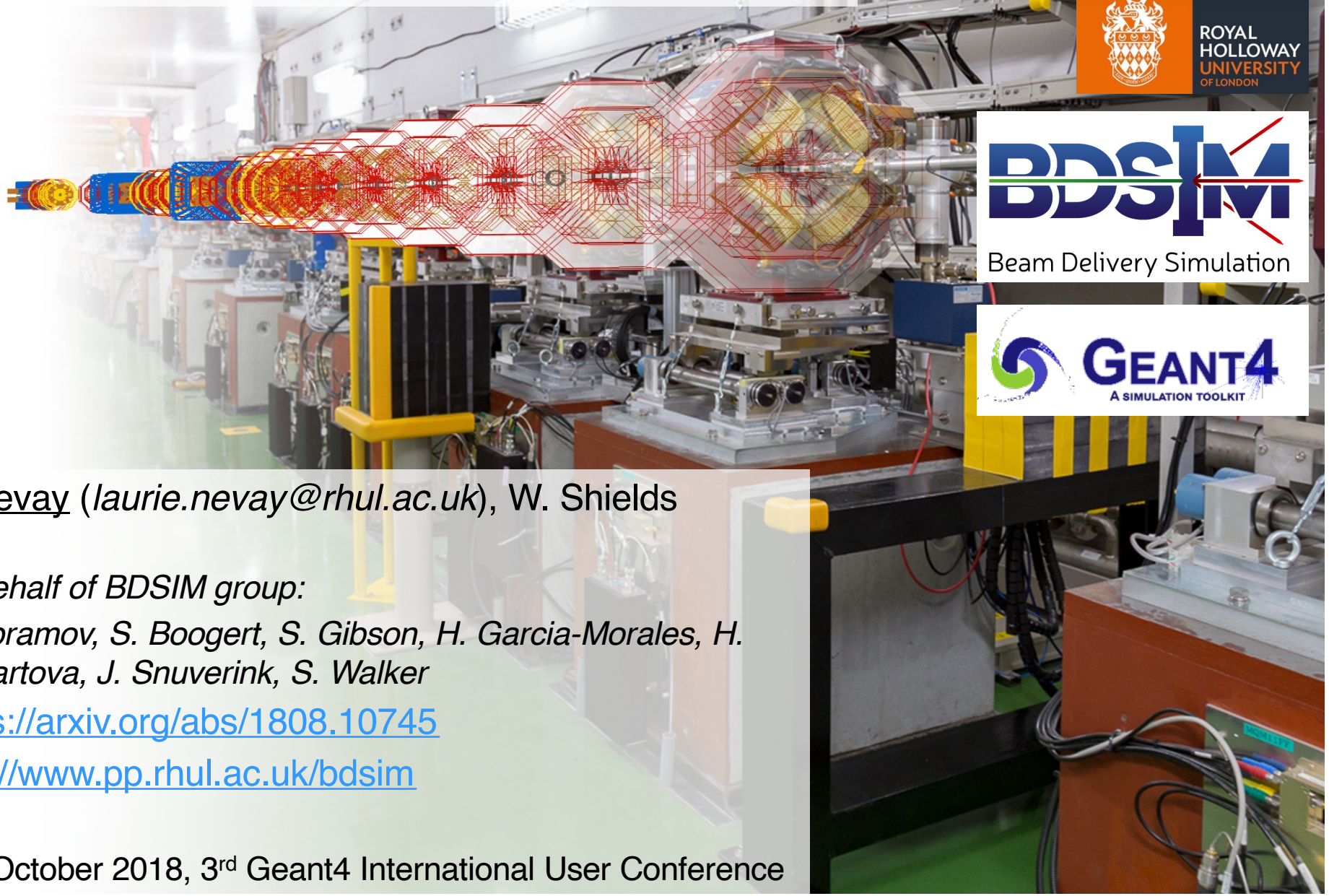
Start-to-End Accelerator Simulations in Geant4 with BDSIM



L. Nevay (*laurie.nevay@rhul.ac.uk*), W. Shields

on behalf of BDSIM group:

