McStas introduction and demo

[1,2]Peter Willendrup, [1]Mads Bertelsen

[1]ESS DMSC
[2]DTU Physics

Agenda

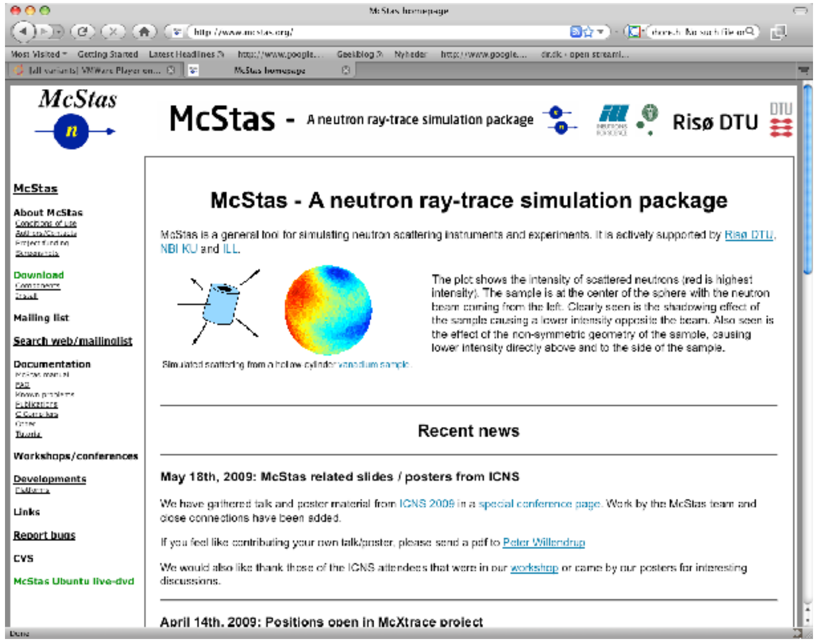
- A brief introduction to McStas
- How McStas works under the hood
- A demo

McStas Introduction

- **Flexible**, general simulation utility for neutron scattering experiments.
- Original design for **M**onte **c**arlo **S**imulation of **t**riple **a**xis **s**pectrometers
- Developed at DTU Physics, ILL, PSI, Uni CPH, ESS DMSC
- V. 1.0 by K Nielsen & K Lefmann (1998) RISØ
- Currently ~2-3 people full time plus students and user-contributions

GNU GPL license

Open Source



Project website at

http://www.mcstas.org

mcstas-users@mcstas.org mailinglist

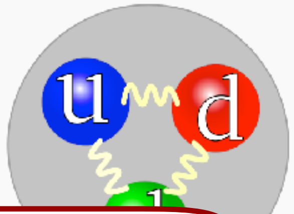
McXtrace - since jan 2009 similar for X-rays

- Synergy, knowledge transfer, shared infrastructure, repo etc.

Used in many places

McStas: Transports cold and thermal Neutrons using Monte Carlo ray-tracing

Neutron

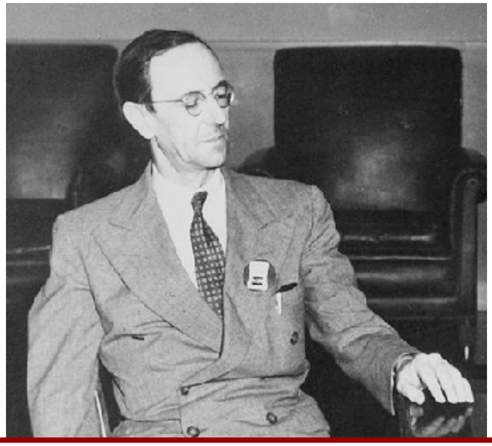


Life time: $\tau_{1/2} = 890s$
Mass: $m = 1.675 \times 10^{-27} kg$
Charge: $Q = 0$
Spin: $s = \hbar/2$
Magnetic moment: $\mu/\mu_n = -1.913$

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

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

Subatomic particle discovered by Sir James Chadwick in 1932



… Non-relativistic velocities, and Born-approximation “non-quantum" treatment.

Similar-featured Monte Carlo ray-tracers:

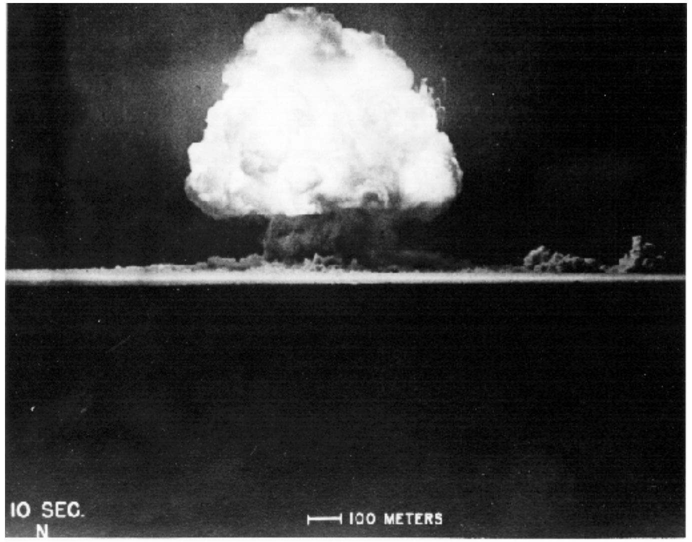
- (NISP*, LANL/ P. Seeger)
- McStas - this talk and collab.
- Vitess, HZB - FZJ
- RESTRAX/SIMRES, NPI
- (IDEAS, ORNL)
- McVine, ORNL
- RAMP, ISIS STFC
- Prompt*, CSNS

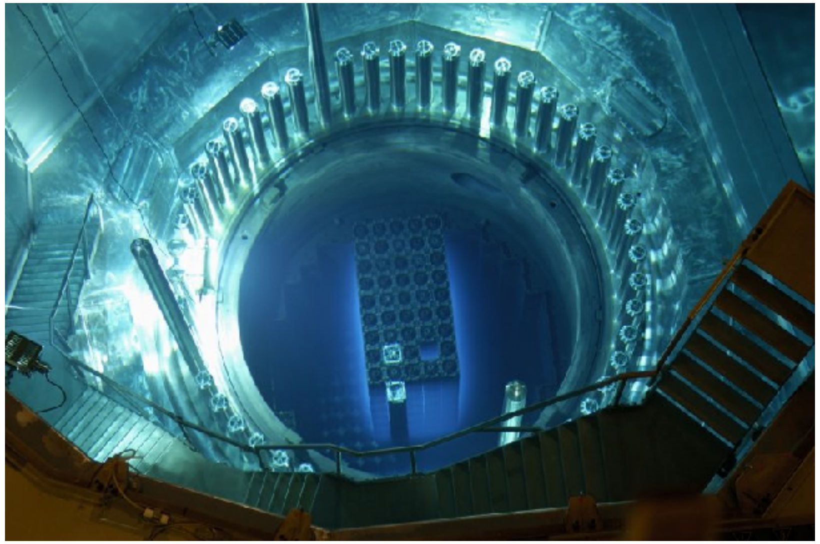
	Energy	Wavelength	n-Wavevector	Velocity	Frequency
cold neutrons:	E = 1 meV	l = 9.0446 Å	k = 0.6947 1/Å	v = 437 m/s	v = 0.2418 THz
	E = 5 meV	l = 4.0449 Å	k = 1.5534 1/Å	v = 978 m/s	v = 1.2090 THz
thermal neutrons:	E = 25 meV	l = 1.8089 Å	k = 3.4734 1/Å	v = 2187 m/s	v = 6.045 THz
	E = 50 meV	l = 1.2791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

*no “linear chain” approach

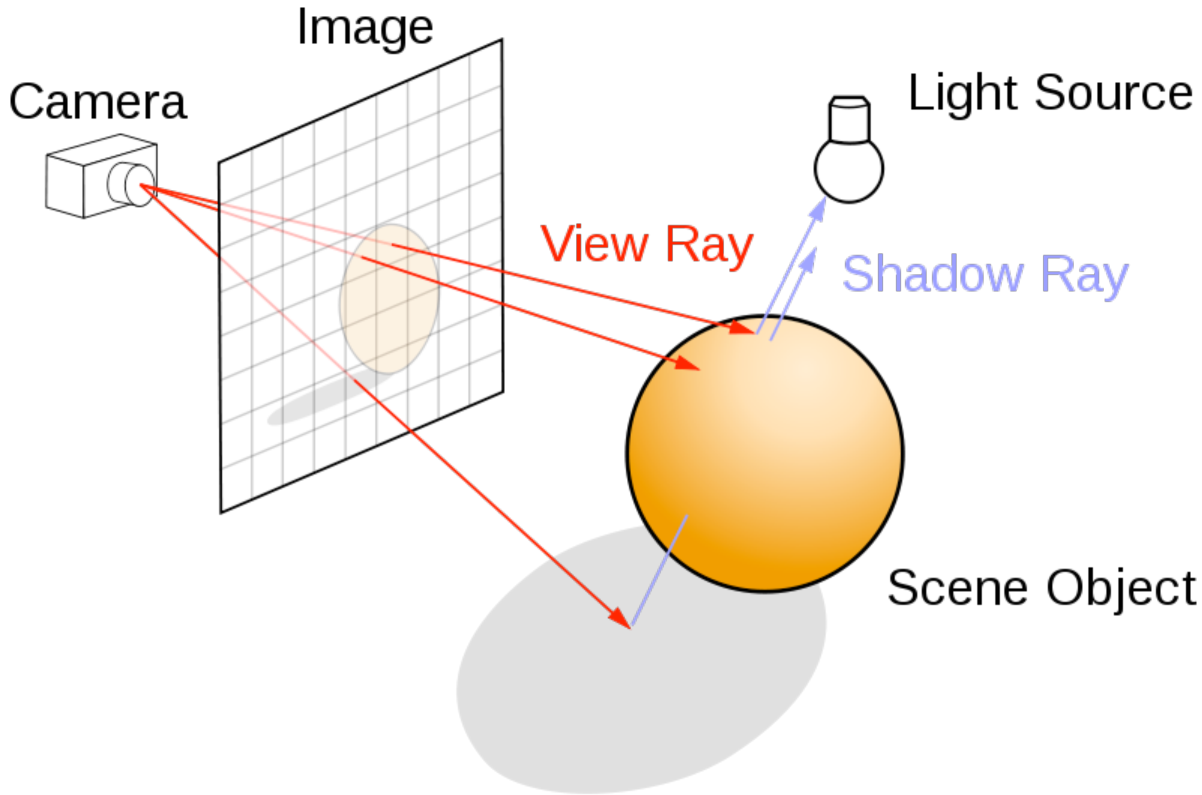
McStas builds on: Monte Carlo techniques

