

McStas Introduction



2021 Virtual ISIS McStas School, April 13th-15th 2021



Agenda

- A (very) brief introduction to neutrons, Monte Carlo & raytracing
- Components of neutron instruments
- How McStas works under the hood
- Components and instruments
- A demo



McStas
 n →

McStas: Neutrons, Monte Carlo & ray-tracing

Neutron

The **quark** structure of the neutron. (The color assignment of individual quarks is not important; what is important is that all three colors are present.)

| | |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Classification | Baryon |
| Composition | 1 up quark, 2 down quarks |
| Statistics | Fermionic |
| Interactions | Gravity, Weak, Strong, Electromagnetic |
| Symbol | $n^- n^0 N^0$ |
| Antiparticle | Antineutron |
| Theorized | Ernest Rutherford ^{[1][2]} (1920) |
| Discovered | James Chadwick ^[1] (1932) |
| Mass | $1.674\,927\,351(74) \times 10^{-27} \text{ kg}$ $939.565\,378(21) \text{ MeV}/c^2$ ^[3] $1.008\,664\,916\,00(43) \text{ u}$ ^[3] |
| Mean lifetime | $881.5(15) \text{ s}$ (free) |
| Electric charge | 0 e |
| Electric dipole moment | $< 2.9 \times 10^{-26} \text{ e-cm}$ |
| Electric polarizability | $1.16(15) \times 10^{-3} \text{ fm}^3$ |

| | |
|------------------|----------------------------------------|
| Life time: | $\tau_{1/2} = 890s$ |
| Mass: | $m = 1.675 \times 10^{-27} \text{ kg}$ |
| Charge: | $Q = 0$ |
| Spin: | $s = \hbar/2$ |
| Magnetic moment: | $\mu/\mu_n = -1.913$ |

$$E = \frac{1}{2}mv^2 = \frac{\hbar^2k^2}{2m} \quad \lambda = 2\pi/k$$

$$E = 81.81 \cdot \lambda^{-2} = 2.07 \cdot k^2 = 5.23 \cdot v^2$$

Subatomic particle discovered by Sir James Chadwick in 1932

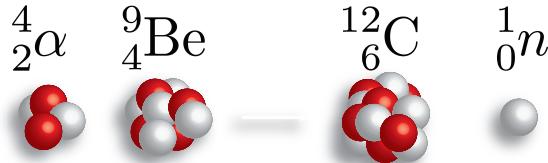


| | Energy | Wavelength | n-Wavevector | Velocity | Frequency |
|-------------------|----------------------------------------------|------------------------------------------------------------------|----------------------------------------------------------|--------------------------------------------------|------------------------------------------------------|
| cold neutrons: | $E = 1 \text{ meV}$ $E = 5 \text{ meV}$ | $\lambda = 9.0446 \text{ \AA}$ $\lambda = 4.0449 \text{ \AA}$ | $k = 0.6947 \text{ 1/\AA}$ $k = 1.5534 \text{ 1/\AA}$ | $v = 437 \text{ m/s}$ $v = 978 \text{ m/s}$ | $v = 0.2418 \text{ THz}$ $v = 1.2090 \text{ THz}$ |
| thermal neutrons: | $E = 25 \text{ meV}$ $E = 50 \text{ meV}$ | $\lambda = 1.8089 \text{ \AA}$ $\lambda = 1.2791 \text{ \AA}$ | $k = 3.4734 \text{ 1/\AA}$ $k = 4.9122 \text{ 1/\AA}$ | $v = 2187 \text{ m/s}$ $v = 3093 \text{ m/s}$ | $v = 6.045 \text{ THz}$ $v = 12.090 \text{ THz}$ |

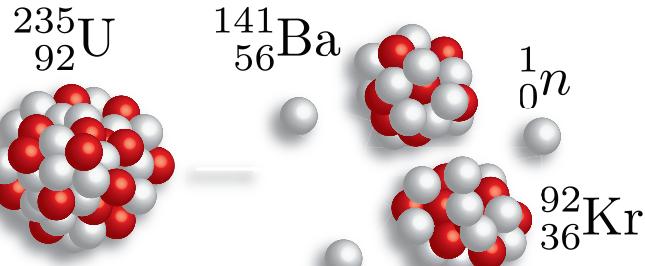


Neutron facilities

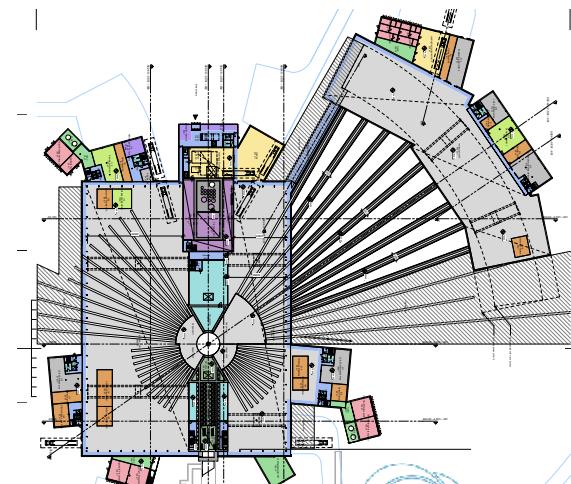
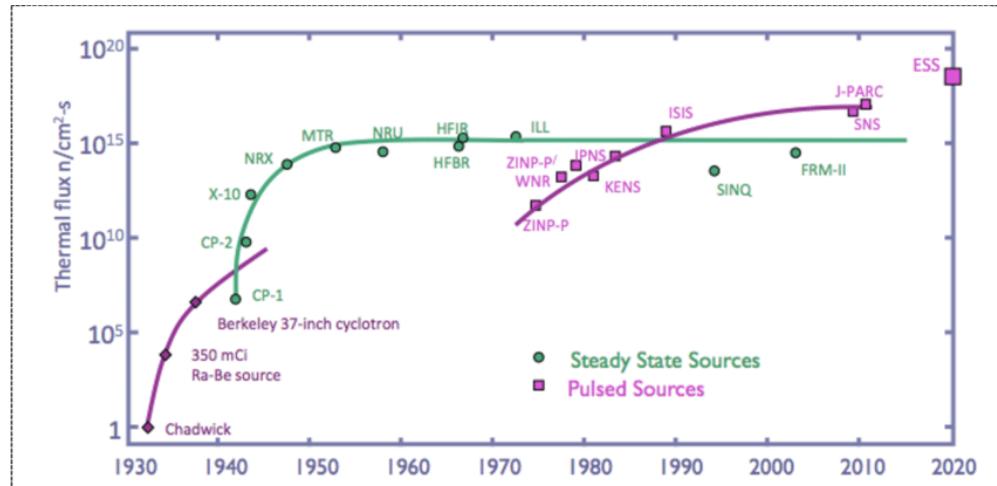
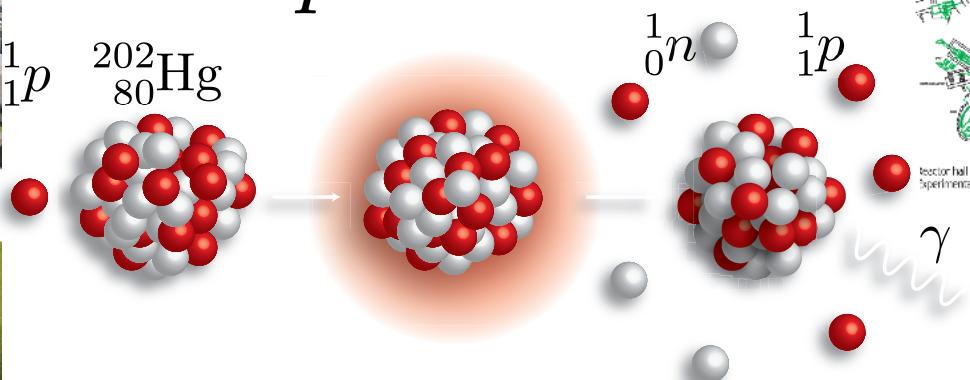
Chadwick