A. Abramov, S. Boogert, S. Gibson, H. Garcia-Morales, H. Pikhartova, J. Snuverink, S. Walker

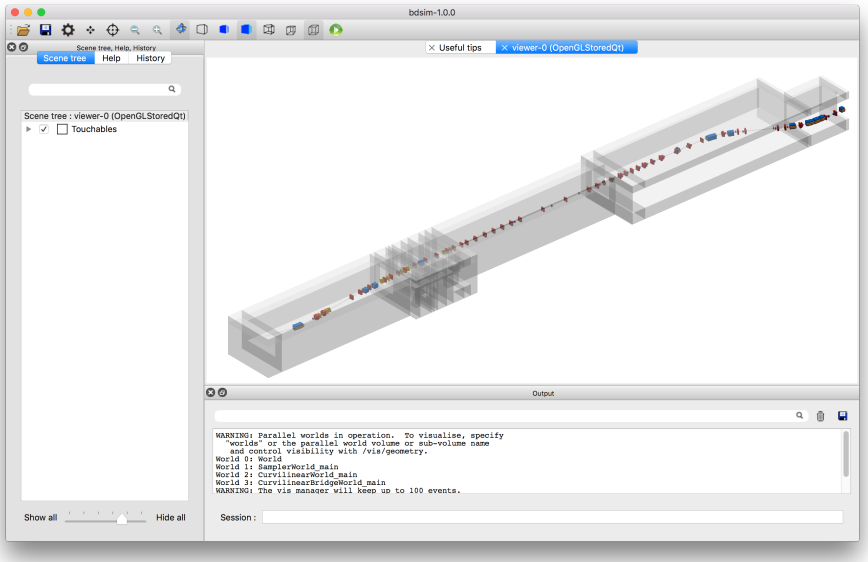
https://arxiv.org/abs/1808.10745

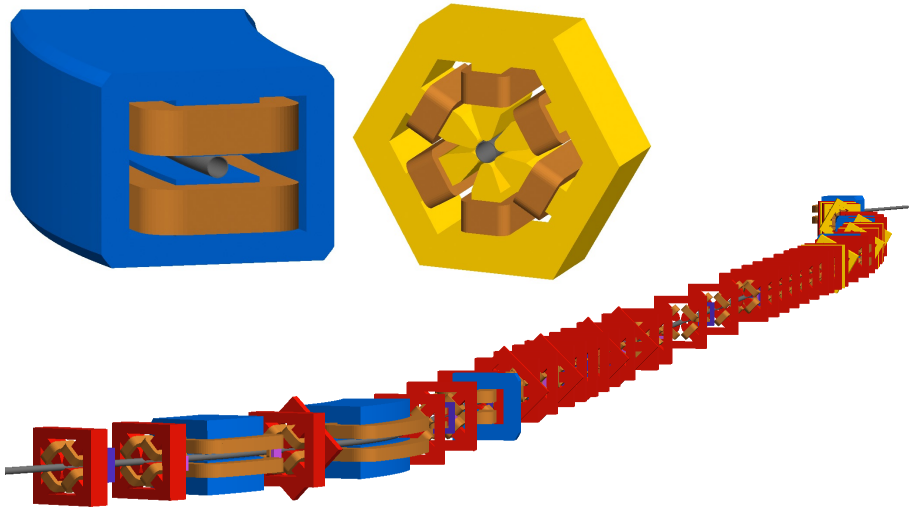
http://www.pp.rhul.ac.uk/bdsim

30th October 2018, 3rd Geant4 International User Conference

Beam Delivery Simulation

- Create Geant4 model of an accelerator from optical description in minutes
  - uses CLHEP, ROOT and Geant4
- Library of scalable generic accelerator geometry in Geant4 C++
  - you can learn a lot with generic geometry
  - scalable and safe from overlaps
- Can overlay other geometry and fields for more detail
- Thick lens 1st order matrices used for in-vacuum tracking
  - replaces Geant4's 4th order Runge-Kutta

