McStas

Simulating Polarized Neutron Scattering Experiments
and Equipment with McStas

Erik Bergbäck Knudsen, DTU Physics

McStas?

Monte **C**arlo **S**imulations of **T**riple **A**xis **S**pectrometers

Started at Risø, Denmark in 1998

Portable code (Unix/Linux/Mac/Windows)





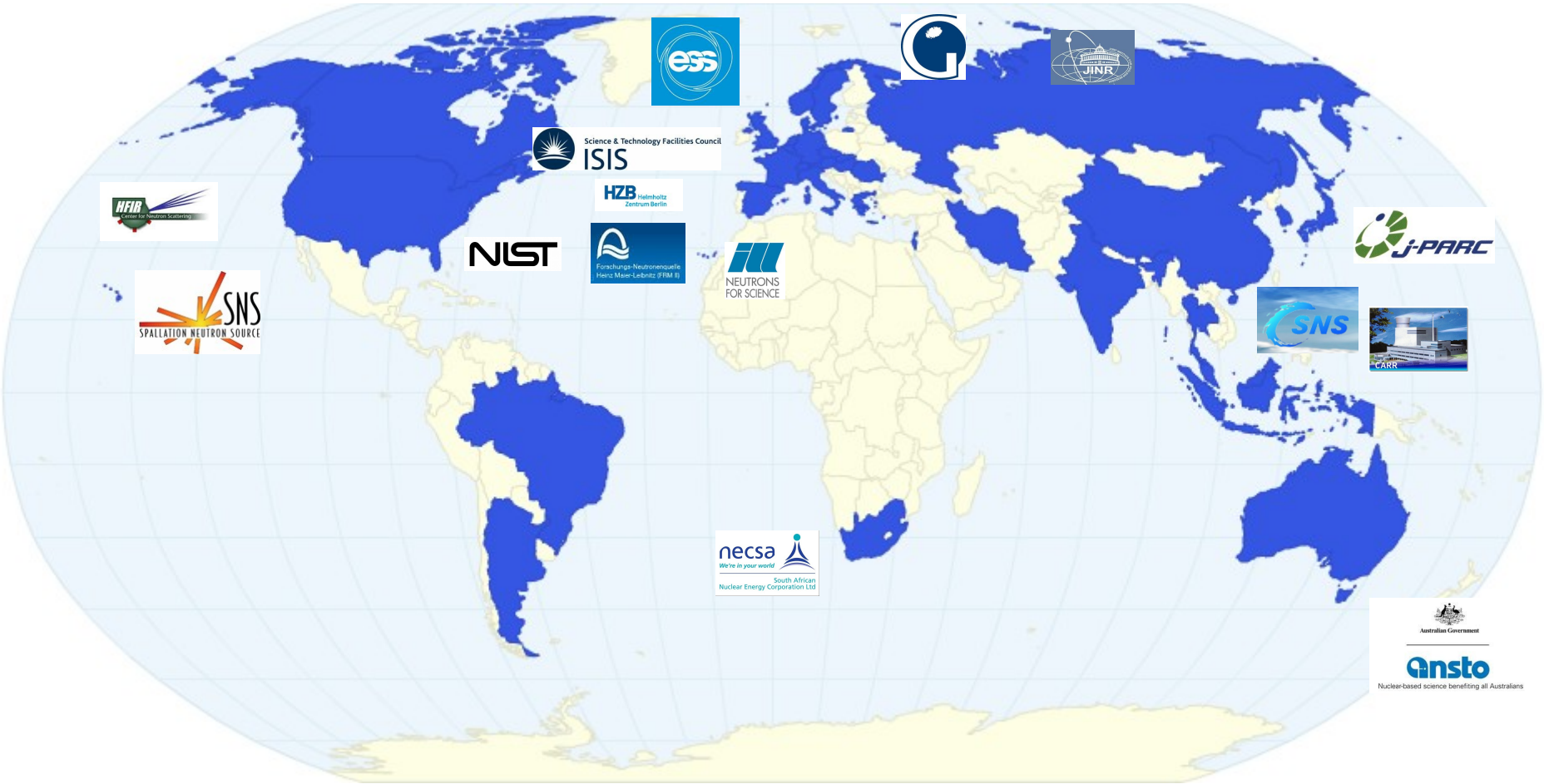
Ran on everything from iPhone to 1000+ node cluster!

Open Source under GPL!

Permanent staff maintaining the code @ DTU & ILL.

Domain-specific-language (DSL) based on compiler technology (LeX+Yacc)

Where?


ess
JINR
Science & Technology Facilities Council
ISIS
HZB Helmholtz Zentrum Berlin
HFIR
NIST
Forschungs-Neutronenquelle Heinz Maier-Leibnitz (FRM II)
NEUTRONS FOR SCIENCE
j-PARC
SNS
SPALLATION NEUTRON SOURCE
SNS
CARR
necsa
We're in your world
South African Nuclear Energy Corporation Ltd
Australian Government
ansto
Nuclear-based science benefiting all Australians

GitHub

McStasMcXtrace/McCode - Mozilla Firefox

GitHub, Inc. (US) | https://github.com/McStasMcXtrace/McCode

This repository | Search | Pull requests | Issues | Marketplace | Gist

McStasMcXtrace / McCode

Watch 7 | Star 11 | Fork 19

Code | Issues 58 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

http://www.mccode.org

Edit

c | neutron | x-ray | scientific-computing | simulation | raytracing | Manage topics

6,962 commits | 15 branches | 6 releases | 14 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

ebknudsen Merge branch 'master' of github.com:McStasMcXtrace/McCode — Latest commit 8cedd4e 6 days ago

cmake	A couple of @MCCODE_PREFIX@ needed	12 days ago
common	descriptory files that are now in the comps-package	27 days ago
doc	Update manual for 2.4.1	12 days ago
docpkg/manuals	Update manual for 2.4.1	12 days ago
mcstas-comps	Remove 1.12c style parameter line	10 days ago
mcstas	Increase symbol table length to accomodate MACS instrument from Mads ...	3 months ago
mcxtrace-comps	this had apprently not been committed	13 days ago
mcxtrace	Make the Component Blah AT (x,y,z) location status in trace mode incl...	4 months ago
meta-pkgs	We only build 32 bit...	15 days ago
obsolete-files	Removing Yosemite scripts as the build server is being migrated to DM...	6 months ago

Website

McStas homepage - Mozilla Firefox

McStas homepage

www.mcstas.org

Most Visited | Getting Started | Portscout Port Up... | Getting Started | Bright: New beamlin... | Astro-Update Hea... | Portscout Port Up...

McStas - A neutron ray-trace simulation package

McStas

About McStas
- Conditions of use
- Authors/Contacts
- Project funding
- Screenshots

Download
- Components
- Linux Install (deb/rpm)
- Mac OS X Install
- Unix Install (src code)
- Windows Install
- Other Downloads (share)

Mailing list

Search web/mailinglist

Documentation
- McStas manual
- FAQ
- Known problems
- Publications
- C Compilers
- Other
- Tools
- Tutorial

Workshops/conferences

Developments

Links

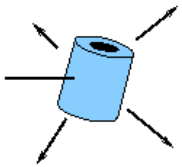
Report bugs

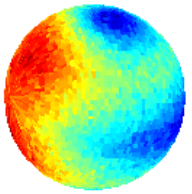
Git

McStas Ubuntu live-dvd

McStas - A neutron ray-trace simulation package

McStas is a general tool for simulating neutron scattering instruments and experiments. It is actively supported by DTU Physics, NBI KU, ESS, PSI and ILL.





Simulated scattering from a hollow-cylinder vanadium sample.

The plot shows the intensity of scattered neutrons (red is highest intensity). The sample is at the center of the sphere with the neutron beam coming from the left. Clearly seen is the shadowing effect of the sample causing a lower intensity opposite the beam. Also seen is the effect of the non-symmetric geometry of the sample, causing lower intensity directly above and to the side of the sample.

Recent news

June 26th, 2017: Update-release 2.4.1 available

Dear all,

McStas 2.4.1 has been released and is ready for download via http://downloads.mcstas.org/mcstas-2.4.1

Release changes are listed below, and the full list of project changes is also available at http://mcstas.org/CHANGES_McStas.

Greetings from the McStas team - hope you will enjoy this new release! :-)

Peter Willendrup

Changes in McStas v.2.4.1, June 26th, 2017

McStas 2.4.1 is the sixth release in the 2.x series and fixes various issues with McStas 2.4, plus provides a small set of new developments.

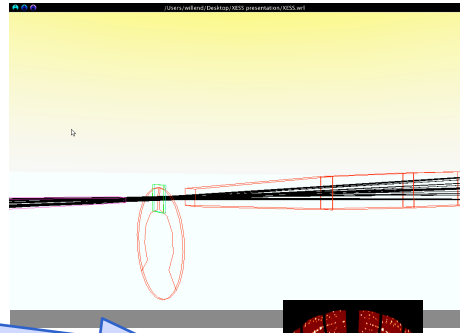
Thanks:

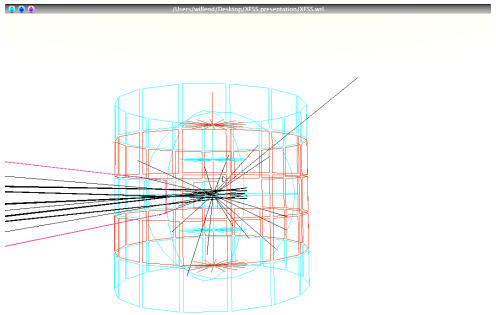
- Thanks to all contributors of components, instruments etc.! This is what Open Source and McStas is all about!

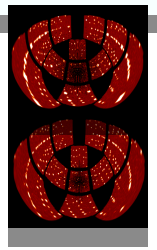
Fixes of issues from last release:

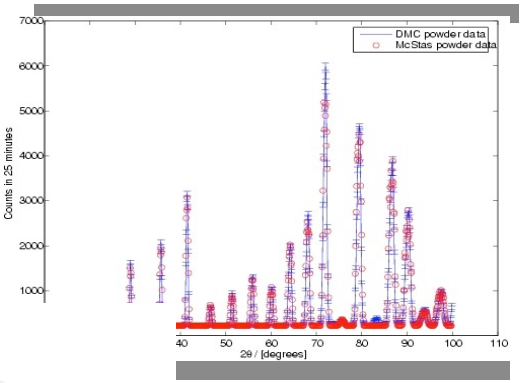
Use cases

Instrumentation
Virtual experiments
Data analysis
Teaching
eLearning-platforms

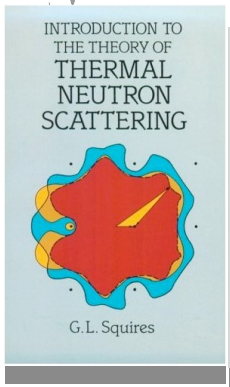


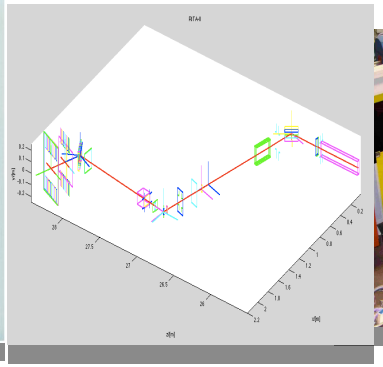


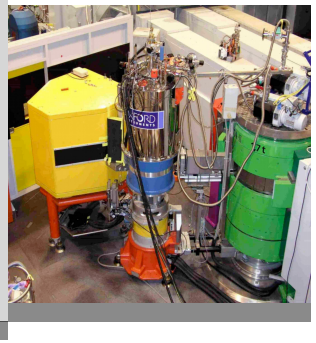


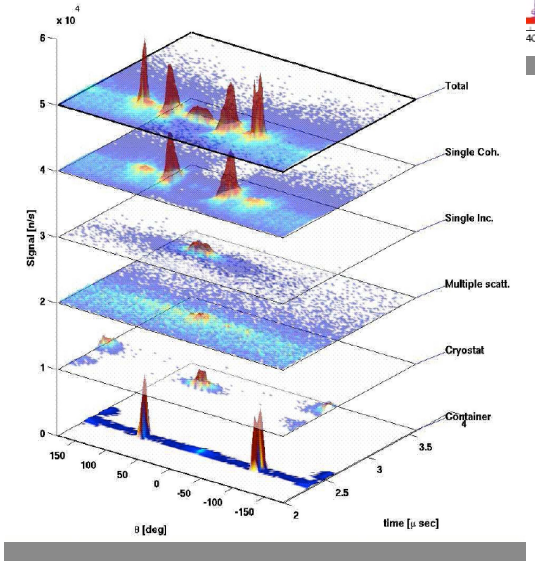


(DTU, KU schools & workshops)



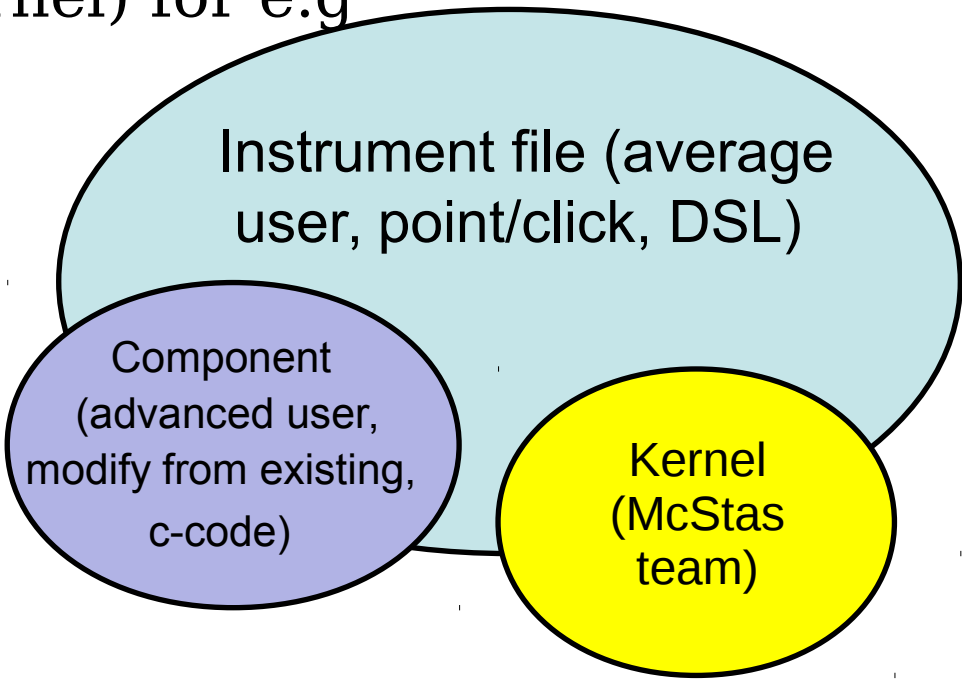




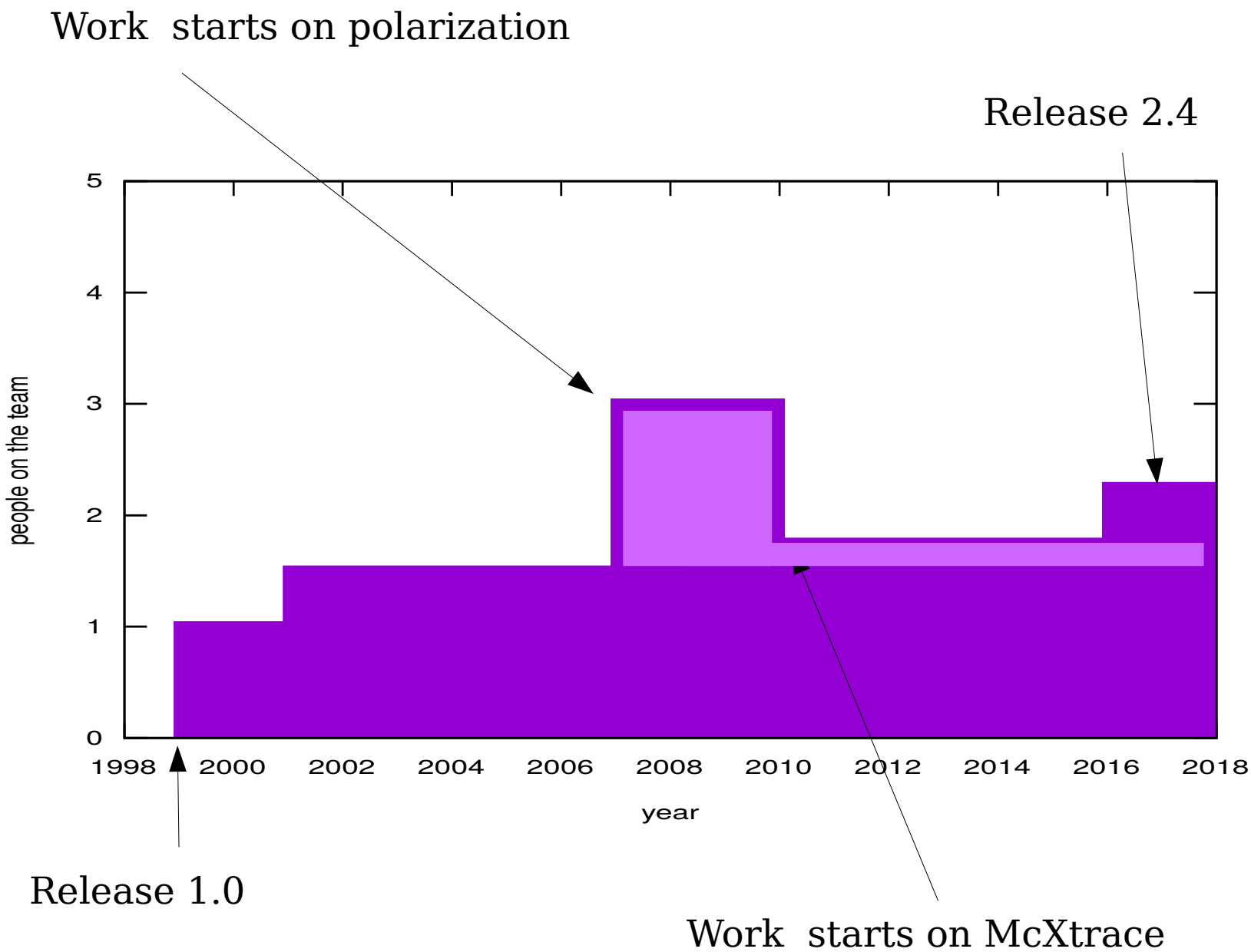


McStas Introduction

- 'Component' files (~150) inserted from library
  - Sources
  - Optics
  - Samples
  - Monitors
  - If needed, write your own comps
- DSL + ISO-C code gen.
- Library of common functions (Kernel) for e.g
  - I/O
  - Random numbers
  - Physical constants
  - Propagation
  - Precession in fields
  - ...



Workforce


Work starts on polarization
Release 2.4
people on the team
0
1
2
3
4
5
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
year
Release 1.0
Work starts on McXtrace

X-ray sister

McStas

McXtrace homepage - Mozilla Firefox

McXtrace homepage

mcxtrace.org

Most Visited | Getting Started | Portscout Port Up... | Getting Started | Bright: New beamlin... | Astro-Update Hea... | Portscout Port Up...

McXtrace

McXtrace - An X-ray ray-trace simulation package

ESRF SAXSLAB GANESHA DTU

McXtrace

Project Status
Project Partners
Project People
Goal

Installation
Mac OSX Installation
Windows Installation

Mailing List

Links

Publications

Minutes of Meetings

McXtrace Art

About McXtrace

Documentation
Commands
Tutorial

Download
Components

Search

McXtrace - Monte Carlo Xray Tracing, is a joint venture by

DTU ESRF SAXSLAB GANESHA

Funding from NABIIT, DSF and the above parties.

Our code is based on technology from McStas.

Code repository (shared with 'McStas') is located at github.com/McStasMcXtrace

For information on our progress, please subscribe to our user mailinglist..

- To download the latest release: download area
- For installation instructions : installation
- For a quick list of the available commands: commands

McXtrace News

The latest McXtrace news.

McXtrace 1.4 is released!

McStas Procedure

1. Describe your instrument in the McStas language (In a text file).
2. Automatically convert beamline into ANSI c
3. Compile
4. Run

1. Optimized for your platform
2. Only includes what you use

Instrument file

```
DEFINE INSTRUMENT simple()

TRACE

COMPONENT origin = Progress_bar()
AT (0, 0, 0) RELATIVE ABSOLUTE

// insert components here (e.g. Insert -> Source -> ...)
COMPONENT source_simple = Source_simple(
    yheight=0.1, xwidth=0.1, focus_xw=0.1, focus_yh=0.1,dist=5000,
    lambda0=5, dlambda=1,gauss=1)
AT (0, 0, 0) RELATIVE PREVIOUS

COMPONENT lmonitor = L_monitor(
    yheight=0.1, xwidth=0.1,
    Lmin=1, Lmax=20, filename="lmonitor")
AT(0,0,1e-6) RELATIVE PREVIOUS

COMPONENT emonitor = E_monitor(
    yheight=0.1, xwidth=0.1,
    Emin=0, Emax=10, filename="emonitor")
AT(0,0,1e-6) RELATIVE PREVIOUS

COMPONENT psdmonitor = PSD_monitor(
    yheight=0.1, xwidth=0.1, filename="psdmonitor")
AT(0,0,1e-6) RELATIVE PREVIOUS

END
```

Written by you!

