McStas hands-on exercise: Reactor source diffractometer (and optionally 3-axis spectrometer)

Peter Willendrup and Erik Knudsen, DTU Physics
Linda Udby, Niels Bohr Institute, University of Copenhagen

July 16, 2012

Postal adress:
Physics Department
Technical University of Denmark
Fysikvej 307
DK-2800 Kongens Lyngby, Denmark

email:
pkwi@fysik.dtu.dkmailto:pkwi@fysik.dtu.dk,erkn@fysik.dtu.dkmailto:erkn@fysik.dtu.dk

Abstract

This document is a tutorial about McStas and neutron scattering for beginners.

The text below is also included as a chapter in the McStas manual.

1 Introduction

This tutorial has been written to help out novel users of McStas and neutron scattering instruments. McStas is a software package for simulating neutron scattering experiments using a Monte Carlo ray-tracing technique. This paper aims at helping the user to gain insight into basic neutron scattering as well as neutron raytracing using the McStas software package [1],[2],[4].

1.1 Prerequisites

Needed knowledge and equipment to work through the tutorial is

- Undergraduate knowledge of mathematics and physics.
- A computer with McStas installed (refer to the McStas homepagehttp://mcstas.risoe.dk [4] for details) or a bootable McStas Ubuntu live DVD (installation to harddisk possible, but not required).
- This tutorial.

1.2 Goals and tasks

The goals and tasks of this tutorial are

- To teach you about the most basic neutron scattering.

- To let you understand some of the typical components in a neutron scattering instrument.
- To teach you basic usage of the McStas neutron simulation package.
- To let you create your first McStas instruments, a two axis diffractometer and a triple axis spectrometer.
- To teach you how to modify your instrument for a specific task.
- To help you learn to debug instruments.
- To help you aquire and analyze data from McStas simulations.

2 Basic neutron scattering

You may recall the Bragg law from your high school physics

$$n\lambda = 2d\sin(\theta),$$

giving the scattering condition for a wave of wavelength λ against a series of lattice planes with lattice spacing d, rotated the angle θ off the lattice plane normal. n is an integer giving the spectral order of the scattered wave. In neutron science one often refers to the *scattering vector*, $\vec{\kappa}$ of a given reflection, where

$$\kappa = |\vec{\kappa}| = n\frac{2\pi}{d}.$$

This gives us the scattering vector formulation of the Bragg law

$$\kappa = 2k\sin(\theta),$$

where $k = \frac{2\pi}{\lambda}$. The Bragg law / scattering condition is illustrated in Figure 1. Most of the neutron

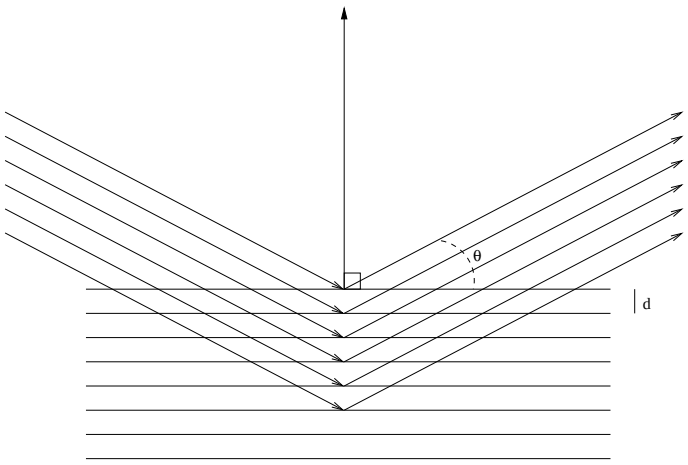

Figure 1: Illustration of the Bragg Law.

processes we will study in this paper are elastic, meaning that the wavelength of the neutron is unaltered by the process.

3 Basic understanding of instrument components

In the McStas formulation of a neutron scattering instrument, all objects apart from the neutron ray are referred to as components. This includes for instance