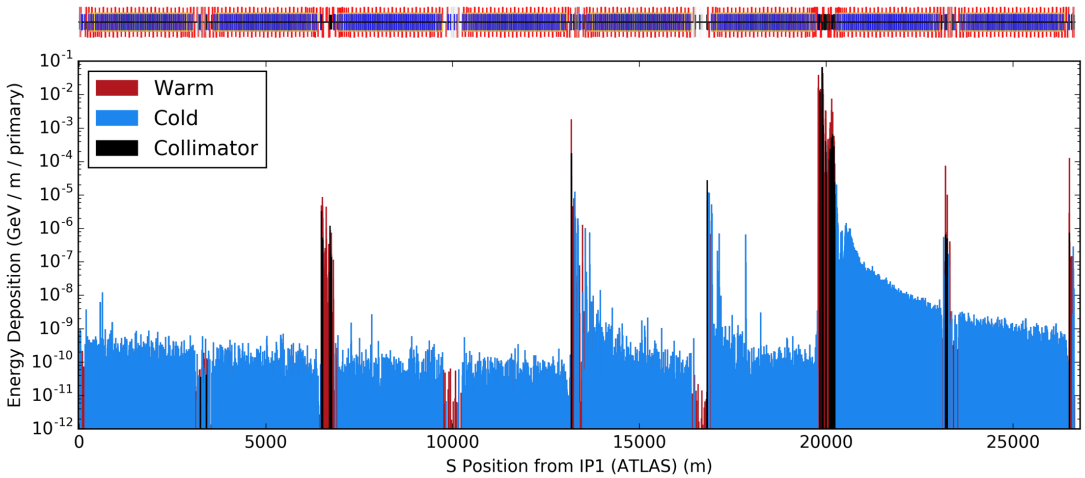




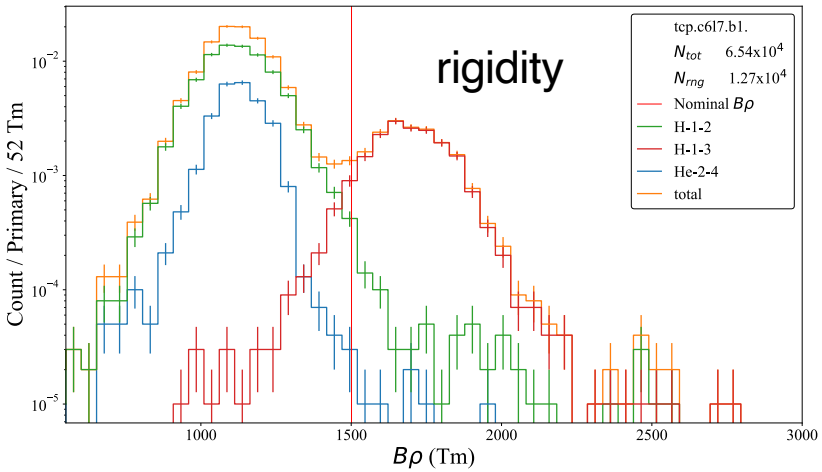
Current Uses

- Being used widely in high energy physics

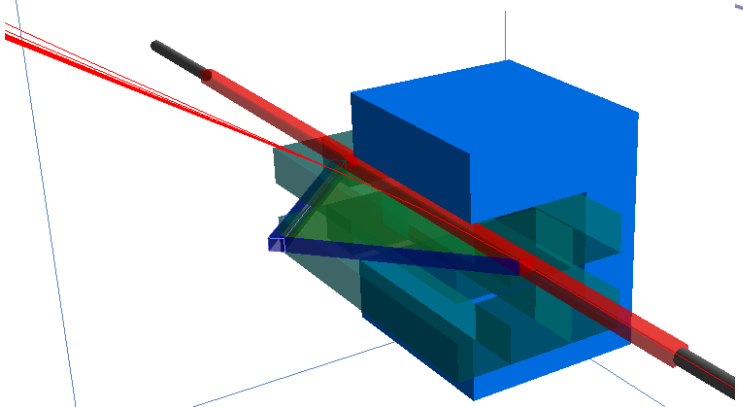
27km LHC model with ~20k beam line elements