- 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
- **Not much focus on crystalline / ordered material and coherent scattering of neutrons** due to the focus on high energies





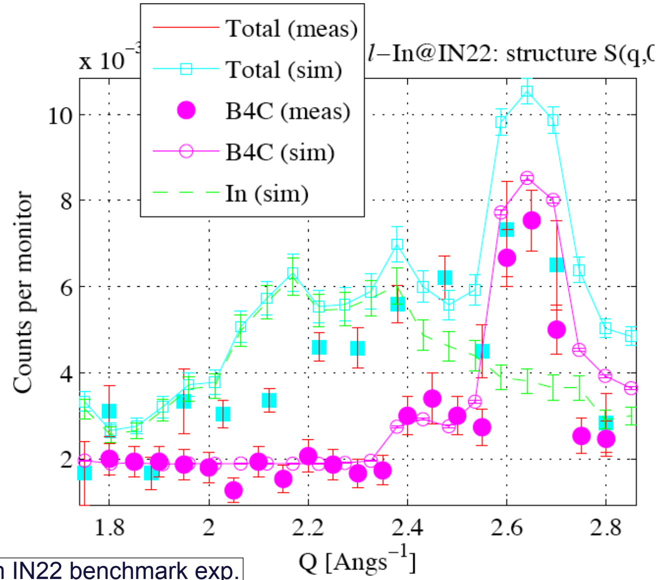
McStas builds on: Ray-tracing methods



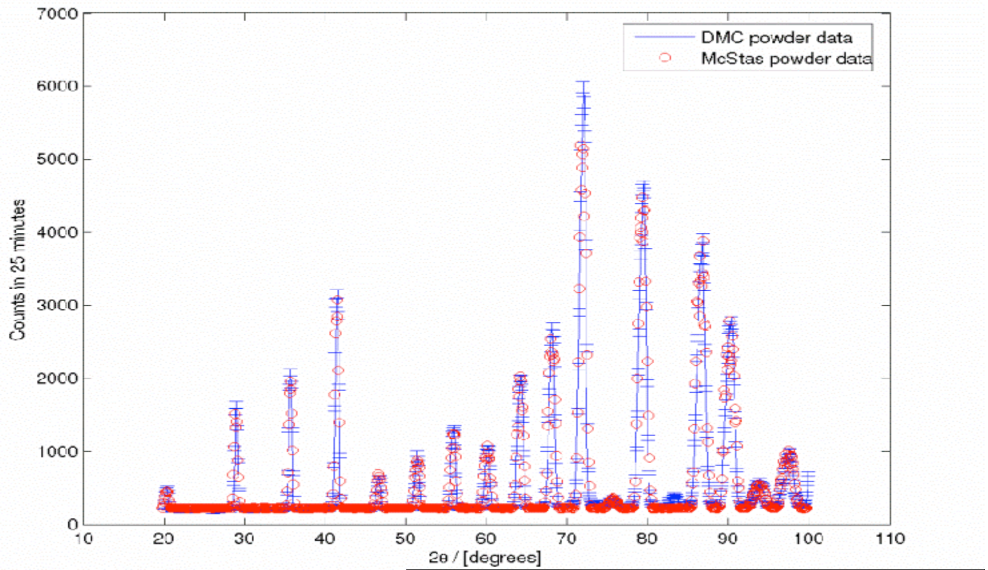
- When neutrons move in “free space”, we use ray-tracing - but in most cases in direction source -> detector (Restrax from NPI Řež has a sample-to-source mode)
- Of course parabolas rather than straight lines are uses to implement gravity

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



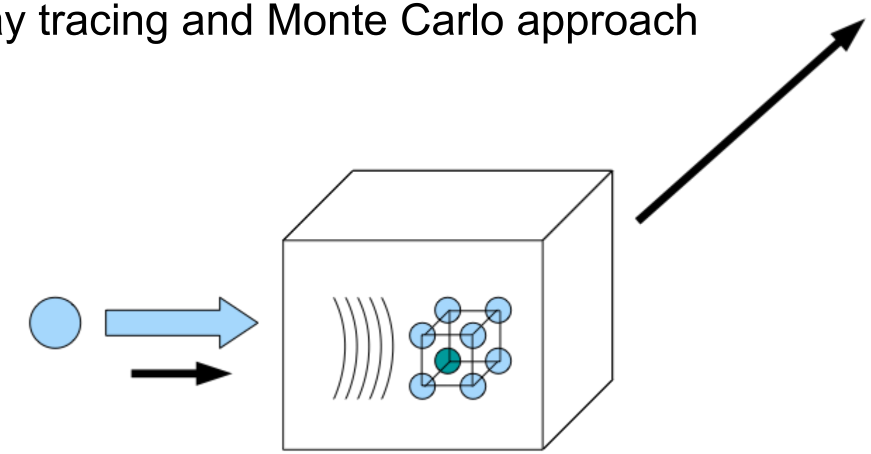
E. Farhi, P. Willendrup, from IN22 benchmark exp.



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

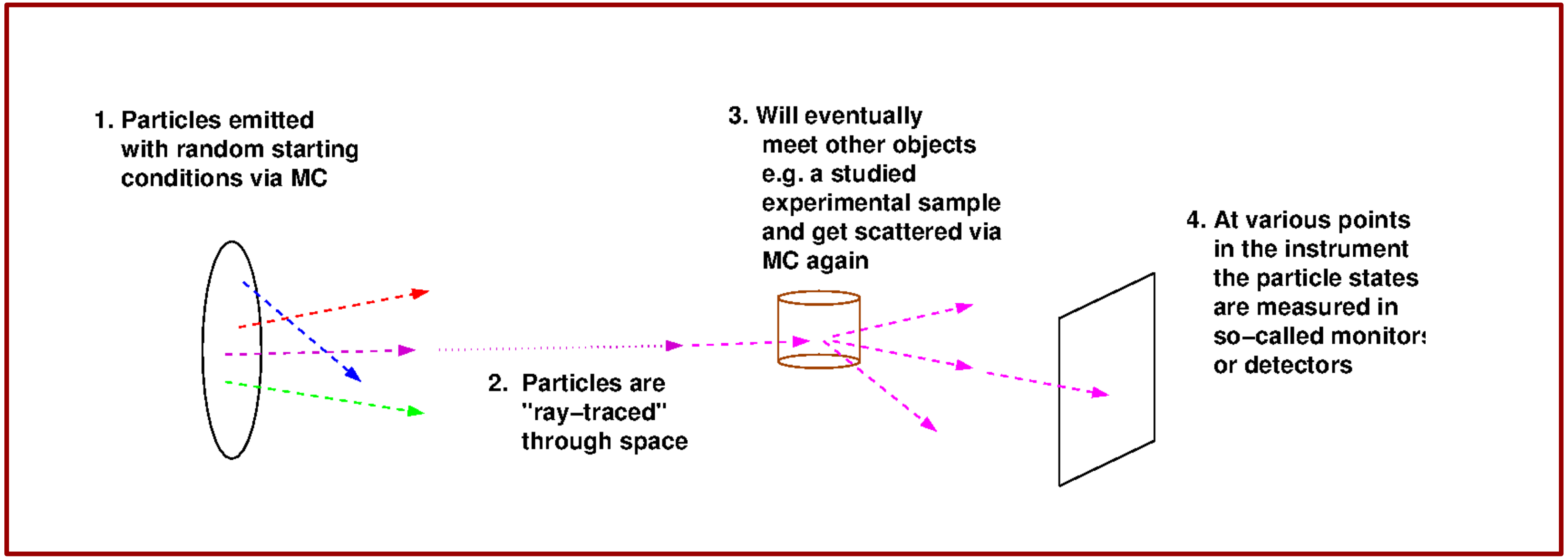
Elements of Monte-Carlo raytracing

- Instrument Monte Carlo methods implement coherent (and stochastic) 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
  - I.e. inside scattering matter
- Uses both particle and wave picture of the neutron and switches back and forward between deterministic ray tracing and Monte Carlo approach

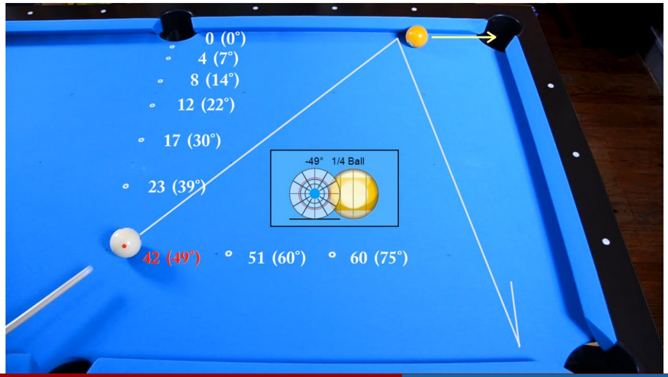


- Result: A realistic and efficient transport of neutrons in the thermal and cold range

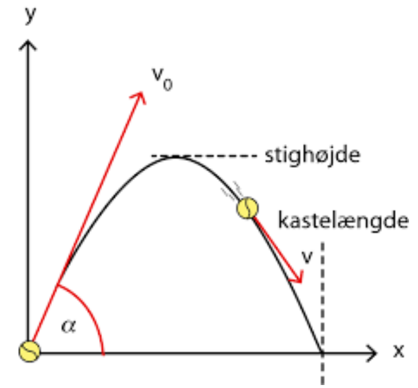
In the big picture, McStas is this…



- Classical Newtonian mechanics, i.e.

