

Peter Willendrup

Simulation optimisation, brilliance basics, FOM's and tools



Topics

- Optimising your simulation (statistics etc.)
 - Variance reduction / biasing
- Source brilliance and brilliance transfer
- Deciding for a figure of merit
- Tools: scans, iFit and guide_bot



Important points to remember

1. Your simulation will only contain elements you provided / defined
2. ... to the precision you defined
3. Answers the questions you posed
4. Background essentially only from “sample”, or sample-near objects



2021 Virtual
ISIS
McStas
School

Ensuring efficiency of the simulation is an important start...

- ◆ Apply focusing techniques
 - ◆ At the source (spatially, temporally, in wavelength...)
 - ◆ At the sample, if possible
- ◆ (carefully!) Apply SPLIT - but only if immediately followed by Monte Carlo choices, e.g. in sample
- ◆ Alternatively use MCPL o/i which allows repetition - beware of biases!



Ensuring efficiency of the simulation is an important start...

- ◆ Apply focusing techniques
 - ◆ At the source (spatially, temporally, in wavelength...)
 - ◆ At the sample, if possible
- ◆ (carefully!) Apply SPLIT - but only if immediately followed by Monte Carlo choices, e.g. in sample
- ◆ Alternatively use MCPL o/i which allows repetition - beware of biases!

All of this can be considered “variance reduction” or biasing

Which hammer?

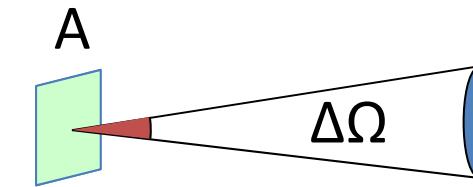


- ◆ Use **MPI** parallelisation - included in macOS install from 2.4, easy to get on Linux... Ready to roll on IDAaaS.
- ◆ Choice of compiler and optimisation flags can at times give ~a factor of 2 in speed, clang vs gcc vs Intel C
- ◆ Think of moving to McStas 3.0 and GPU's could help you... (see thursday talk)
- ◆ - Always consider if you are asking the right question if runtimes reach days/weeks... -> **Pick the right hammer!**

Sledge-hammer / brute force!



Neutron flux - and brilliance / brightness

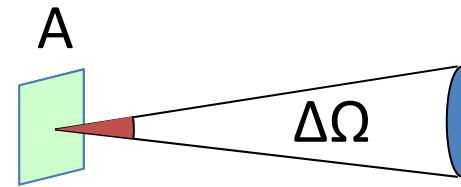


$B = N \text{ per time per } A \text{ per } \Delta\Omega$

units = $\text{n/s/cm}^2/\text{sr}$



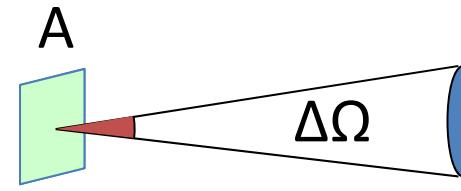
Neutron flux - and brilliance / brightness