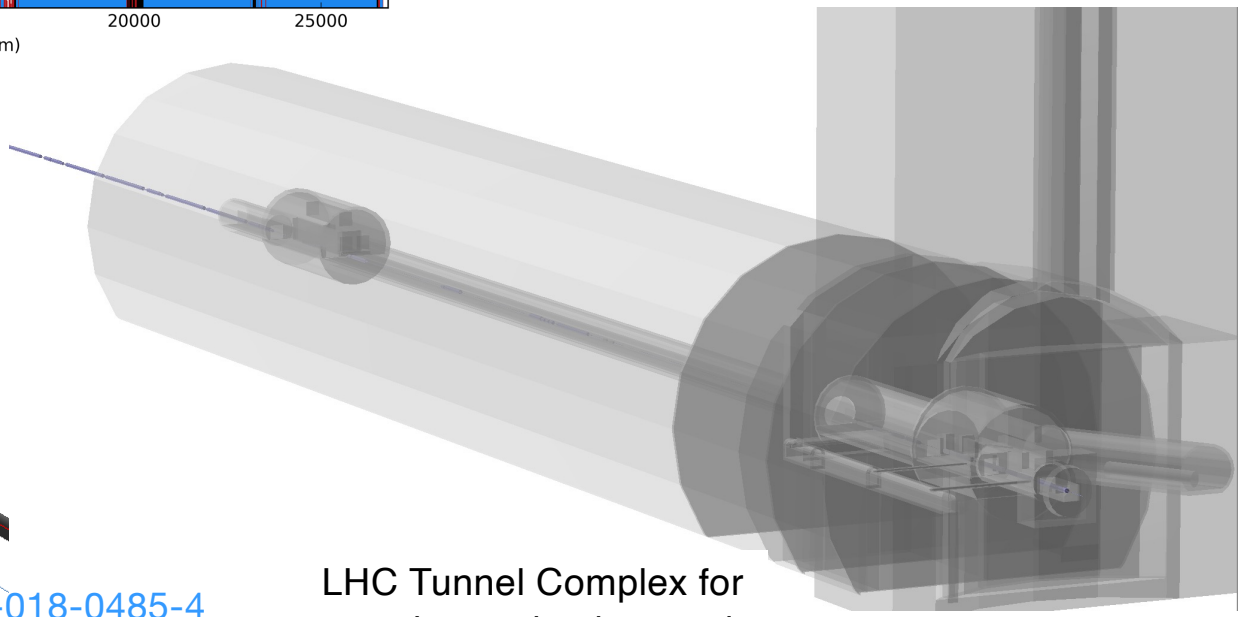
Ion fragmentation in LHC collimation system



AWAKE spectrometer calibration



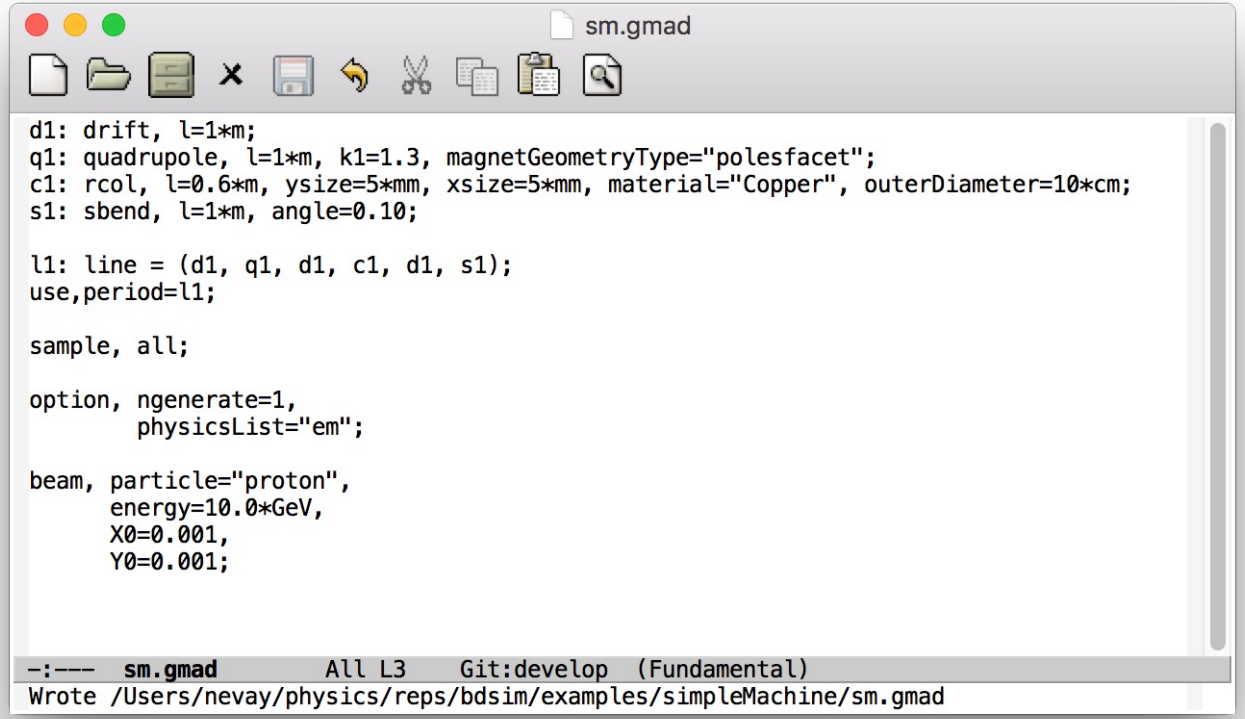
https://www.nature.com/articles/s41586-018-0485-4