Component file

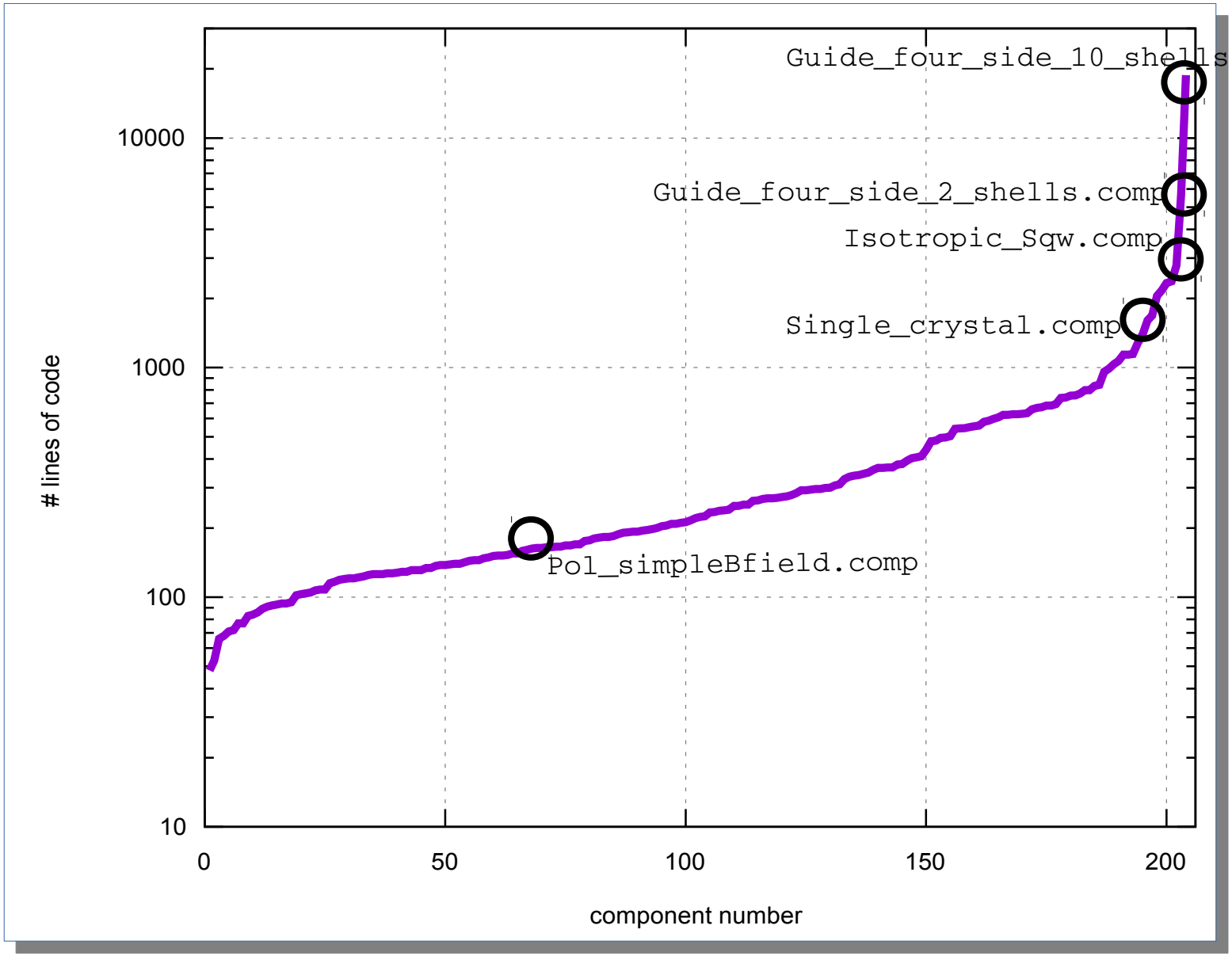
```
/*******************************************************************************
*
* Mcstas, neutron ray-tracing package
*         Copyright 1997-2002, All rights reserved
*         Risoe National Laboratory, Roskilde, Denmark
*         Institut Laue Langevin, Grenoble, France
*
* Component: Source_flat
*
* %I
* Written by: Kim Lefmann
* Date: October 30, 1997
* Modified by: KL, October 4, 2001
* Modified by: Emmanuel Farhi, October 30, 2001. Serious bug corrected.
* Version: $Revision: 1.22 $
* Origin: Risoe
* Release: McStas 1.6
*
* A circular neutron source with flat energy spectrum and arbitrary flux
*
* %D
* The routine is a circular neutron source, which aims at a square target
* centered at the beam (in order to improve MC-acceptance rate).  The angular
* divergence is then given by the dimensions of the target.
* The neutron energy is uniformly distributed between E0-dE and E0+dE.
*
* Example: Source_flat(radius=0.1, dist=2, xw=.1, yh=.1, E0=14, dE=2)
*
* %P
* radius: (m)   Radius of circle in (x,y,0) plane where neutrons
*               are generated.
* dist:   (m)   Distance to target along z axis.
* xw:     (m)   Width(x) of target
* yh:     (m)   Height(y) of target
* E0:     (meV) Mean energy of neutrons.
* dE:     (meV) Energy spread of neutrons.
* Lambda0 (AA)  Mean wavelength of neutrons.
* dLambda (AA)  Wavelength spread of neutrons.
* flux    (1/(s*cm**2*st)) Energy integrated flux
*
* %E
*******************************************************************************/

DEFINE COMPONENT Source_simple
DEFINITION PARAMETERS ()
SETTING PARAMETERS (radius, dist, xw, yh, E0=0, dE=0, Lambda0=0, dLambda=0, flux=1)
OUTPUT PARAMETERS ()
STATE PARAMETERS (x,y,z,vx,vy,vz,t,s1,s2,p)
DECLARE
%{
  double pmul, pdir;
%}
INITIALIZE
%{
  pmul=flux*PI*1e4*radius*radius/mcget_ncount();
%}
```

```
TRACE
%{
 double chi,E,Lambda,v,r, xf, yf, rf, dx, dy;

 t=0;
 z=0;

 chi=2*PI*rand01();                          /* Choose point on source */
 r=sqrt(rand01())*radius;                    /* with uniform distribution. */
 x=r*cos(chi);
 y=r*sin(chi);

 randvec_target_rect(&xf, &yf, &rf, &pdir,
        0, 0, dist, xw, yh, ROT_A_CURRENT_COMP);

 dx = xf-x;
 dy = yf-y;
 rf = sqrt(dx*dx+dy*dy+dist*dist);

 p = pdir*pmul;

 if(Lambda0==0) {
   E=E0+dE*randpm1();              /*  Choose from uniform distribution */
   v=sqrt(E)*SE2V;
 } else {
   Lambda=Lambda0+dLambda*randpm1();
   v = K2V*(2*PI/Lambda);
 }

 vz=v*dist/rf;
 vy=v*dy/rf;
 vx=v*dx/rf;
%}

MCDISPLAY
%{
  magnify("xy");
  circle("xy",0,0,0,radius);
%}

END
```

Written by advanced users/developers

Components

Write a new component or update an existing one – It's really not that big a task.



Generated c-code

```
/* Automatically generated file. Do not edit.
 * Format:     ANSI C source code
 * Creator:    McStas <http://neutron.risoe.dk>
 * Instrument: My_Instrument.instr (My_Instrument)
 * Date:       Sat Apr  9 15:27:56 2005
 */


/* THOUSANDS of lines removed here.... */

  /* TRACE Component Source. */
  SIG_MESSAGE("Source (Trace)");
  mcDEBUG_COMP("Source")
  mccoordschange(mcposrSource, mcrotrSource,
    &mcnlx, &mcnly, &mcnlz,
    &mcnlvx, &mcnlvy, &mcnlvz,
    &mcnlt, &mcnlsx, &mcnlsy);
  mcDEBUG_STATE(mcnlx, mcnly, mcnlz, mcnlvx, mcnlvy, mcnlvz,mcnlt,mcnlsx,mcnlsy, mcnlp)
#define x mcnlx
#define y mcnly
#define z mcnlz
#define vx mcnlvx
#define vy mcnlvy
#define vz mcnlvz
#define t mcnlt
#define s1 mcnlsx
#define s2 mcnlsy
#define p mcnlp
  STORE_NEUTRON(2,mcnlx, mcnly, mcnlz, mcnlvx,mcnlvy,mcnlvz,mcnlt,mcnlsx,mcnlsy,
  mcScattered=0;
  mcNCounter[2]++;
#define mccompcurname Source
#define mccompcurindex 2
{   /* Declarations of SETTING parameters. */
MCNUM radius = mccSource_radius;
MCNUM dist = mccSource_dist;
MCNUM xw = mccSource_xw;
MCNUM yh = mccSource_yh;
MCNUM E0 = mccSource_E0;
MCNUM dE = mccSource_dE;
MCNUM Lambda0 = mccSource_Lambda0;
MCNUM dLambda = mccSource_dLambda;
MCNUM flux = mccSource_flux;
#line 58 "Source_simple.comp"
{
 double chi,E,Lambda,v,r, xf, yf, rf, dx, dy;

 t=0;
 z=0;

 chi=2*PI*rand01();                          /* Choose point on source */
 r=sqrt(rand01())*radius;                    /* with uniform distribution. */
 x=r*cos(chi);
 y=r*sin(chi);

 randvec_target_rect(&xf, &yf, &rf, &pdir,
        0, 0, dist, xw, yh, ROT_A_CURRENT_COMP);
```

Written by McStas!

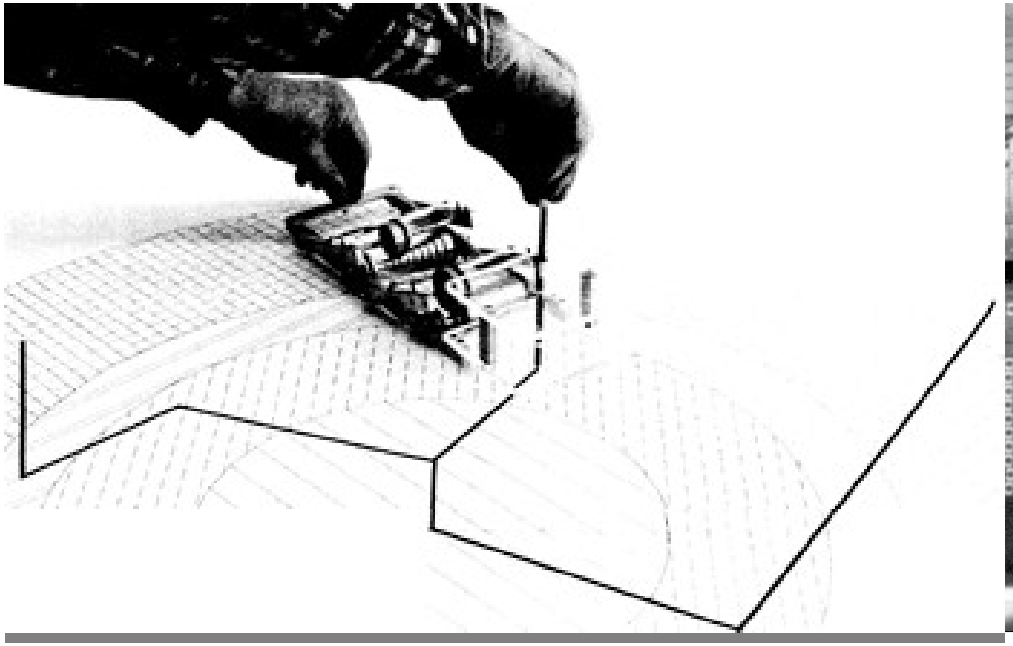
Monte Carlo Technique

- During WW2, “numerical experiments” were applied at Los Alamos for solving mathematical complications of computing fission, criticality, neutronics, hydrodynamics, thermonuclear detonation etc.

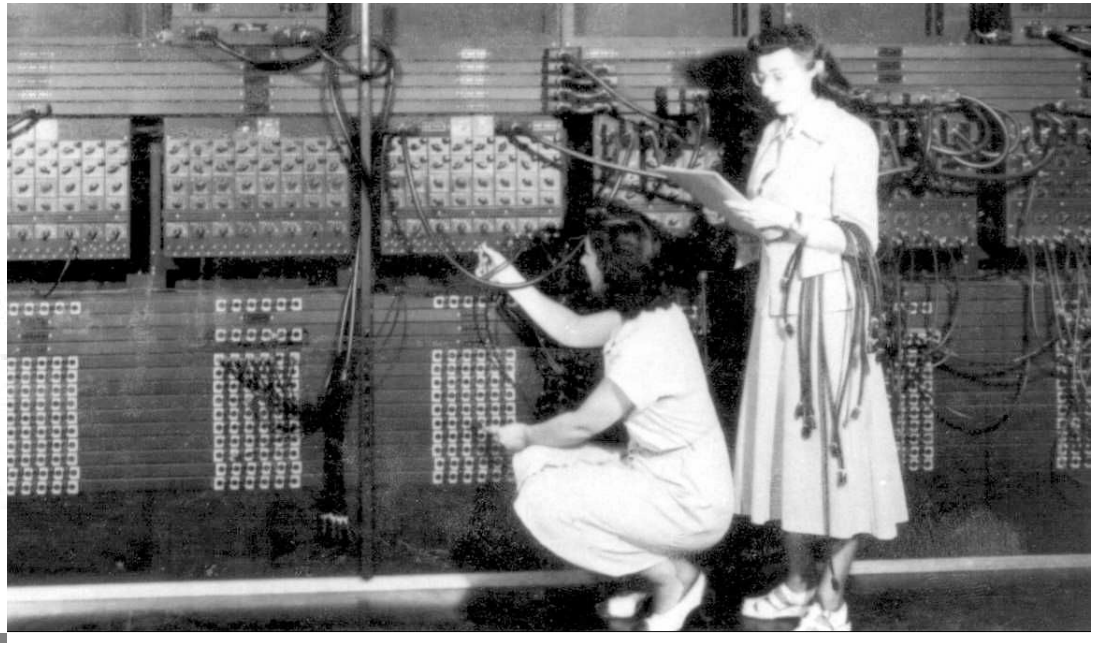


- Notable fathers: John v. Neumann, Stanislav Ulam, Nicholas Metropolis
- Named “Monte Carlo” after Ulam’s fathers frequent visits to the Monte Carlo casino in Las Vegas
- Initially “implemented” by letting large numbers of women use tabularized random numbers and hand calculators for individual particle calculations
- Later, analogue and digital computing devices were used

Early Monte Carlo



FERMIAC

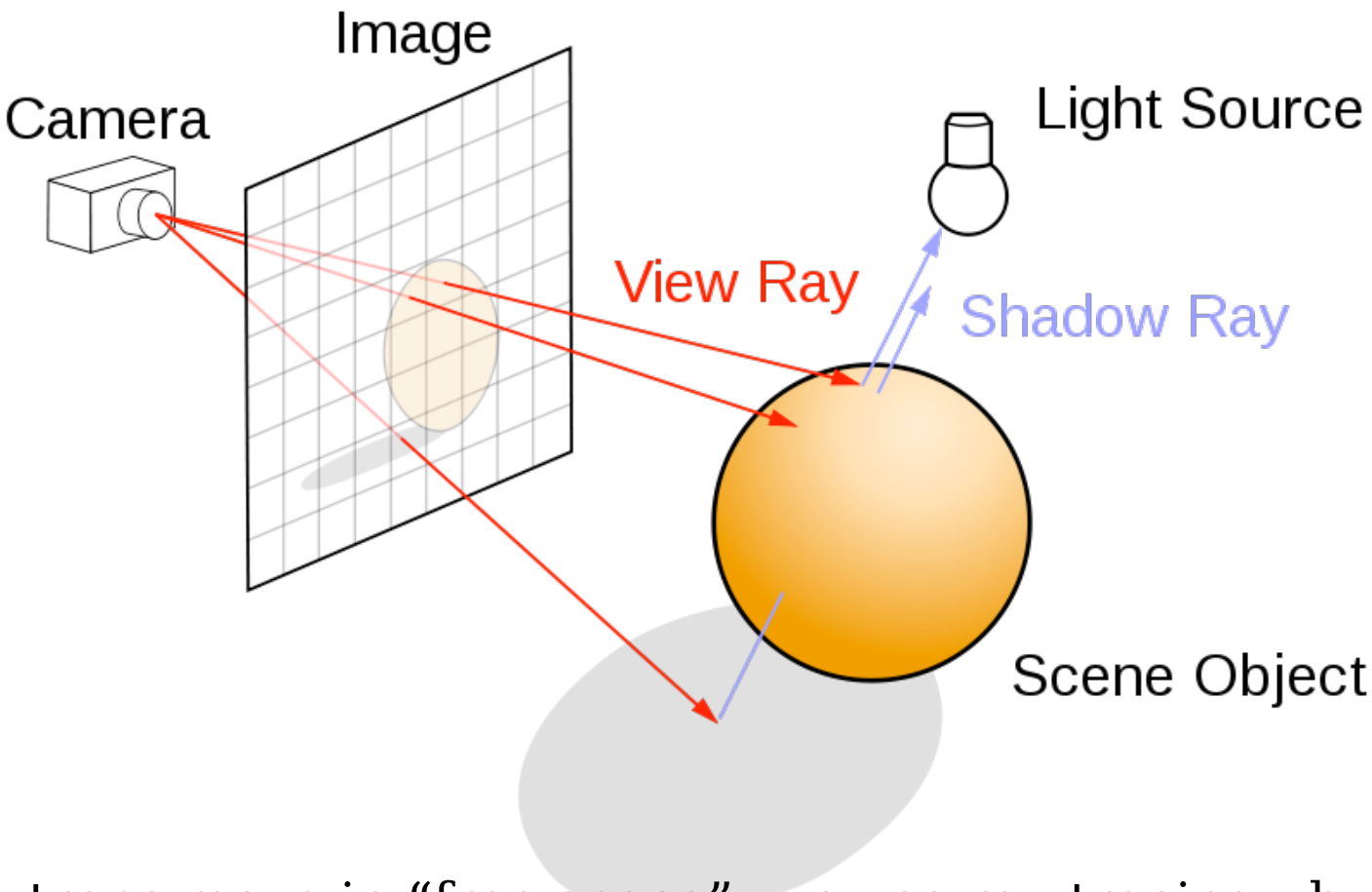


ENIAC

Monte Carlo III

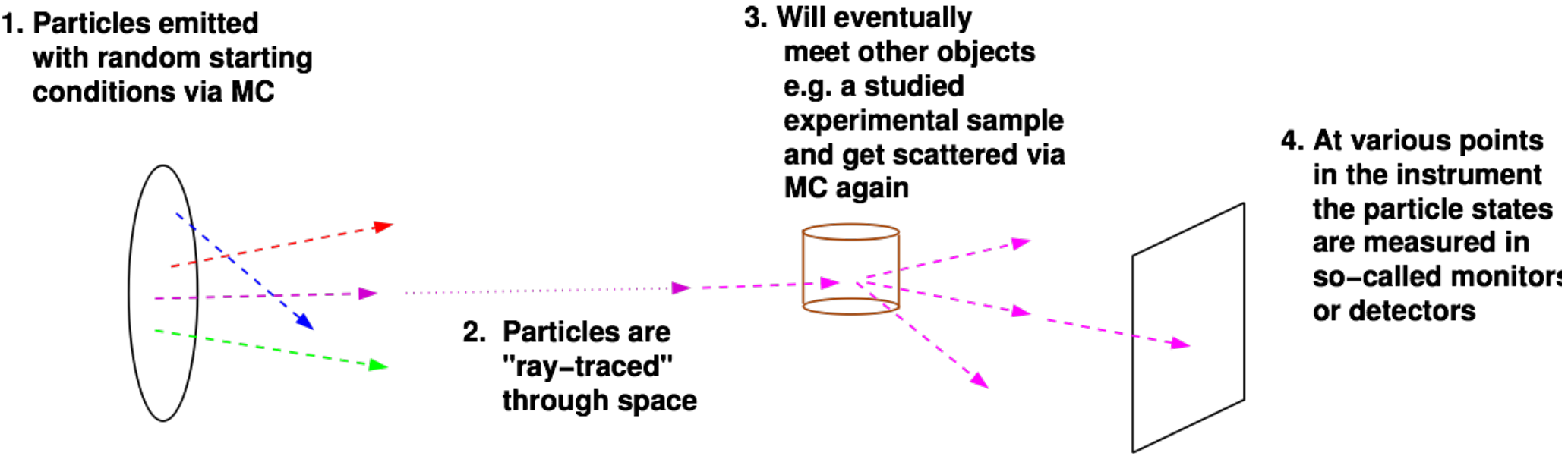
- Los Alamos has since then developed and perfected many different monte carlo codes leading to what is today known as the codes MCNP5 and MCNPX
- State of the art is MCNPX (or soon the merged MCNP6 code) that features numerous particles
- MCNP was originally Monte Carlo Neutron Photon, later N-Particle
- Mainly used for high-energy particle descriptions in weapons, power reactors and routinely used for estimating dose rates and needed shielding
- Other similar types of codes are FLUKA, PHITS, and to some extent Geant4
- In general, these codes do not handle coherent scattering of neutrons due to the focus on high energies

Ray tracing



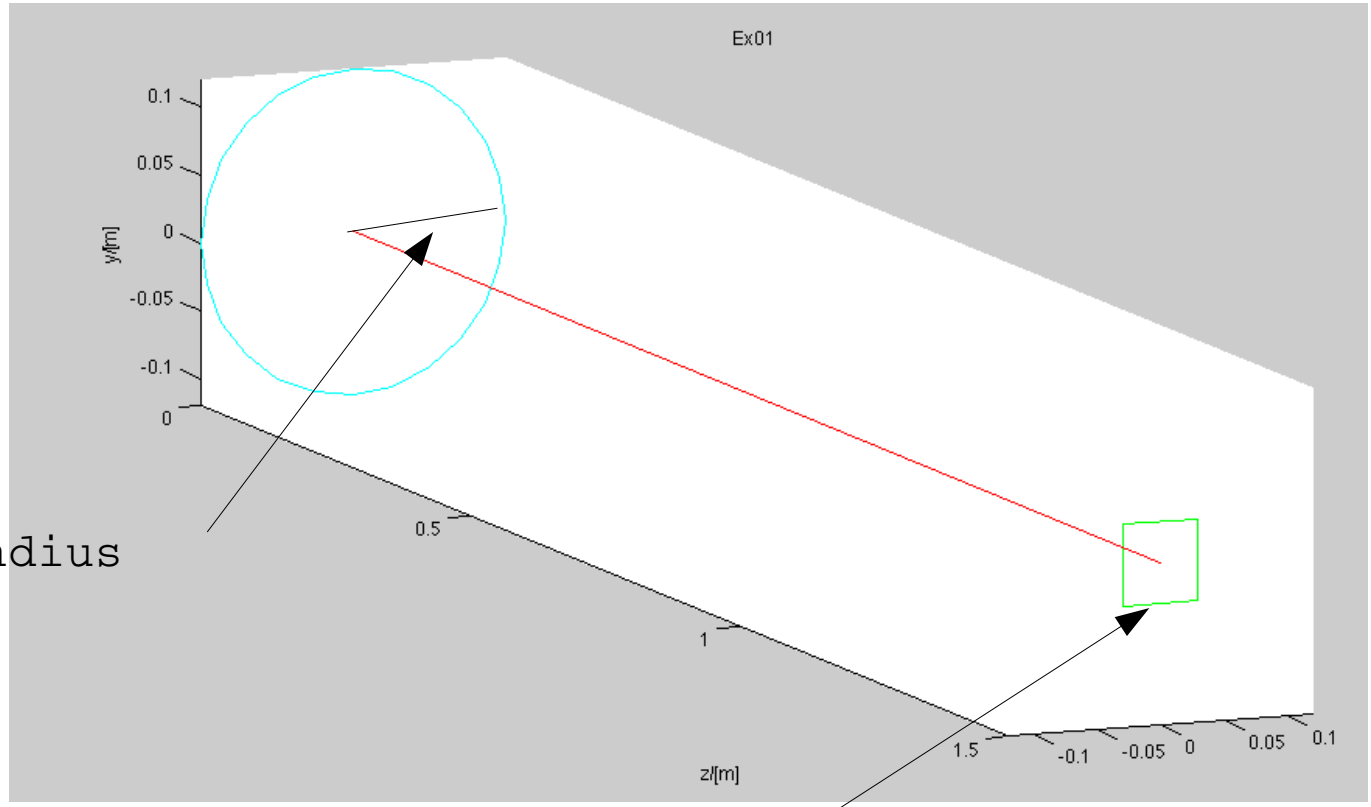
- When neutrons move in “free space”, we use ray-tracing - but in most cases in direction source -> detector
- Parabolas rather than straight lines are uses to implement gravity