- **Source** The exit of a neutron production facility, where neutron rays of certain velocities are emitted into some portion of space.
- **Monochromator** (Idealized) crystal that is used to select neutrons of a single wavelength[1] λ_0 to probe the sample with (monochromator) or to analyze with (analyzer).
- **Sample** An object altering the neutron physical properties in some sense, examples used here are:
  - Vanadium. Scatters incoming neutron rays incoherently.
  - PowderN. Can be thought of as a large number of crystals, each scattering neutron rays according to Braggs law, thereby producing N concentric Debye Scherrer cones. This sample also has the posibility of adding inchoherent, eleastically scattered neutron rays.
- **Monitors** Objects *monitoring* or registering neutron ray characteristics. In the exercises below are used different types of detectors or monitors:
  - Monitor. Single monitor, detecting the number of neutrons flying through a plane. (User defined opening size).
  - PSD_monitor. Square monitor, detecting the number of neutron rays passing through a plane, divided into pixels. Square regions of a plane. (User defined resolution and opening size).
  - PSD_monitor_4PI. As PSD_monitor but shaped like a sphere.
  - L_monitor. Wavelength monitor, measuring the different wavelengths of the passing neutron rays. (L is for λ).
  - Monitor_nD. General monitor for detecting all sorts of physical properties of the neutron ray. In our cases used with options:
    - 'single' - as PSD_monitor but only one small square.
    - 'banana' - as PSD_monitor but shaped like a curved, horizontal band.
- **Collimators** Devices controling the direction and divergence of the neutron ray.
  - Collimator_linear A series of parallel absorbing neutron plates that limits the beam divergence. Typical values are $6'$ to $120'$.

More information on the McStas components is available by using the `mcdoc` program (You may need to set the BROWSER system variable to your webbrowser of choice):

- `mcdoc -s` , Shows a html list of all the components
- `mcdoc Monitor.comp` , Shows the documentation for a given component
- `mcdoc -M` , brings up the McStas manual in PDF format
- `mcdoc -c` , brings up the McStas component manual in PDF format

4 Basic McStas

In short, the core of the McStas system is a precompiler. From a user-provided instrument description, components are assembled into a single piece of `ansi-c` code. Using a compiler, *e.g.* `gcc`, the c code is compiled into an executable program which can be run on your computer. Optionally, the program takes input arguments to tune the setup of your instrument/simulation. This section will take you through a

[1] In reality, the monochromator selects a normal distribution of wavelegths around λ_0, and perhaps higher orders as well ($n = 2, 3, ...$ in Braggs law)

simple example instrument to teach you the basic instrument language of McStas. (Instrument filename is vanadium_example.instr, can be loaded using the `Neutron Site/Tutorial` menu item of the `mcgui`, see below). Please study *carefully* the instructive comments, marked by `/* ... */` characters

```
/* The line below defines the 'name' of our instrument */
/* Here, we have a single input parameter, ROT         */
DEFINE INSTRUMENT vanadium_example(ROT=0)

/* The DECLARE section allows us to declare variables  */
/* in c syntax. Here, coll_div (collimator divergence) */
/* is set to 60 arc minutes...                         */
DECLARE
%{
  double coll_div = 60;
%}

/* Here comes the TRACE section, where the actual      */
/* instrument is defined....                           */
TRACE

/* The Arm() class component defines reference points  */
/* and directions in 3D space. Every component instance*/
/* must have a unique name. Here, arm is used. This    */
/* Arm() component is set to define the origin of our  */
/* global coordinate system (AT (0,0,0) ABSOLUTE)      */
COMPONENT arm = Arm() AT (0,0,0) ABSOLUTE

/* Next, we need some neutrons. Let's place a neutron  */
/* source. Refer to documentation of Source_flat to    */
/* understand the different input parameters.          */
/* The source component is placed RELATIVE to the arm  */
/* component, meaning that modifying the position or   */
/* orientation of the arm will also affect the source  */
/* component (and other components after that one...)  */
COMPONENT source = Source_simple(radius = 0.015, dist = 1,
  xw=0.024, yh=0.015, E0=5, dE=0.2)
 AT (0,0,0) RELATIVE arm

/* Here we have a collimator - placed to improve beam  */
/* divergence. The component is placed at a distance   */
/* RELATIVE to a previous component...                 */
COMPONENT collimator = Collimator_linear(len = 0.2,
  divergence = coll_div, xwidth = 0.04, yheight=0.06)
  AT (0, 0, 0.4) RELATIVE arm

/* We also need something to 'shoot at' - here a sample*/
/* made from vanadium - an isotrope scatterer. Options */
/* are available to restrict the solid angle in which  */
```

```
/* neutrons are emitted (no need to simulate neutrons  */
/* that we know for sure will not reach the rest of    */
/* instrument).                                        */
/* Other options for smart targeting are available -   */
/* refer to component documentation for info.          */
COMPONENT target = V_sample(thickness = 0.004, radius = 0.012,
  yheight = 0.015, focus_r = 0, pack = 1,
  target_x = 0, target_y = 0, target_z = 1)
  AT (0,0,1) RELATIVE arm

/* Here, a secondary arm - or reference point, placed  */
/* on the sample position. The ROT parameter above     */
/* defines rotation of this arm (and components        */
/* relative to the arm)                                */
COMPONENT arm2 = Arm()
  AT (0,0,0) RELATIVE target
  ROTATED (0,ROT,0) relative arm

/* For data output, let us place a detector. This      */
/* detector is not very realistic, since it is sphere  */
/* shaped and has a 10 m radius, but has the advantage */
/* that EVERYTHING emitted from the sample will be     */
/* picked up. Notice that this component changes       */
/* orientation with the ROT input parameter of the     */
/* instrument.                                         */
COMPONENT PSD_4pi = PSD_monitor_4PI(radius=10, nx=101, ny=51,
  filename="vanadium.psd")
  AT (0,0,0) RELATIVE arm2
END
```