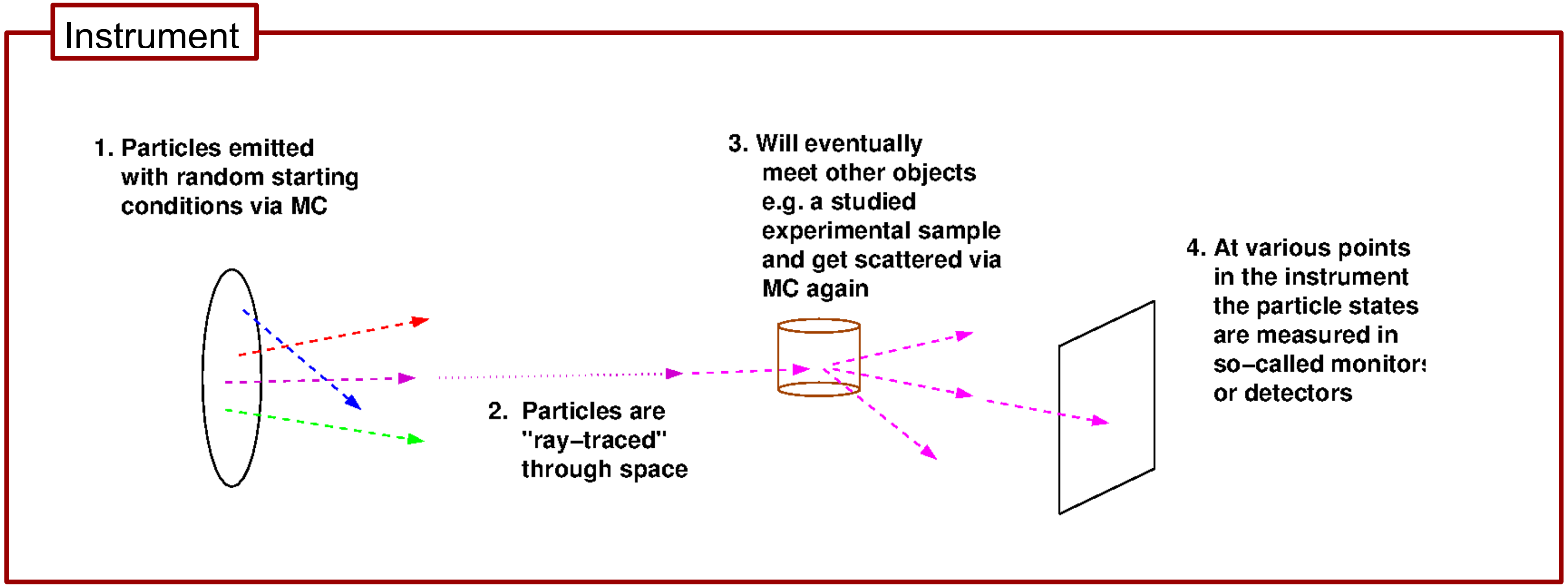


and



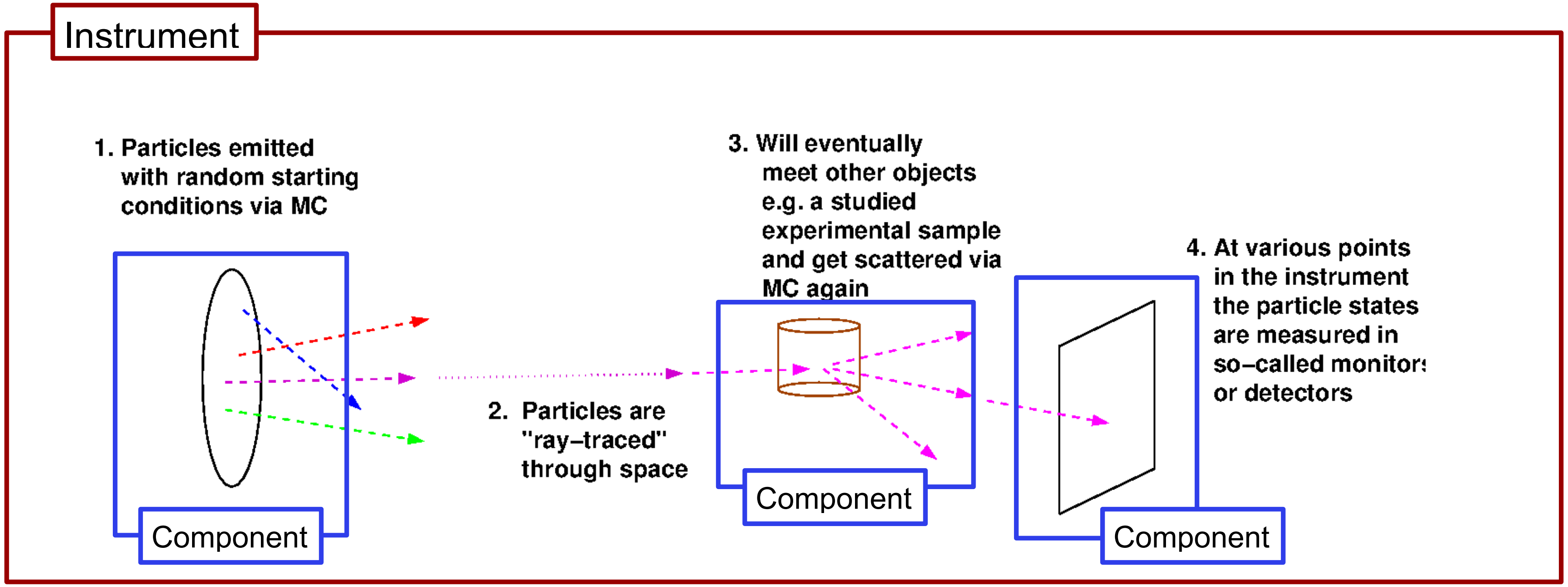
- (independent, particles though…)

In the big picture, McStas is this…



The instrument defines our “lab coordinate system”

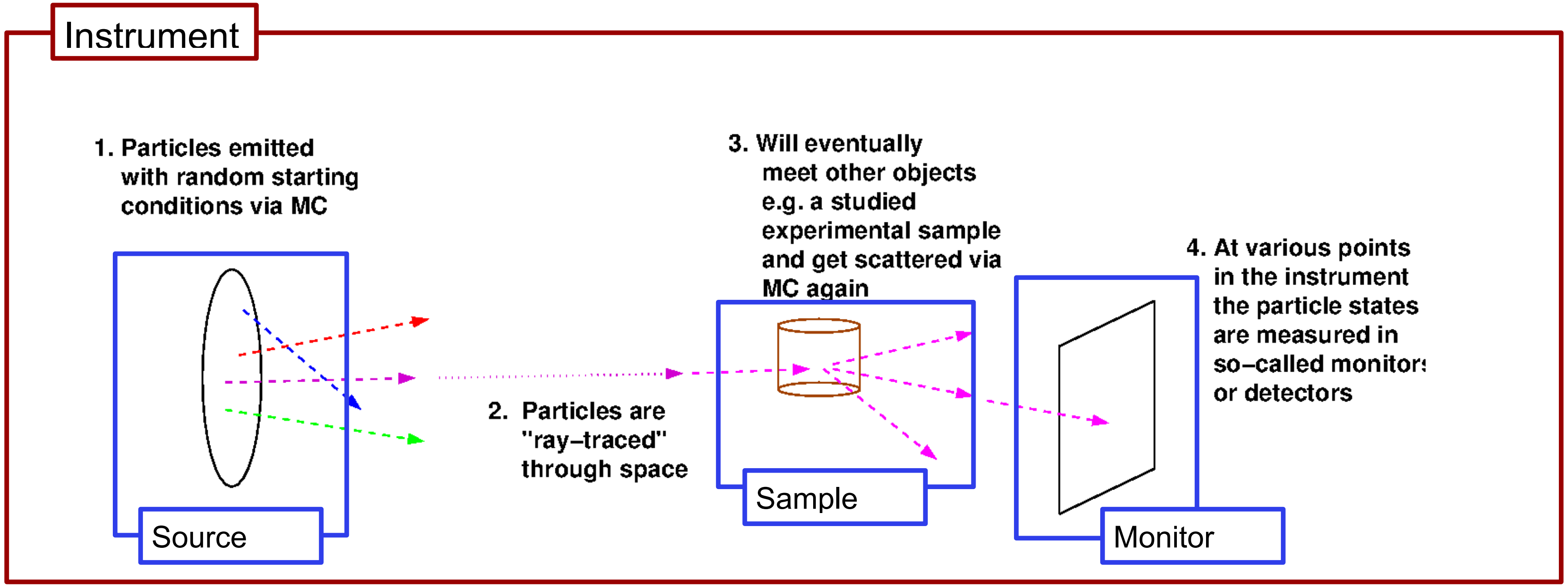
In the big picture, McStas is this…



The instrument defines our "lab coordinate system"