Fission

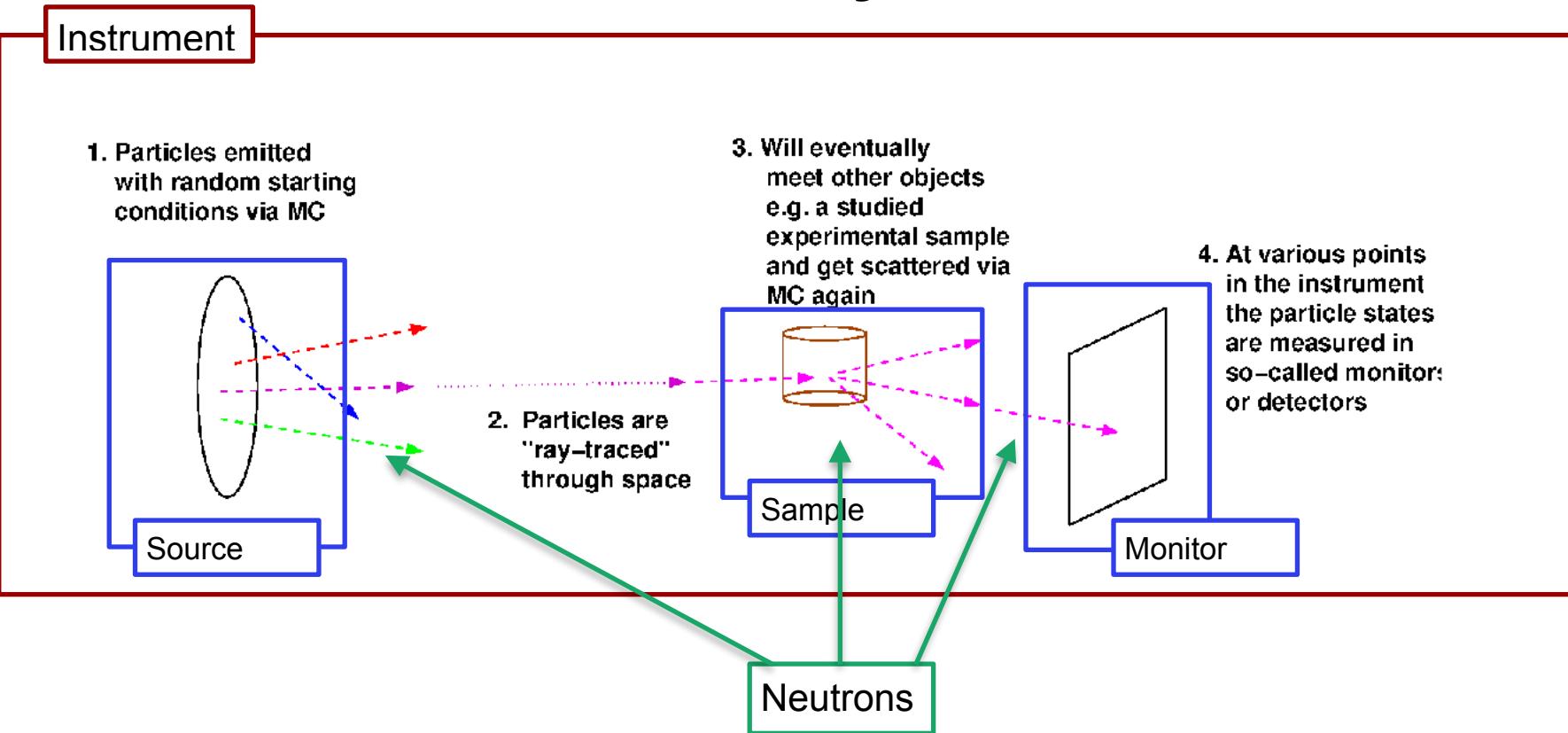


Spallation





McStas is a Monte Carlo ray-tracer





When to use Monte Carlo methods

Dimensionality of phase space must be large ($d > 5$)

Overall complexity is beyond reasonable analytical methods

Each event can be computed easily and independently MC is
the '*lazy guy*' method – think microscopic

Examples:

Estimate π from a circle/square (“*Buffon needle*”)

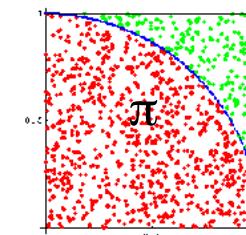
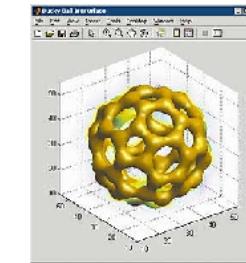
Area under/inside a curve/volume (integration)

Molecular Dynamics

spin-system phase transitions (*Ising* model)

nuclear reactions

ray-tracing (light, particles)



Number of points for which
 $\{ x^2+y^2 \leq 1, (x,y) \in [0,1] \}$
 Ratio circle/square $\rightarrow \pi/4$



How to implement Monte Carlo methods ?

Good random generator:
from thermal electronic noise (hardware)

or quasi-random generators => *quasi-Monte-Carlo*

We encounter a probability $0 < p < 1$.

Crude Monte-Carlo (yes/no choice):

We shoot n events $\xi \in [0,1]$

We keep events that satisfy $\xi < p$

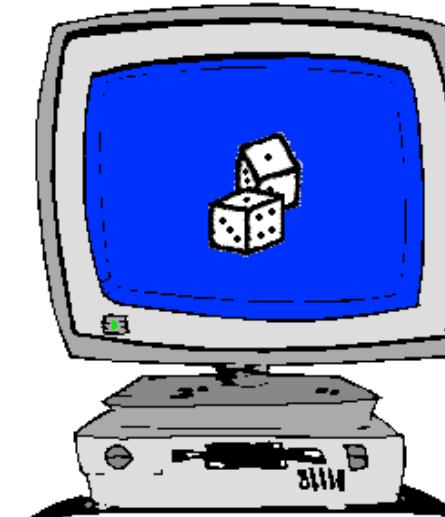
np events low statistics

Importance sampling (fuzzy choice – event weighting):

Keep n events, no more random number...

But associate a **weight** p to each of them (we set $\xi = p$)

Retain statistical accuracy ($1/\sqrt{n}$)





Origin of Monte Carlo methods

First application using computers:

Metropolis, Ulam and Von Neumann at Los Alamos, 1943

Neutron Scattering and Absorption in U and Pu, Origin of MCNP

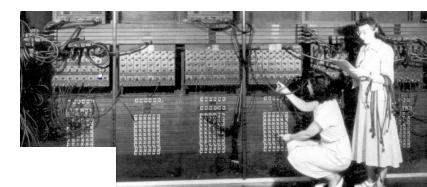
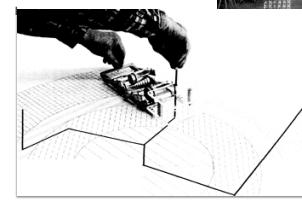


Name:

Monte Carlo casino, a random generator (Ulam's father played poker)



Bradbury Science Museum, LANL

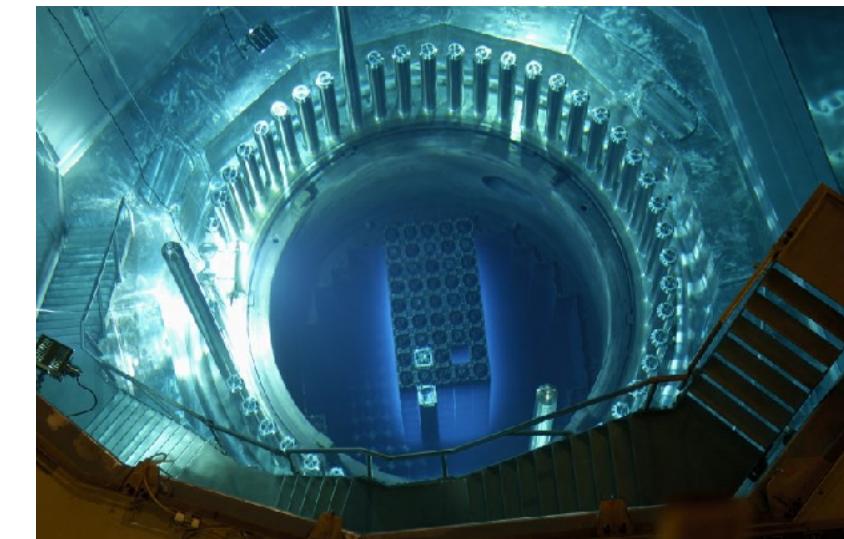
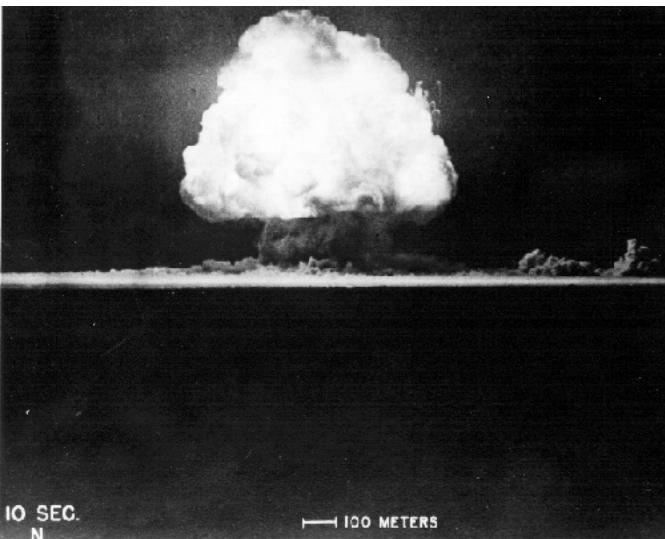


ENIAC
US Army BRL → UPENN



Monte Carlo techniques

- 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 MCNP6 that features numerous (even exotic) 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
- Does not to date handle crystalline / ordered material and coherent scattering of neutrons due to the focus on high energies





Examples of Monte Carlo programs

Each time physics takes place (scattering, absorption, ...) random choices are made.

Light ray-tracing: PoV-RAY and others ...

*Nuclear reactor simulations (neutron transport):
MCNP, Tripoli, GEANT4, FLUKA*

Neutron Ray-Tracing propagation:

McStas <www.mcstas.org>, Vitess, Restrax, NISP, IDEAS, McVine

Neutrons are described as $(\mathbf{r}, \mathbf{v}, \mathbf{s}, t)$, and are transported along instrument models.