Enlightened by the above example, you are probably now ready to learn a few more important details and tips about McStas.

- **Neutron representation:** A neutron 'history' or package is an entity representing a large number of neutrons. It has the following physical properties:
  - Spatial coordinates, $\vec{x}$ or x, y, z.
  - Velocity components, $\vec{v}$ or v_x, v_y, v_z.
  - Spin components, $\vec{s}$ or s_x, s_y, s_z.
  - Time, t.
  - Neutron weight factor, p.
- **Neutron histories/Intensities:** McStas simulates neutron histories rather than direct neutron counts, *i.e.* when a Monte Carlo choice is made in a given component (*e.g.* a random number is generated to decide a new direction of the neutron ray), the neutron *weight factor* is adjusted accordingly. As you may have guessed already, the weight factor is the average number of of observed neutrons of a given behaviour. The transition to direct neutron intensites is made by adjusting the initial neutron weight of the source component, so that the sum of all simulated weight factors equals the absolute intensity of neutrons emitted in one second. This means that the intensity of the neutron beam at a given position is the initial neutron weight multiplied by the product of all the Monte Carlo weight factors occuring from the source to the given position. When observing McStas output, I is the intensity, not N.

- **3D space:** The 3D space in which the instrument is defined, usually has a single component which is placed ABSOLUTEly in space, *e.g.* at (0,0,0). All other components can be placed RELATIVE to this component.
- **Changing coordinate system:** Each component has its own local coordinate system. As the neutron travels from one component to the other, the local component coordinate system changes. The definition is that z is the direction toward the next component, and that the y direction is vertical. Our coordinate system is right-handed, making x horisontal and pointing left, looking in the direction of z.
- **Component order matters!** It is important to understand that McStas is component order dependent. The basic idea is to follow the neutron as it travels from one component to the next in the instument description. This means that if you place one component *geometrically* before another component, but *orderly* after the other component, neutrons may never reach your 'first' component. This means that some designs can be difficult to achieve, though generally a solution can be found.
- **Use Arm()'s!** The Arm() component is very good for defining changed orientation of the instrument, *e.g.* for axis turning points etc. Placing many Arm()'s will improve future flexibility of your instrument.
- **Use PSD_monitor()'s!** The PSD_monitor() component is a Position Sensitive Detector. This component can be used to image the shape of your beam as it travels through the instrument. This is very useful for debugging purposes. Other monitors, for instance wavelength monitors can also be useful.

In the McStas manual, available by clicking herehttp://mcstas.risoe.dk/documentation/manual/mcstas-1.8-manual.pdf if you are using an internet browser to view this document, description of usage of the different McStas tools is printed. The main McStas programs are

- *mcstas* - Core application.
- *mcgui* - Main graphical user interface.
- *mcdisplay* - Ray trace / debugging application.
- *mcplot* - Data / display application.
- *mcdoc* - Documentation application.

Here are a few hints on using the tools:

- To start mcgui, execute `mcgui` in a terminal window (`mcgui.pl` on Windows).
- To handle instrument files (opening, editing, compiling), use `File` menu of `mcgui`.
- To simulate and plot data, use the `Simulation` menu of `mcgui`.
- To use the distributed example McStas instruments, use the `Neutron Site` menu of `mcgui`.
- For further help on usage, use the items of the `mcgui` menu of `Help` menu or read the chapter *Running McStas* of the McStas manual [2].