The components define devices or features available in our instrument

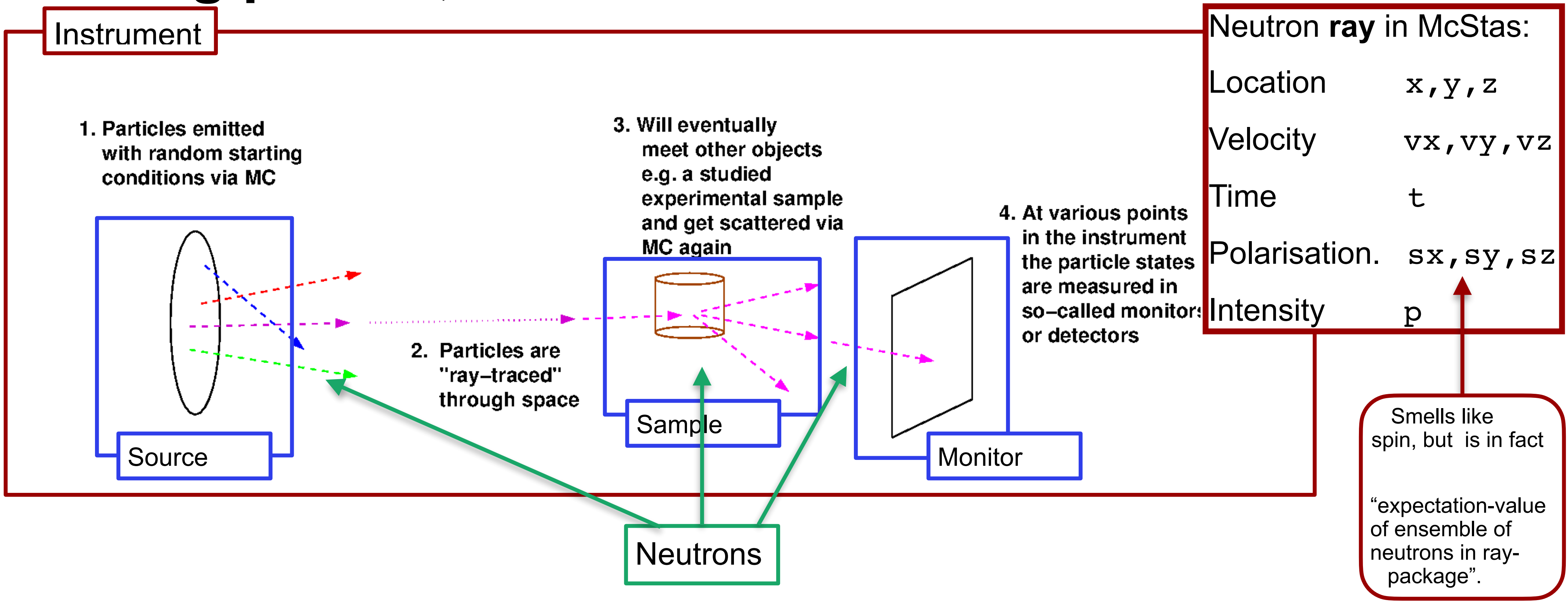
In the big picture, McStas is this…



The instrument defines our “lab coordinate system”

The components define devices or features available in our instrument - they have different function

In the big picture, McStas is this…



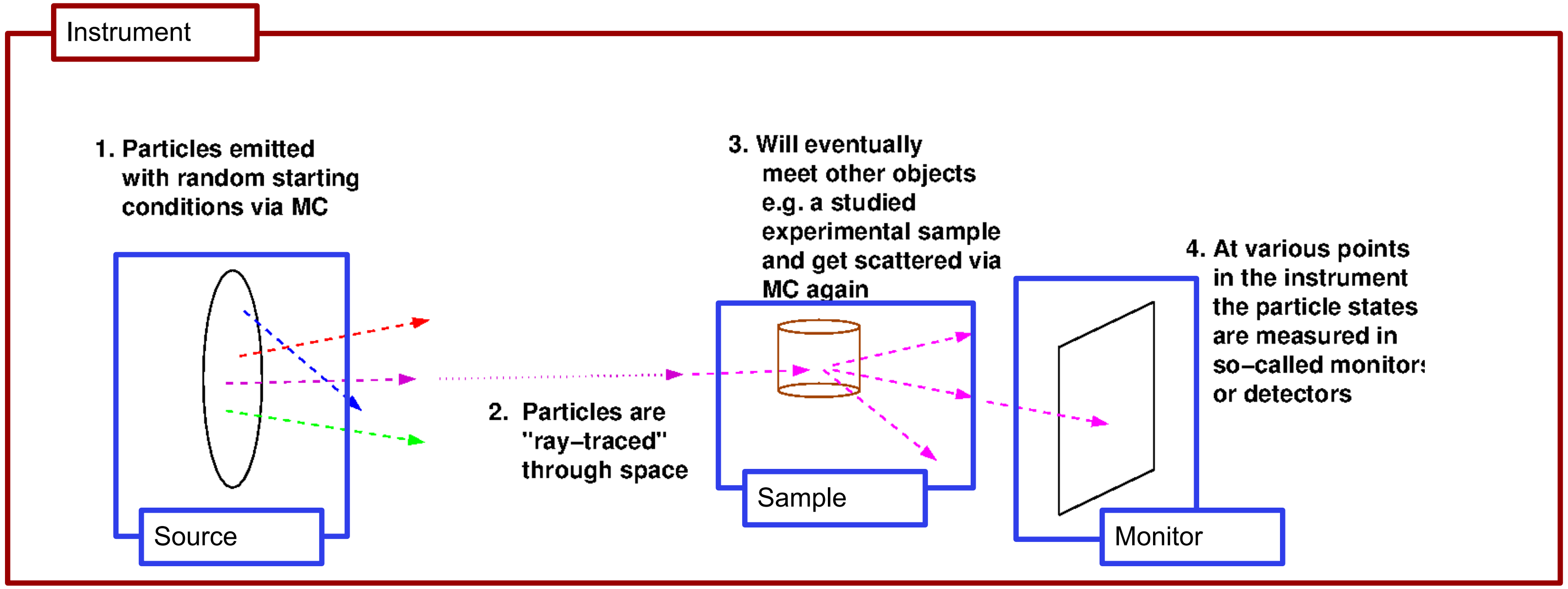
The instrument defines our "lab coordinate system"

The components define devices or features available in our instrument - they have different function

Neutron particles are passed on from one component to the next, changing state under way

In a given component, the neutron intensity is adjusted by a multiplicative factor (probability)

Transport of weight through the instrument…



Neutron **ray** in McStas:	
Location	`x,y,z`
Velocity	`vx,vy,vz`
Time	`t`
Polarisation.	`sx,sy,sz`
Intensity	`p`

p_0 p_j p_n

$$p_j = w_j p_{j-1}$$

$$p_j = p_0 \prod_{k=1}^{j} w_k$$

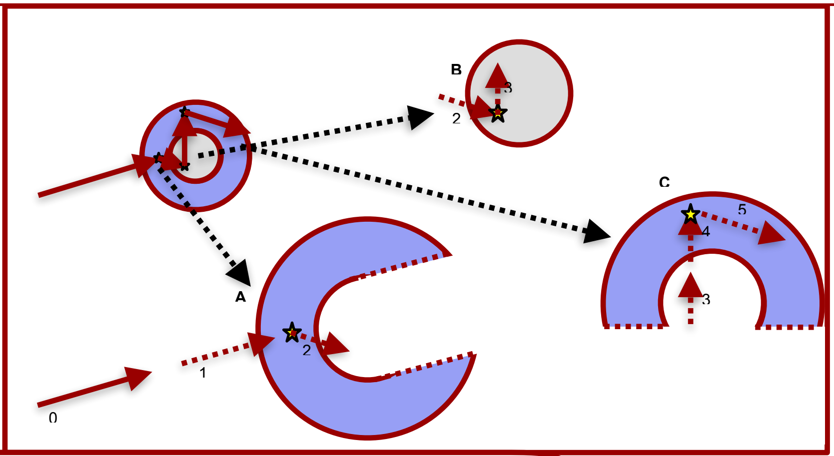
The weight multiplier of the j'th component, w_j, is calculated by the probability rule $f_{\mathrm{MC,b}} w_j = P_{\mathrm{b}}$ where Pb is the physical probability for the event "b", and $f_{MC,b}$ is the probability that the Monte Carlo simulation selects this event.