McStas ray tracing



- Important efficiency mechanisms:
  - “Focusing” - e.g. source to beamport only (4π vs. limited solid angle only)
  - Rather vs. single particle description, absorption handled though statistics and downscaling the ray weight

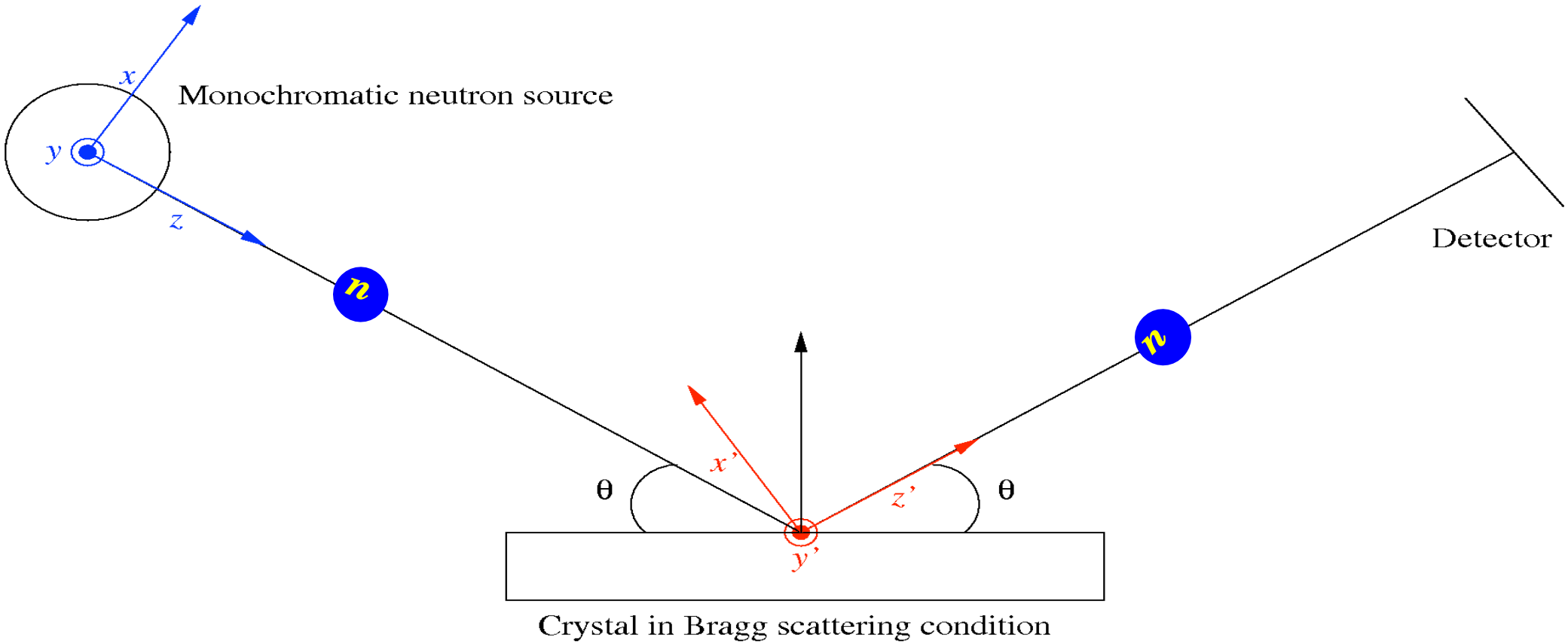
McStas variance reduction/focusing



Source `radius`

Source `focus_xw` x `focus_yh` @ `dist` m

McStas ray tracing

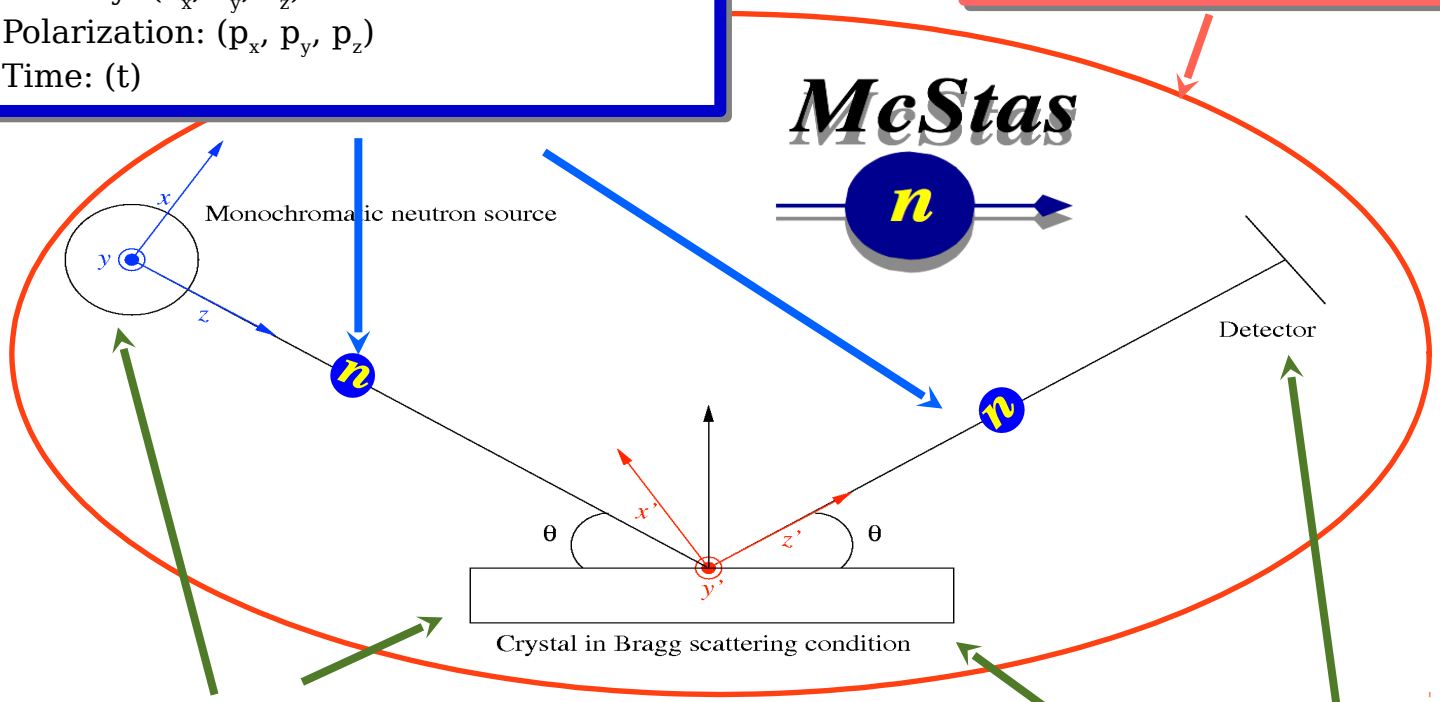


McStas ray tracing

Neutron ray/package:

Weight: (p) # neutrons left in the package
Position: (x, y, z)
Velocity: (v_x, v_y, v_z)
Polarization: (p_x, p_y, p_z)
Time: (t)

Instrument: positioning + transformation between component coordinate systems, e.g. neutron source, crystal, detector

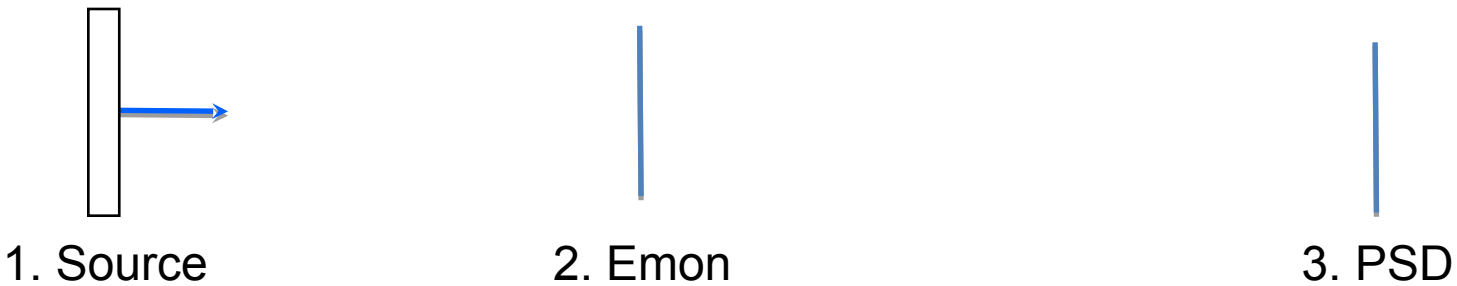


Components: Here the neutron physics happen, neutron weight adjusted according to scattering probabilities etc.

Local, internal coordinate system!

Component order

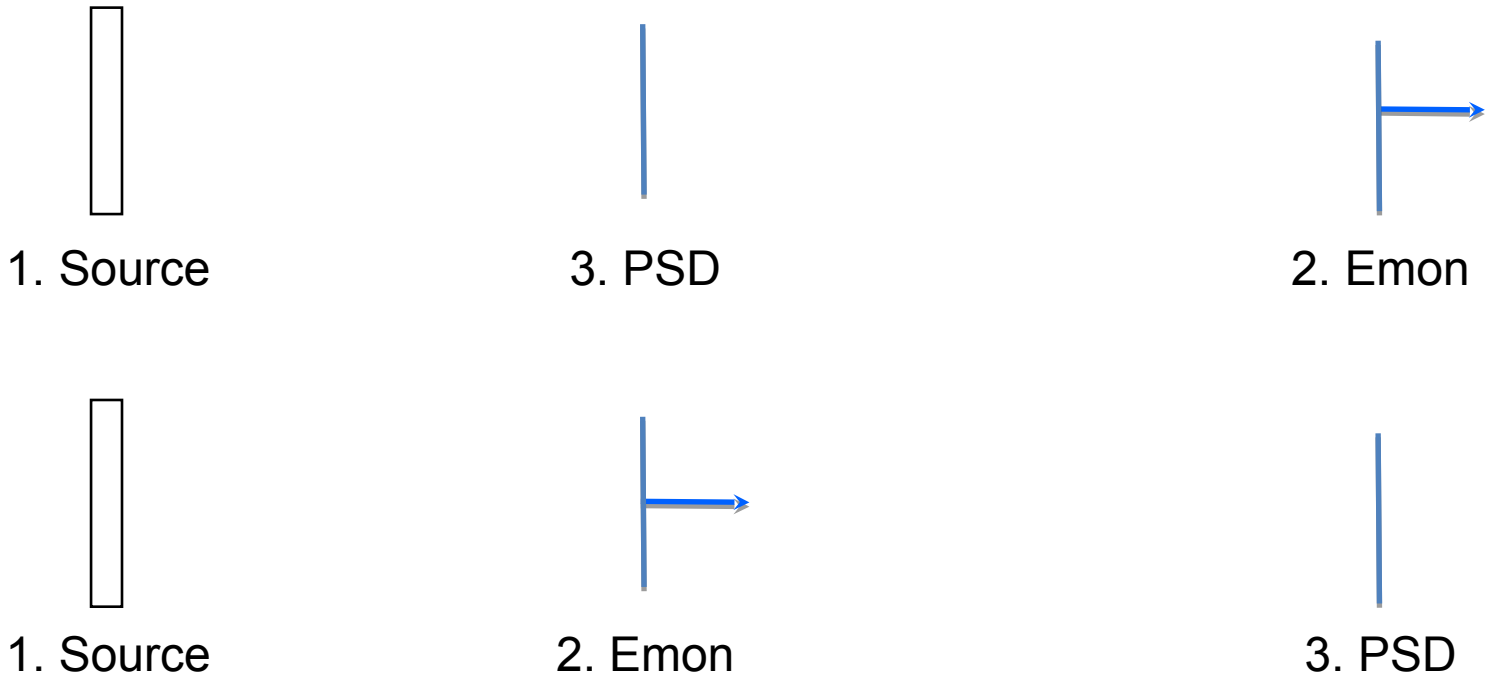
Order does matter:



Starting at the source

Component order

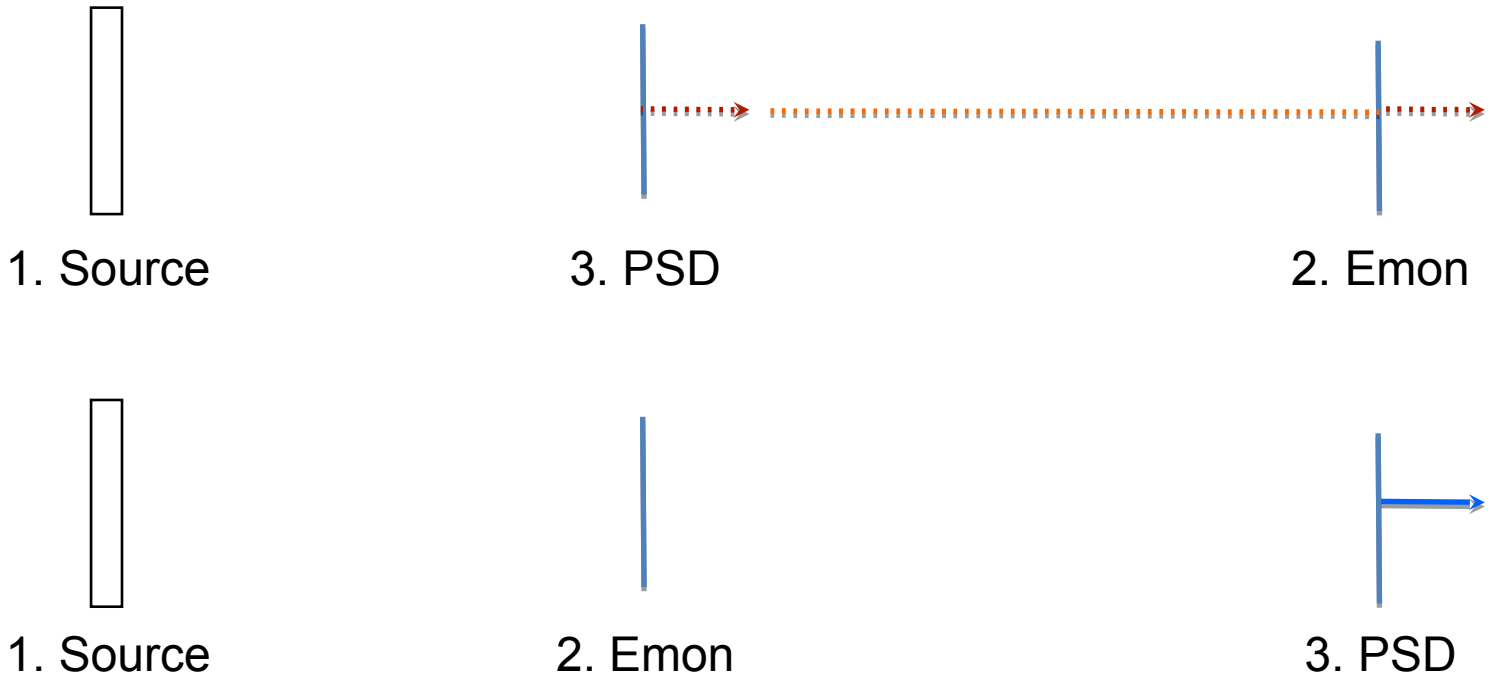
Order does matter:



Moving to first comp in the list

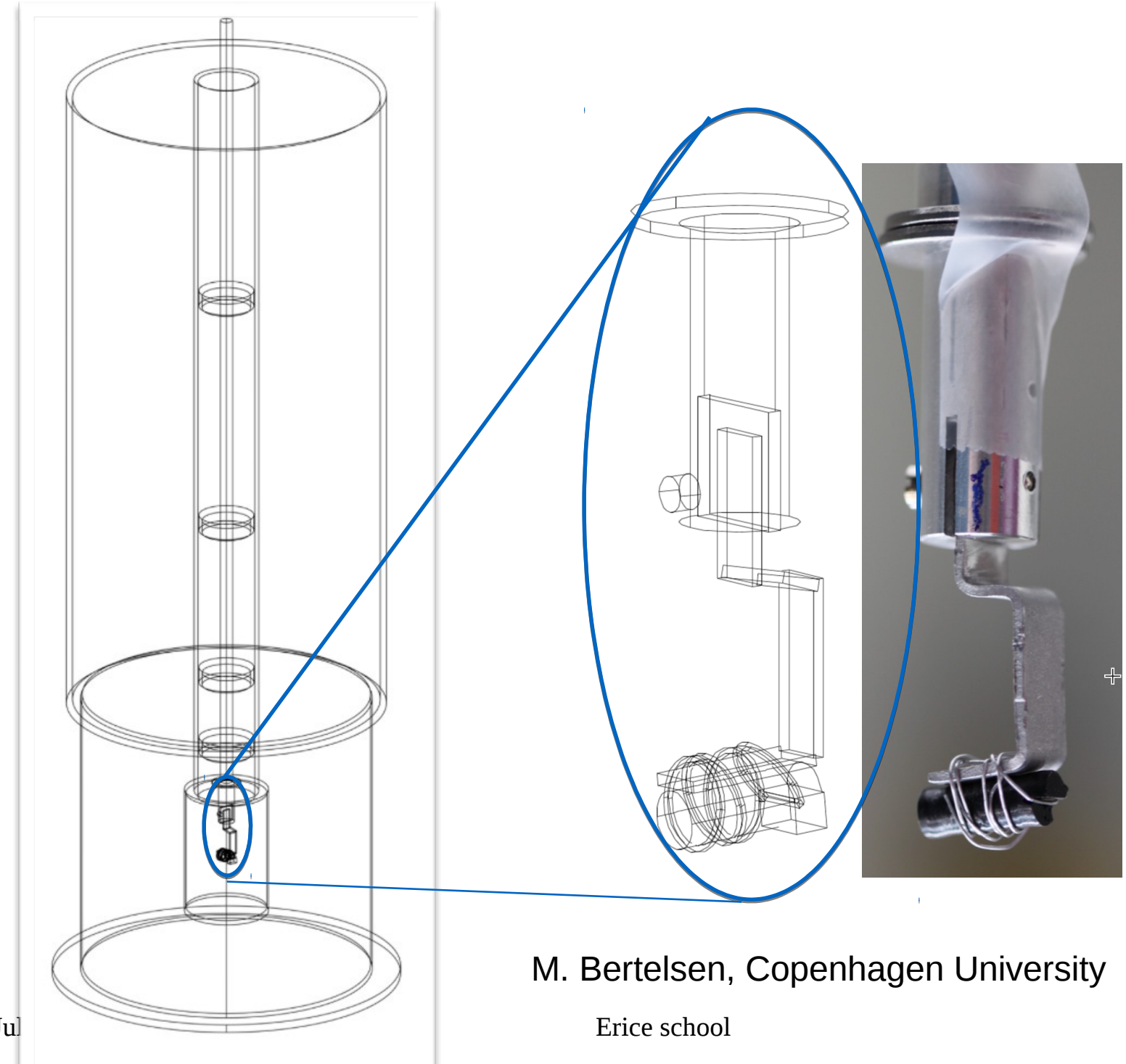
Component order

Order does matter:



Moving to 3rd comp in list requires “moving back in time”.
Default behavior is to ABSORB this type of neutron.
For monitors use restore_neutron=1 in this case.
For homegrown comps use ALLOW_BACKPROP macro.