5 Exercises

Throughout the rest of this paper, you will have to do the work! Through a series of small exercises, you will set up and use two simple neutron scattering instruments: a diffractometer and a triple axis spectrometer. To get an idea of what your final instrument might look like, see the sample instrument portrayed in Figure 2.

In subsection 5.7 is shown what the final exercise instrument file might look like. It is advisable to only use this file when stuck as learning by doing (yourself) is preferable to copying.

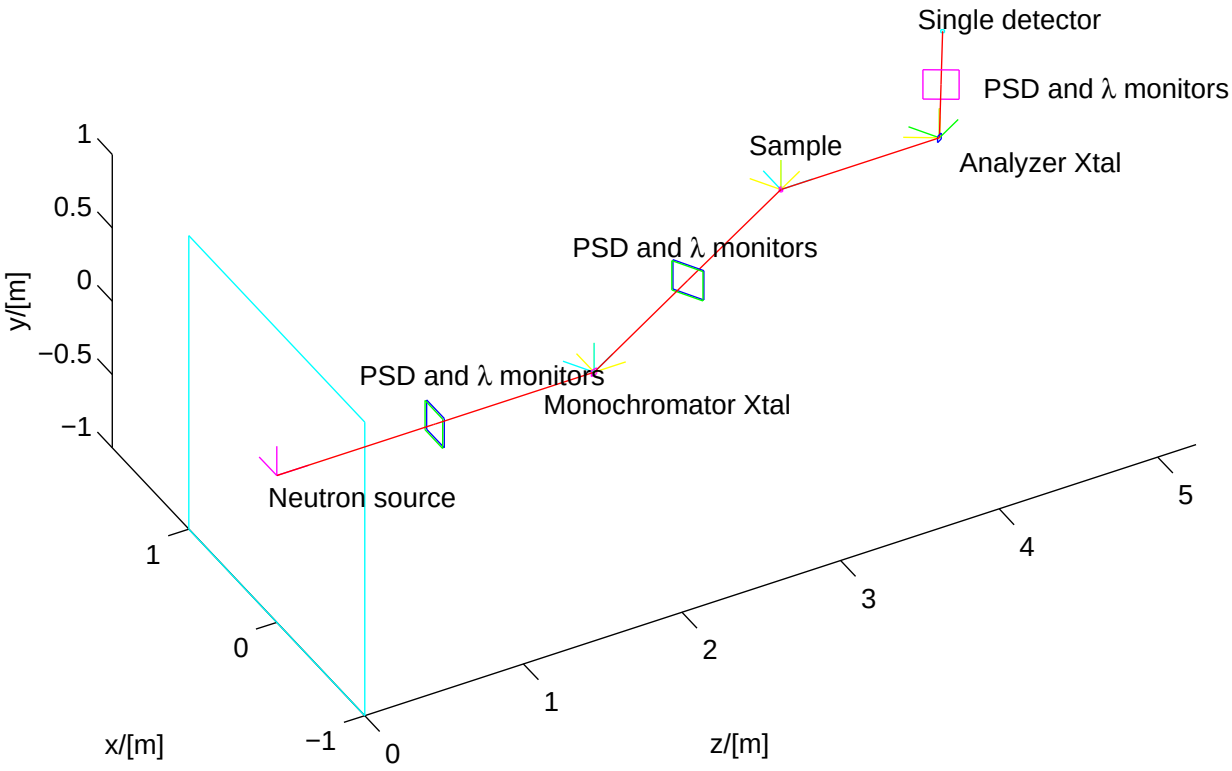

Figure 2: Illustration of a triple axis diffractometer.

5.1 Exercise: Source and PSD

1. Start the GUI (Graphical User Interface) by writing the command `mcgui` in a terminal. To open a terminal in Ubuntu, go to Applications → Accessories → Terminal or press Ctrl+Alt+t.
2. Click the Edit/New button on the GUI.
3. Insert a template instrument through the menu Insert → Instrument template or by pressing Alt+i twice. Set up an instrument, consisting only of an arm (keep the arm 'Progress_bar' that is already in the file), a source (Source_Maxwell_3) and two monitors (a PSD_monitor and an L_monitor). Insert the source at (0,0,0) relative to the origin by placing the curser at the right location (after the 'Progress_bar') in the instrument file and going to the 'Insert' menu to select the source. Likewise, place the monitors at (0,0,1) relative to the origin. As you input each component you should read the their documentation to find the needed input parameters. The component library can be accessed by clicking the Help (McDoc) button on McGui and choosing 'Component library index'.

This will open up an webpage with a list of all the components.