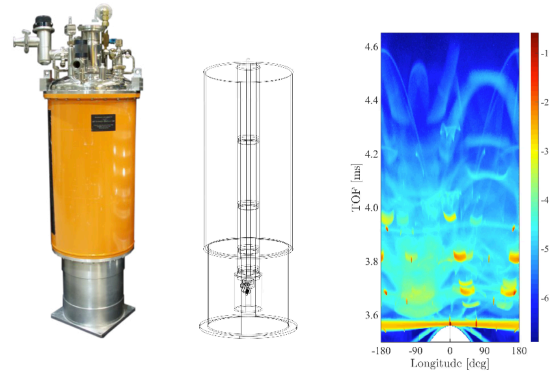
In case of "branching", i.e. multiple outcomes, it is clear that $\sum_b f_{\mathrm{MC,b}} = 1$

McStas is by design a "linear chain" of components

```
{SPLIT} COMPONENT name = comp(parameters) {WHEN condition}
 AT (...) [RELATIVE [reference|PREVIOUS] | ABSOLUTE]
 {ROTATED {RELATIVE [reference|PREVIOUS] | ABSOLUTE} }
 {GROUP group_name}
 {EXTEND C_code}
 {JUMP [reference|PREVIOUS|MYSELF|NEXT] [ITERATE number_of_times | WHEN condition] }
```

- But:
  - We have syntaxes/logic to e.g.GROUP components. (Think: XOR and similar logic)
  - Material-assemblies may be arranged in "concentric" onion-shell assemblies
  - The Union subsystem (Mads Bertelsen) has been added, defining region(s) of the instrument where geometry and materials are decoupled and we completely deviate from the linear approximation
  - NCrystal may be used to describe materials, also within Union. cfg="materials_galore.ncmat"

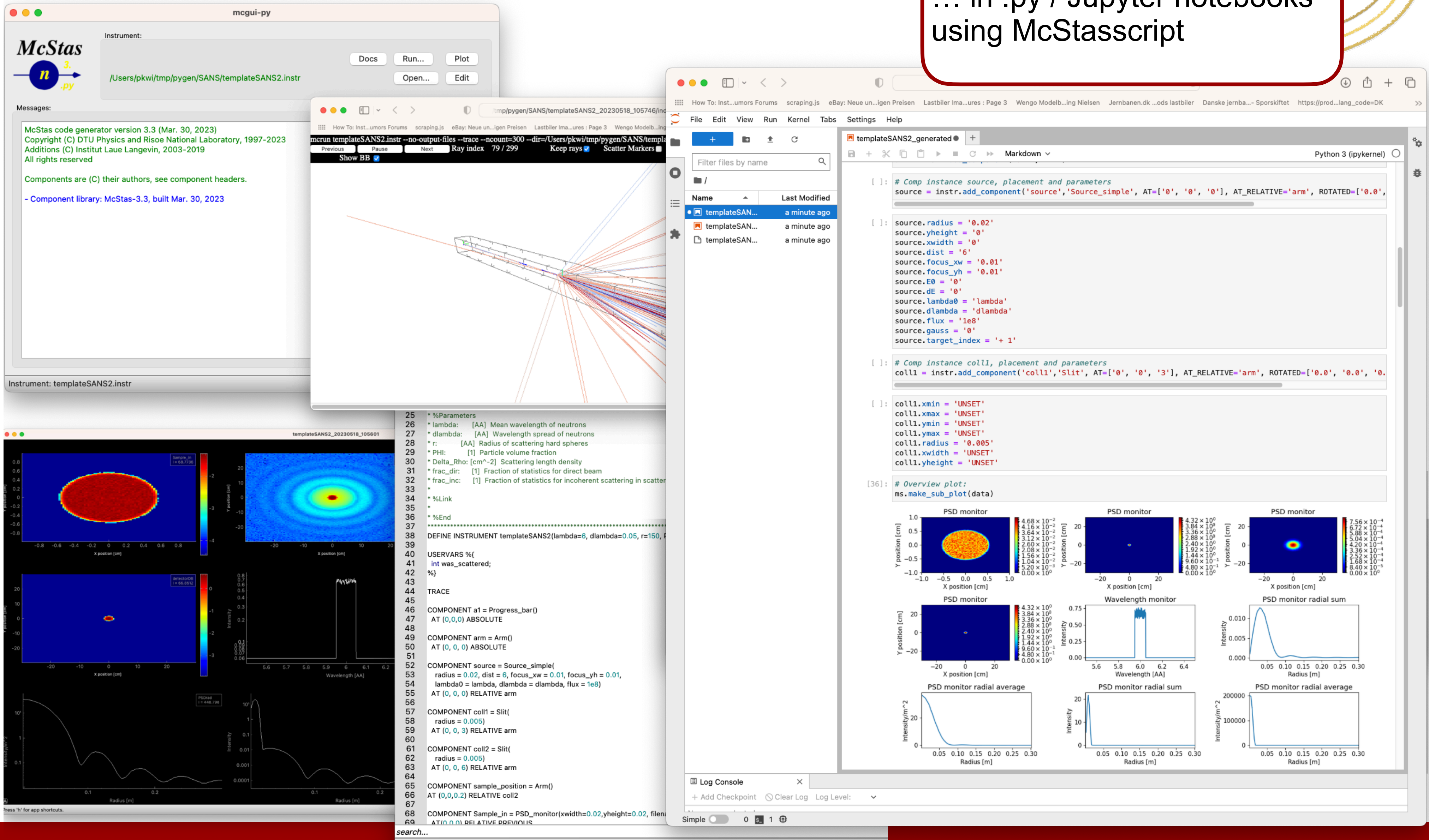




Union-framework

McStas: simulation toolkit for neutron scattering instruments, V.E.

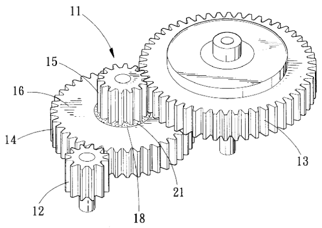
… in .py / Jupyter notebooks using McStasscript



Work in GUI or fav. editor using **DSL** or …

Code generation

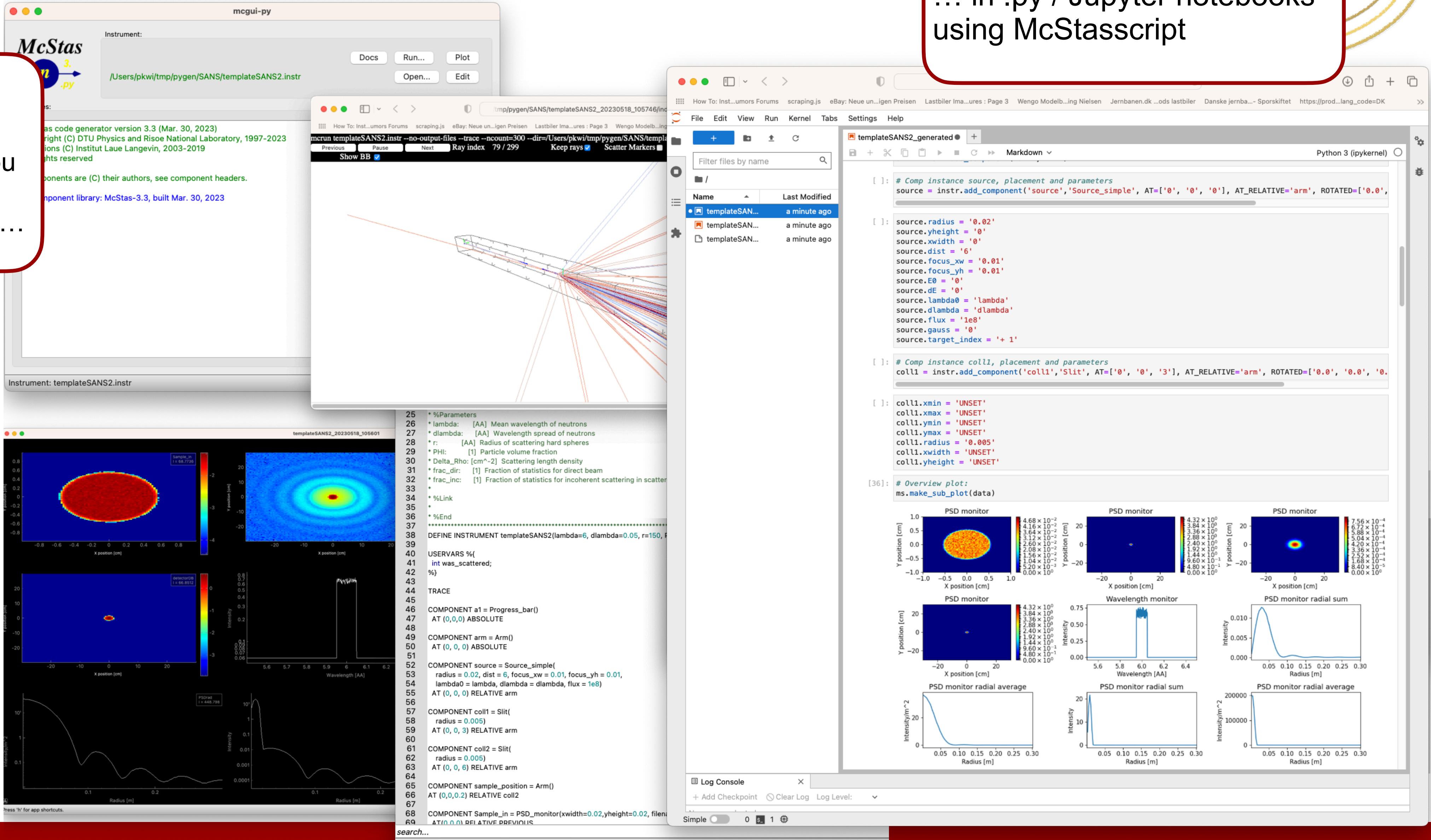
DSL → ISO C



McStas: simulation toolkit for neutron scattering instruments, V.E.