B is independent of distance
- property of the source



Neutron flux - and brilliance / brightness

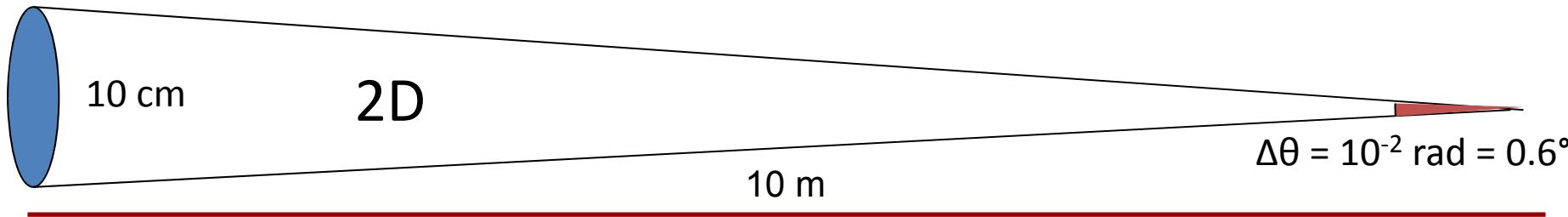


Brilliance/Brightness B $\text{n}/\text{s}/\text{cm}^2/\text{sr}$

Flux Ψ $\text{n}/\text{s}/\text{cm}^2$

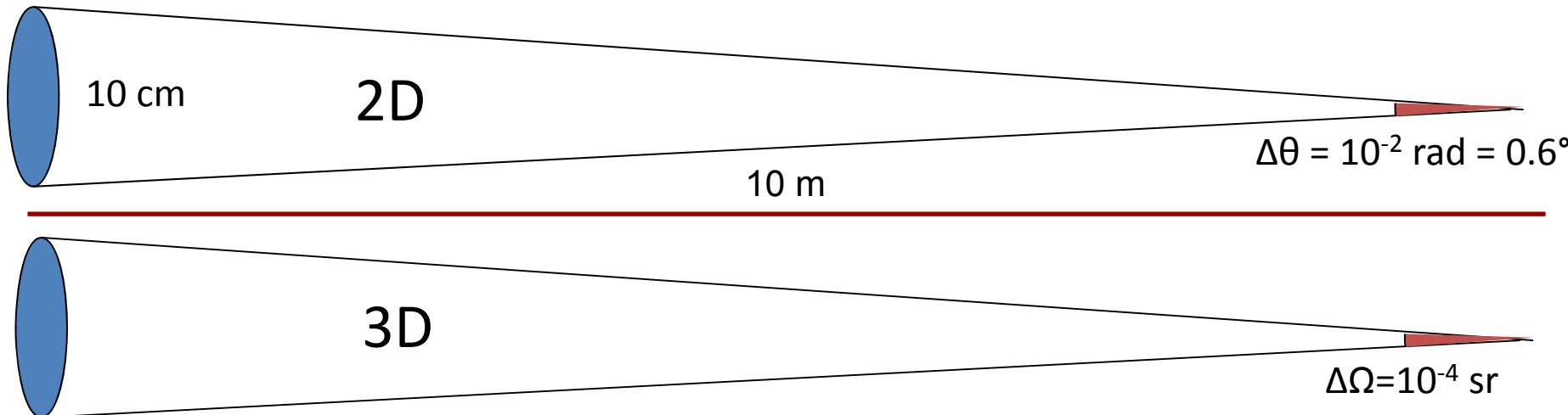


Neutron flux - and brilliance / brightness





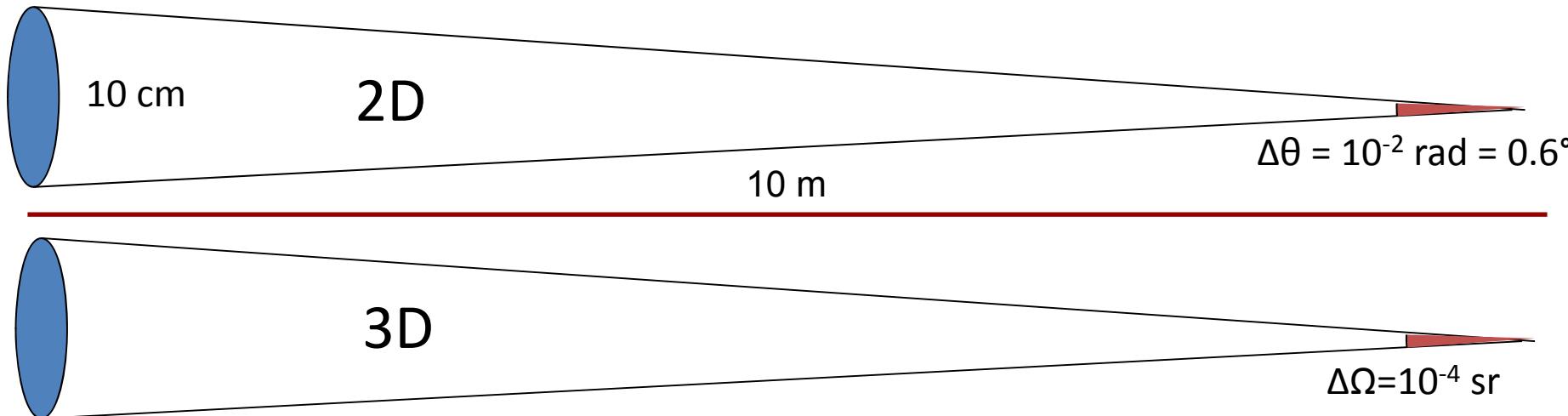
Neutron flux - and brilliance / brightness