For the source we will help you out. Try

```
COMPONENT source = Source_Maxwell_3(
    size = 0.1, l_low = 0.1, l_high = 10, dist = 1, xw = 0.1,
    yh = 0.1, T1 = 50, T2=50, T3=50, I1=1e14, I2=0, I3=0)
AT (0, 0, 0) RELATIVE Origin
```

Read the Source_Maxwell_3 documents using McDoc to understand the suggested parameters.

4. Save your instrument file with a meaningful name and extension (such as 'TAStutorial.instr'). When the file is saved for the first time, McGui is automatically given the name of the instrument file. Run a simulation by pressing the Run button in McGui. After compiling the instrument file, McGui will open a window with questions on the simulation such as neutron count, *i.e.* how many times a neutron ray is simulated. For now, simply give the name of the directory to store you simulation run in and press the Start button. The instrument simulation will run and its output will be given in the McGui window. When the simulation finishes, plot the results by pressing the Plot button. You should now have two plots, one from the PSD monitor and one from the wavelength monitor. If either or both of the monitors have detected no neutrons, start looking for mistakes in your instrument file.

5. Narrow down the interval of wavelengths emitted from the source to *e.g.* l_low=3.999 and l_high=4.001. You do that by changing the value in the component in the instrument file either by writing the values directly or by making varible input by writing i.e l_low=Llow and putting the variable Llow parameter in the DEFINE INSTRUMENT section like this:

   ```
   DEFINE INSTRUMENT TAStutorial(Llow=0.1, Lhigh=15)
   ```

   Rerun your simulation to check the effect, you will be prompted for the value of the input parameters. Reset the wavelength interval to [0.1 15] Å.

6. Estimate the solid angle covered by your PSD. Try to understand the neutron intensity as illustrated by the plot of the registered events in the PSD in the two previous runs. Try running the simulation with half or double the number of neutron rays. Try also to vary the source focus area and to understand what you observe.

5.2 Exercise: Insert a monochromator

1. Keep your current components but focus the source on tracing only at a $2\times2\text{cm}^2$ area 2 m after the source.

   In the following we will insert a monochromator component at this position and a new set of PSD and L_monitor after the monochromator. In order to be able to rotate the monochromator it is helpful to insert two Arm() components a the rotation point at the monochromator. One of the Arm's is used to rotate the monochromator, while the other rotates the instrument components that follow.

2. Insert the two new arms at (0,0,2) relative to the Origin. Put the Monochromator_flat component (use the component library index to get the needed parameters) at (0,0,0) relative to the monochromator arm. Also, add two new input parameters of your instrument, which we will call OMM (Ω_M) and TTM ($2\theta_M$). These will define the angles of rotation at the monochromator as portrayed in Figure 3.

3. Given $\lambda = 4$Å, and knowing that for the monochromator $\kappa = 1.8734$ Å^{-1} (Pyrolytic Graphite (002) reflection), use Bragg's law to determine the correct Bragg angle (*i.e.* OMM/TTM) of the monochromator. Add the OMM and TTM input parameters to the DEFINE line at the beginning

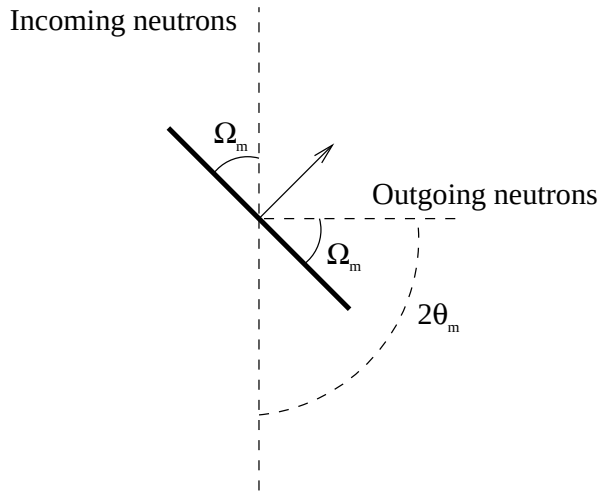

Figure 3: Illustration of the monochromator orientation.

of the instrument file and give them the defailt values you just calculated. Rotate the two arm components by OMM and TTM around the y-axis.