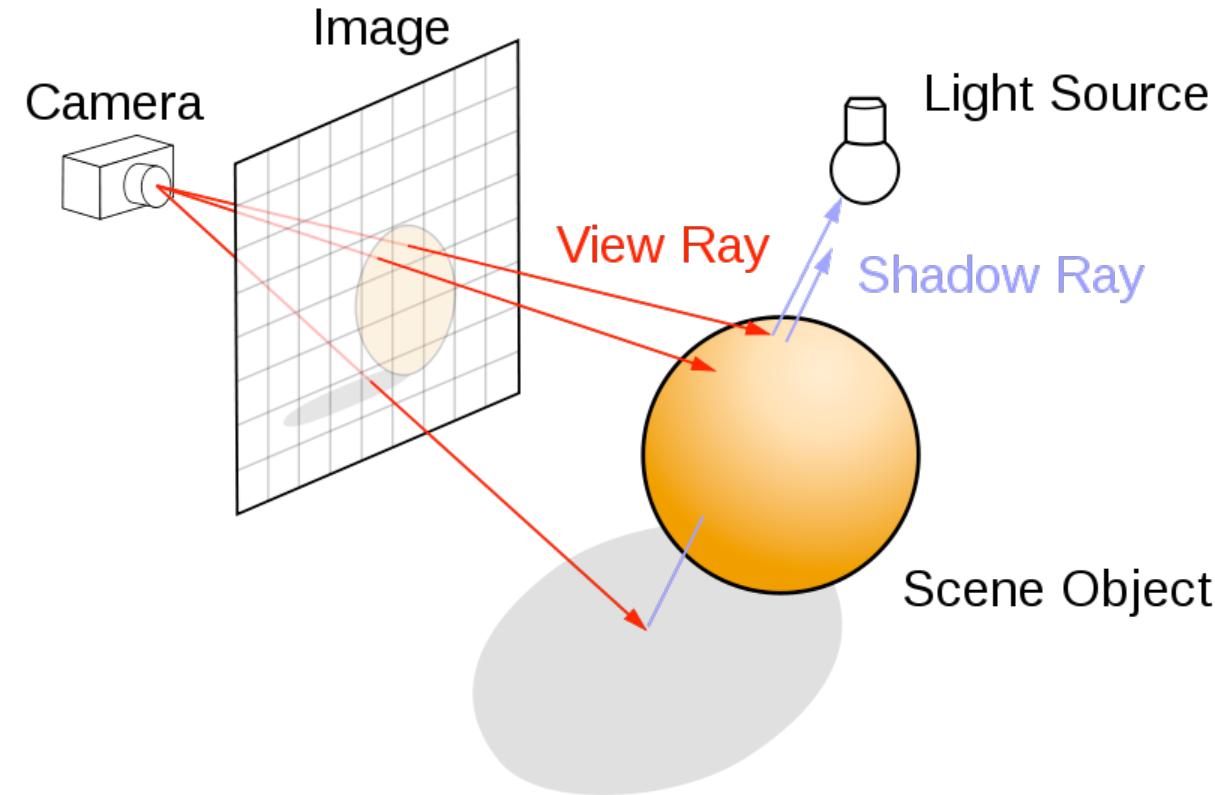
Propagation simply uses Newton rules, incl. gravitation.

X-ray tracing

Shadow, McXtrace, RAY, ...



Ray-tracing methods

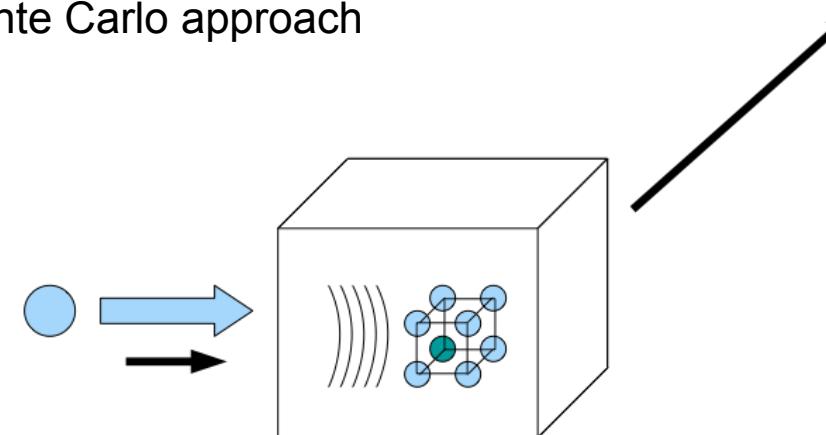


- When neutrons move in “free space”, we use ray-tracing - but in most cases in direction source -> detector
- Of course parabolas rather than straight lines are used to implement gravity



Elements of Monte-Carlo raytracing

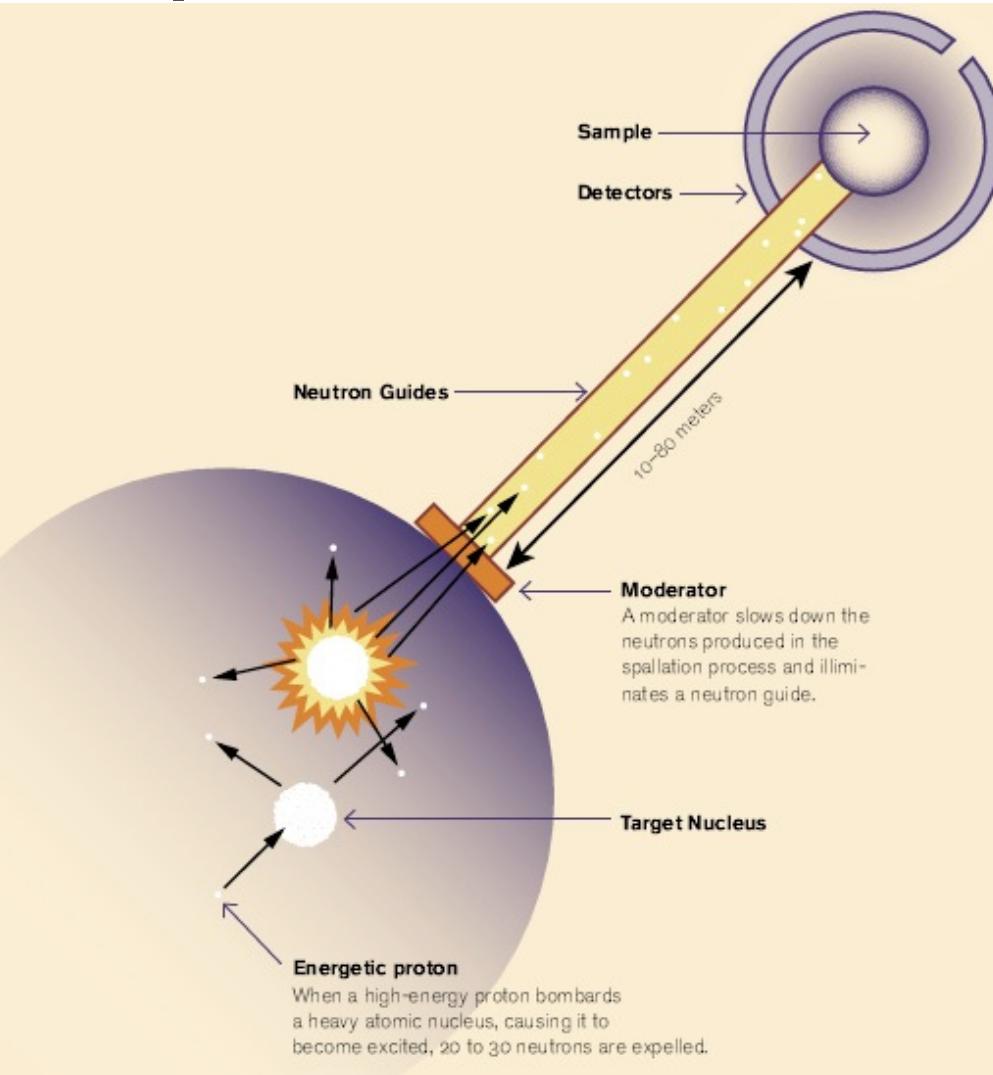
- Instrument Monte Carlo methods implement coherent scattering effects
- Uses deterministic propagation where this can be done
- Uses Monte Carlo sampling of “complicated” distributions and stochastic processes and multiple outcomes with known probabilities are involved
 - I.e. inside scattering matter
- Uses the particle-wave duality of the neutron to switch back and forward between deterministic ray tracing and Monte Carlo approach



- Result: A realistic and efficient transport of neutrons in the thermal and cold range