Union



M. Bertelsen, Copenhagen University

Union

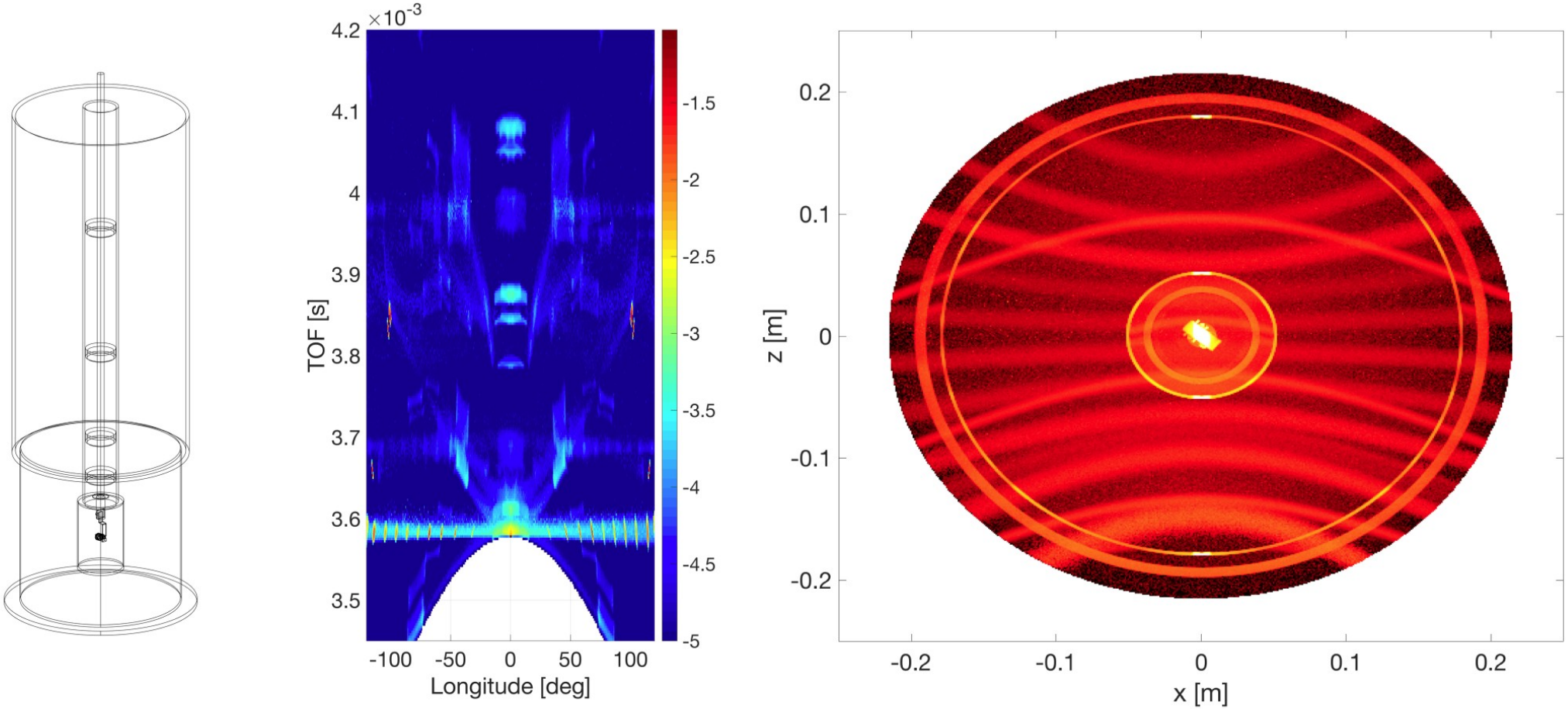


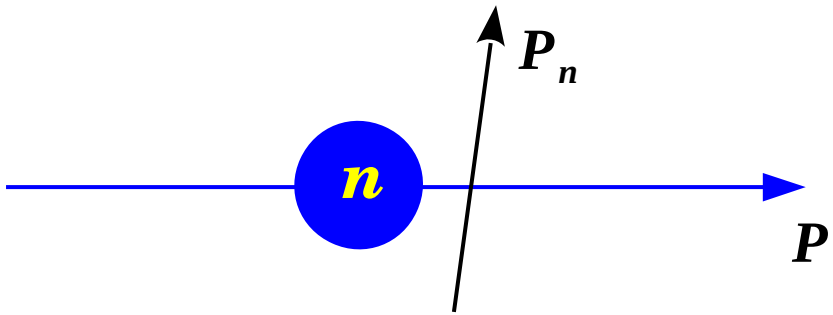
Figure 1: Left: Depiction of cryostat/sample model. Center: Time of flight banana monitor showing spurions. Right: Histogram over scattering locations in cryostat as seen from above.

Mcstas "particle" model

Neutron ray/package:

Weight: (p) # neutrons left in the package
Position: (x, y, z)
Velocity: (v_x, v_y, v_z)
Polarization: (s_x, s_y, s_z)
Time: (t)

$$\boldsymbol{P}_n = \frac{1}{p}\sum_i^p \boldsymbol{P}_i, \quad n = ray\,number$$



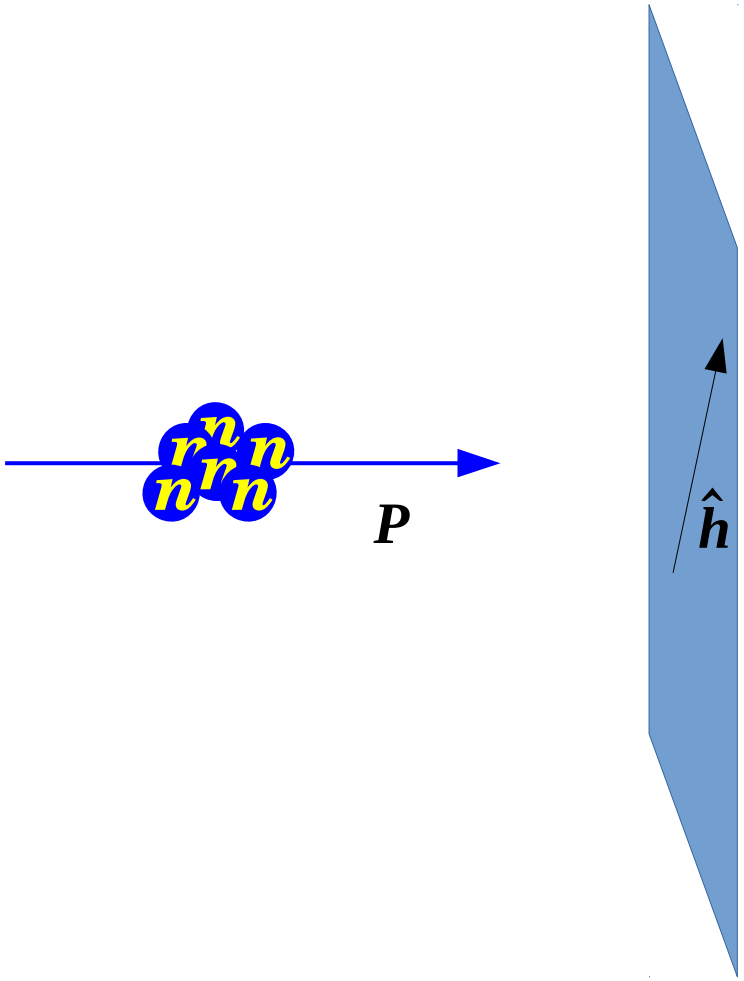
$$\boldsymbol{P}_i = 2\left(\langle \hat{\boldsymbol{s}}_{x,i}\rangle \hat{\boldsymbol{i}}_{x,i} + \langle \hat{\boldsymbol{s}}_{y,i}\rangle \hat{\boldsymbol{i}}_{y,i} + \langle \hat{\boldsymbol{s}}_{z,i}\rangle \hat{\boldsymbol{i}}_{z,i}\right)$$

$$\boldsymbol{P} = \frac{1}{N}\sum_{n=0}^{N} \boldsymbol{P}_n$$

From G. Williams: *"Polarized neutrons", Oxford Science Publ.*, 1988

McStas detectors/monitors

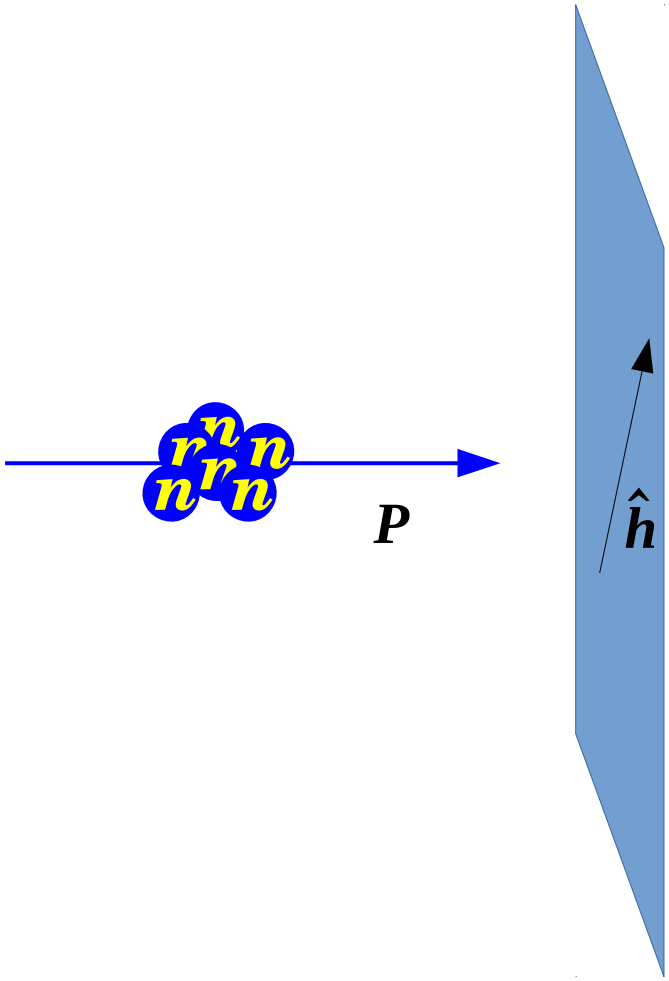
Monitoring: How and What do we monitor?

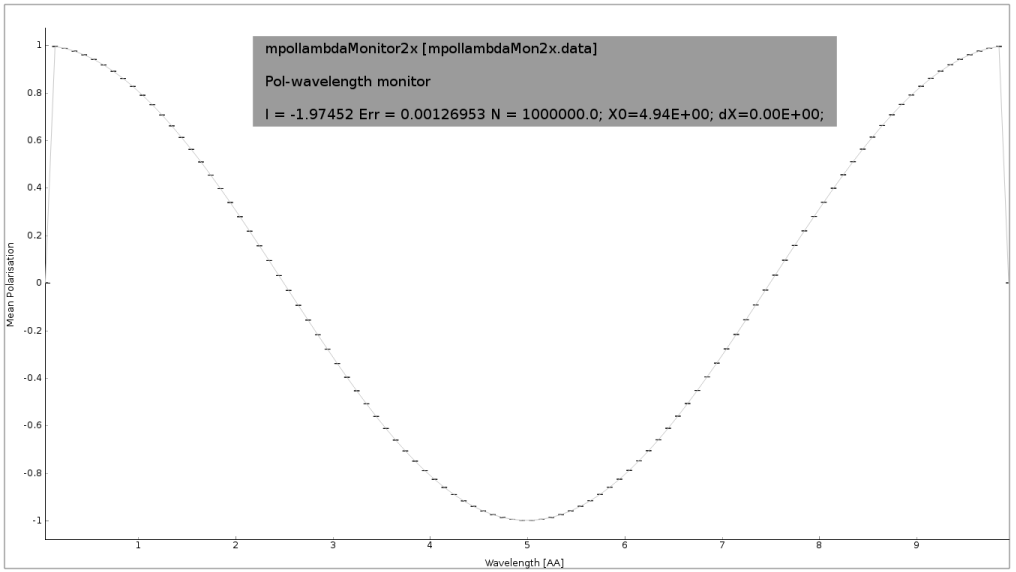


$$\boldsymbol{P}_{\hat{\boldsymbol{h}}} = \frac{\sum_{n=0}^{N} p_n \boldsymbol{P}_{\boldsymbol{n}} \cdot \hat{\boldsymbol{h}}}{\sum_{n}^{N} p_n}$$

McStas detectors/monitors

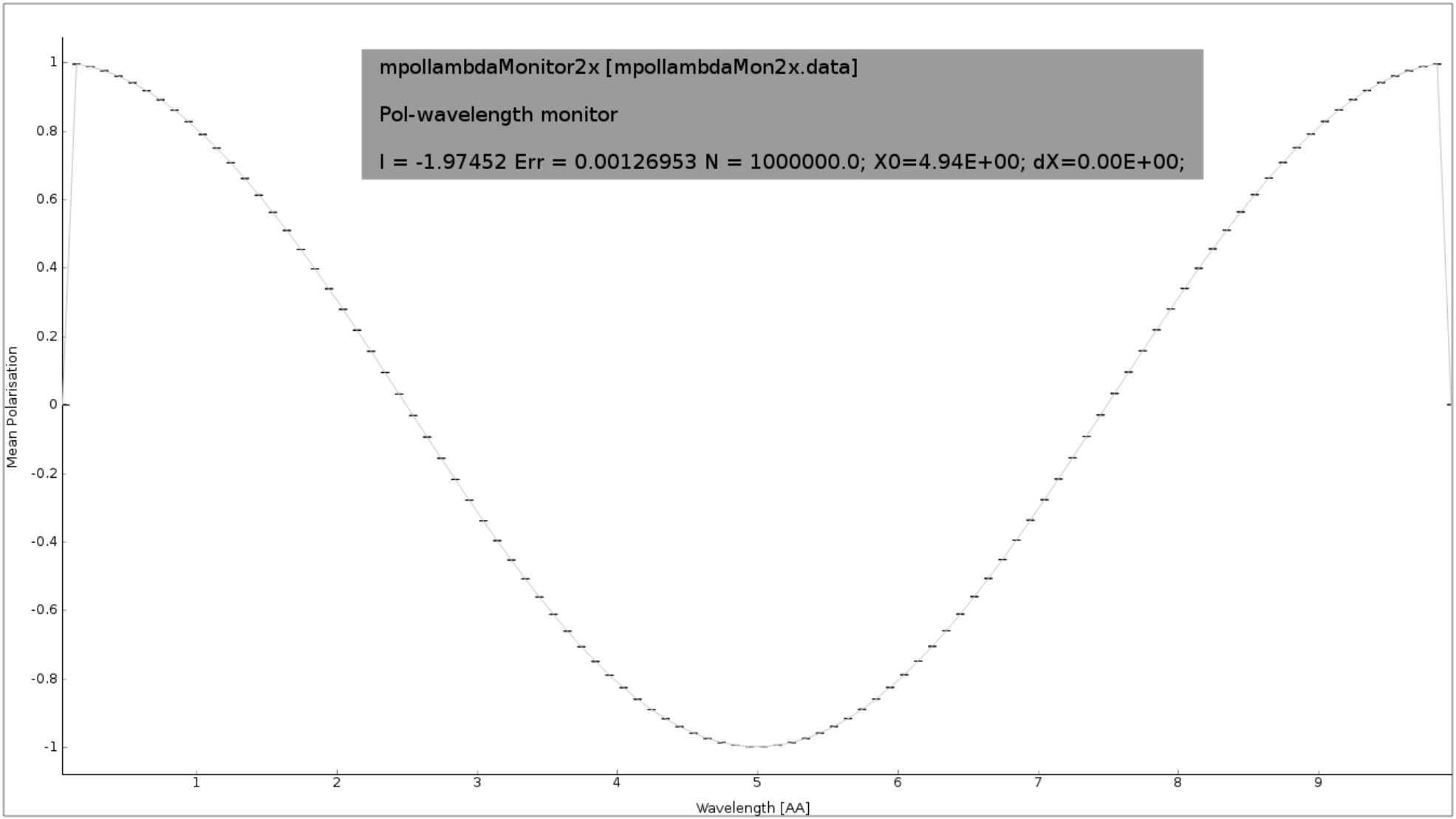
Monitoring: How and What do we monitor?





McStas detectors/monitors

Monitoring: How and What do we monitor?



Polarization monitors

Available monitors:

- `Pol_monitor.comp`: 0D
- `PolLambda_monitor.comp`: 2D
- `MeanPolLambda_monitor.comp`: 1D

McStas precession algorithm

Magnetic fields in McStas

- The challenge:
  - Fast beam/ray transport: # $rays > 10^6$
  - Unknown magnetic field and field strength
  - >1 Magnet → nested fields.

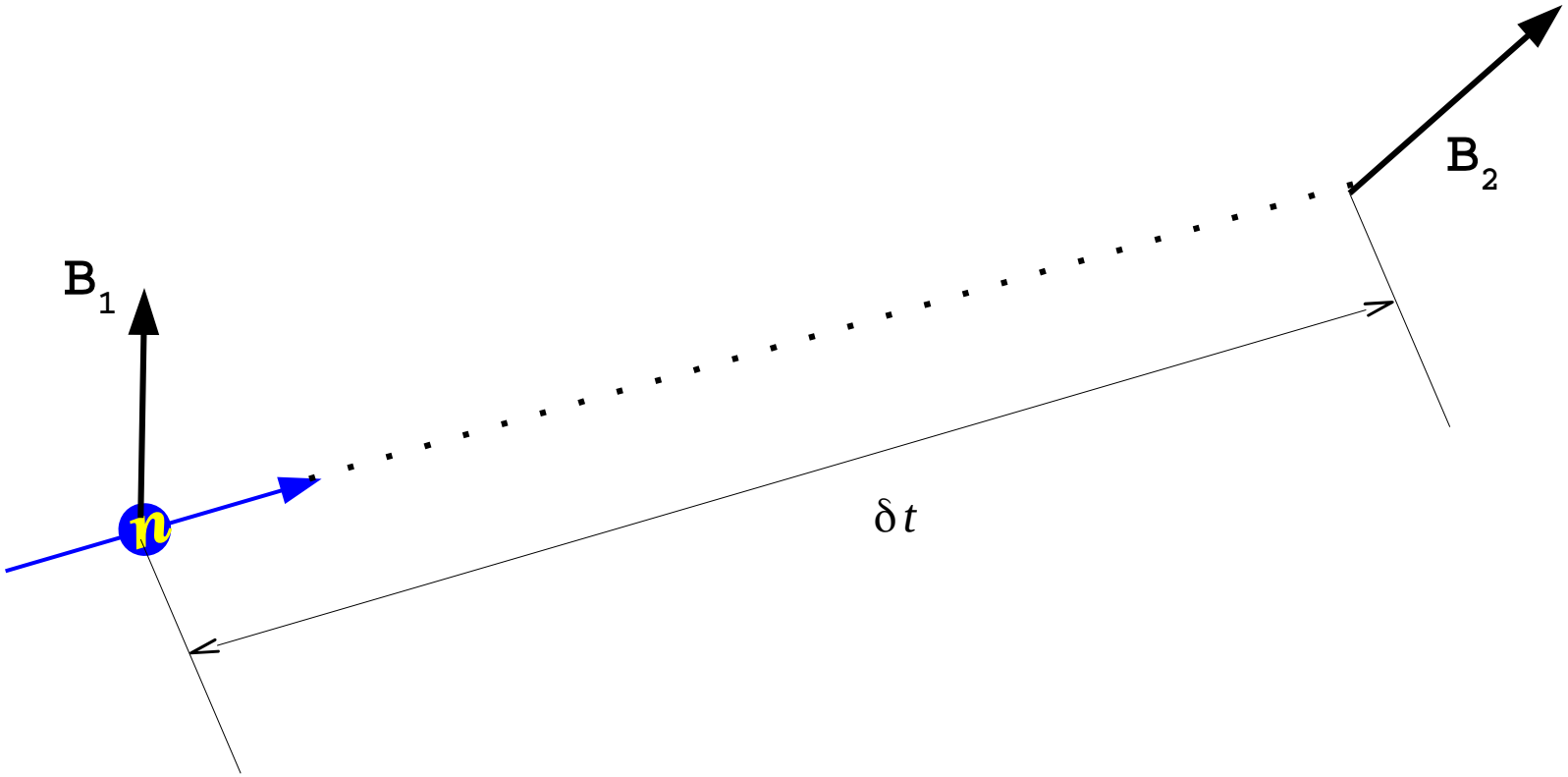
McStas precession algorithm

while $n_t < t_{target}$ **do**
- store neutron;
- sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- propagate neutron: $\delta t (< \Delta t)$;
- sample magnetic field: $\mathbf{B}_2 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- **while** $|\mathbf{B}_1 - \mathbf{B}_2| > \delta B_{threshold}$ **do**
  - restore neutron;
  - $\delta t := \delta t / 2$;
  - propagate neutron: $\delta t (< \Delta t)$;
  - sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- precess polarization: $\mathbf{P_n}$ by ω around $\frac{\mathbf{B}_1 + \mathbf{B}_2}{2}$;

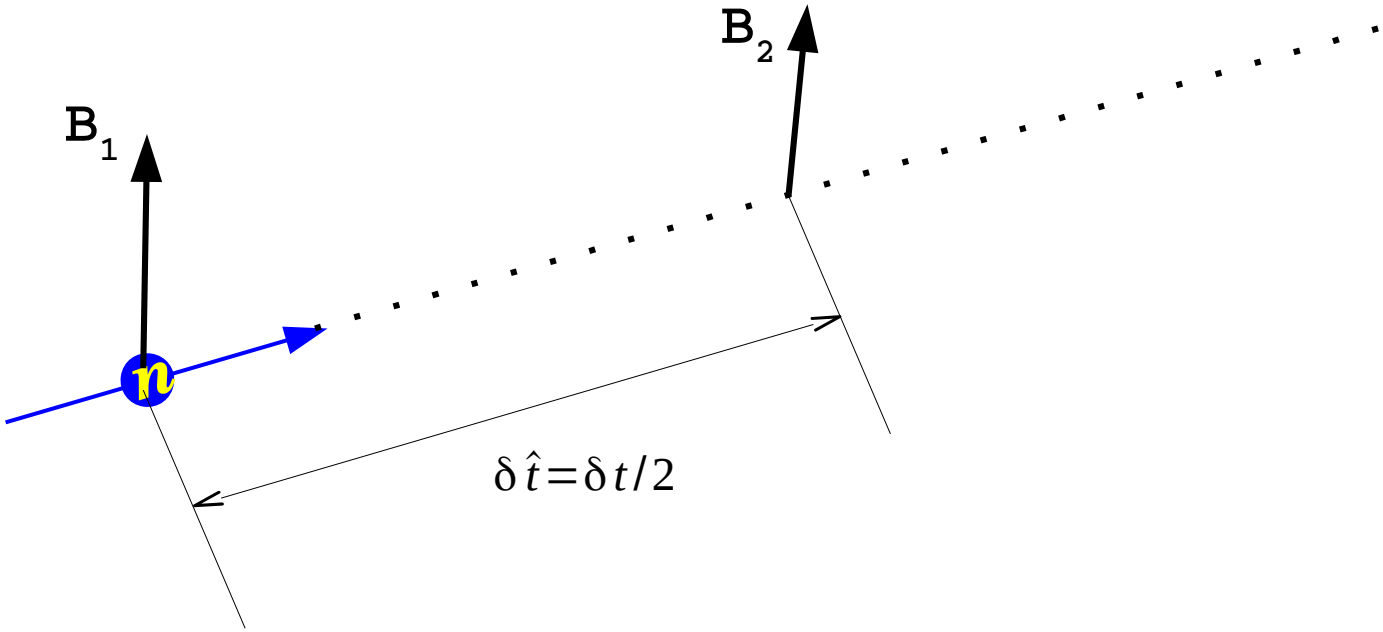
Algorithm 1: SimpleNumMagnetPrecession: Simplistic algorithm for tracking polarization of a Monte-Carlo neutron in a magnetic field. The neutron's state is stored as a position (n_x, n_y, n_z), a velocity $\mathbf{v}$, time n_t, and polarization vector $\mathbf{P_n}$.