4. Place the two 'samplesized' monitors 1.5 m after the monochromator relative to the TTM Arm. Give the wavelength monitor 1000 channels
5. Do a scan of OMM a couple of degrees to each side of the calculated Bragg value to verify the finding, while keeping TTM fixed. This is done by replacing the fixed value of OMM in the Run Simulation window with two numbers separated by a comma, *e.g.* 20,25. These numbers represent the minimum and maximum values of OMM. A number of steps must also be given and this is done by changing the # steps value from 1 to *e.g.* 20. Check the position of the peak on the PSD and the wavelength on the L_monitor.
6. What should κ be set to to get the monochromator reflection at exactly OMM= 45° (TTM= 90°)? Adjust κ for the monochromator and verify the calculation by a scan. Check the wavelength distribution by plotting the the central scanpoint.
7. Determine the Bragg angle to scatter second order ($n = 2$) neutrons from the κ just calculated and verify it by scanning OMM. Set OMM to this value. Perform the simulation and check the wavelength distribution. Comment.
8. Before you go on, change the minimum and maximum wavelengths of the source to a suitably narrow interval around the wavelength which you record at the sample position. There is no need to produce neutron rays that will not be scattered at the monochromator.

5.3 Exercise: Insert a sample

1. Now, insert a `V_sample()` component just after the last PSD_monitor and L_monitor and a Beamstop component 0.2 m after the sample. The `V_sample()` component simulates a Vanadium sample. Such a sample scatters incoherently, *i.e.* in all directions. At the same position as the sample, insert a PSD_monitor_4PI component of radius 0.5 m. Read the documentation for details on input parameters. Run a simulation. Notice the number of hits.

5.4 Exercise: Insert a sample with two powder lines

1. Next, let us insert something more interesting. Remove the V_sample and PSD_monitor_4PI and insert a PowderN component with the following parameters:

```
COMPONENT sample = PowderN(radius=0.01,h=0.01, d_phi=0.1, pack=0.5,
   DW=0.9, frac=0.5, reflections="mylist.dat",
   Vc=3.86*3.86*11.82, sigma_abs=0, sigma_inc=2, barns=1)
  AT (0, 0, 1.5) RELATIVE arm2
```

   You should also create a list of reflections and save it as `mylist.dat`. The list should have the following contents:

```
# column_j 3 multiplicity 'j'
# column_q 1  Scattering vector modulus [Angs^-1]
# column_F2 2 Scattering factor |F^2| in [barns]
2 1000 8
2.6 1000 4
```

2. Insert a banana shaped detector:

```
COMPONENT BananaDetector = Monitor_nD(
   xwidth=1.5, yheight = 0.09,
   options="banana, theta limits [-55 -35] bins=360, file =detector.dat")
  AT (0,0,0) RELATIVE sample
```

   and make sure that the previously inserted beamstop is before the banana detector in the component list. Test that Bragg peak neutrons reach the detector by running a simulation.

   Afterwards, choose trace instead of simulate in the Run simulation window and see the instrument, you have created. To see only the trace of the neutrons, which hit the detector, select the detector component in the list given in the Inspect component box. Also, insert the following after the sample component:

```
EXTEND %{
if (!SCATTERED) ABSORB;
%}
```

   This will insert a command in c-language into the compiled instrument file. The command says that if the simulated neutron does not scatter at the sample, it is absorbed. This removes the direct beam and should be used with caution. You can now remove the Beamstop component. You can also insert the same statement after the monochromator.

5.5 Exercise: Insert a real sample

1. Instead of `mylist.dat` use `Na2Ca3Al2F14.laz`. Change the limits of the banana detector to [-130 -10] and change the sample component to read:

```
COMPONENT sample = PowderN(
    reflections = "Na2Ca3Al2F14.laz", d_phi = 0.1, radius = 0.004,
    h = 0.03, DW = 0.9, barns = 1, pack = 0.7, frac = 0, tfrac=0)
  AT (0, 0, 1.5) RELATIVE arm2
```

   Make a simulation to check that a nice powder pattern is observed. If it is time for a coffee break, close the plot from the quick run and instead do a long simulation, *e.g.* 20 million neutron rays, and have coffee.

2. If you don't have time for coffee, you can enhance the statistics by using SPLIT at the sample component since there are many MC choices there:

```
SPLIT 10 COMPONENT sample = PowderN(
    reflections = "Na2Ca3Al2F14.laz", d_phi = 0.1, radius = 0.004,
    h = 0.03, DW = 0.9, barns = 1, pack = 0.7, frac = 0, tfrac=0)
  AT (0, 0, 1.5) RELATIVE arm2
```

   Comment on the pattern as function of scattering angle!