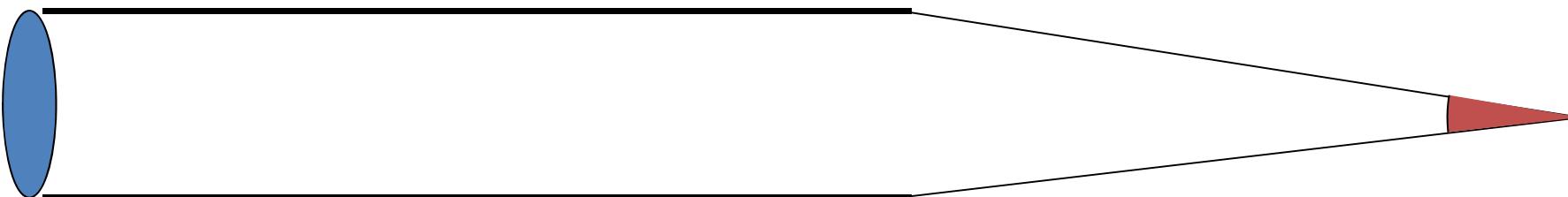
$$\text{Flux} = \text{Source Brightness} \times \text{Solid Angle}$$
$$\Phi = B \times \Delta\Omega$$



Neutron flux - and brilliance / brightness



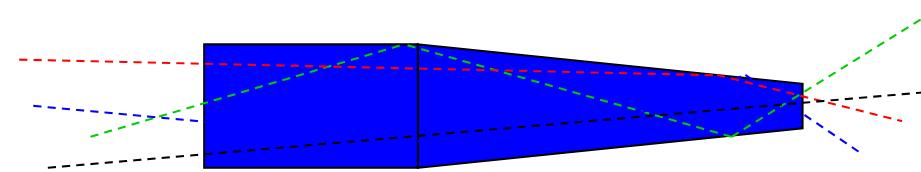
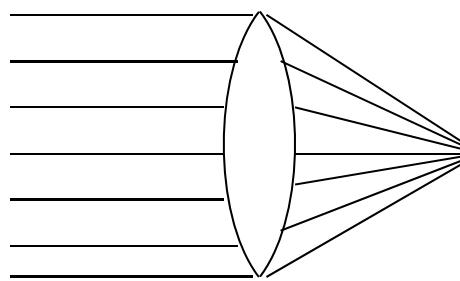
$$\text{Flux} = \text{Source Brightness} \times \text{Solid Angle}$$
$$\Phi = B \times \Delta\Omega$$



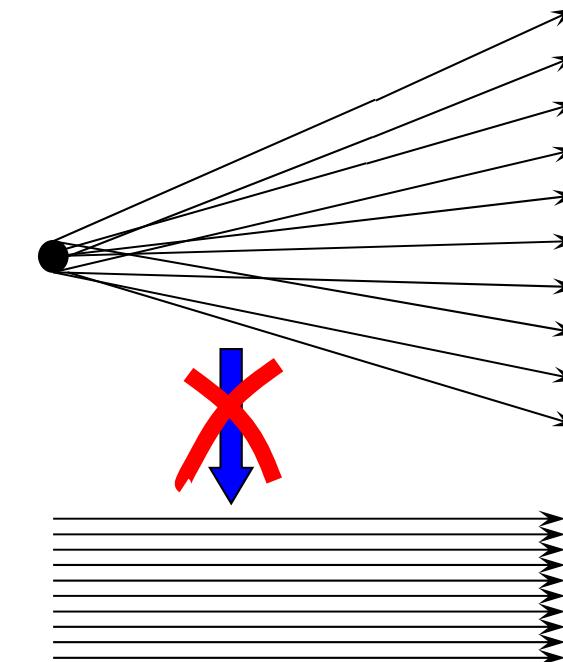


Liouvilles Theorem

- Conservation laws:
 - neutrons can't be created from thin air
 - neither can “phase space density”
- There is no such thing as a free lunch
 - Beam manipulation transfers distribution between time, area, divergence, energy
- Most common application:
 - Focusing increases divergence
 - improve flux, lose angular resolution