Components of neutron instruments

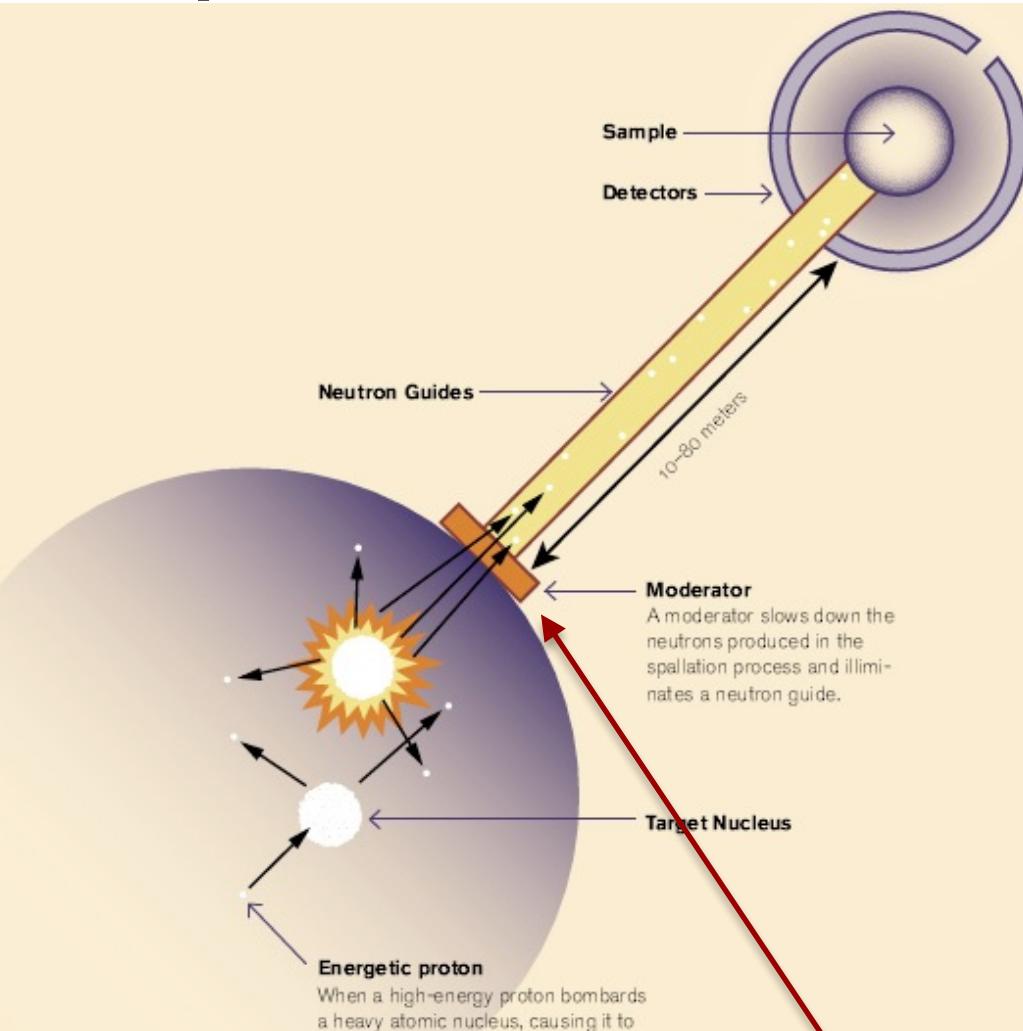


DTU
European Spallation Source

Components of neutron instruments



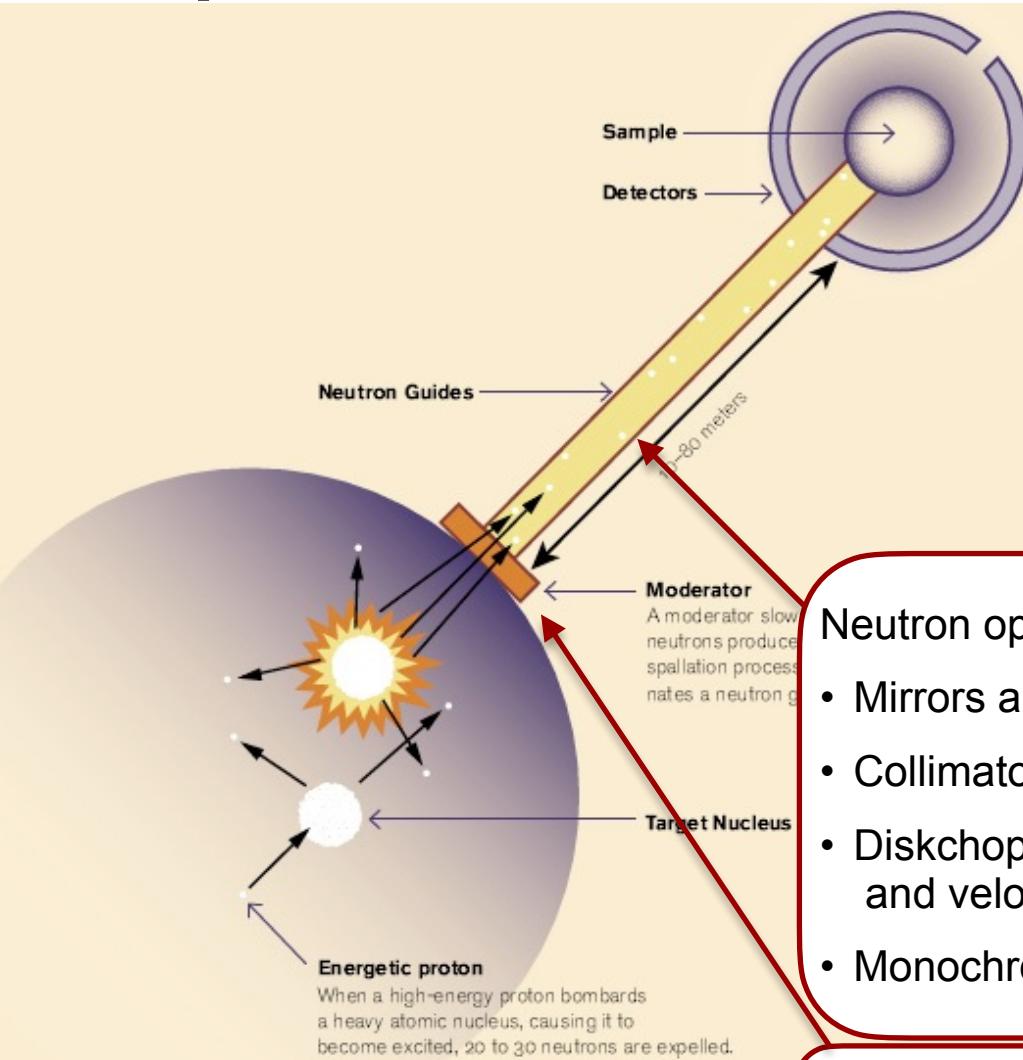
2021 Virtual
ISIS
McStas
School



In McStas the moderator is the “source”

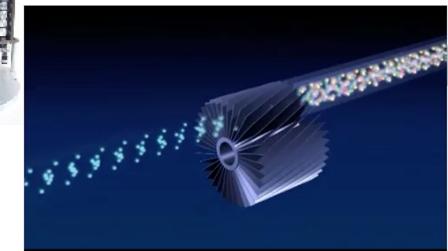
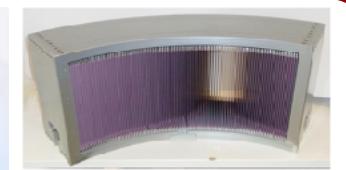


Components of neutron instruments



Neutron optics include things like:

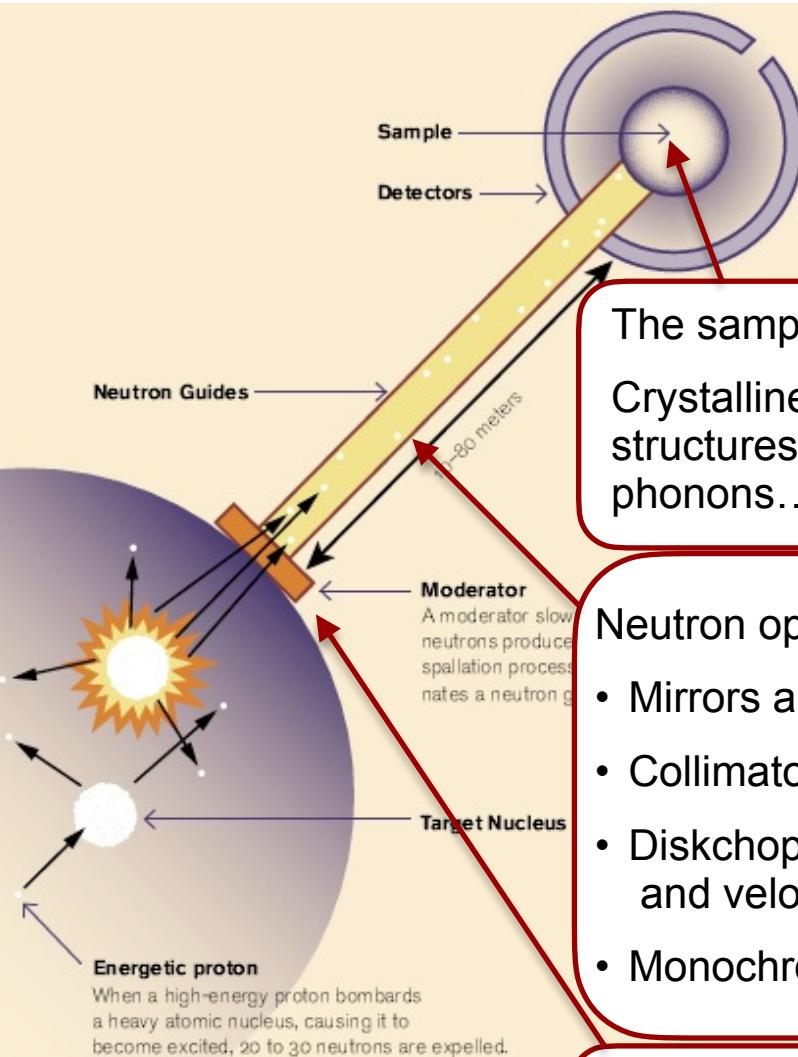
- Mirrors and guides
- Collimators and slits
- Diskchoppers, Fermi choppers and velocity selectors
- Monochromators/Analysers