From: Knudsen et.al., *J. Neutron Research*, 2014

McStas precession algorithm



McStas precession algorithm



McStas precession algorithm

while $n_t < t_{target}$ **do**
- store neutron;
- sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- propagate neutron: $\delta t (< \Delta t)$;
- sample magnetic field: $\mathbf{B}_2 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- **while** $|\mathbf{B}_1 - \mathbf{B}_2| > \delta B_{threshold}$ **do**
  - restore neutron;
  - $\delta t := \delta t / 2$;
  - propagate neutron: $\delta t (< \Delta t)$;
  - sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- precess polarization: $\mathbf{P_n}$ by ω around $\frac{\mathbf{B}_1 + \mathbf{B}_2}{2}$;

Algorithm 1: SimpleNumMagnetPrecession: Simplistic algorithm for tracking polarization of a Monte-Carlo neutron in a magnetic field. The neutron's state is stored as a position (n_x, n_y, n_z), a velocity $\mathbf{v}$, time n_t, and polarization vector $\mathbf{P_n}$.

From: Knudsen et.al., *J. Neutron Research*, 2014

McStas precession algorithm

while $n_t < t_{target}$ **do**
- store neutron;
- sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- propagate neutron: $\delta t (< \Delta t)$;
- sample magnetic field: $\mathbf{B}_2 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- **while** $|\mathbf{B}_1 - \mathbf{B}_2| > \delta B_{threshold}$ **do**
  - restore neutron;
  - $\delta t := \delta t/2$;
  - propagate neutron: $\delta t (< \Delta t)$;
  - sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- precess polarization: $\mathbf{P_n}$ by ω around $\frac{\mathbf{B}_1 + \mathbf{B}_2}{2}$;

Algorithm 1: SimpleNumMagnetPrecession: Simplistic algorithm for tracking polarization of a Monte-Carlo neutron in a magnetic field. The neutron's state is stored as a position (n_x, n_y, n_z), a velocity $\mathbf{v}$, time n_t, and polarization vector $\mathbf{P_n}$.

From: Knudsen et.al., *J. Neutron Research*, 2014

McStas precession algorithm

while $n_t < t_{target}$ **do**

- store neutron;
- sample magnetic field
- propagate neutron: $\delta t (< \Delta t)$;
- sample magnet
- **while** $|\mathbf{B}_1 - \mathbf{B}_2| > \delta B_{threshold}$ **do**
  - restore neutron;
  - $\delta t := \delta t / 2$;
  - propagate neutron: $\delta t (< \Delta t)$;
  - sample magnetic field: $\mathbf{B}_1 = \mathbf{B}(n_x, n_y, n_z, n_t)$;
- precess polarization: $\mathbf{P_n}$ by ω around $\frac{\mathbf{B}_1 + \mathbf{B}_2}{2}$;

```
void mc_pol_set_timestep(double dt);
```

```
void mc_pol_set_angular_accuracy(double domega);
```

Algorithm 1: SimpleNumMagnetPrecession: Simplistic algorithm for tracking polarization of a Monte-Carlo neutron in a magnetic field. The neutron's state is stored as a position (n_x, n_y, n_z), a velocity $\mathbf{v}$, time n_t, and polarization vector $\mathbf{P_n}$.

From: Knudsen et.al., *J. Neutron Research*, 2014

McStas polarization components

Magnetic fields:
- Pol_FieldBox.comp
- Pol_constBfield.comp
- Pol_simpleBfield.comp
- Pol_simpleBfield_stop.comp
- Pol_triafield.comp

Optics:
- Monochromator_pol.comp
- Pol_bender.comp
- Pol_guide_vmirror.comp
- Pol_mirror.comp
- Pol_pi_2_rotator.comp
- Transmission_polarisatorABSnT.comp
- Pol_bender_tapering.comp

Monitors:
- Pol_monitor.comp
- MeanPolLambda_monitor.comp
- PolLambda_monitor.comp

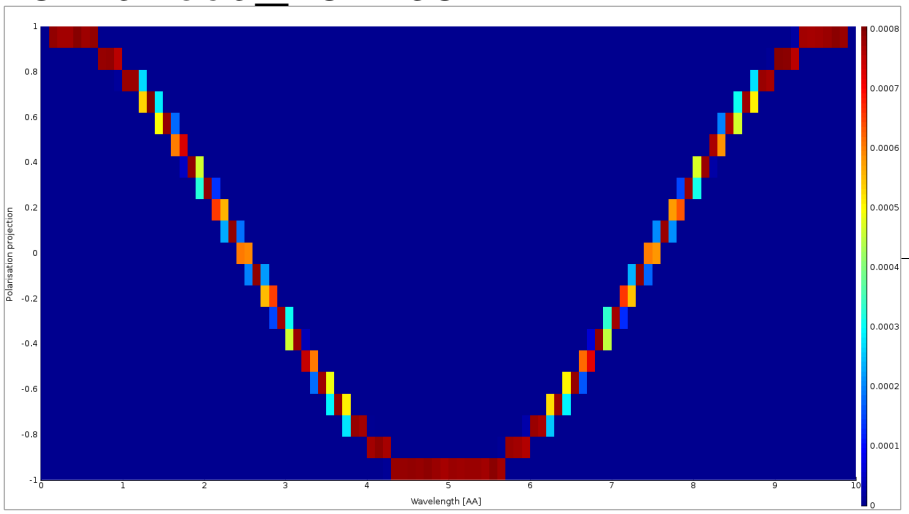
Idealized components:
- PolAnalyser_ideal.comp
- Set_pol.comp

Contrib:
- Foil_flipper_magnet.comp

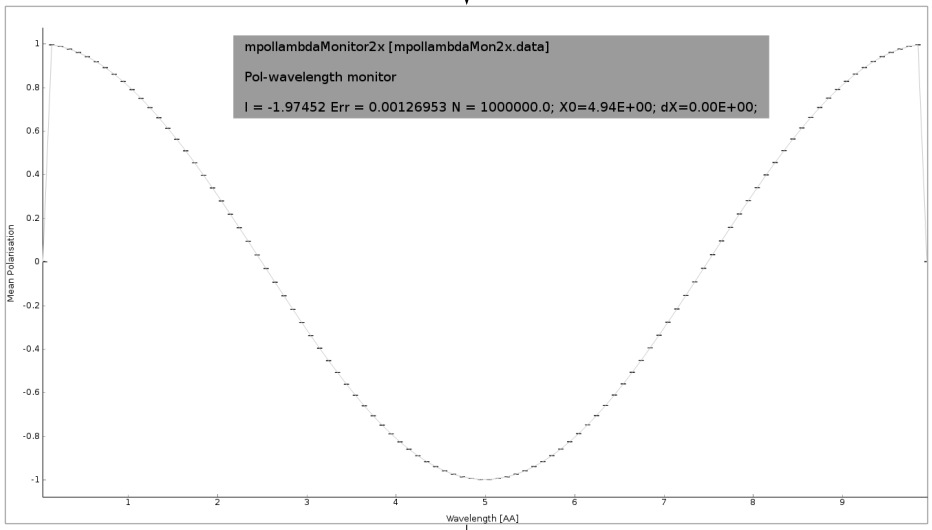
McStas polarization monitors

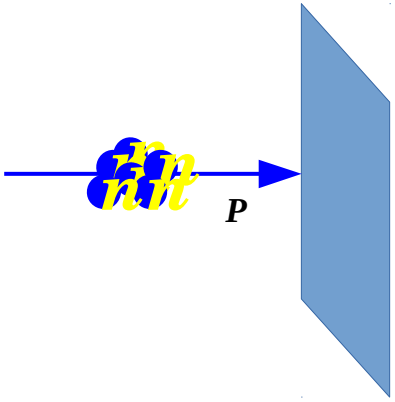
Monitors

PolLambda_monitor



MeanPolLambda_monitor



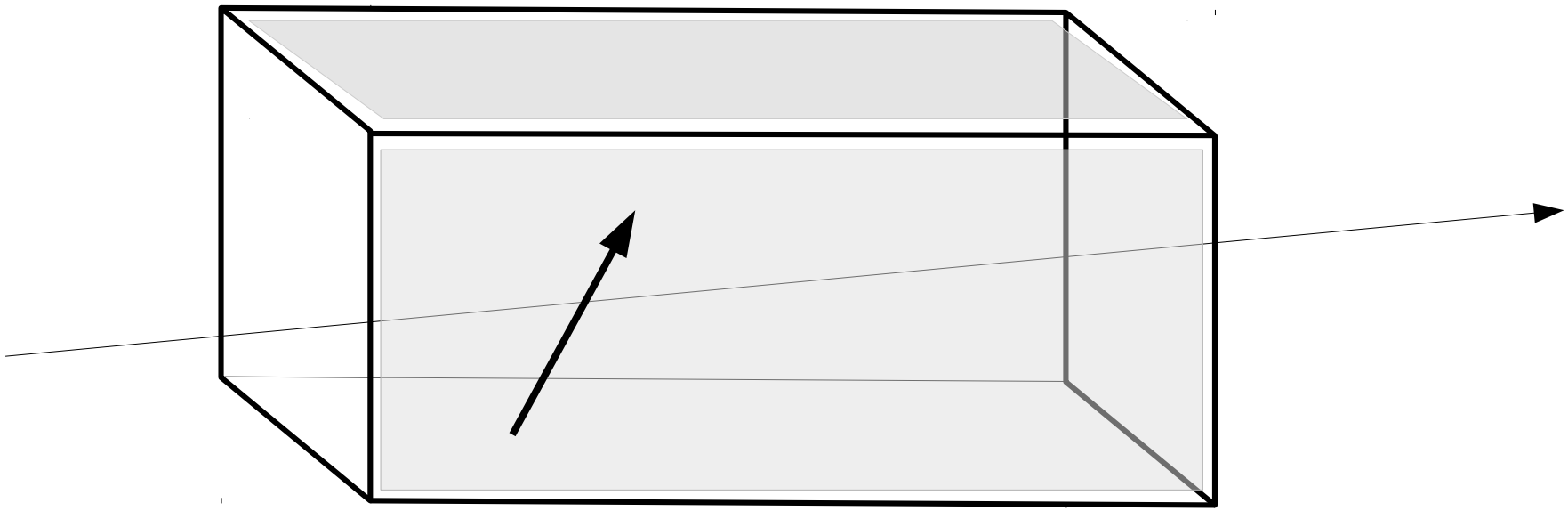


Pol_monitor

$$\boldsymbol{P}\|(m_x, m_y, m_z) = 0.87$$

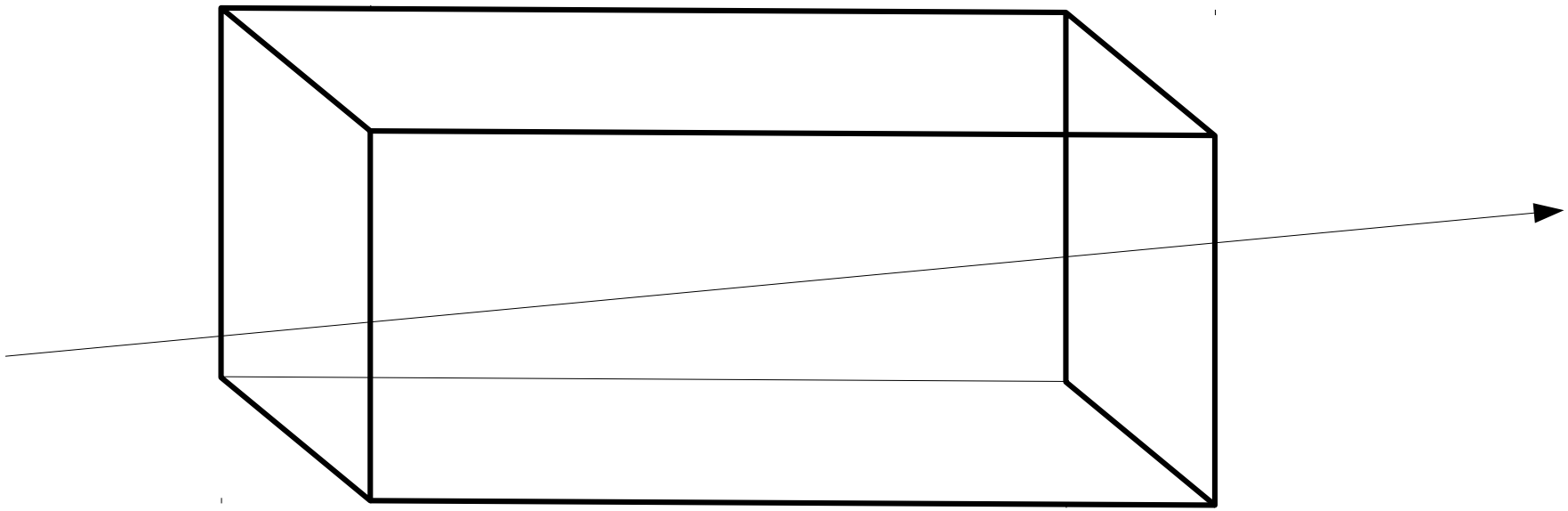
McStas magnetic fields

- `Pol_constBfield.comp`
  Single constant Magnetic field in a “box”.
  - user may specify a wavelength to flip.
  - blocking walls



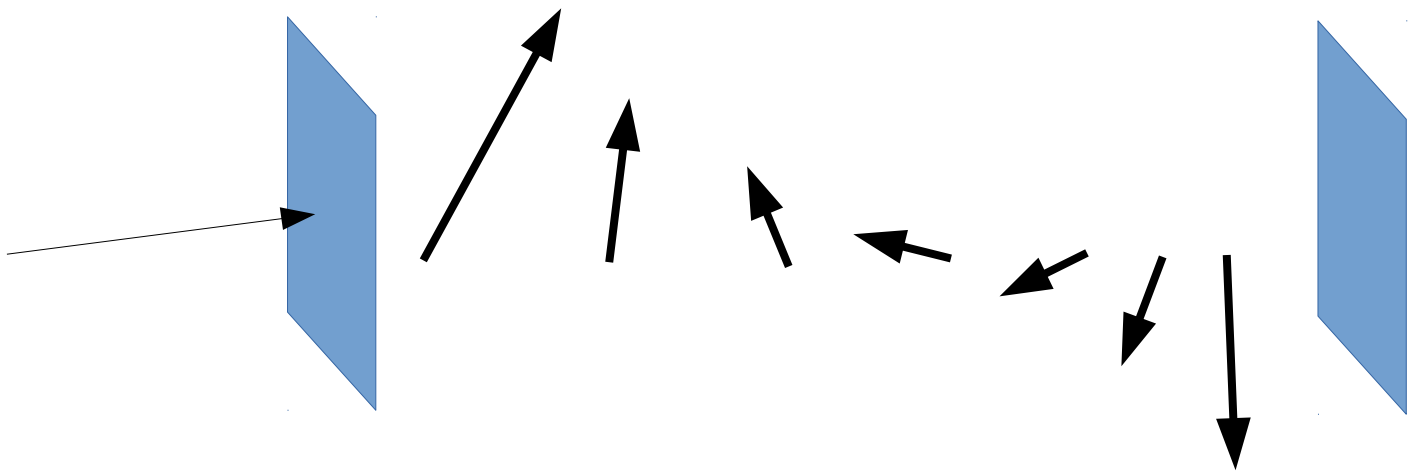
McStas magnetic fields

- `Pol_FieldBox.comp`
  Single Magnetic field in a “box”
  - optional user supplied field c-function



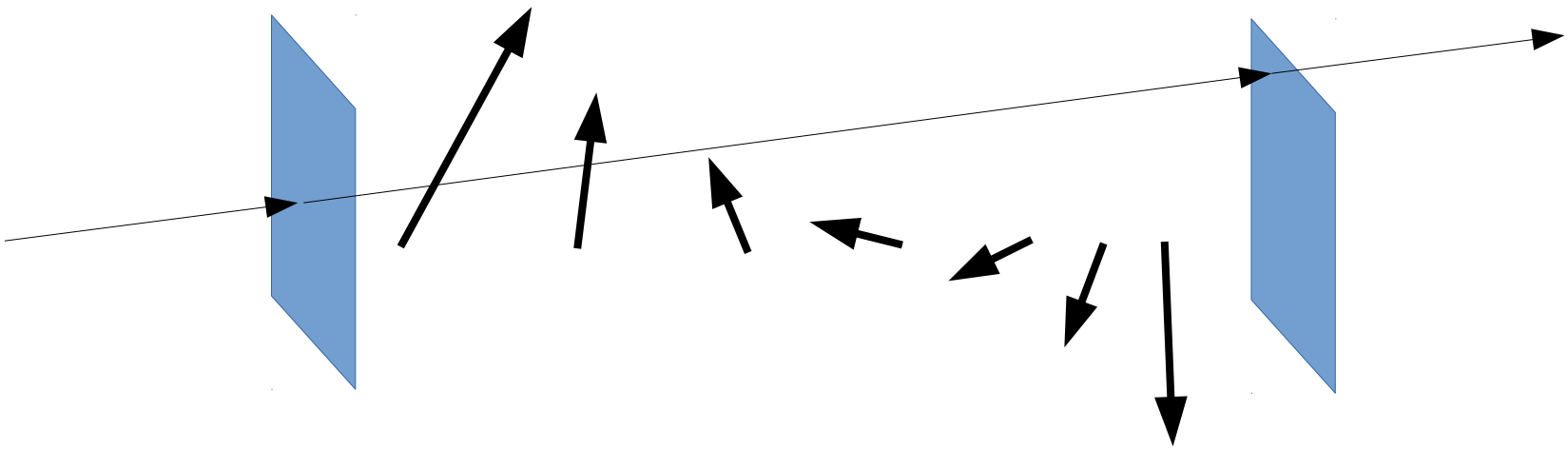
McStas magnetic fields

- `Pol_simpleBfield.comp`
- `Pol_simpleBfield_stop.comp`
  - **- Entry/Exit contruction allows for nested magnetic field descriptions.**
    - Any magnetic fields through user supplied c-function
    - Tabled magnetic fields

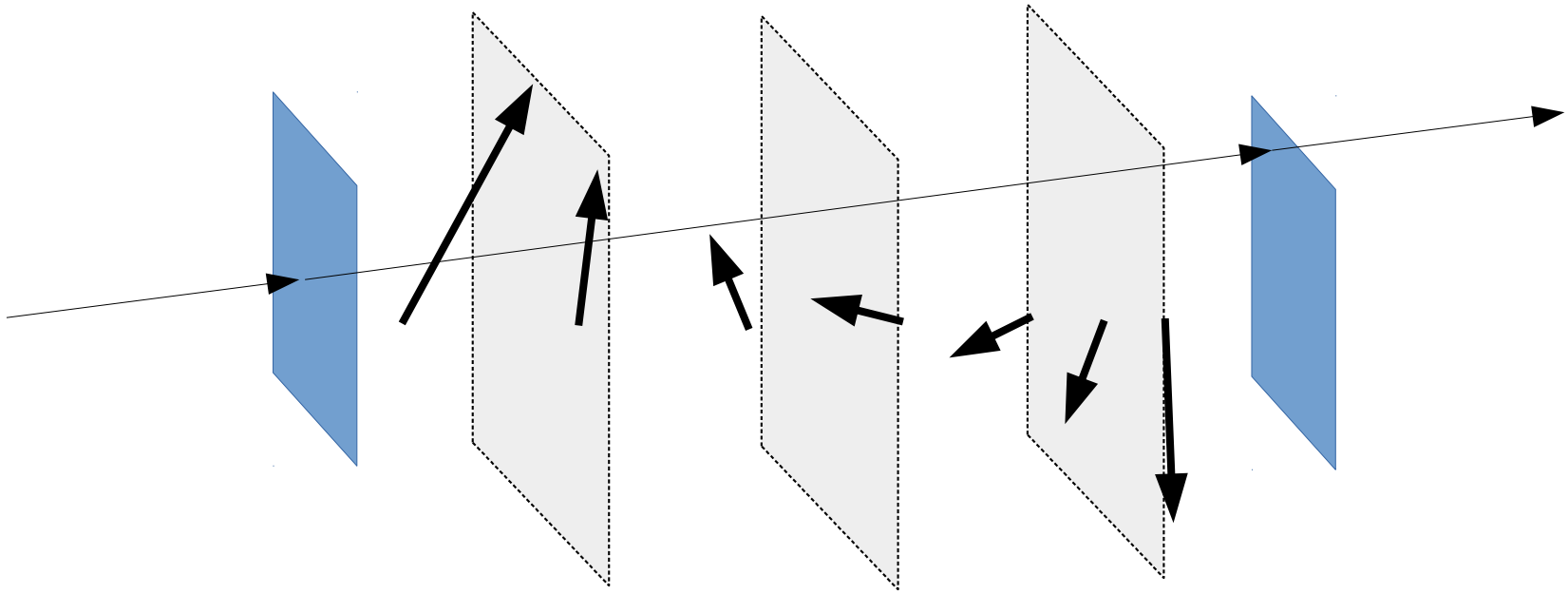


McStas Polarization Capabilities IV

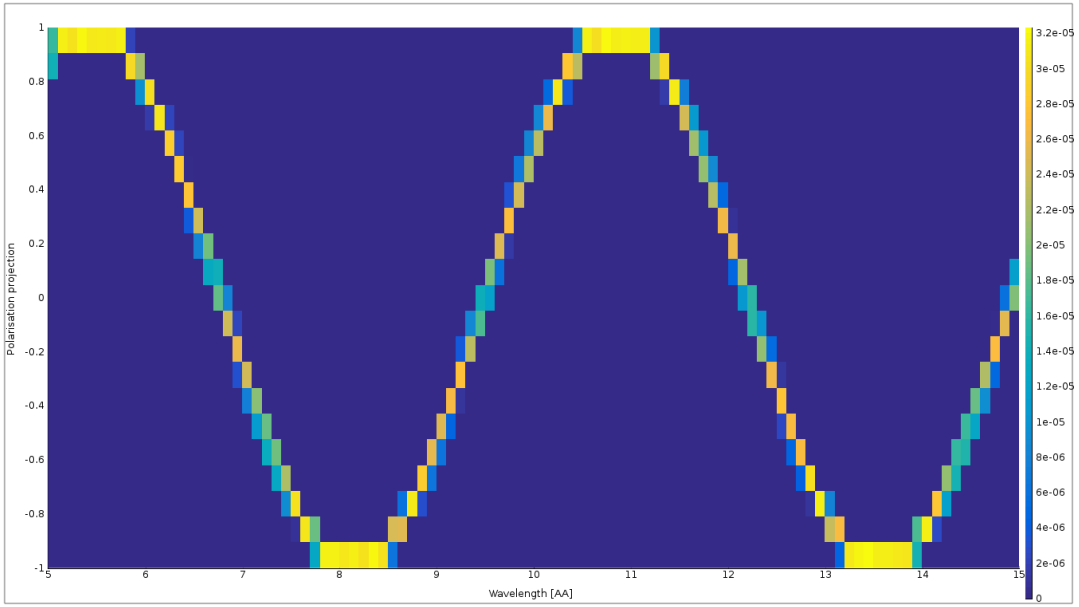
- `Pol_simpleBfield.comp`
- `Pol_simpleBfield_stop.comp`
  - **- Entry/Exit contruction allows for nested magnetic field descriptions.**
    - Any magnetic fields through user supplied c-function
    - Tabled magnetic fields

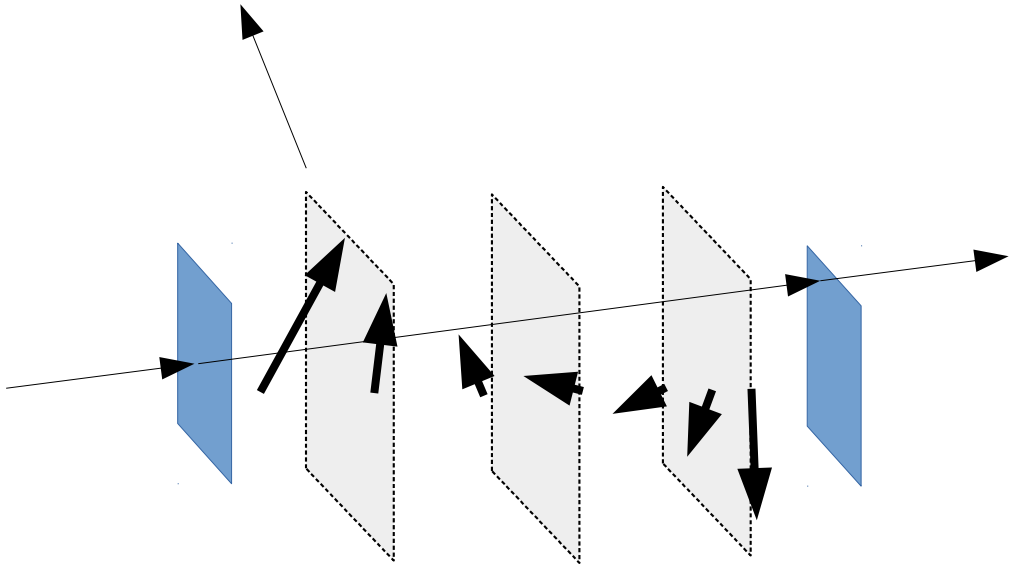


Pol_monitors along the way...