5.6 Exercise: Insert an analyzer

1. Comment out the banana detector and beamstop with /* and */ before and after the components, respectively. We are in stead going to put a single detector of an appropriate size at the perimeter location of the bananadetector.
2. Add an arm at the sample and an angle to rotate the part of the instrument located after the sample, *e.g.* TT (Two Θ). Put a single detector (use e.g. a PSD) at a location corresponding to the radius of the bananadetector but relative to the TT arm. Look at your results from the last simulation to determine an approximate scan range for the sample scattering angle (TT), *e.g.* -25,-35, which will make a scan through some powderlines.
3. Between sample and detector, set up an analyser crystal by copying and modifying your monochromator component. Add new arms and angles: OMA and TTA; A is for Analyzer. Adjust the analyser to Bragg condition for the chosen wavelength and make sure that the analyser crystal rotates accordingly to keep the Bragg condition. Re-scan TT and notice the difference to the scan performed in the previous task. Try also scanning around -TT and notice the difference to the other scan. Can you explain the difference? Contrary to a real experiment you can take a look at the energy which is hitting the detector - try that by using an appropriate monitor.
4. Keeping what you learned at the other lectures in mind, how could you improve the q-resolution? Think about wavelength and divergence of the beam and play with your instrument to make it better.

5.7 Example instrument file

```
/*******************************************************************************
*         McStas instrument definition URL=http://www.mcstas.org
*
* Instrument: test (rename also the example and DEFINE lines below)
*
* %Identification
* Written by: Your name (email)
* Date: Current Date
* Origin: Your institution
* Release: McStas CVS-080208
* Version: 0.2
* %INSTRUMENT_SITE: Institution_name_as_a_single word
*
* Instrument short description
*
* %Description
* Instrument longer description (type, elements, usage...)
```

```
*
* Example: mcrun test.instr <parameters=values>
*
* %Parameters
* Par1: [unit] Parameter1 description
*
* %Link
* A reference/HTML link for more information
*
* %End
*******************************************************************************/

/* Change name of instrument and input parameters with default values */
DEFINE INSTRUMENT test(OMM=36.607, TTM=73.214, TT=-66.94, OMA=36.607, TTA=73.214)

/* The DECLARE section allows us to declare variables or  small      */
/* functions in C syntax. These may be used in the whole instrument. */
DECLARE
%{
%}

/* The INITIALIZE section is executed when the simulation starts     */
/* (C code). You may use them as component parameter values.         */
INITIALIZE
%{
%}

/* Here comes the TRACE section, where the actual      */
/* instrument is defined as a sequence of components.  */
TRACE

/* The Arm() class component defines reference points and orientations  */
/* in 3D space. Every component instance must have a unique name. Here, */
/* Origin is used. This Arm() component is set to define the origin of  */
/* our global coordinate system (AT (0,0,0) ABSOLUTE). It may be used   */
/* for further RELATIVE reference, Other useful keywords are : ROTATED  */
/* EXTEND GROUP PREVIOUS. Also think about adding a neutron source !    */
/* Progress_bar is an Arm displaying simulation progress.               */
COMPONENT Origin = Progress_bar()
  AT (0,0,0) ABSOLUTE

COMPONENT Source_Maxwell_3 = Source_Maxwell_3(
    size = 0.05, l_low = 3.9, l_high = 4.1, dist = 10, xw = 0.01,
    yh = 0.01, T1 = 150.42, T2 = 38.74, T3 = 14.84, I1 = 3.67E11, I2 = 3.64E11,
    I3 = 0.95E11)