Liouville theorem
 $\rightarrow \Delta x \Delta \phi = \text{const}$

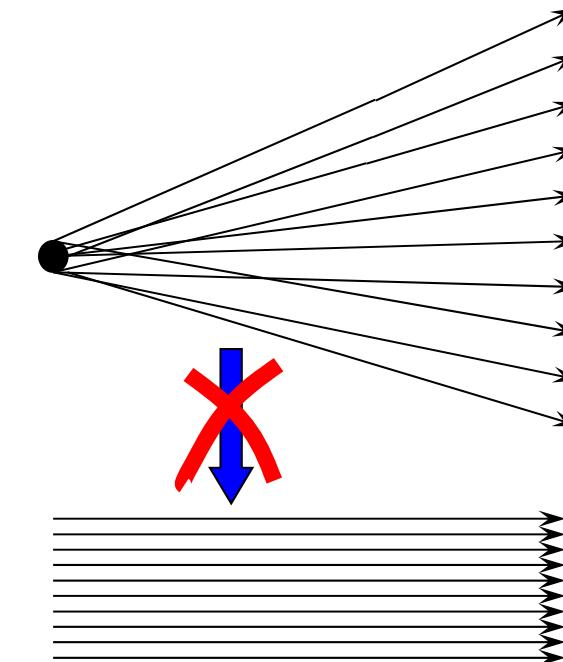




Liouvilles Theorem

- Conservation laws:
 - neutrons can't be created from thin air
 - neither can “phase space density”
- There is no such thing as a free lunch
 - Beam manipulation transfers distribution between time, area, divergence, energy
- Most common application:
 - Focusing increases divergence
 - improve flux, lose angular resolution