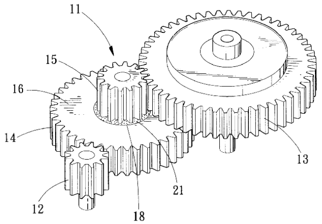
… in .py / Jupyter notebooks using McStasscript

Run-times of “small” problems are often done in seconds to minutes, but you may also use a “bigger hammer” if needed…



Work in GUI or fav. editor using **DSL** or …

DSL —Code generation→ ISO C



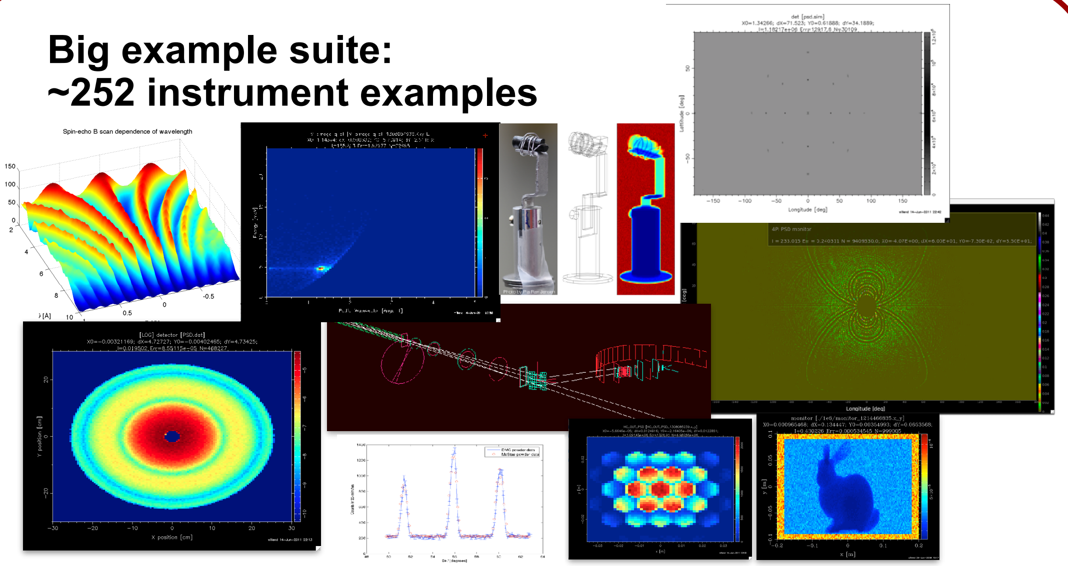
McStas: simulation toolkit for neutron scattering instr…

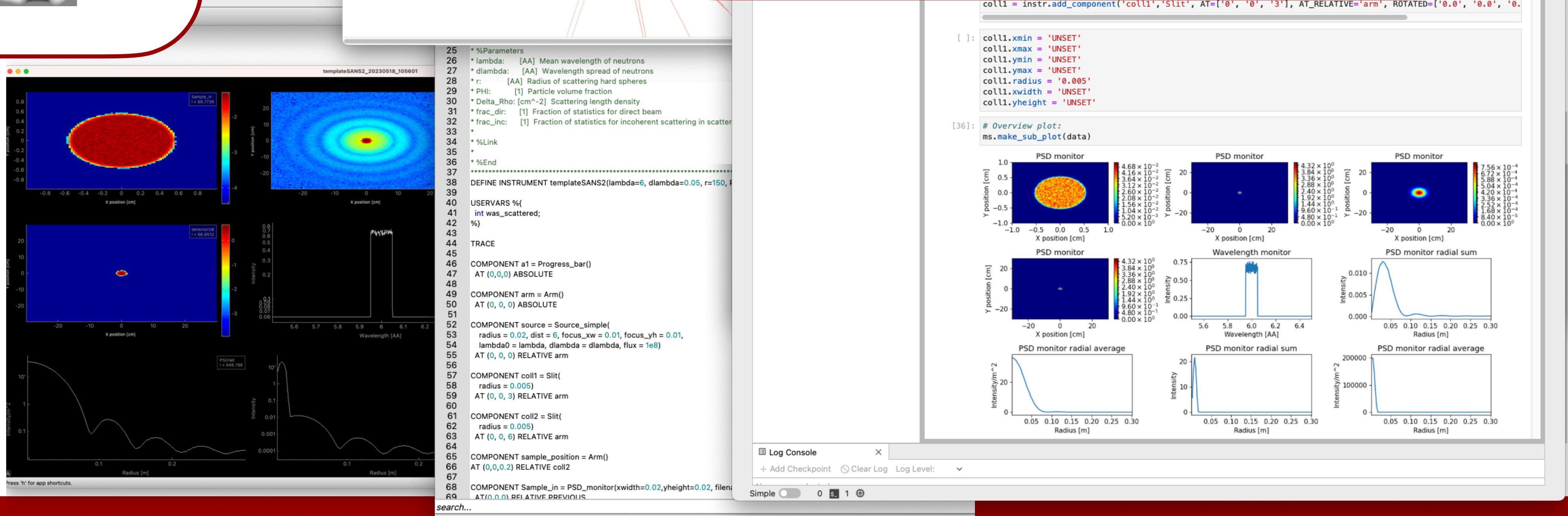
DTU
EUROPEAN SPALLATION SOURCE

Very portable. 1…N CPU's via MPI

+ NVIDIA GPU's

Big example suite:
~252 instrument examples

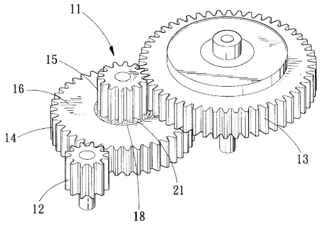




Work in GUI or fav. editor using **DSL** or …

Code generation

DSL → ISO C

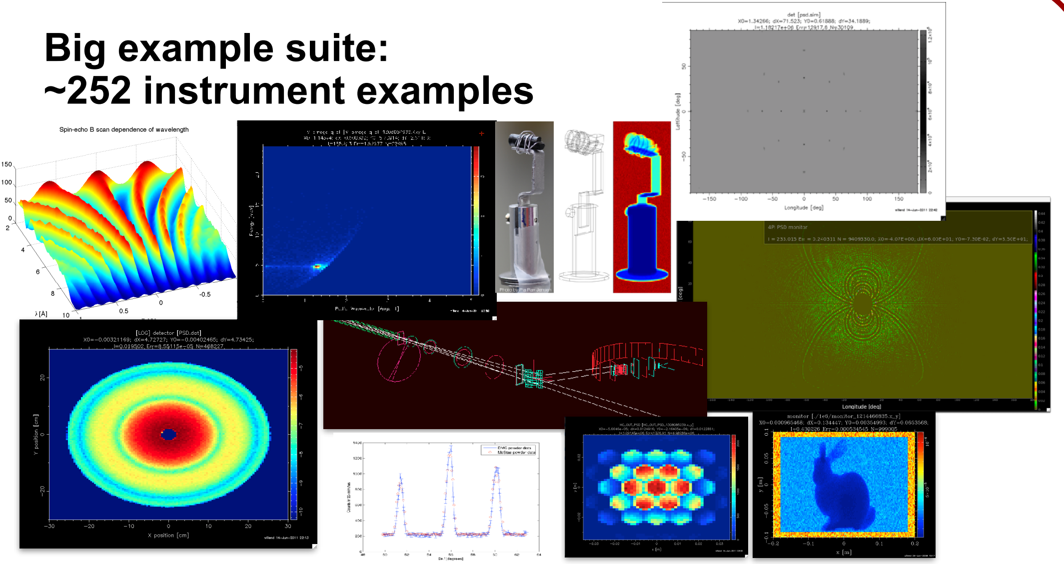


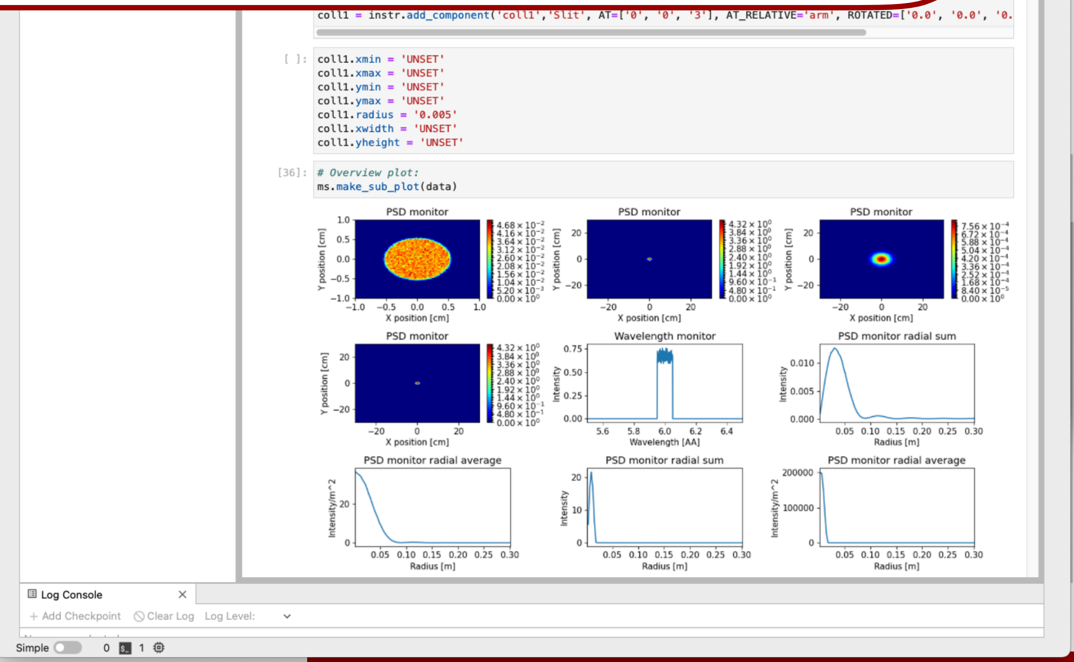
McStas: simulation toolkit for neutron scattering instr…