  AT (0, 0, 0) RELATIVE Origin

COMPONENT L_monitor = L_monitor(
    filename = "test2.psd", xmin = -0.1, xmax = 0.1, ymin = -0.1,
    ymax = 0.1, Lmin = 2.1, Lmax = 6)
```

```
  AT (0, 0, 1) RELATIVE Origin

COMPONENT PSD_monitor = PSD_monitor(
    nx = 90, ny = 90, filename = "test.psd", xmin = -0.1,
    xmax = 0.1, ymin = -0.1, ymax = 0.1)
  AT (0, 0, 1) RELATIVE Origin

COMPONENT arm1 = Arm()
  AT (0, 0, 2) RELATIVE Origin
  ROTATED (0, OMM, 0) RELATIVE Origin

SPLIT 10 COMPONENT Monochromator_flat = Monochromator_flat(
    zmin = -0.1, zmax = 0.1, ymin = -0.1, ymax = 0.1,
    mosaich = 30, mosaicv = 30)
  AT (0, 0, 0) RELATIVE arm1
EXTEND %{
if (!SCATTERED) ABSORB;
%}

COMPONENT arm2 = Arm()
  AT (0, 0, 2) RELATIVE Origin
  ROTATED (0, TTM, 0) RELATIVE Origin

COMPONENT L_monitor2 = L_monitor(
    filename = "test3.psd", xmin = -0.1, xmax = 0.1, ymin = -0.1,
    ymax = 0.1, Lmin = 2.1, Lmax = 6)
  AT (0, 0, 1) RELATIVE arm2

COMPONENT PSD_monitor2 = PSD_monitor(
    nx = 90, ny = 90, filename = "test4.psd", xmin = -0.1,
    xmax = 0.1, ymin = -0.1, ymax = 0.1)
  AT (0, 0, 1.01) RELATIVE arm2

SPLIT 10
COMPONENT sample = PowderN(
    reflections = "Na2Ca3Al2F14.laz", d_phi = 0.1, radius = 0.004,
    h = 0.03, DW = 0.9, barns = 1, pack = 0.7, frac = 0, tfrac=0)
  AT (0, 0, 2) RELATIVE arm2

/*COMPONENT sample = PowderN(
    reflections = "mylist.dat", d_phi = 0.1, radius = 0.01,
    Vc = 3.86*3.86*11.82, sigma_abs = 0, sigma_inc = 2, h = 0.01,
    DW = 0.9, barns = 1, pack = 0.5, frac = 0.5)
  AT (0, 0, 2) RELATIVE arm2*/
EXTEND %{
if (!SCATTERED) ABSORB;
%}

COMPONENT arm3 = Arm()
  AT (0, 0, 0) RELATIVE sample
```

```
  ROTATED (0, TT, 0) RELATIVE sample

/*COMPONENT BananaDetector = Monitor_nD(
    xwidth = 2, yheight=0.09,
    options="banana, theta limits [-130 -10] bins=360, file =detector.dat")
  AT (0, 0, 0) RELATIVE sample

COMPONENT STOP2 = Beamstop(radius=0.3)
AT (0,0,3.5) RELATIVE arm2*/

COMPONENT PSD_monitor3 = PSD_monitor(
    nx = 90, ny = 90, filename = "test5.psd", xmin = -0.132,
    xmax = 0.132, ymin = -0.02, ymax = 0.02)
  AT (0, 0, 0.5) RELATIVE arm3

COMPONENT arm4 = Arm()
  AT (0, 0, 1) RELATIVE arm3
  ROTATED (0, OMA, 0) RELATIVE arm3

SPLIT 10 COMPONENT AnalyzerCrystal = Monochromator_flat(
    zmin = -0.1, zmax = 0.1, ymin = -0.1, ymax = 0.1,
    mosaich = 30, mosaicv = 30)
  AT (0, 0, 0) RELATIVE arm4

COMPONENT arm5 = Arm()
  AT (0, 0, 0) RELATIVE AnalyzerCrystal
  ROTATED (0, TTA, 0) RELATIVE arm3

COMPONENT Analyzer = Monitor_nD(
    xwidth = 0.01, yheight=0.1,
    options="single, file = analyzer.dat")
  AT (0, 0, 1) RELATIVE arm5



/* This section is executed when the simulation ends (C code). Other    */
/* optional sections are : SAVE                                         */
FINALLY
%{
%}
/* The END token marks the instrument definition end */
END
```

6 Suffix

Well done, you have come to the end of the McStas tutorial. Hopefully, most of the goals of the tutorials have been fulfilled. Otherwise, feel free to contact the authorsmailto:peter.willendrup@risoe.dk,kim.lefmann@risoe.dk of this paper or the McStas users mailinglistmailto:mcstas-users@mcstas.org for further help.

References

[1] K. Lefmann and K. Nielsen: *McStas, a general software package for neutron ray-tracing simulations*, Neutron News, **10** pp. 20-23, 1999

[2] P. Willendrup, E. Farhi K. Lefmann et. al.: *User and Programmers Guide to the Neutron Ray-Tracing Package McStas, Version 1.11*, Risø National Laboratory, Roskilde, Denmark, January 2007

[3] P. Willendrup, E. Farhi K. Lefmann et. al.: *Component Manual for the Neutron Ray-Tracing Package McStas, Version 1.11*, Risø National Laboratory, Roskilde, Denmark, January 2007

[4] McStas homepage: http://www.mcstas.org