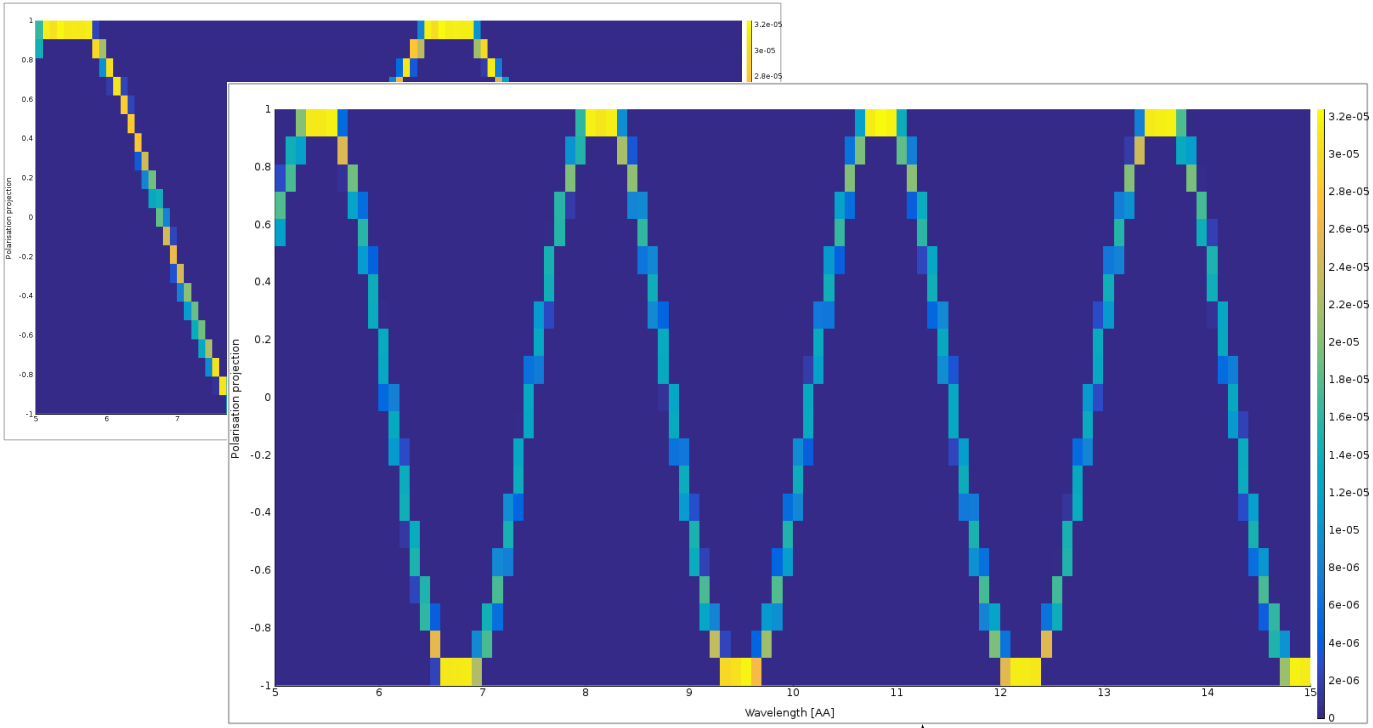


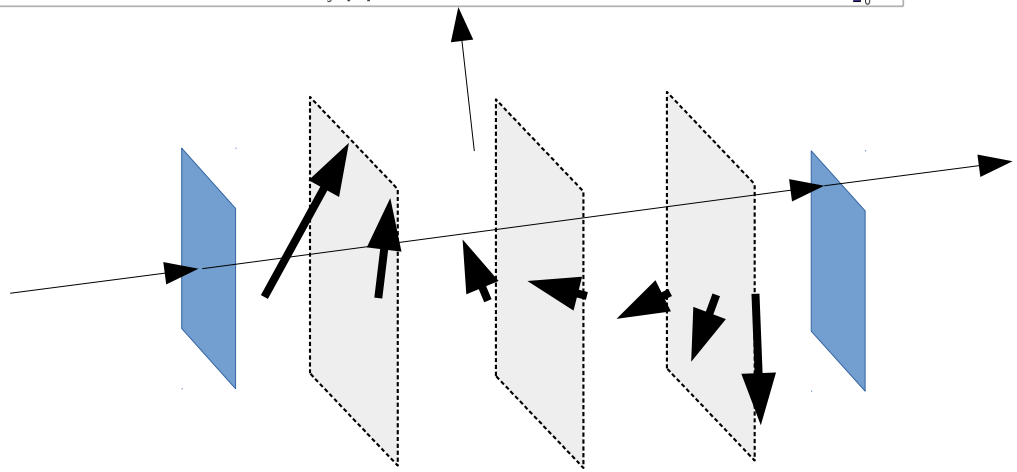
Pol_monitors along the way...



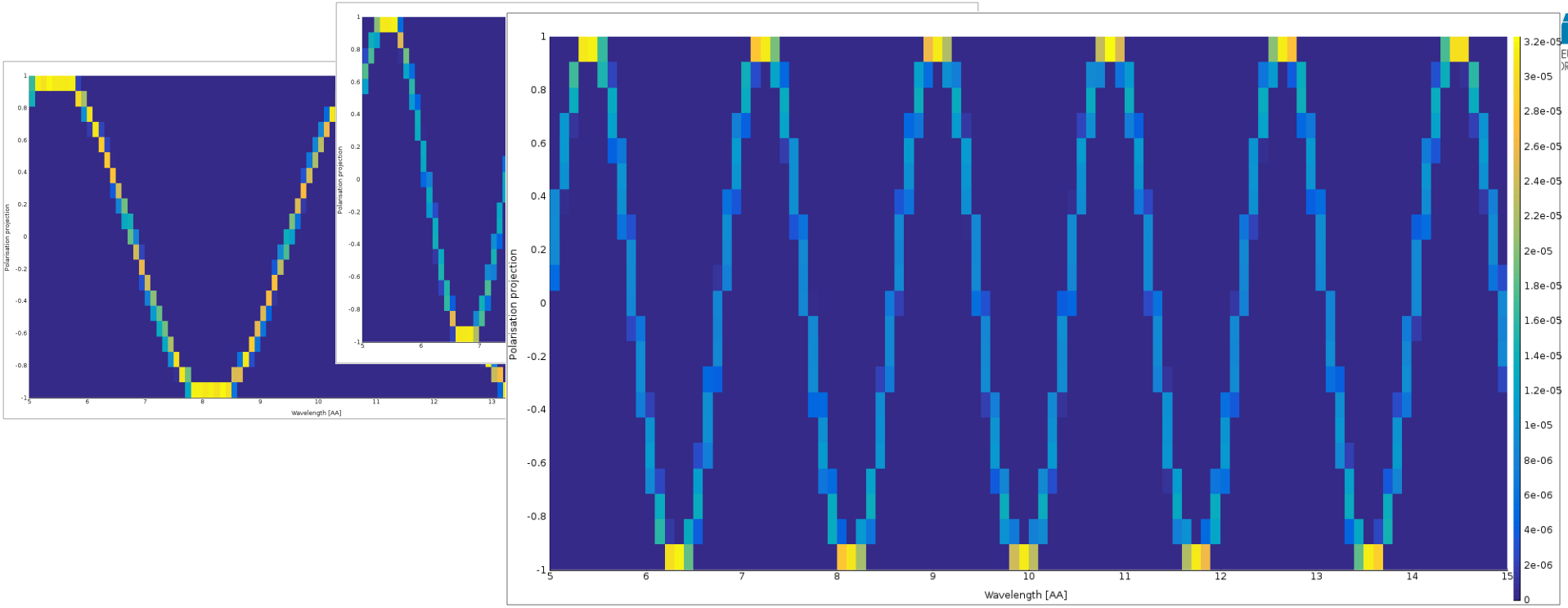


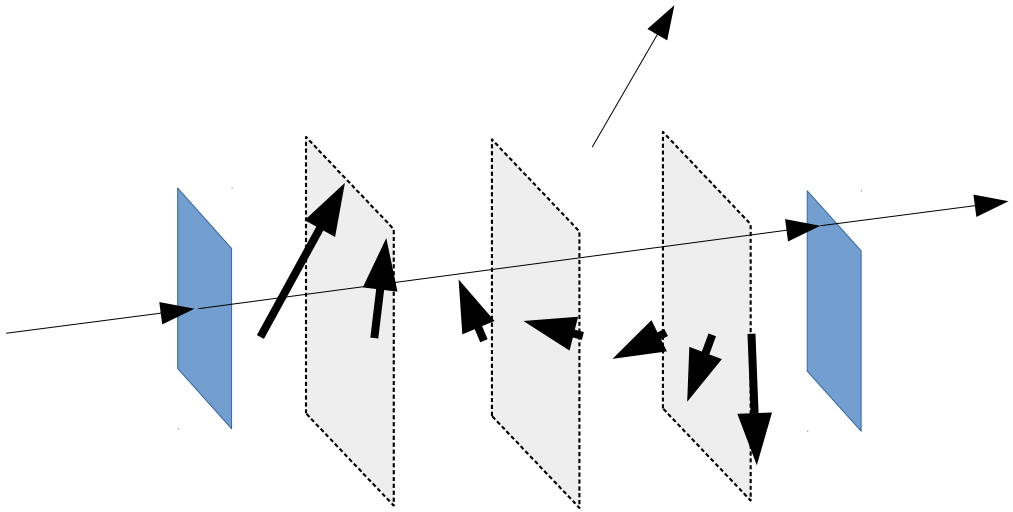
Pol_monitors along the way...



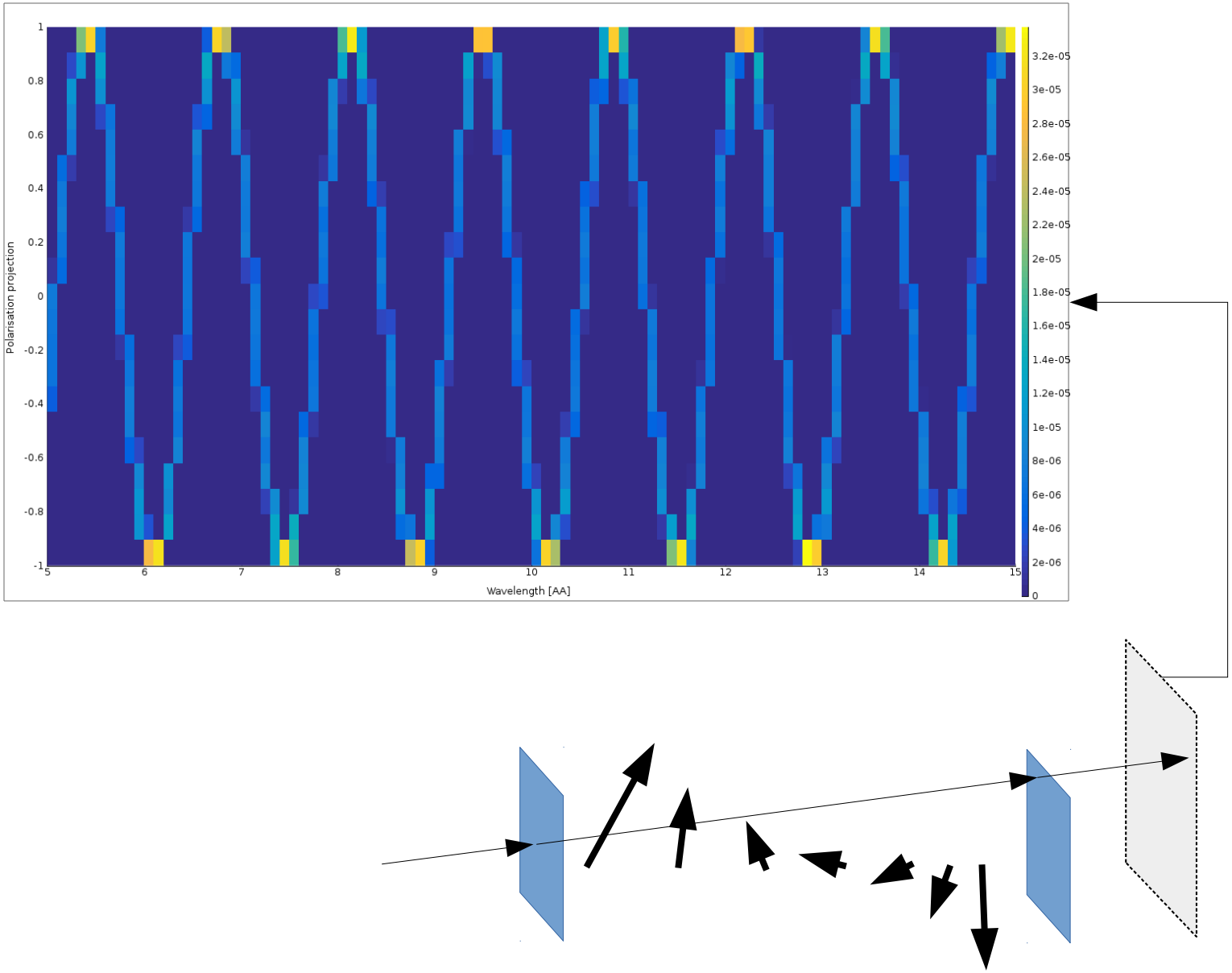


Pol_monitors along the way...

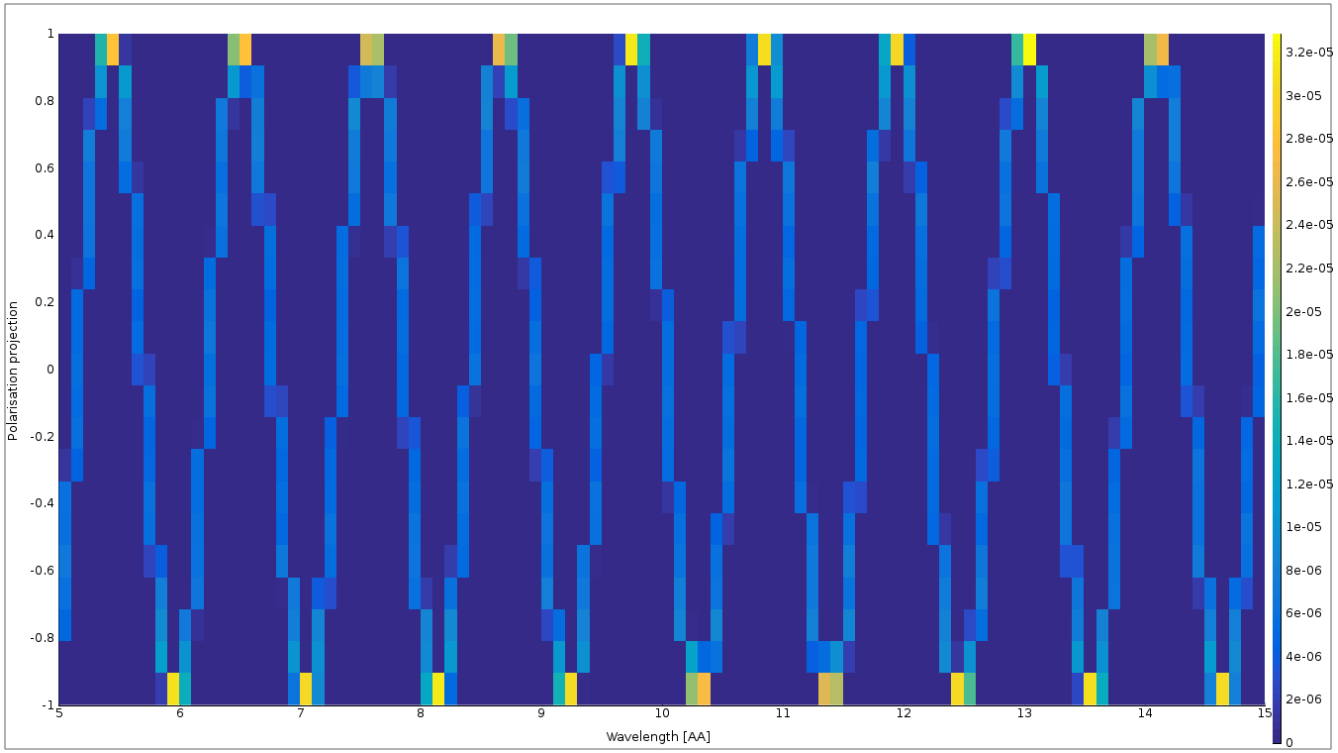


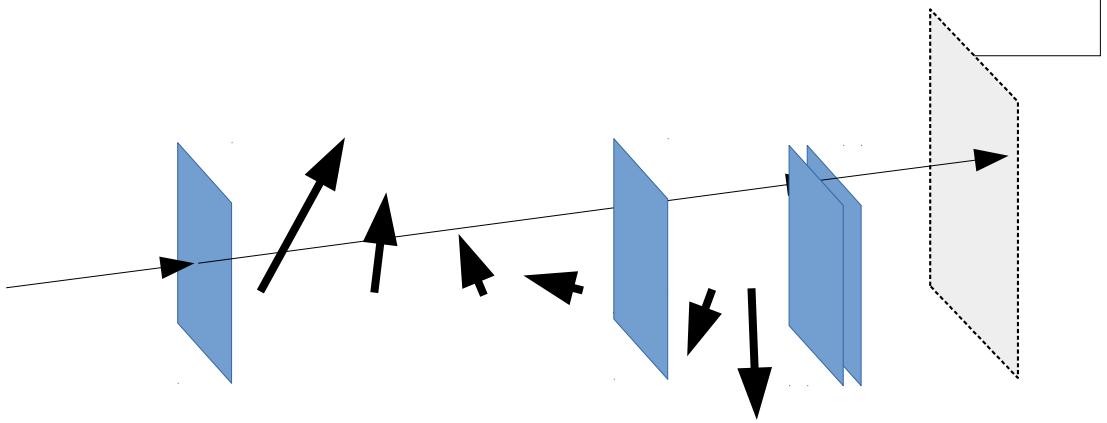


Nested fields



Nested fields





How does one go about it?

Example walk-through: SE-template

How does one go about it?