In McStas the moderator is the “source”

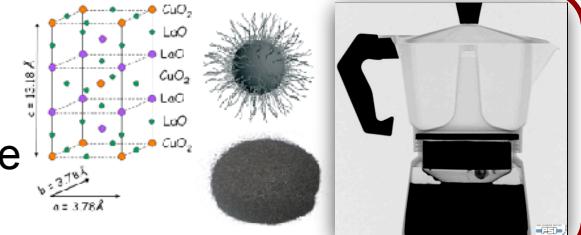


Components of neutron instruments



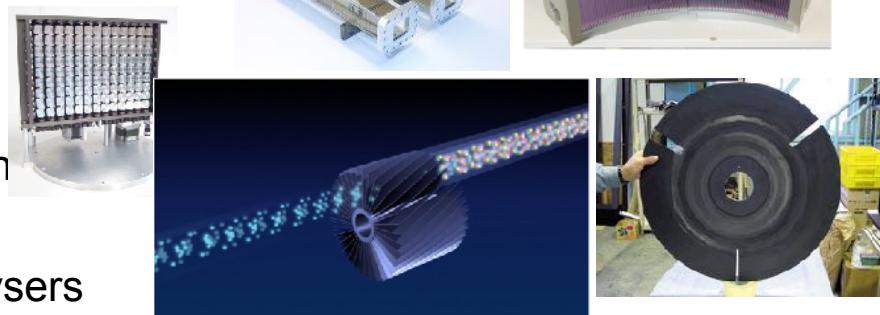
The sample:

Crystalline, powders, liquids, micelles, structures to image, inelastic features like phonons...



Neutron optics include things like:

- Mirrors and guides
 - Collimators and slits
 - Diskchoppers, Fermi ch and velocity selectors
 - Monochromators/Analysers



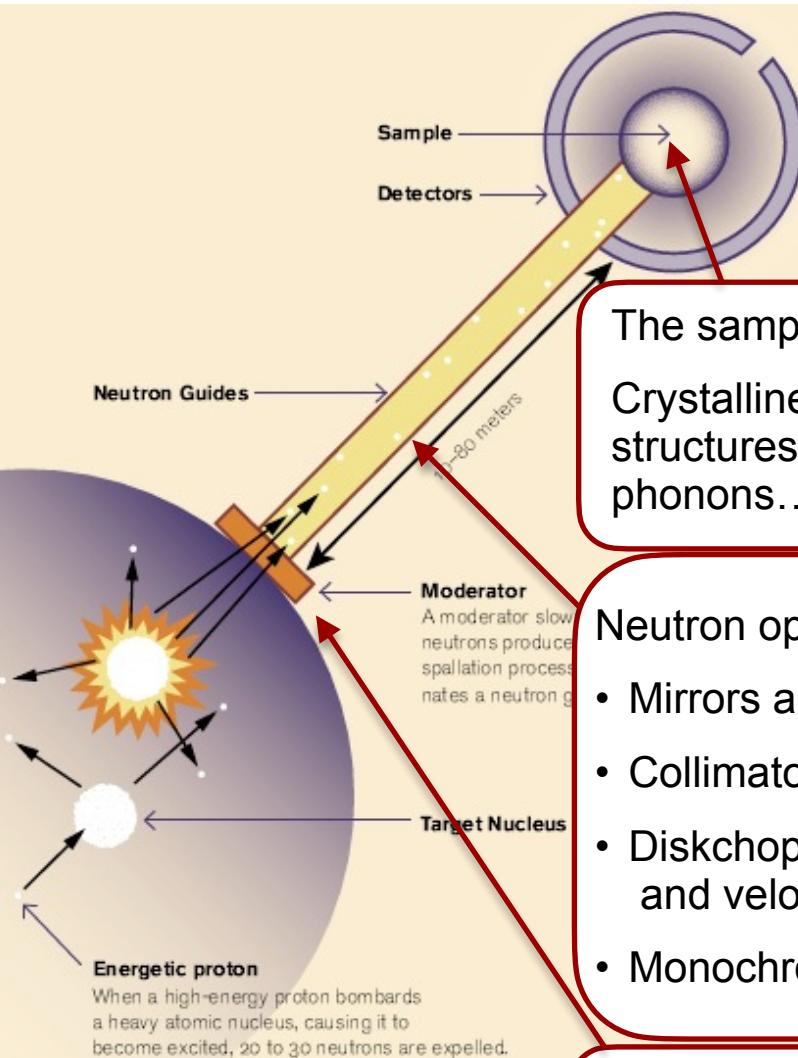
In McStas the moderator is the “source”



McStas

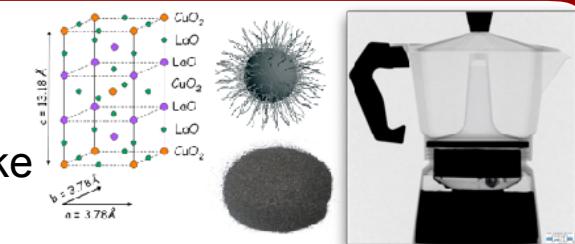
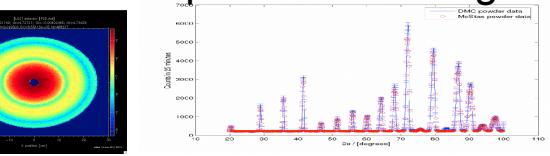


Components of neutron instruments



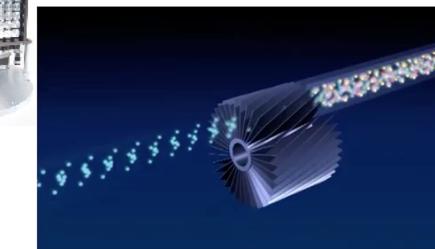
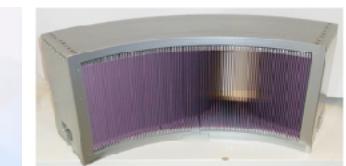
The sample:

Crystalline, powders, liquids, micelles, structures to image, inelastic features like phonons...



Neutron optics include things like:

- Mirrors and guides
- Collimators and slits
- Diskchoppers, Fermi choppers and velocity selectors
- Monochromators/Analysers



In McStas the moderator is the “source”



McStas Introduction

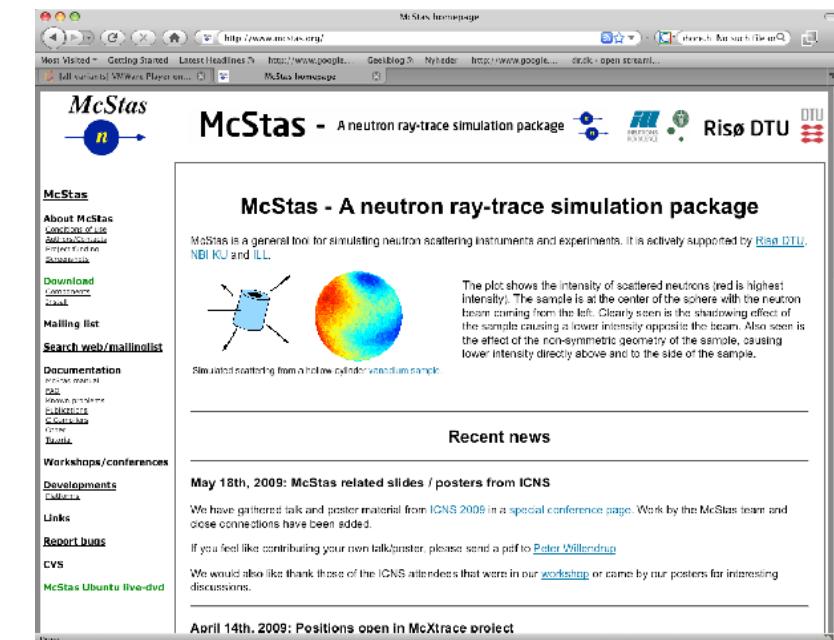
- Flexible, general simulation utility for neutron scattering experiments.
- Original design for Monte carlo Simulation of triple axis spectrometers
- Developed at DTU Physics, ILL, PSI, Uni CPH, ESS DMSC
- V. 1.0 by K Nielsen & K Lefmann (1998) RISØ
- Currently 2.5+1 people full time plus students

Project website at
<http://www.mcstas.org>

mcstas-users@mcstas.org mailinglist



GNU GPL license
Open Source



The screenshot shows the McStas homepage. The left sidebar contains links for About McStas, Documentation, Workshops/conferences, and Recent news. The main content area features a title "McStas - A neutron ray-trace simulation package" and a sub-section "McStas - A neutron ray-trace simulation package". It includes a plot showing scattered neutron intensity from a sample and a link to "Simulated scattering from a hollow cylinder vanadium sample". Below this is a "Recent news" section with entries for May 18th, 2009, and April 14th, 2009.