Liouville theorem
 $\rightarrow \Delta x \Delta \phi = \text{const}$

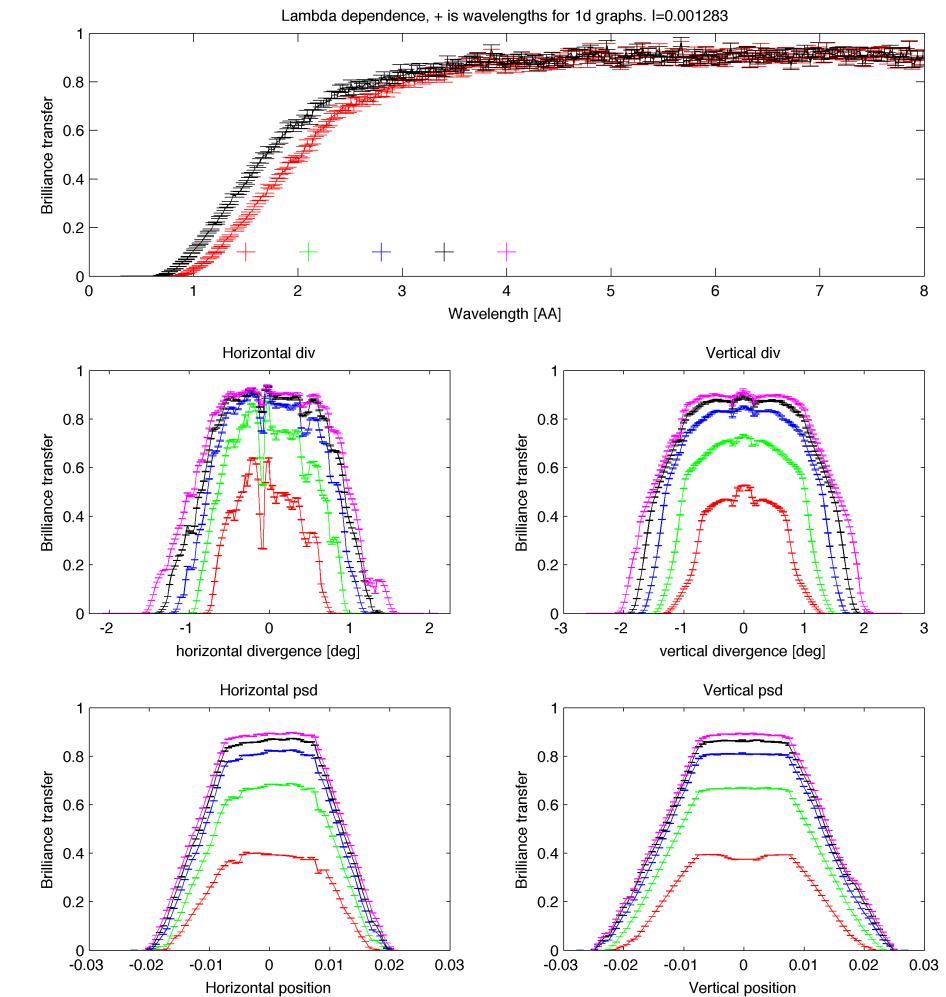
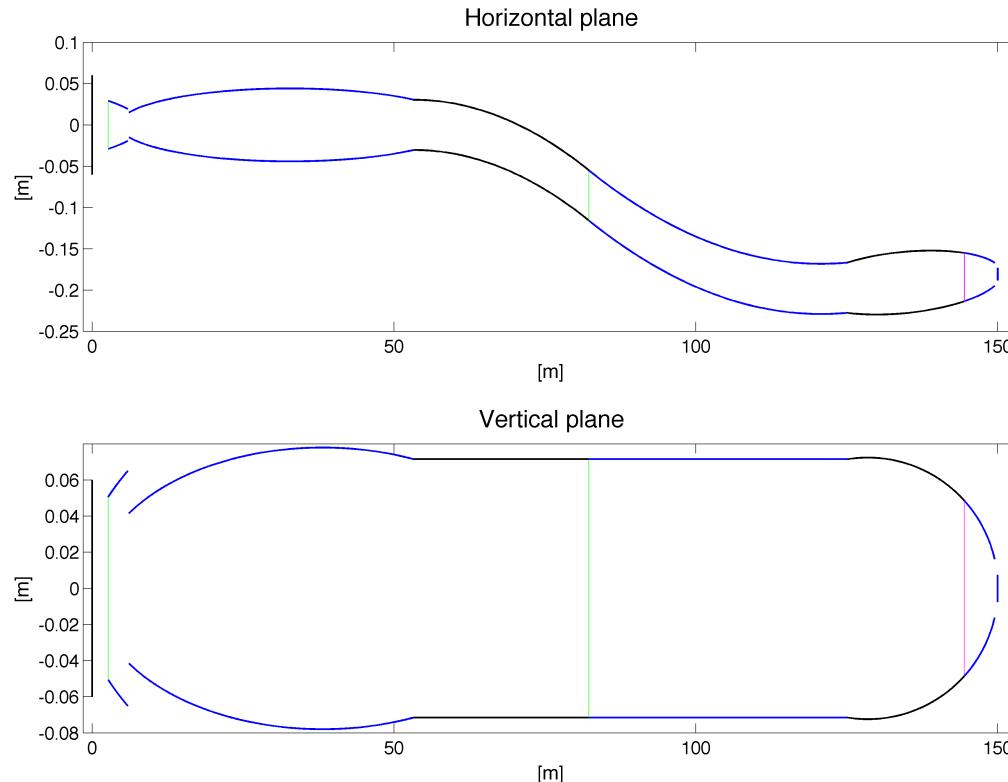


Integrated flux $\int \Psi dA d\Omega$ can never increase



A good optic maximises *transfer of brilliance*

- Example, elliptic guide designed by guide_bot
 See Thursday talk by Mads Bertelsen



How to implement this in McStas-code? See example folder...

- **BTsimple**, measures phase space before optic...

1. /* Measure incoming phase-space */
COMPONENT BT_in = L_monitor(xwidth=0.02, yheight=0.02, filename="BT_in.dat",
Lmin=lambda-dlambda, Lmax=lambda+dlambda, nL=101, restore_neutron=1)
WHEN ((VertDiv <= maxvd) && (HorDiv <= maxhd)) AT (0,0,2.0) RELATIVE origin

How to implement this in McStas-code?

See example folder...

- **BTsimple**, measures phase space **before** optic...

1.

```
/* Measure incoming phase-space */  
COMPONENT BT_in = L_monitor(xwidth=0.02, yheight=0.02, filename="BT_in.dat",  
    Lmin=lambda-dlambda, Lmax=lambda+dlambda, nL=101, restore_neutron=1)  
WHEN ((VertDiv <= maxvd) && (HorDiv <= maxhd)) AT (0,0,2.0) RELATIVE origin
```

Directional range

Spatial ranges

Spectral range

How to implement this in McStas-code?