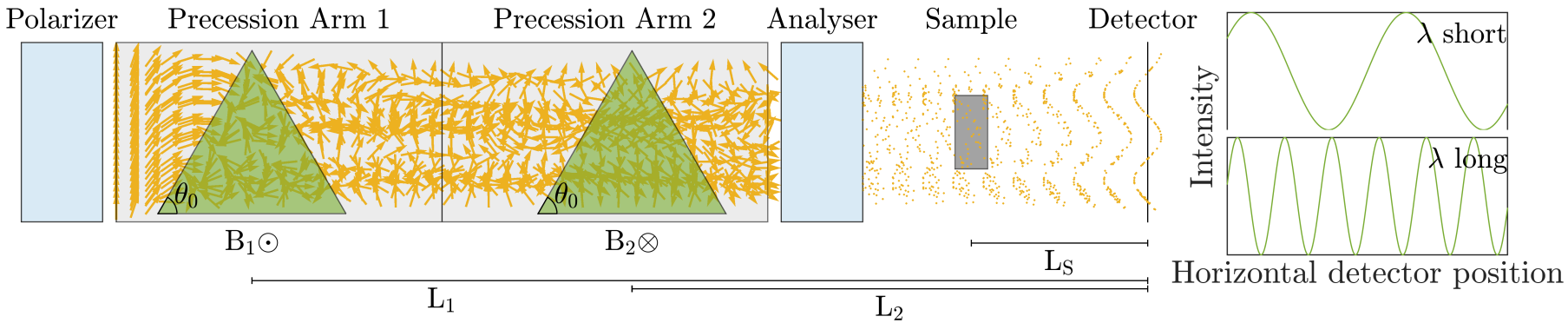
Example walk-through: SE-template

Getting help

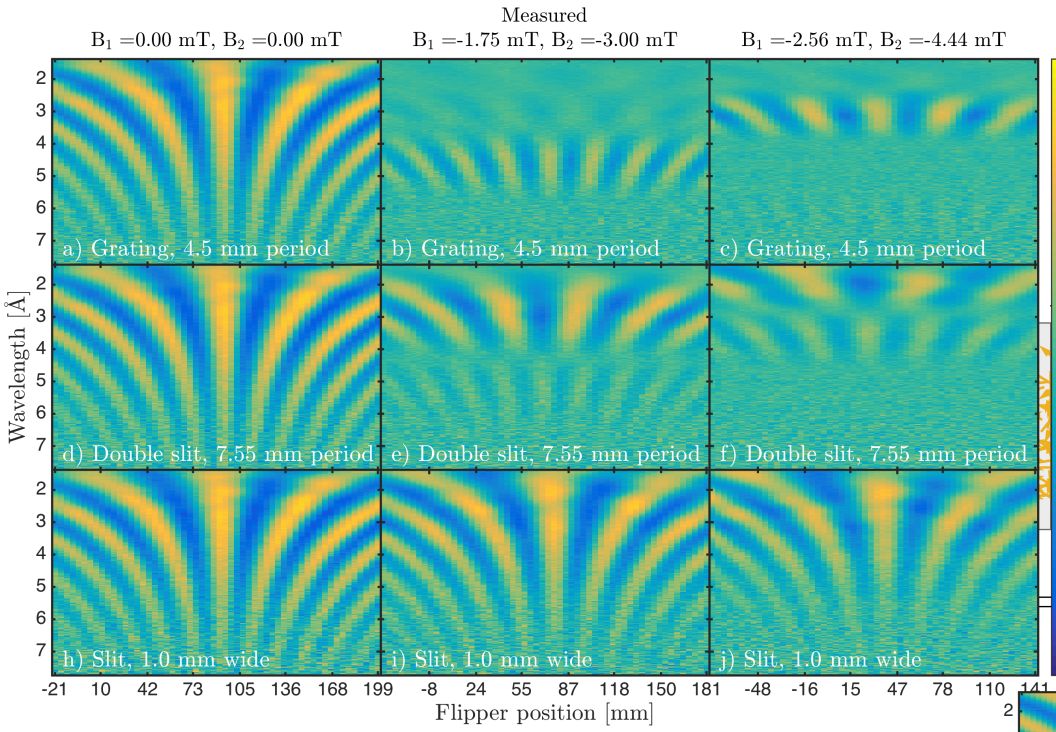
- Check example header.
- Use `mcdoc`
- Read/check the manual
- User mailing list: `mcstas-users@mcstas.org`
- Give us a call/write us an email!

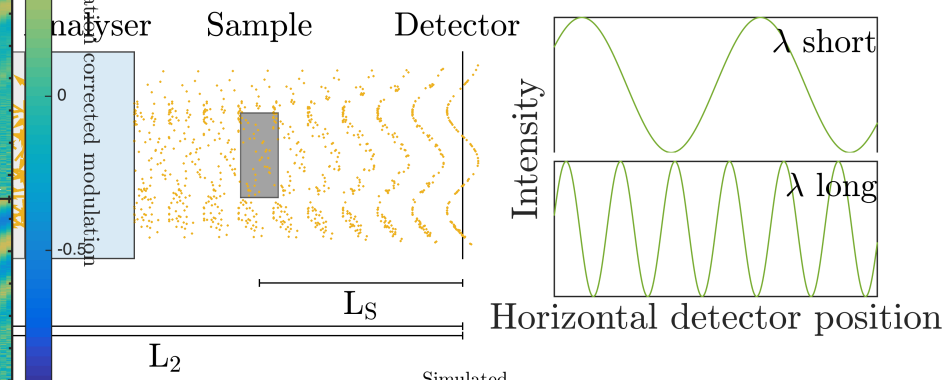
McStas example SEMSANS

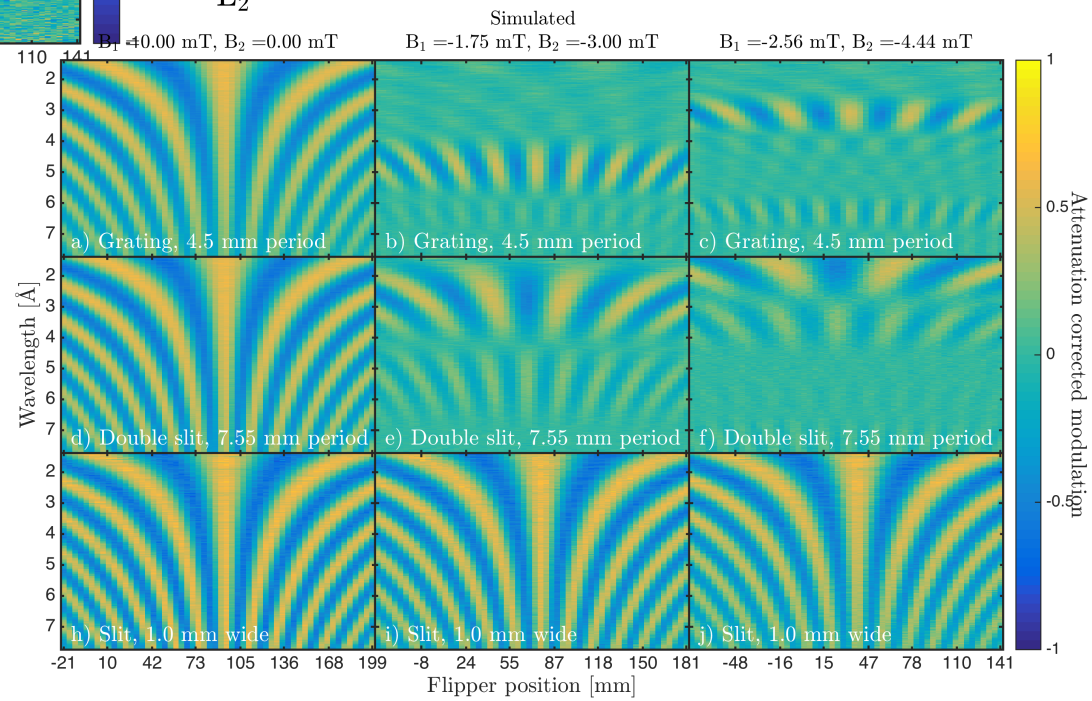
Courtesy: M. Sales et.al.



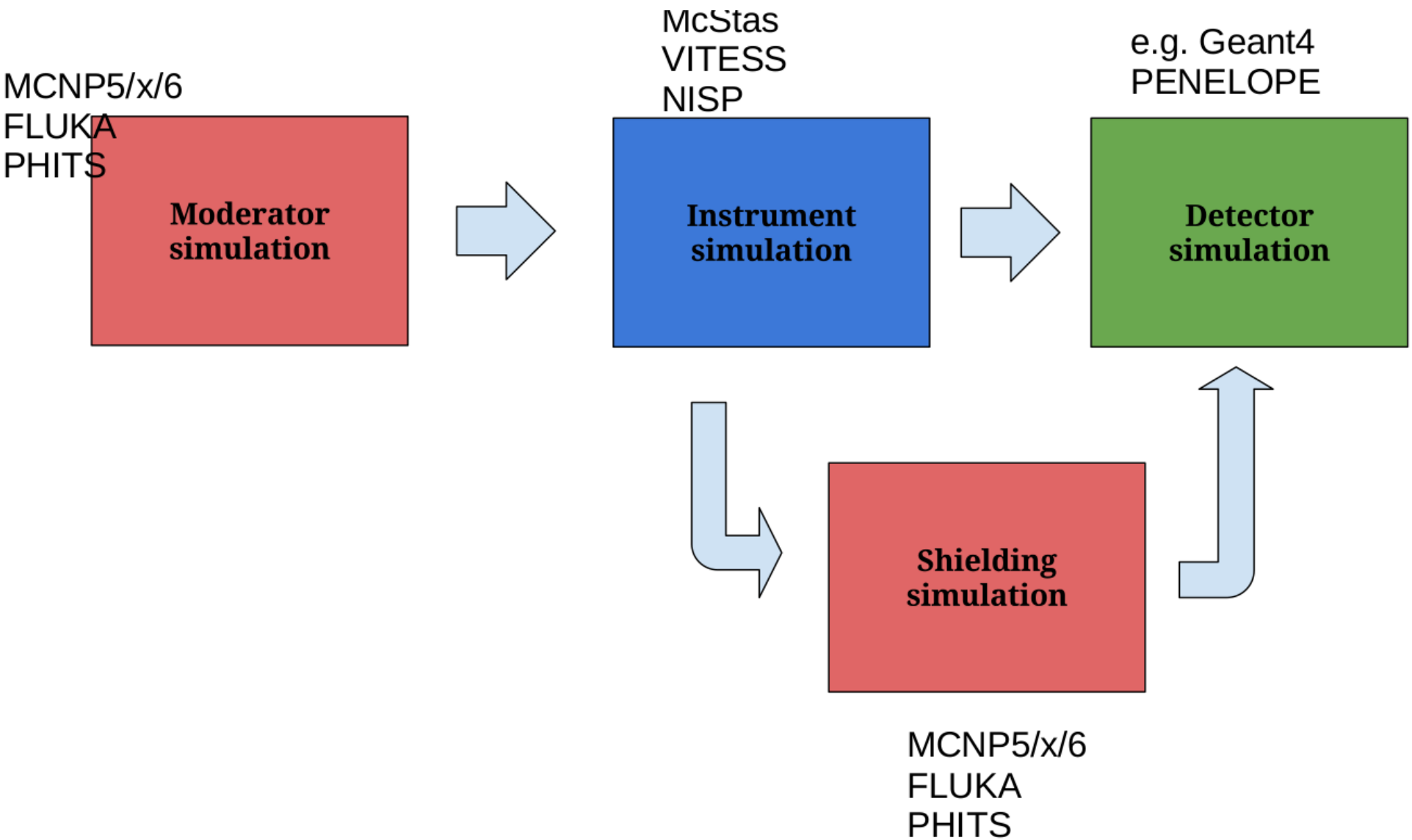
McStas example SEMSANS



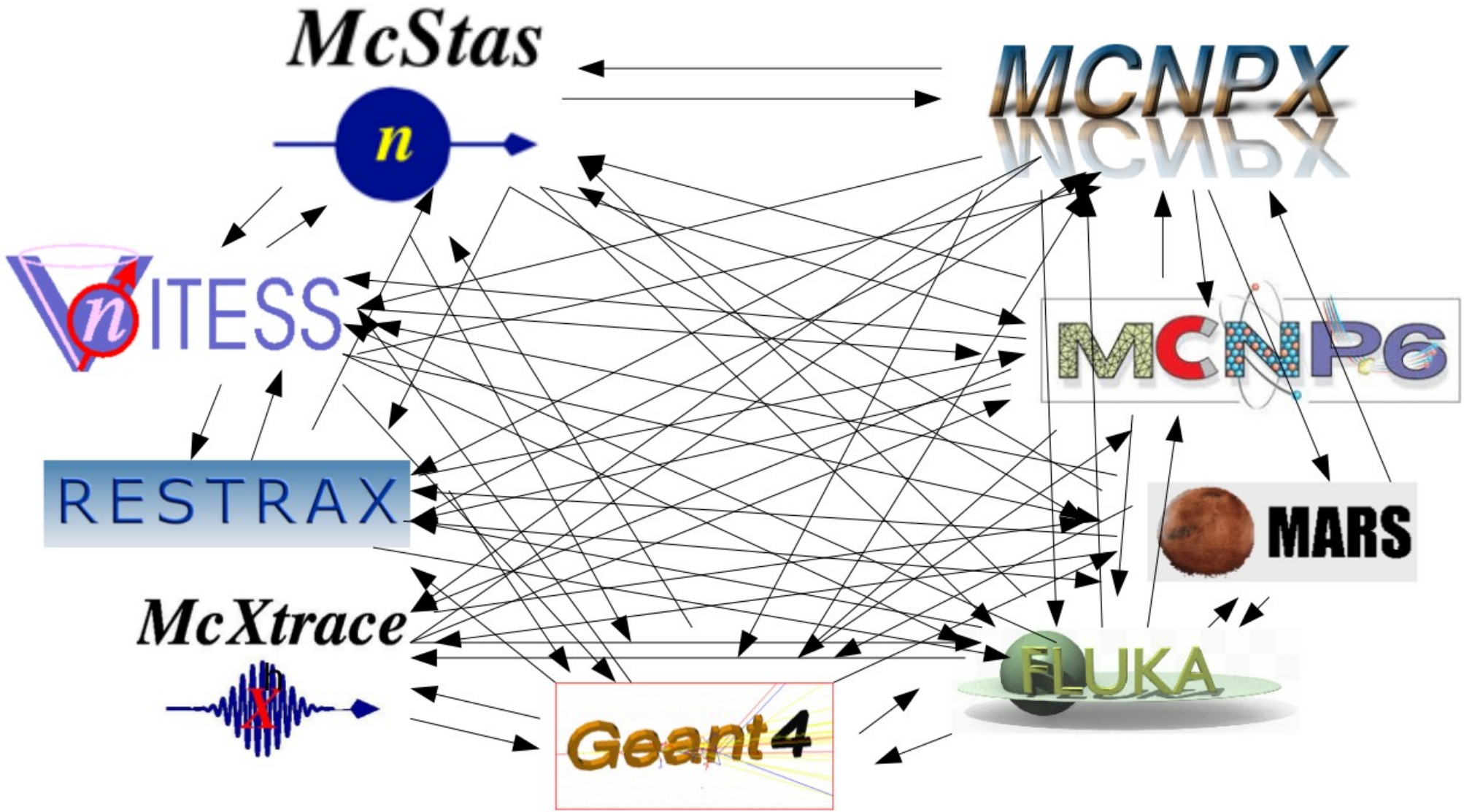




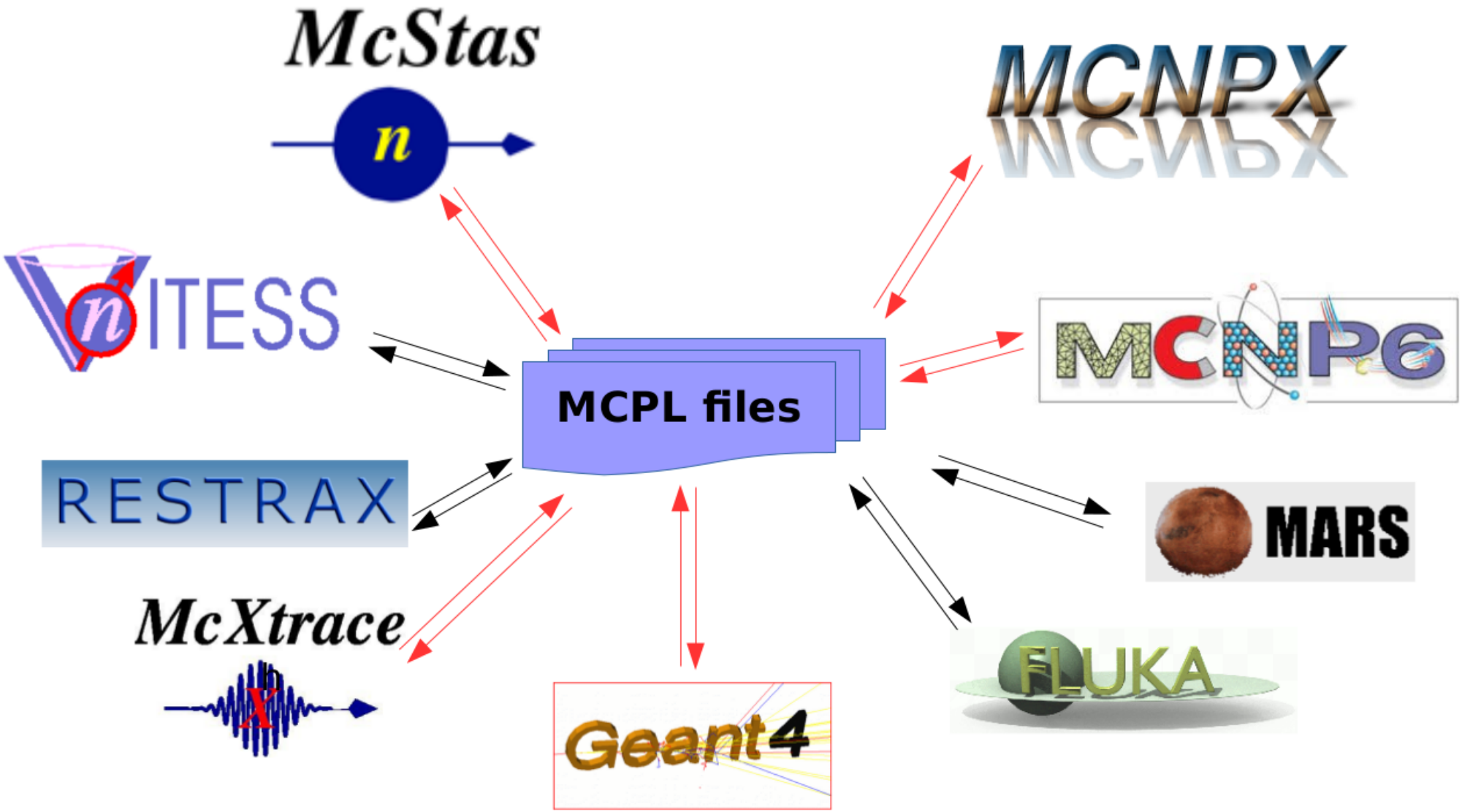
A goal: real sims with background



The problem


McStas
MCNPX
VITESS
MCNP6
RESTRAX
MARS
McXtrace
FLUKA
Geant4

The solution



McStas components on the way

Things on the way

Magnetic fields:
- Pol_FieldBox.comp
  - **Tabled fields**
- Pol_constBfield.comp
- Pol_simpleBfield.comp
  - **3D entry/exit windows**
- Pol_simpleBfield_stop.comp
- Pol_triafield.comp

Optics:
- Monochromator_pol.comp
- Pol_bender.comp
- Pol_guide_vmirror.comp
- Pol_mirror.comp
- Pol_pi_2_rotator.comp
- Transmission_polarisatorABSnT.comp
- Pol_bender_tapering.comp
- Pol_McRadia.comp
  - **Dynamic coupling to RADIA**

Monitors:
- Pol_monitor.comp
- MeanPolLambda_monitor.comp
- PolLambda_monitor.comp
- Pol_PSD_monitor.comp

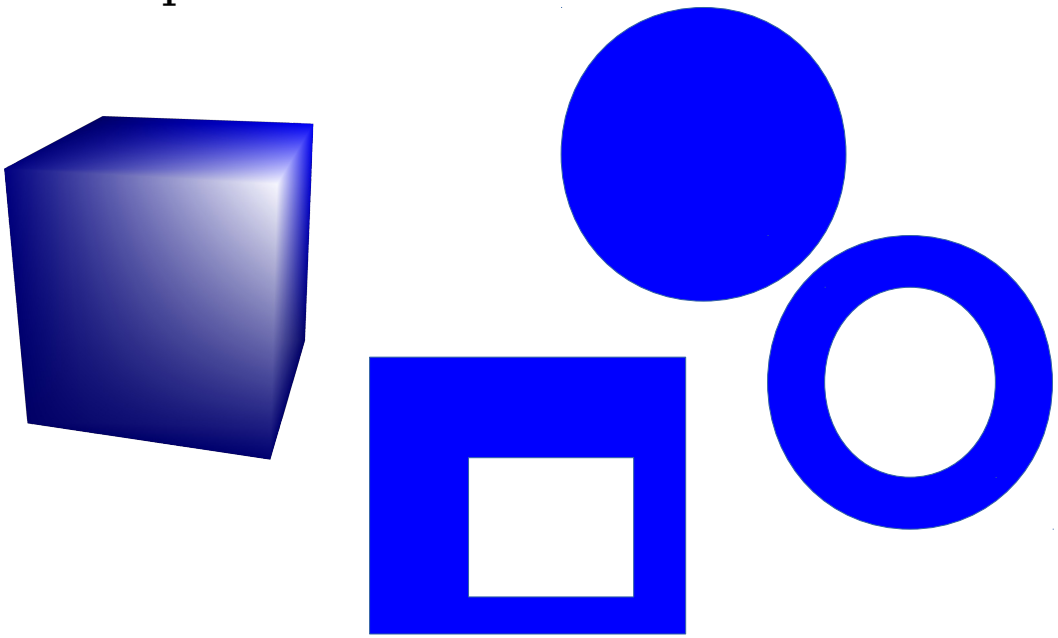
Idealized components:
- PolAnalyser_ideal.comp
- Set_pol.comp
- Pol_SF_ideal.comp

Contrib:
- Foil_flipper_magnet.comp

Sample component
- Magnetic_single_crystal.comp

McStas components on the way

Generalized Simple B-Fields: constant, functional, tabled, … but in more general shapes



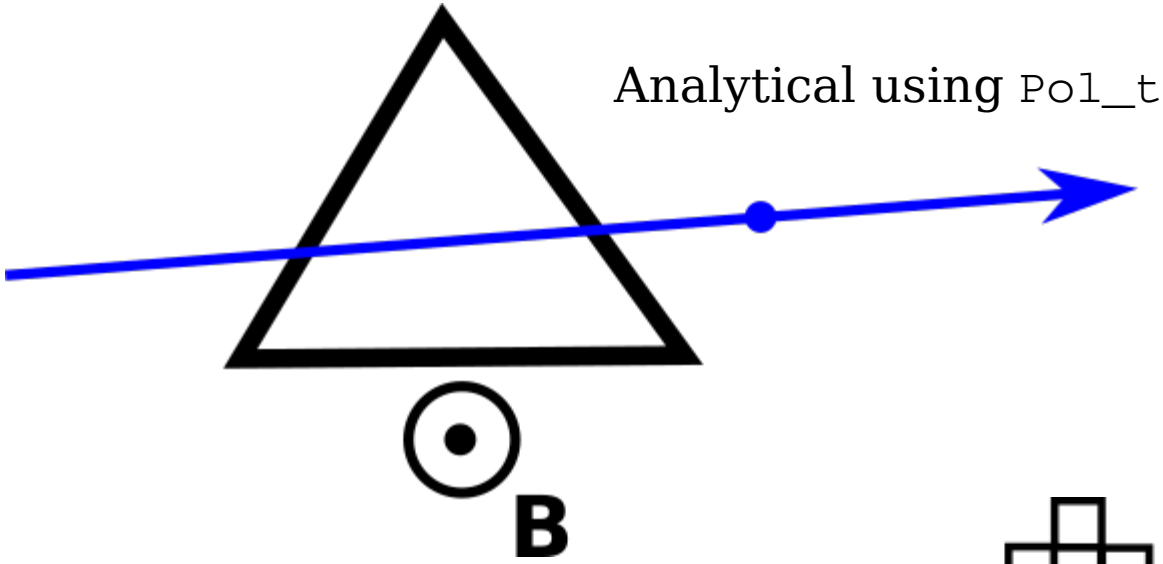
RF-flipper

He3-objects

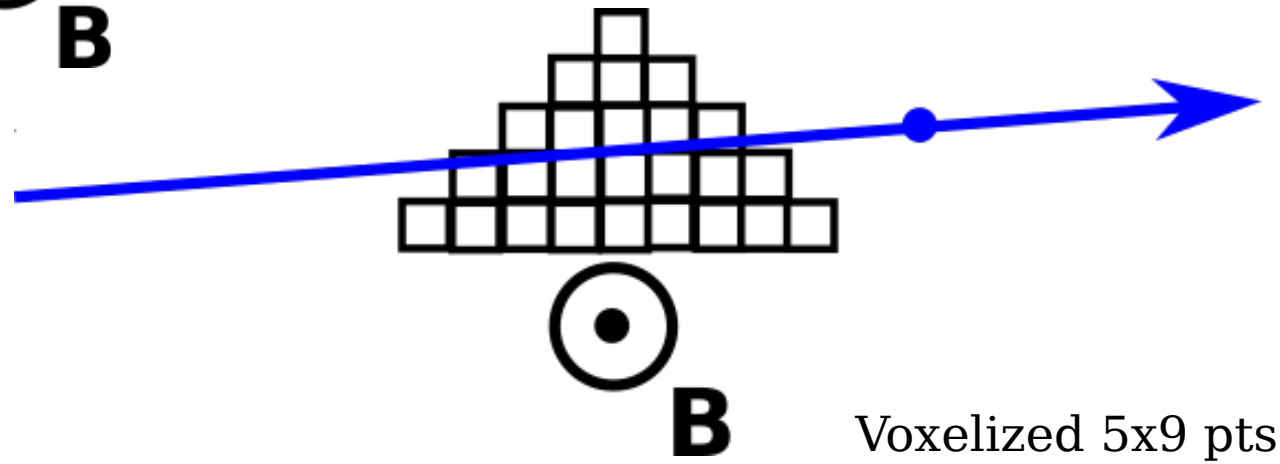
McStas components on the way

McRadia compared with analytical field description

Requires a mathematica license.