2021 Virtual
ISIS
McStas
School

McXtrace - since jan 2009 similar for X-rays

McStas Introduction

Main Page – McXtraceWiki

Most Visited Getting Started Latest Headlines http://www.google... Geekblog Nyheder http://www.google... dr.dk open streami... Log in / create account

article discussion edit history

Main Page

McXtrace

[edit]

McXtrace - Monte Carlo Xray ray-tracing is a joint venture by

Risø DTU DTU ESRF JJ X-RAY Danish Science Design Engineering Production

Funding from NABIIT, DSF and the above parties.

Our code will be based on technology from McStas

For information on our progress, please subscribe to our user mailinglist.
mailto:webmaster@mcxtrace.org

This page was last modified 13:15, 25 February 2009. This page has been accessed 2,049 times. Privacy policy About McXtraceWiki Disclaimers Powered By MediaWiki

- Synergy, knowledge transfer, shared infrastructure



Used in many places

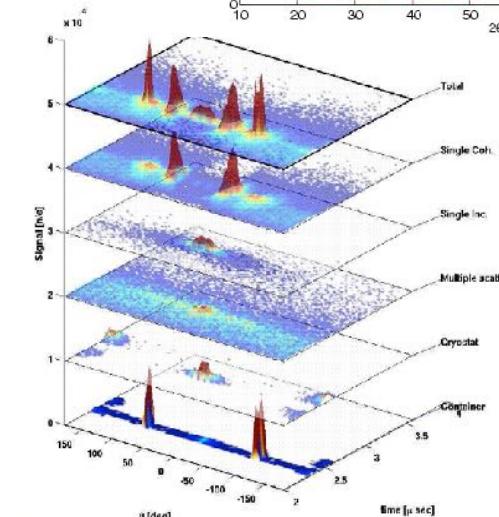
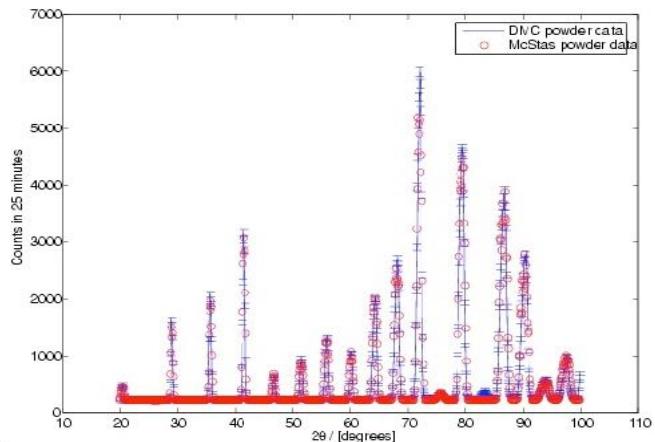
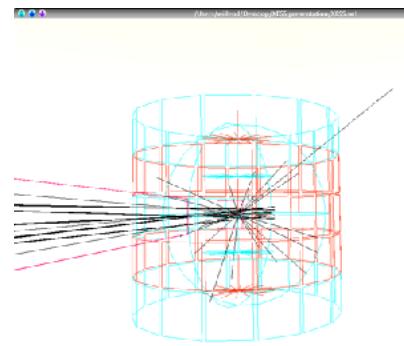
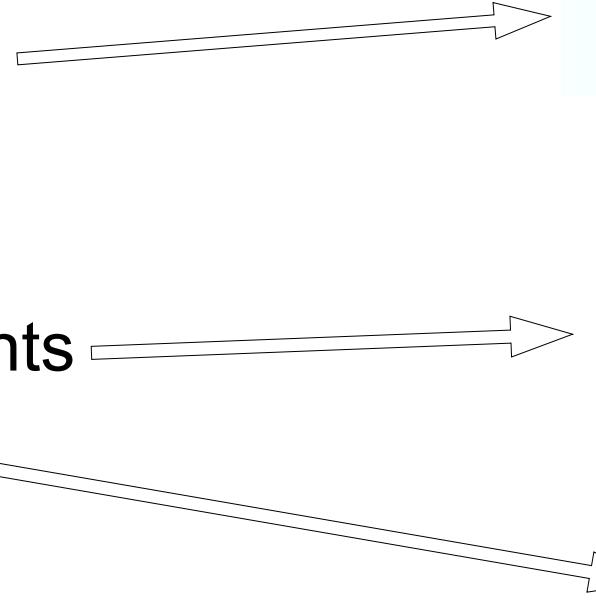
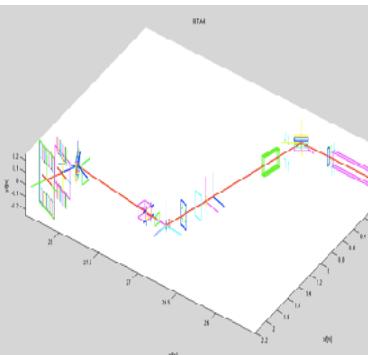
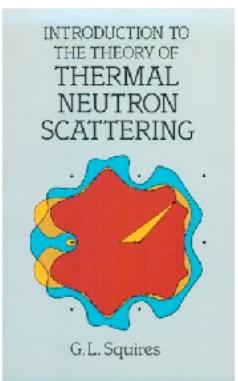




2021 Virtual
ISIS
McStas
School

What is McStas used for?

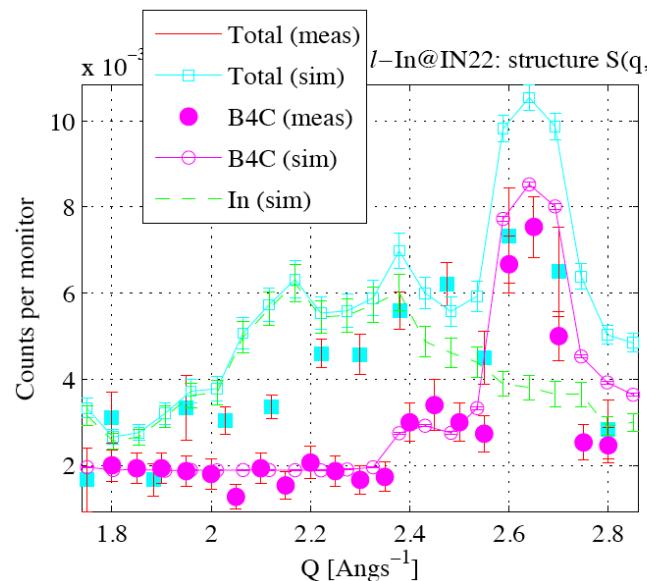
- Instrumentation
- Planning
- Construction
- Virtual experiments
- Data analysis
- Teaching
(KU, DTU)



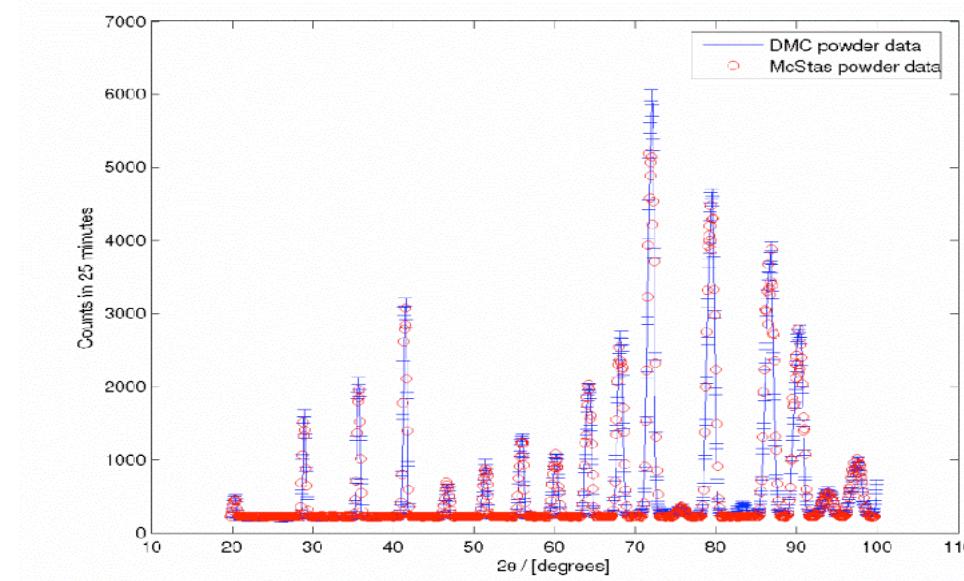
Reliability - cross comparisons



- Much effort has gone into this
- Here: simulations vs. exp. at powder diffract. DMC, PSI
- The bottom line is
- McStas agree very well with other packages (NISP, Vitess, IDEAS, RESTRAX, ...)
- Experimental line shapes are within 5%
- Absolute intensities are within 10%
- Common understanding: McStas and similar codes are reliable



