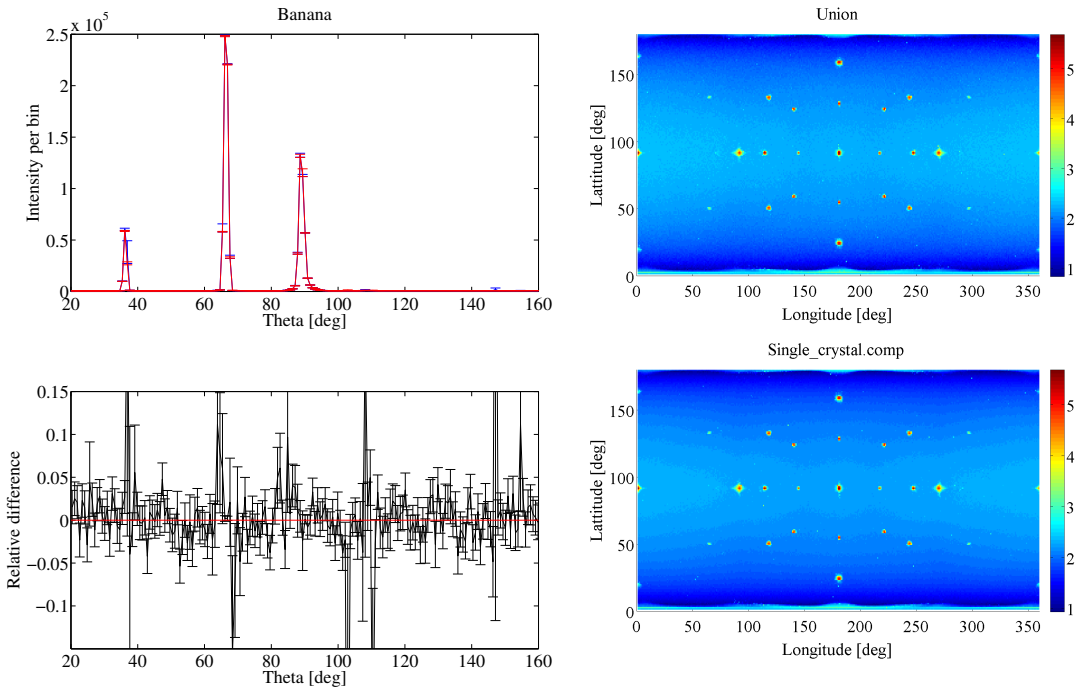


Figure 16: Left: Banana detector results for Union components (blue) and Single_crystal (red). Right: Scattering into 4π, Union components top and Single_crystal bottom, log10 of intensities on colorscale.

9 Adding a new physical process

It is of highest priority that adding a new physical process to the Union project is as easy as possible. It does however have some requirements not present when adding a normal McStas component. The main part is to create a process component, which will define the functions for determining scattering probability and the function called for scattering events. The component initialize and input can be used as normal. In many cases calculations done in the function that determines scattering probability can be useful for calculating the final wavevector after the scattering event, and there is thus the possibility to transfer information between the two using a structure. At the end of the component file the information is packed into a global variable, but this part will be identical for all process components and is thus not a concern. A new_process template is provided to get new users started.

In addition to making the component, one has to add a declaration of the structure used for transporting in the Union_functions.c file under the data_transfer_union.

10 Adding a new geometry

The work required to add a new geometry is significantly larger, and will thus probably be reserved for the developer. The reason is that in addition to the obvious requirement of adding a function describing the intersection between a ray and the new geometry, functions are needed for the following tasks.

- Check if a point is within the new geometry
- Transform necessary vectors from the geometry components frame of reference to the Union_master components frame of reference
- Simple mcdisplay code
- Check if the new geometry overlaps with all other types of geometries
- Check if the new geometry is completely inside all other types of geometries

- Check if all other geometries are completely inside the new geometry

As more geometries are added, the work required to add a new geometry thus increases. In addition to the above, a reference to a struct to storage the geometrical parameters needs to be made in Union_functions.c, and the generate_children_lists/generate_overlap_lists functions need simple updates to use the new functions. All memory allocations are ready for geometries where more than two intersection times, but currently set to only two intersections for all volumes.

11 Planned features

So far the overwelming majority of the work have gone into the core of the Union project, the Union_master component. With this release of beta code, the focus will shift to adding more physical processes and geometries, even though there is still work to be done on the core. The next geometries will probably be a sphere and a cone. The next scattering processes will probably be inelastic, as the basic types of elastic scattering is covered.

The tagging system will be expanded, probably with the option to add additional details to the data collected, like the mean and standard deviation of the wavelength, time or similar for each history. In addition the data file should be placed with the other detector files, and it should have a custom name.

Input sanitation is also an important aspect, as the code needs to tell the user what went wrong in case of errors in the input. This is largely non-existent in the current version.

There are also plans for adding a new kind of process, a surface process, which can be effects like supermirror reflectivity and refraction.

The direction of the project and priorities of the above tasks depends on feedback from beta users (all whom read this document), which is highly appreciated.

12 Known bugs

It is still early code, meaning there are probably bugs to be discovered. The biggest problem at the moment is a crash which happens when two planes are overlapping perfectly, as the algorithm can not find which of the two volumes the ray enters when intersecting with this plane, and in most cases that neutron is lost. Remember it is perfectly fine to overlap volumes, so instead of letting two volume just touch, make them overlap a tiny bit, or leave a tiny amount of space in-between. Under normal circumstances, even in the complicated demonstrations shown here, less than one ray out of 1E11 should be lost. When using the Single crystal processes however there can however be significantly more due to the problems described in section 5.1.3. This early code still makes many unnecessary checks to confirm the code is working correctly, meaning the user will be notified if problems occur.

The terminal output from the components is large, and will be reduced in the future.

- Crash if two planes of boxes / ends of cylinders overlap perfectly
- Can not compile if no processes are defined before Union_make_material
- Can not compile if no materials are defined
- Can not compile if no geometries are defined
- Crash if a material_string is set to something not defined
- When using MPI only history for one thread is written to disk / not sorted correctly
- Mcdisplay zoom level not always reasonable
- Near infinite loops in Single crystal process