See example folder...

- **BTsimple**, measures phase space **before** and **after** optic...

```
1. /* Measure incoming phase-space */
COMPONENT BT_in = L_monitor(xwidth=0.02, yheight=0.02, filename="BT_in.dat",
                           Lmin=lambda-dlambda, Lmax=lambda+dlambda, nL=101, restore_neutron=1)
WHEN ((VertDiv <= maxvd) && (HorDiv <= maxhd)) AT (0,0,2.0) RELATIVE origin
```

Same phase-space element
and identical binning...

```
2. /* Measure outgoing phase-space @ sample position */
COMPONENT BT_out = L_monitor(xwidth=0.02, yheight=0.02, filename="BT_out.dat",
                           Lmin=lambda-dlambda, Lmax=lambda+dlambda, nL=101, restore_neutron=1)
WHEN ((VertDiv <= maxvd) && (HorDiv <= maxhd)) AT (0,0,2.0+gL) RELATIVE guide
```

How to implement this in McStas-code?

See example folder...

- **BTsimple**, measures phase space **before** and **after** optic, calculates transferred fraction as fct. of wavelength:

1.

```
/* Measure incoming phase-space */
COMPONENT BT_in = L_monitor(xwidth=0.02, yheight=0.02, filename="BT_in.dat",
    Lmin=lambda-dlambda, Lmax=lambda+dlambda, nL=101, restore_neutrons=0);
WHEN ((VertDiv <= maxvd) && (HorDiv <= maxhd)) AT (0,0,2.0) RELATIVE origin
```

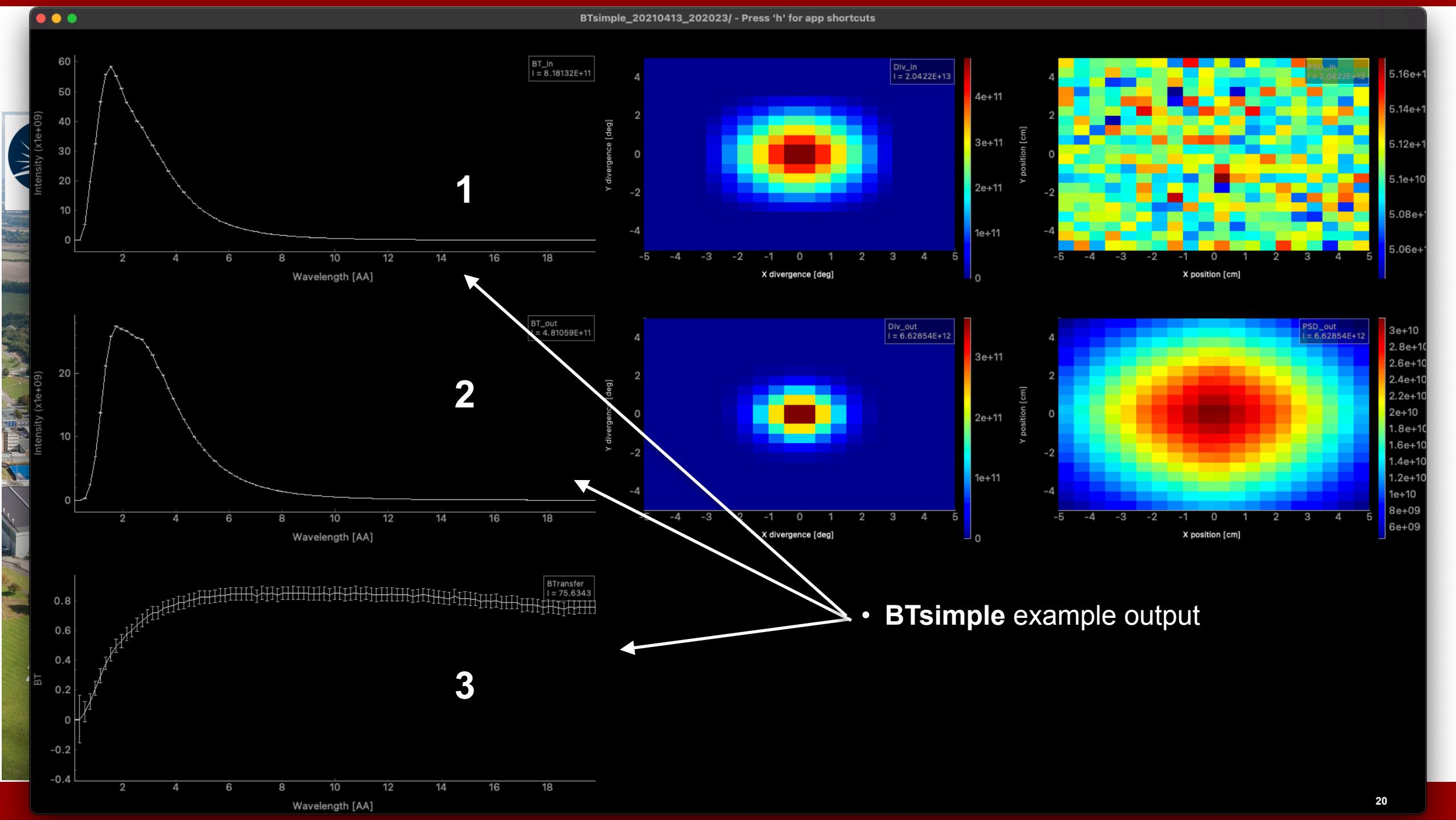
2.

```
/* Measure outgoing phase-space @ sample position */
COMPONENT BT_out = L_monitor(xwidth=0.02, yheight=0.02, filename="BT_out.dat",
    Lmin=lambda-dlambda, Lmax=lambda+dlambda, nL=101, restore_neutrons=0);
WHEN ((VertDiv <= maxvd) && (HorDiv <= maxhd)) AT (0,0,2.0+gL) RELATIVE guide
```