E. Farhi, P. Willendrup

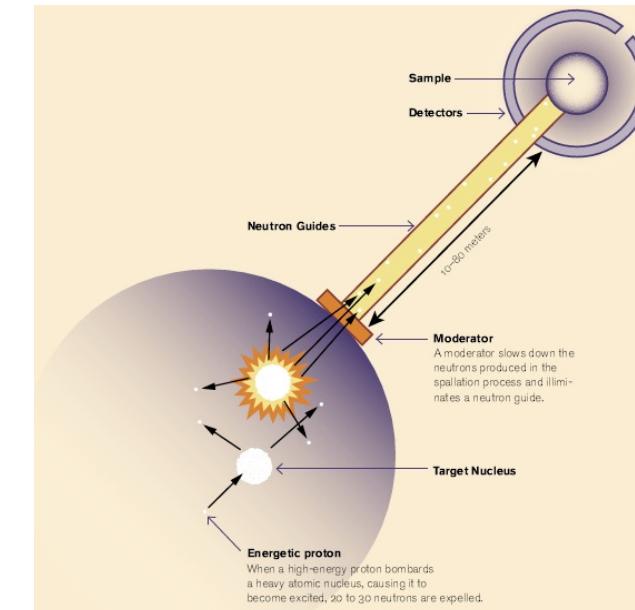


P. Willendrup et al., Physica B, 386, (2006), 1032.



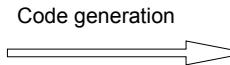
McStas overview

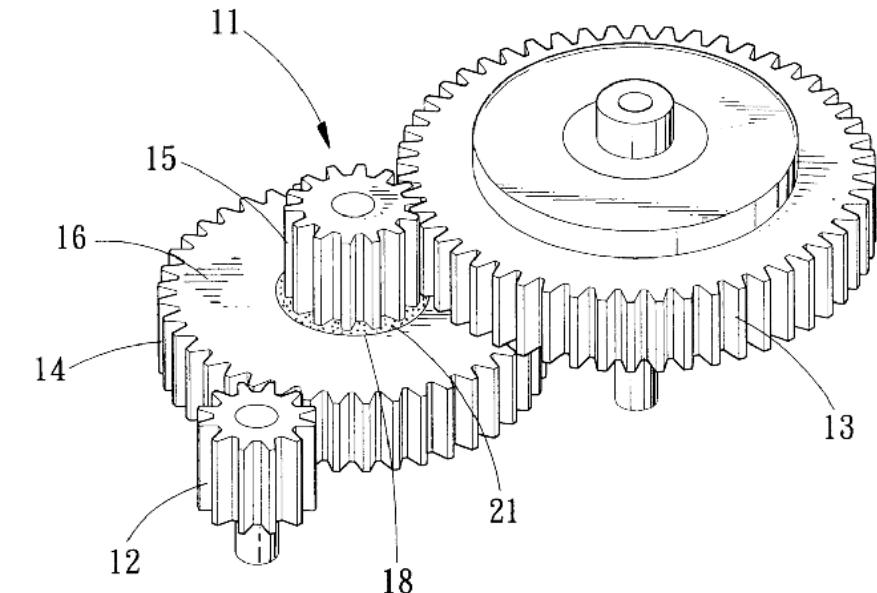
- Portable code (Unix/Linux/Mac/Windoze)
- Ran on everything from iPhone to 1000+ node cluster!
- 'Component' files (~100) inserted from library
 - Sources
 - Optics
 - Samples
 - Monitors
 - If needed, write your own comps
- DSL + ISO-C code gen.





Under-the-hood / inner workings

- Domain-specific-language (DSL) based on compiler technology (LeX+Yacc)
 - Simple Instrument language  ISO C
- Component codes realizing beamline parts (including user contribs)
- Library of common functions for e.g.
 - I/O
 - Random numbers
 - Physical constants
 - Propagation
 - Precession in fields
 - ...





Implementation

- Three levels of source code:
 - Instrument file (All users)
 - Component files (Some users)
 - ANSI c code (no users)



Instrument file

```
DEFINE INSTRUMENT My_Instrument(DIST=10)

/* 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.                                                       */
COMPONENT Origin = Arm()
    AT (0,0,0) ABSOLUTE

COMPONENT Source = Source_simple(
    radius = 0.1, dist = 10, xw = 0.1, yh = 0.1, E0 = 5, dE = 1)
    AT (0, 0, 0) RELATIVE Origin

COMPONENT Emon = E_monitor(
    filename = "Emon.dat", xmin = -0.1, xmax = 0.1, ymin = -0.1,
    ymax = 0.1, Emin = 0, Emax = 10)
    AT (0, 0, DIST) RELATIVE Origin

COMPONENT PSD = PSD_monitor(
    nx = 128, ny = 128, filename = "PSD.dat", xmin = -0.1,
    xmax = 0.1, ymin = -0.1, ymax = 0.1)
    AT (0, 0, 1e-10) RELATIVE Emon

/* The END token marks the instrument definition end */
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 MU-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*s)) 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();
    r=sqrt(rand01())*radius; /* Choose point on source */
    /* with uniform distribution. */

    xf=r*cos(chi);
    yf=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) { /* Choose from uniform distribution */
        E=E0+dr*randpm1();
        v=sqrt(E)*SE2V;
    } else {
        Lambda=Lambda0+dLambda*randpm1();
        v = K2V*(2*PI/Lambda);
    }

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

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

END
```

Written by developers
and possibly you!