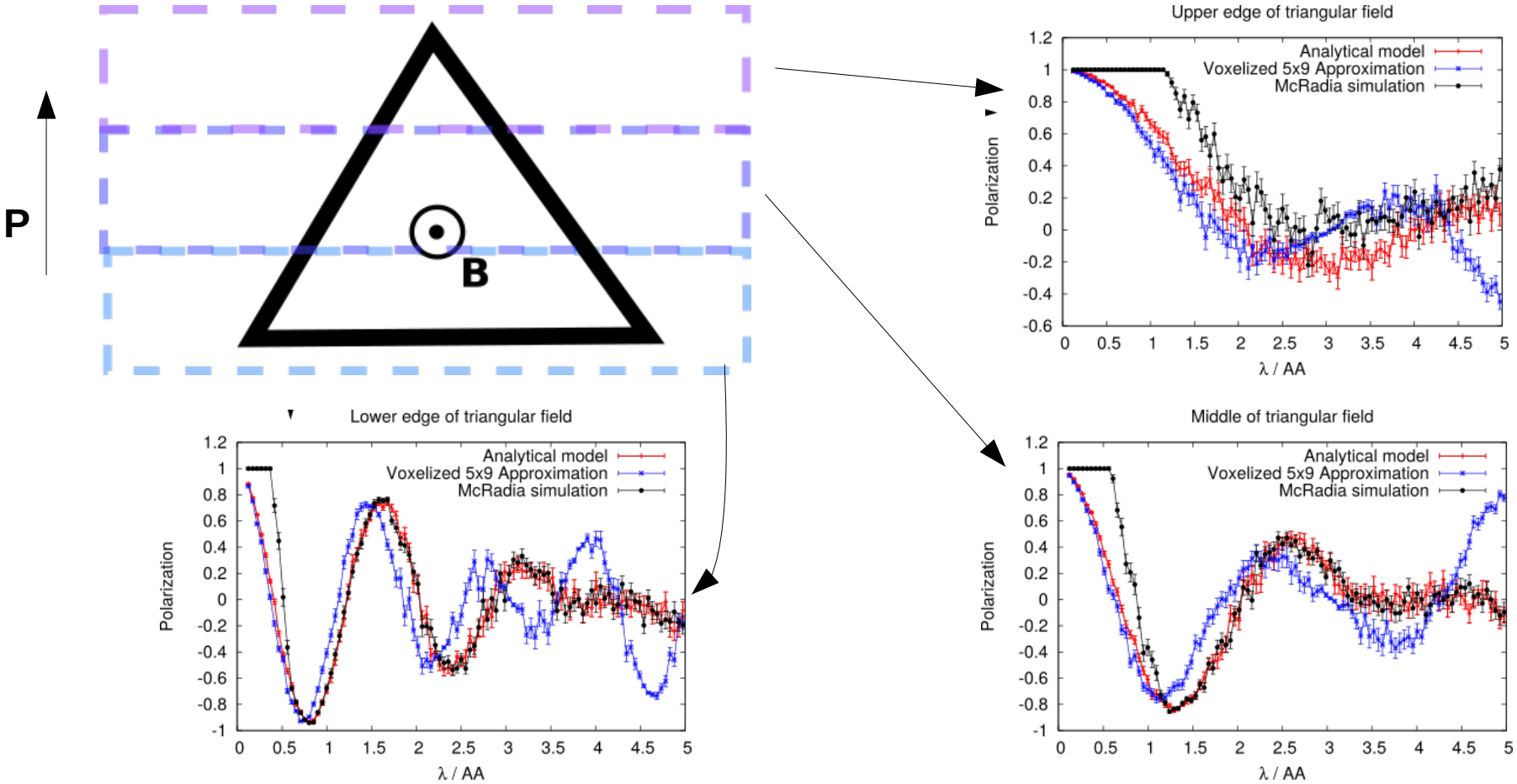
Analytical using `Pol_triafield.comp`



Voxelized 5x9 pts.

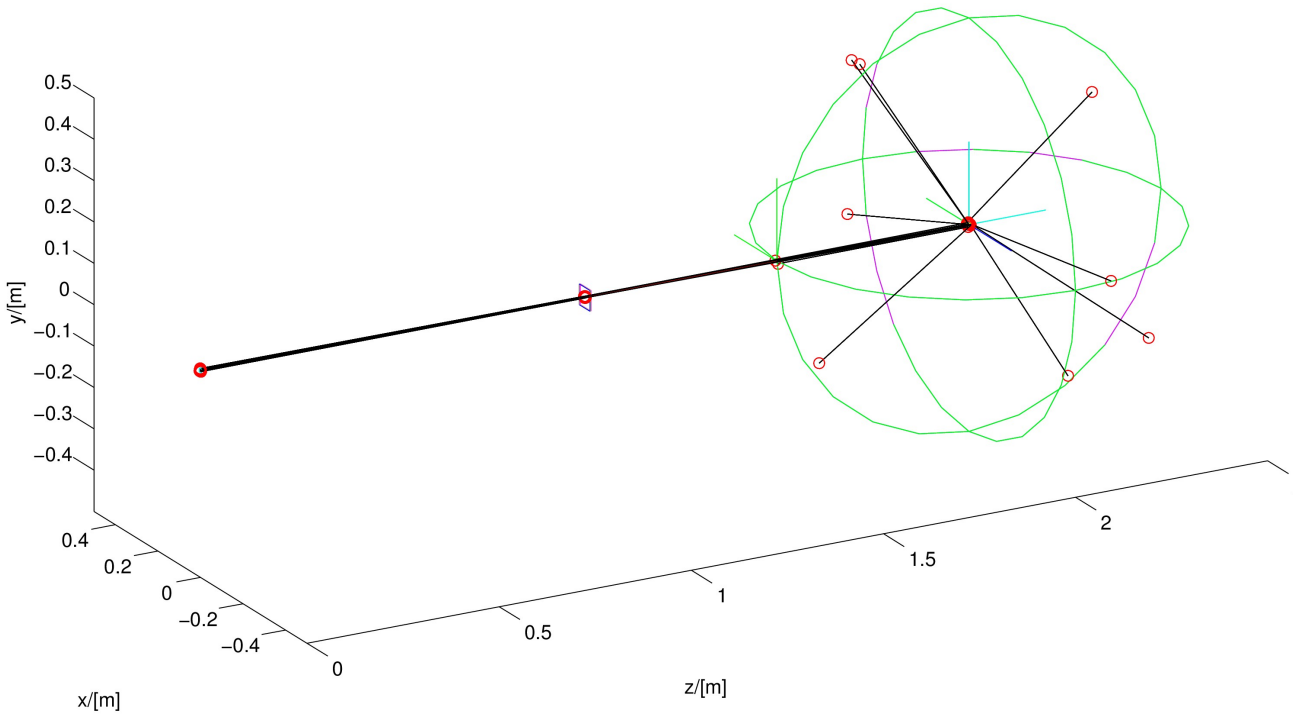
McStas components on the way

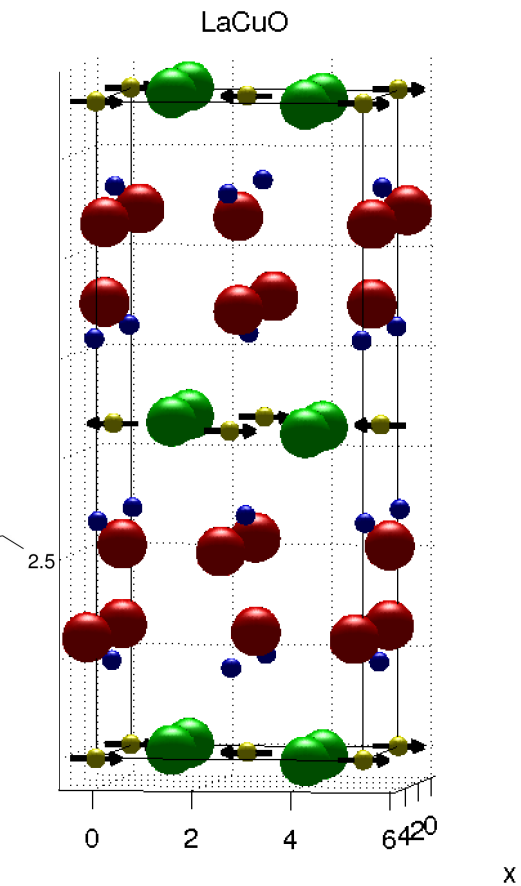
McRadia compared with other field descriptions



McStas components on the way

Magnetic single crystal

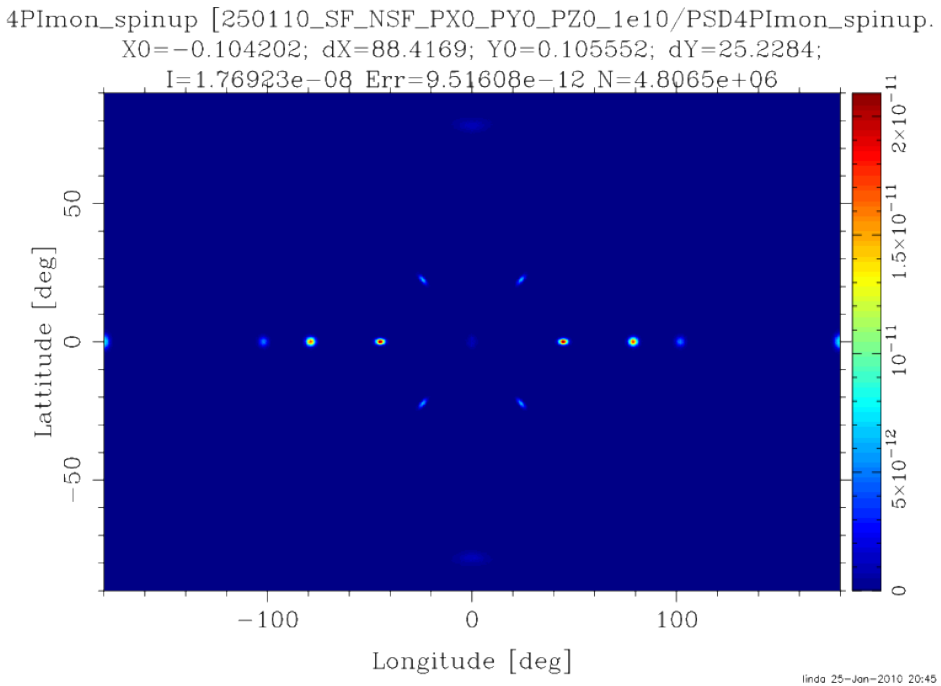


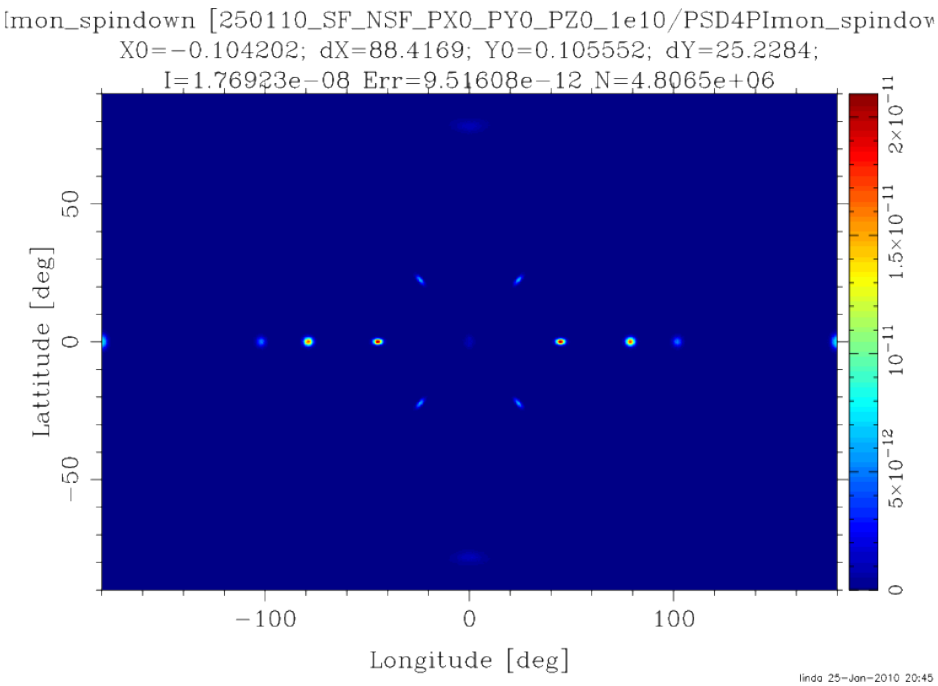


index	iontype	x	y	z	b_{coh}[fm]	g_S	S_x	S_y	S_z	g_L	L_x	L_y	L_z
1	Cu2+	0.5	0.5	0	7.718	2	0	-0.5	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

McStas components on the way

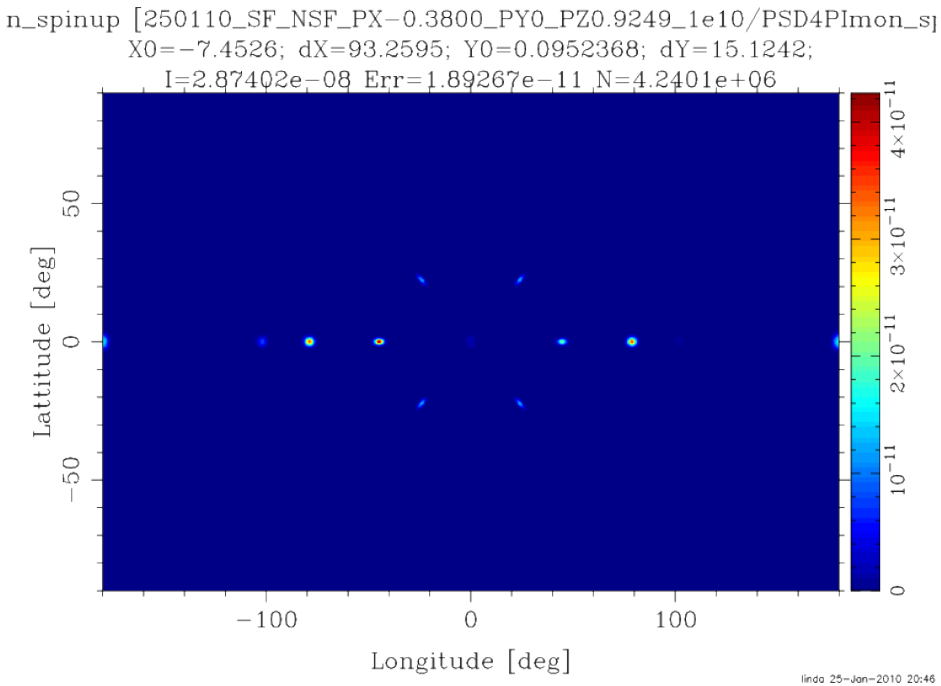
Magnetic single crystal – Unpolarized beam

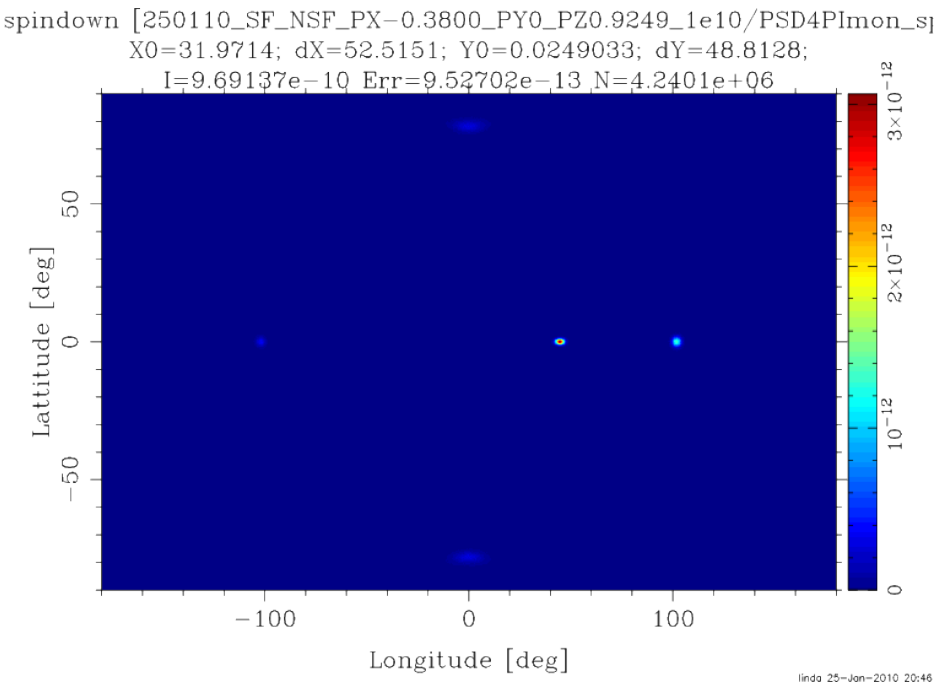




McStas components on the way

Magnetic single crystal – Polarized beam





Magnetic single crystal

The magnetic scattering cross-section for a sample with localised spin+orbital angular moment $g\mathbf{J} = (g_S + g_L)\mathbf{J} = 2\mathbf{S} + \mathbf{L}$ is:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega_\mathrm{f}\mathrm{dE_f}} = \frac{k_f}{k_i}\sum_{i,f} P(\lambda_i)\left|\langle\lambda_f \mid \sum_j e^{i\mathbf{Q}\cdot\mathbf{d}_j}\, U_j^{\sigma_i\sigma_f} \mid \lambda_i\rangle\right|^2 \delta(\hbar\omega + E_i - E_f)$$

where $\mid \lambda_i\rangle$ and $\langle\lambda_f \mid$ are the initial and final states of the sample with energies E_i and E_f respectively, $P(\lambda_i)$ is the distribution of initial states and

$$U_j^{\sigma_i\sigma_f} = \langle\sigma_f \mid b_j - m_j\mathbf{J}_{\perp j}\cdot\boldsymbol{\sigma} \mid \sigma_i\rangle$$

where $\mid \sigma_i\rangle$ and $\langle\sigma_f \mid$ are the initial and final spin states of the neutron, and $\boldsymbol{\sigma}$ are the Pauli spin matrices working on the neutron state.

From: G. Shirane et.al. ,"Neutron Scattering with Triple-Axis Spectrometer", *Cambridge Univ. Press*, 2002

Magnetic single crystal

If $\mathbf{P} = P(\xi, \eta, \zeta) = P\hat{\boldsymbol{\zeta}}$.Thus, the matrix elements of $U^{\sigma_i \sigma_f}$ can now be written

$$
\begin{aligned}
U^{++} &= b - mJ_{\perp\zeta} \\
U^{--} &= b + mJ_{\perp\zeta} \\
U^{+-} &= -m\left(J_{\perp\xi} + iJ_{\perp\eta}\right) \\
U^{+-} &= -m\left(J_{\perp\xi} - iJ_{\perp\eta}\right)
\end{aligned}
$$

where $m = \frac{r_0\gamma}{2} g f(\mathbf{Q})$ with r_0 the classical electron radius, $\gamma = 1.913$, g the Landé splitting factor and $f(\mathbf{Q})$ the magnetic form factor of a particular ion in the sample.

Acknowledgements

DTU
- Peter Willendrup
- Jacob Garde

ILL
- Emmanuel Farhi

Copenhagen University
- Kim Lefmann
- Mads Bertelsen

PSI
- Emmanouela Rantsiou
- Uwe Filges

Kristian Nielsen
Kurt Clausen
Peter Christianse
Klaus Liutenant

And all other contributors!

Outlook/Discussion items

- How can we interact better?
  - Better support for the community?
  - Code-sharing?