Very portable. 1…N CPU's via MPI

+ NVIDIA GPU's

Big example suite:
~252 instrument examples



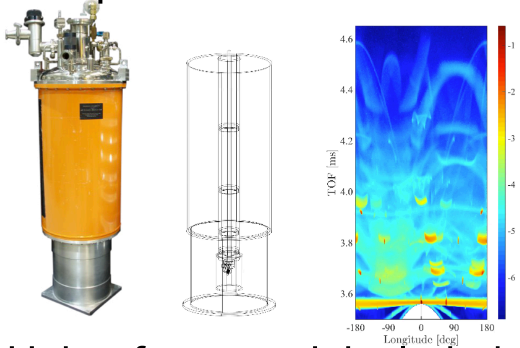


* **Open Source (GPLv3)**
* "Large" user community, "accepted" code, many user contrib.
* Good user support
* Interconnects with e.g. **MCNP**(x), **Geant4**, OpenMC, **Vitess** via direct interfaces or **MCPL**
* May use built-in material models or use e.g. **NCrystal**, **SASmodels** etc.
* Generates NeXus/HDF that loads in **Mantid**

***Made with the "instrument scientist" in mind**

McStas: simulation toolkit for neutron scattering instruments, virtual experiments

Sample-environments



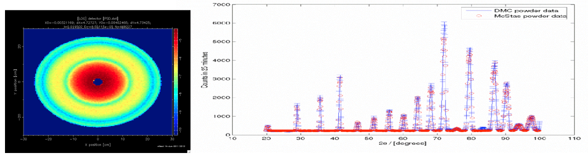
Union-framework included

McStas supports MPI and from 3.x Nvidia GPUs

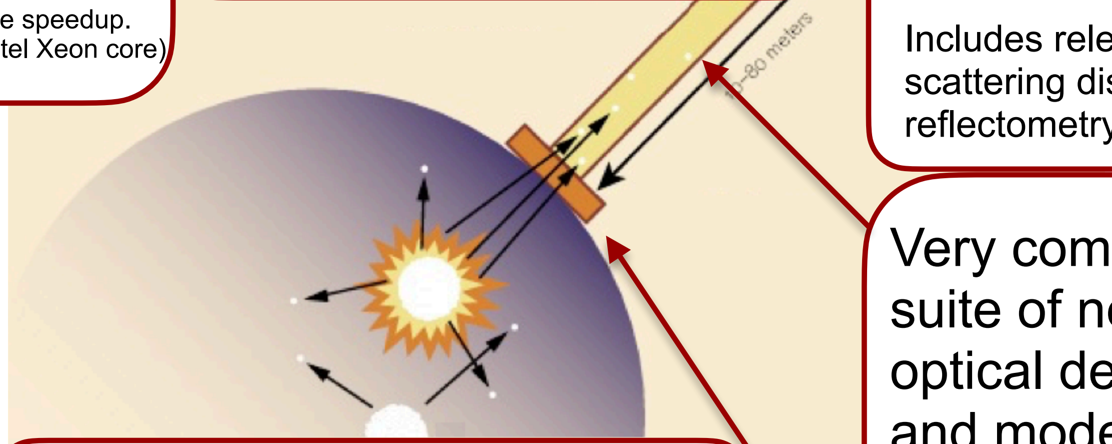
McStas + NVIDIA

- 2 orders of magnitude speedup.
(1x Tesla V100 vs 1 Intel Xeon core)

Detectors

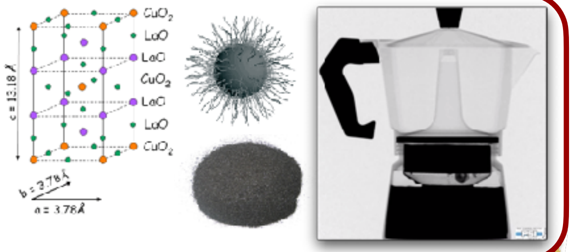


In most cases ideal, but:
* Easy to add point-spread fct.
* He3 model included
* Hooks to e.g. Geant4 via MCPL



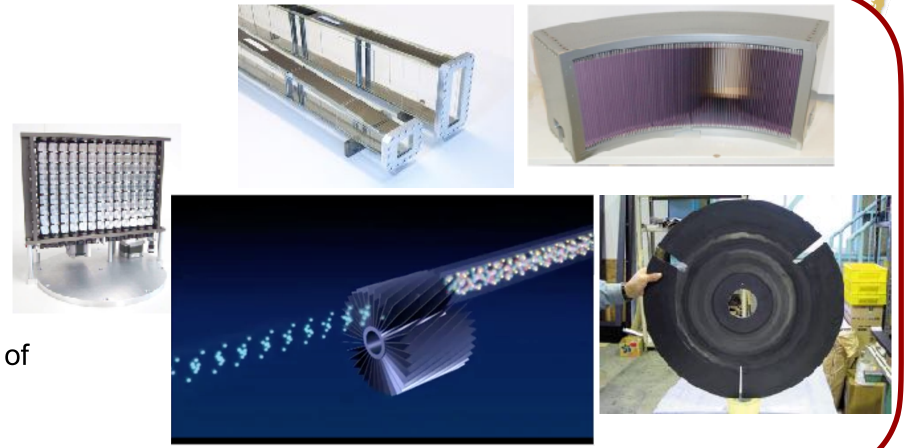
Scientific model-samples

Includes relevant models for 'all' scattering disciplines, e.g. SANS, imaging, reflectometry, pwd+sx diffraction, spectroscopy



Very complete suite of neutron-optical devices and models.

(And "rolling your own" for any type of "component" is straightforward.)



Models instruments at reactors or spallation sources

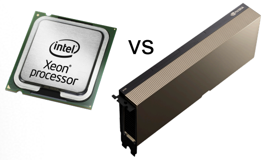
(Needs source term from e.g. MCNP or OpenMC.)

Includes library of neutron moderators at existing and future facilities.

McStas

Peter Willendrup, DTU Physics and ESS DMSC

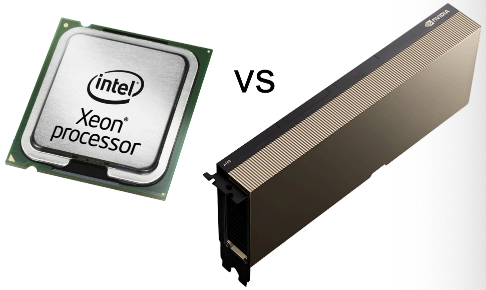
Maximum performance indication on NVIDIA A100 (Ampere)