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")
mccoordchange(mcpoSource, 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,mcnley, mcnlsy, mcnlp);
mcScattered=0;
mcNCounter[2]++;
#define mccompcurname Source
#define mccompcurindex 2
{
    /* Declarations of SETTING parameters. */
MCNUM radius = mccSource_radius;
MCNUM dist = mccSource_dist;
MCNUM xv = mccSource_xv;
MCNUM yh = mccSource_yh;
MCNUM EO = mccSource_EO;
MCNUM dr = 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, xv, yh, ROT_A_CURRENT_COMP);
}
}

```

Written by mcstas!

McStas is a (pre)compiler!

Input is .comp and .instr files +
runtime functions for e.g. random
numbers

Output is a single c-file, which can
be compiled using e.g. gcc.

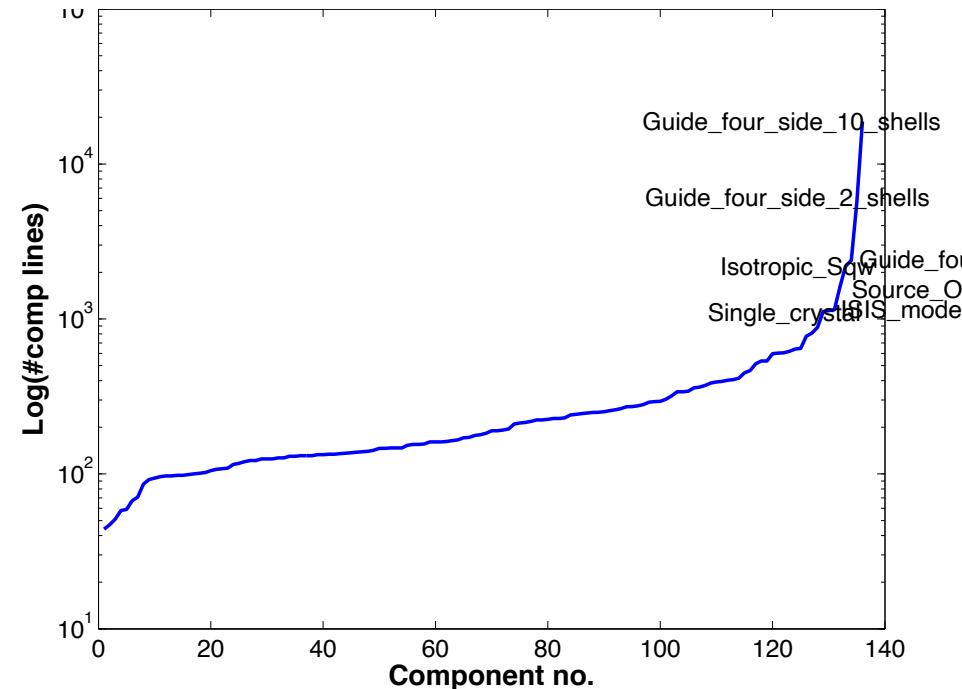
Can take input arguments if
needed.



Writing new comps or understanding existing is not that complex...

- Check our long list of components and look inside... Most of them are quite simple and short... Statistics:

Number of lines of code per component - 199 comps in total

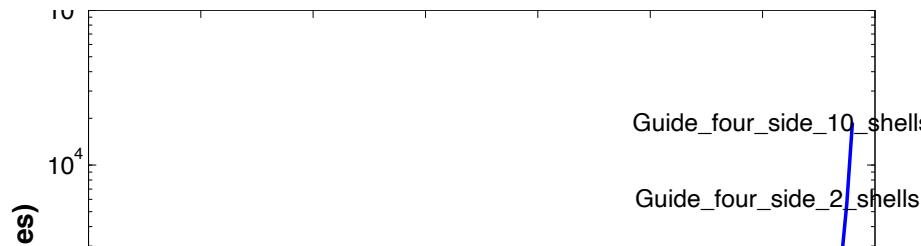




Writing new comps or understanding existing is not that complex...

- Check our long list of components and look inside... Most of them are quite simple and short... Statistics:

Number of lines of code per component - 199 comps in total

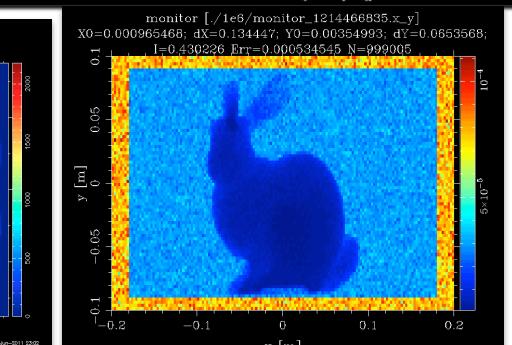
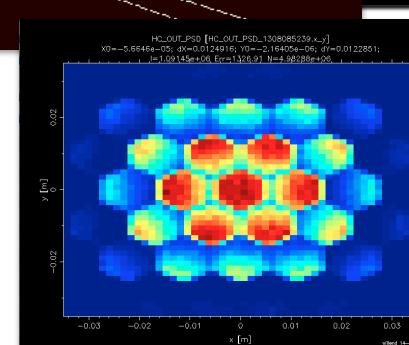
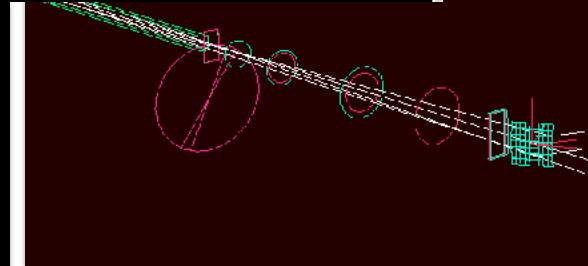
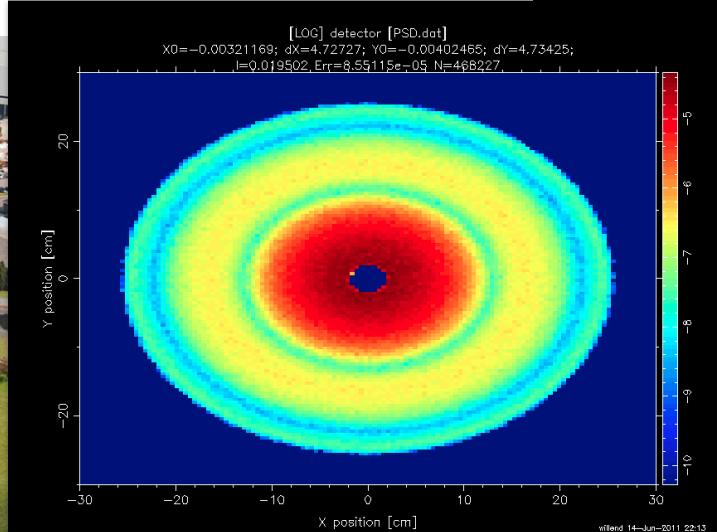
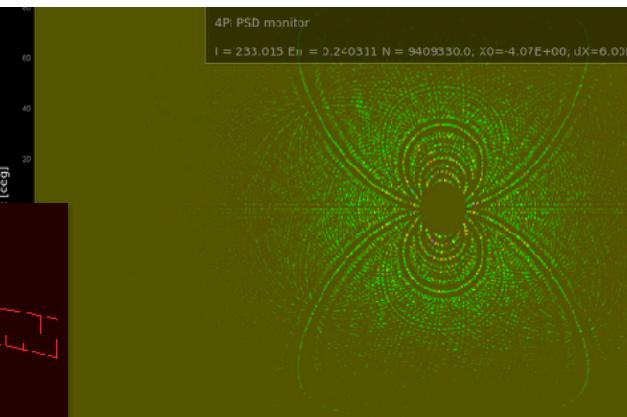
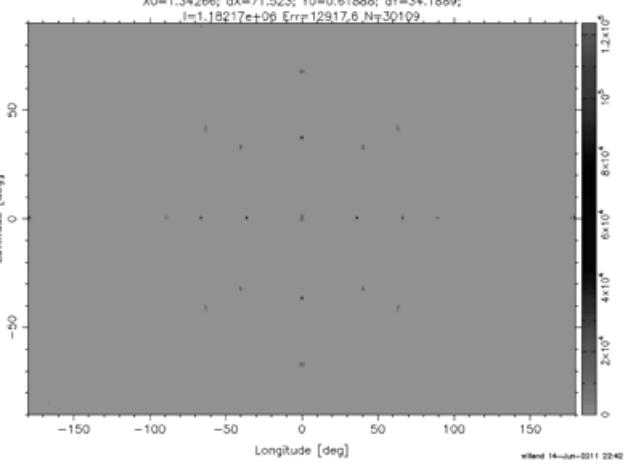
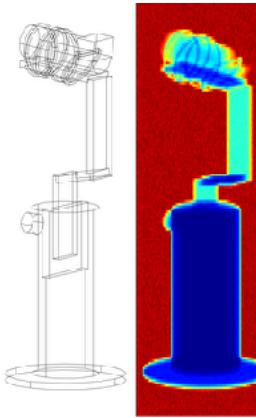
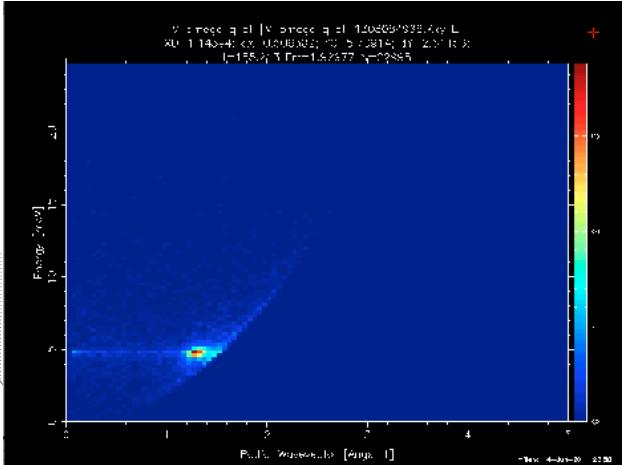
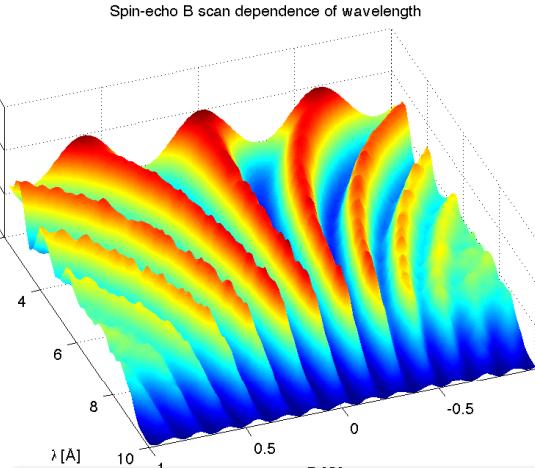


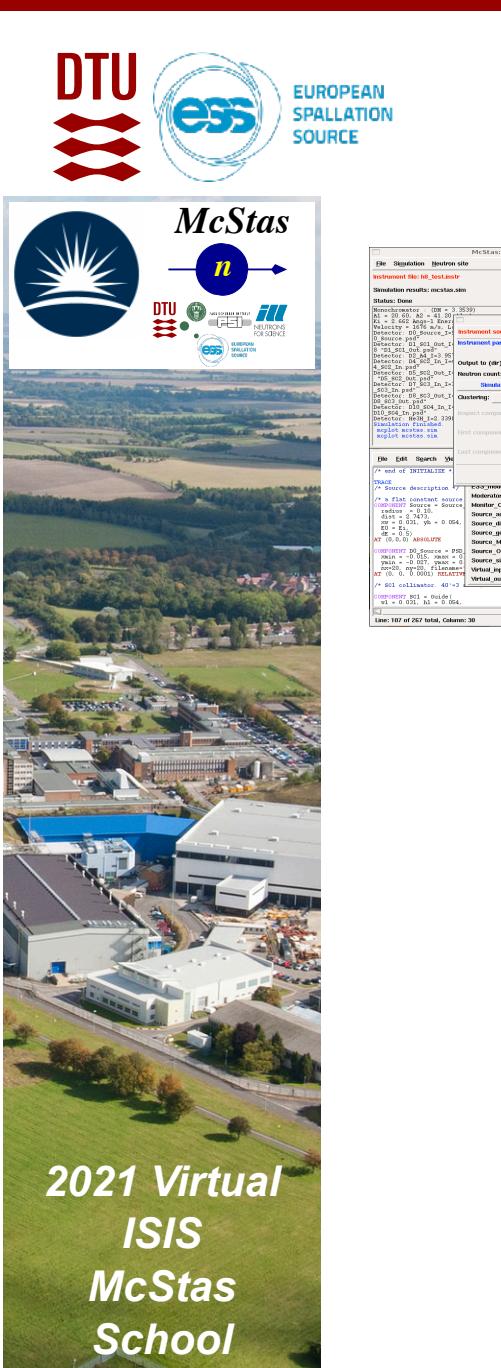
- Well-developed community support
 - 30-40% of existing and new additions are from users
 - No direct refereeing of the code, but these requirements:
 - At least one test-instrument
 - Meaningful documentation headers (in-code docs)
 - Contributions go in dedicated contrib/ section of library



2021 Virtual
ISIS
McStas
School

Example suite: ~140 instruments





2021 Virtual
ISIS
McStas
School



THIS IS NOT
THE END
IT'S JUST
THE BEGINNING