- Performs piece-wise division of the bins w. error propagation

3.

```
FINALLY
%{
/* This adds another "monitor" that measures BT_out / BT_in */
int j;
double* tmpN;
double* tmpp1;
double* tmpp2;
double* tmpd1;
double* tmpd2;
tmpN=MC_GETPAR(BT_out,L_N);
tmpp1=MC_GETPAR(BT_in,L_p);
tmpp2=MC_GETPAR(BT_out,L_p);
tmpd1=MC_GETPAR(BT_in,L_p2);
tmpd2=MC_GETPAR(BT_out,L_p2);
for (j=0;j<101;j++) {
    BT_N[j]=tmpN[j];
    if (tmpp1[j] != 0) {
        BT_p[j]=tmpp2[j]/tmpp1[j];
    } else {
        BT_p[j]=0;
    }
    if ((tmpp1[j] != 0) && (tmpp2[j] != 0)) {
        BT_p2[j]=sqrt((tmpd1[j]/tmpp1[j])*(tmpd1[j]/tmpp1[j])
                      + (tmpd2[j]/tmpp2[j])*(tmpd2[j]/tmpp2[j]));
    } else
        BT_p2[j]=0;
}
// This set of defines is to avoid getting a '.' in the component name
#ifndef NAME_CURRENT_COMP
#define NAME_CURRENT_COMP
#endif
#define NAME_CURRENT_COMP "BTransfer"
#endif
DETECTOR_OUT_1D(
    "Brilliance transfer",
    "Wavelength [AA]",
    "BT",
    "L", lambda-dlambda, lambda+dlambda, 101,
    &BT_N[0],&BT_p[0],&BT_p2[0],
    "Brilliance_transfer.dat");
%}
```





What to optimise? Figure of merit requires thought...

- Brilliance-transfer is probably a good quantity to think of, yet...
- Otherwise a case-specific FOM should be chosen
 - Usually not a matter of simply maximising flux...
 - Depends strongly on the science to be done
 - Wavelength-band, dynamical range of instrument
 - Typical sample size, q and E resolution needed etc.
 - Price (e.g. in the form of shielding requirements and guide coatings)
- See parallel sessions!