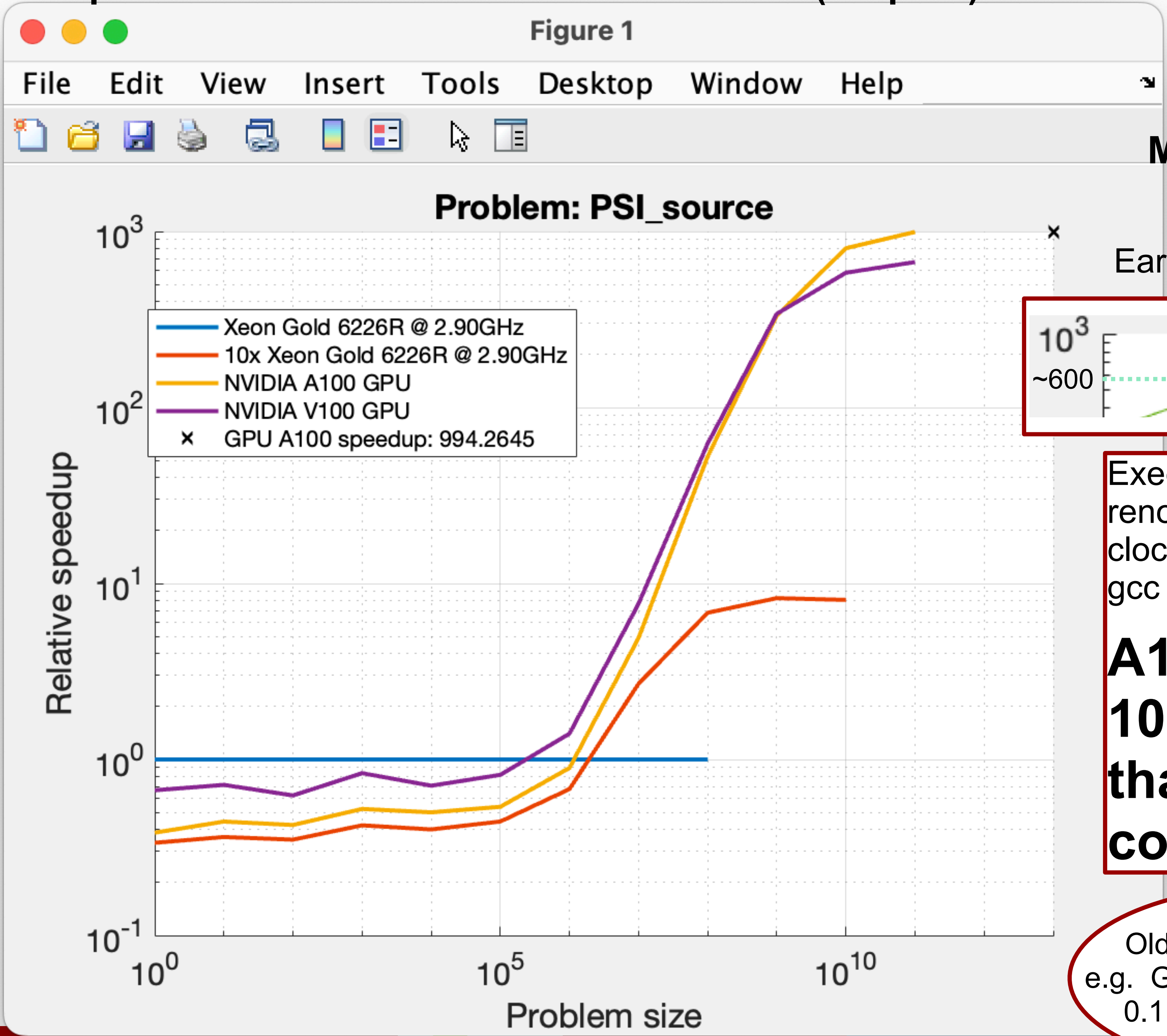


Maximal speedup: ~1000

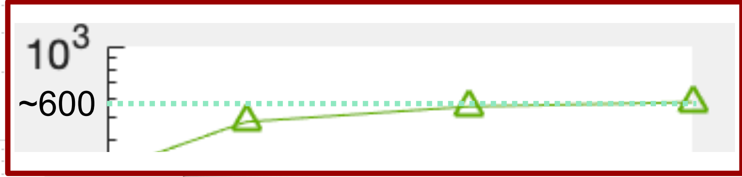
Idealised instrument with source and monitor only - i.e. without any use of the ABSORB macro.

(Good indication of maximal speedup achievable.)





Earlier dataset from V100 ~600



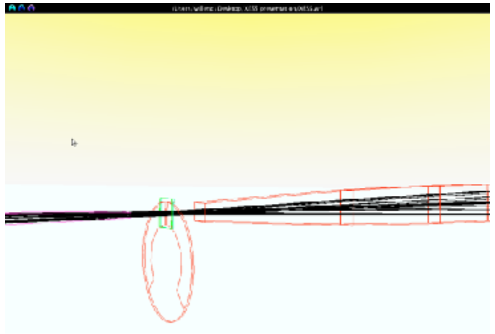
Execution speedups renormalised to wall-clock of single-core gcc standard simulation,

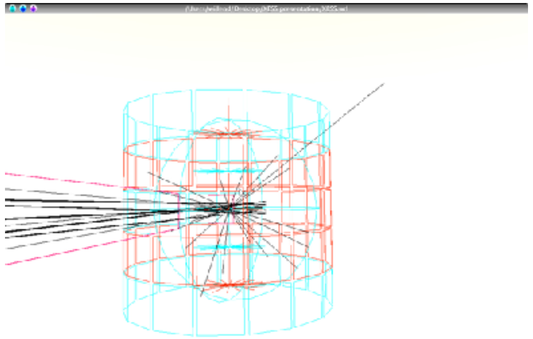
A100 run can be ~ 1000 times faster than a single-core CPU run

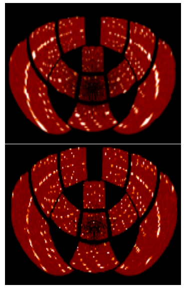
Older "Gamer-GPU" e.g. GeForce 1030 is ~ 0.1 V100 or 0.05 A100

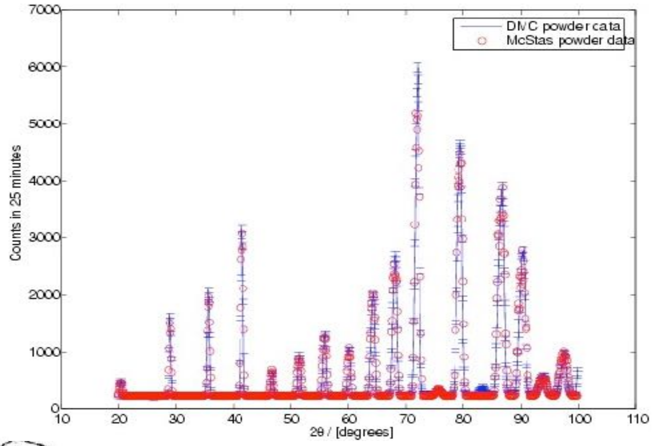
What is McStas used for?

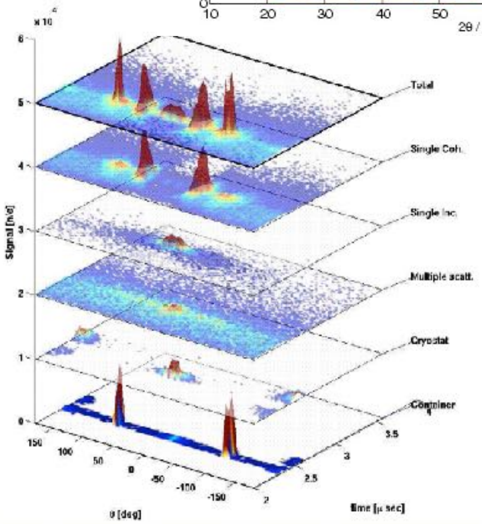
- Instrumentation
- Planning
- Construction
- Virtual experiments
- Data analysis
- Teaching
  (KU, DTU)

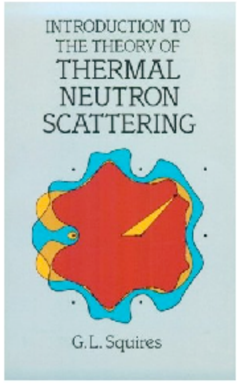


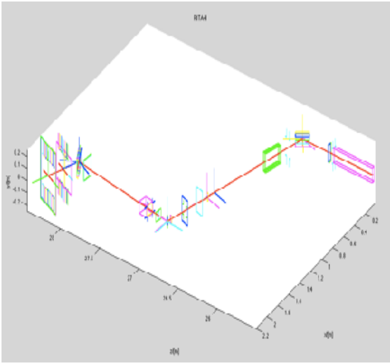








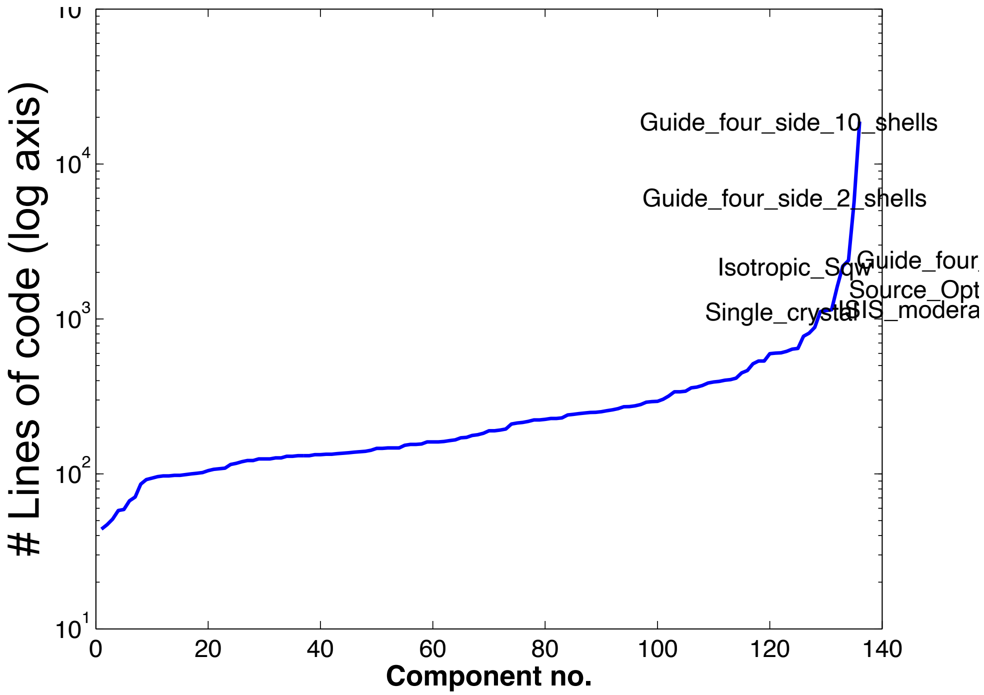




Writing new comps or understanding existing is not that complex...

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

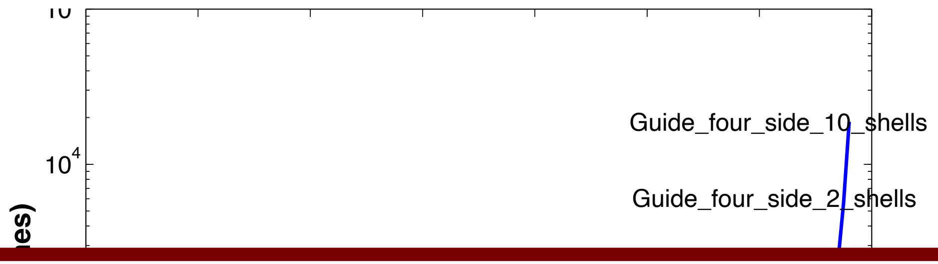
Number of lines of code per component - 240 comps in total



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 - 240 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

Thanks to all users, contributors, developers,

- DEMO TIME