FOM examples for different types of instruments



STS04-41-TR0002, R00

Table 7.2 – Metrics proposed by the individual working groups

Instrument group	Flux metric	Shape metric
Small angle neutron scattering	$\max \left(\int_{1.5\text{\AA}}^{20\text{\AA}} IF(\lambda) \lambda^{2-3} d\lambda \right)$	N/A
Reflectometry	$\max \left(\int_{3\text{\AA}}^{15(20)\text{\AA}} IF(\lambda) \lambda^4 d\lambda \right)$	N/A
Spin-Echo	$\max \left(\int_{4\text{\AA}}^{20\text{\AA}} IF(\lambda) \lambda^{2-3} d\lambda \right)$	N/A
Spectroscopy	$\max \left(\int_{1\text{\AA}}^{10\text{\AA}} PF(\lambda) \lambda^0 d\lambda \right)$	$\min \left(\int_{1\text{\AA}}^{10\text{\AA}} \left \frac{dPF(\lambda)}{d\lambda} \right d\lambda \right)$
Nuclear (classical) diffraction	$\max \left(\int_{0.5(0.2)\text{\AA}}^{4\text{\AA}} PF(\lambda) \lambda^0 d\lambda \right)$	$\min \left(\int_{0.5(0.2)\text{\AA}}^{4\text{\AA}} \left \frac{dPF(\lambda)}{d\lambda} \right d\lambda \right)$
Magnetic diffraction	$\max \left(\int_{1\text{\AA}}^{10\text{\AA}} PF(\lambda) \lambda^0 d\lambda \right)$	$\min \left(\int_{1\text{\AA}}^{10\text{\AA}} \left \frac{dPF(\lambda)}{d\lambda} \right d\lambda \right)$

Here $IF(\lambda)$ is defined as the pulse-integrated flux and $PF(\lambda)$ is the pulse-peak flux for a particular wavelength λ .

STS04-41-TR0002, R00

**Long-Pulse Neutron Instrumentation Workshop,
August 26-28, 2009**
Report

R. K. Crawford (editor), M. Arai, C. Carlile, L. Chapon, G. Granroth, S. Langridge, K. Lefmann, F. Mezei, M. Monkenbusch, G. Muhrer, A. Wiedenmann

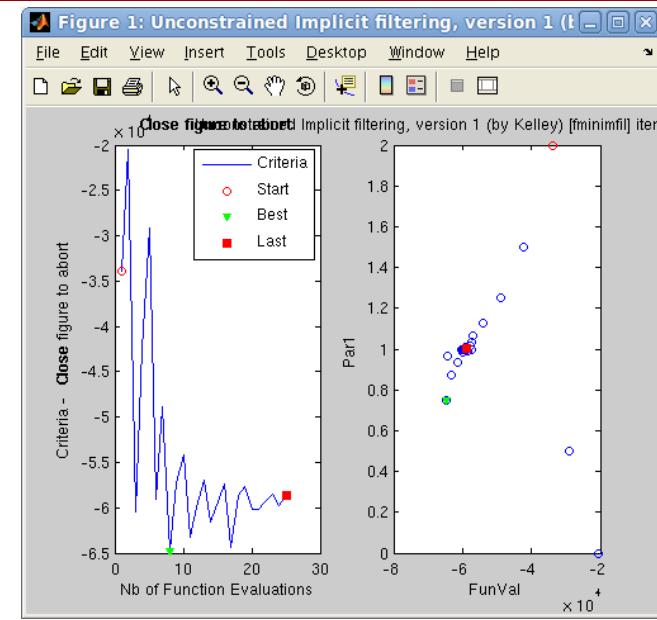
Rev. 00 - August 2010

Prepared by
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, Tennessee 37831-6285
managed by
UT-Battelle, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22



Parameter-variation

- mcrun and mcgui provide
 - Simple equidistant line-scans or colinear scans
`par=a,b -N k.` / `par1=a,b par2=c,d -Nk`
 - mcrun further provides
 - -L list mode (can be non-equidistant, user chooses order, still a form of scan)
`par1=a,b,c,d,e,f par2=g,h,i,j,k,l`
 - Run your simulations via iFit <http://ifit.mccode.org> - see <http://ifit.mccode.org/McStas.html>
 - McStas simulation becomes an object function, monitor(s) an optimisation criterion
 - Includes a host of different optimisers, e.g. particle swarms in parameter-space
 - Matlab-based, perform calculations based on monitor outputs (!! use Matlab <= 2017b !!)
 - Can work as a recompiled binary, but not as stable as a pure Matlab
 - Use official distribution package from <http://ifit.mccode.org/Downloads/iFit.zip>, GitHub code is too ‘bleeding edge’
- guide_bot tool is built on iFit, see Thursday talk. Relies on iFit v. x.y.



iFit