LHC Tunnel Complex for accelerator background in detector

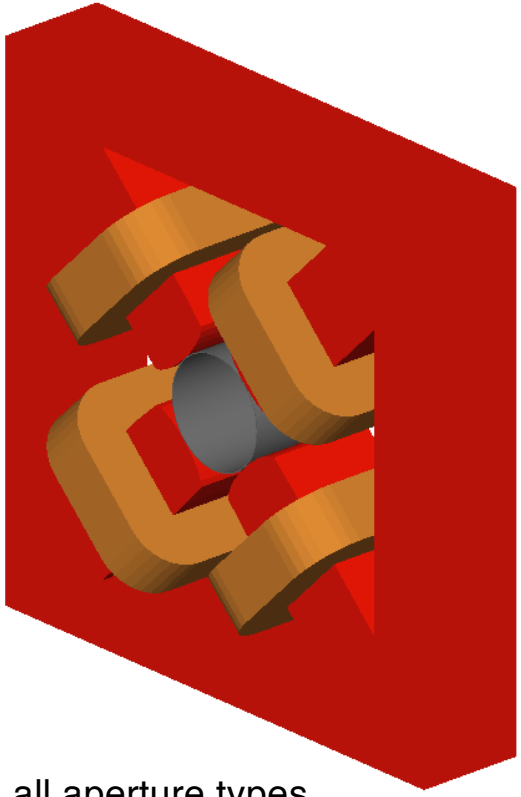
Input and Model Preparation

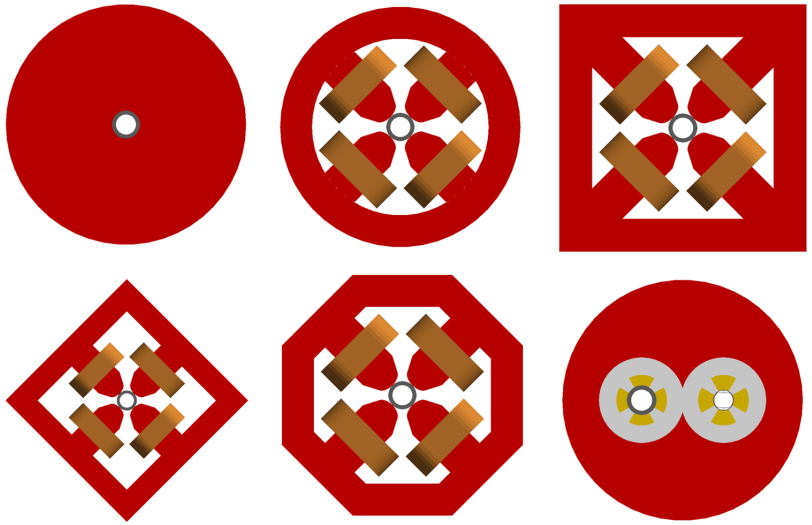
- Model defined by ASCII input (parser using Flex & Bison)
  - "GMAD" - Geant4 + MAD (common accelerator code)
- Can be written by hand for but also convert optical models
  - MADX, MAD8, TRANSPORT converters
  - pybdsim, pymadx, pymad8, pytransport Python utilities on PIP



Generic Geometry

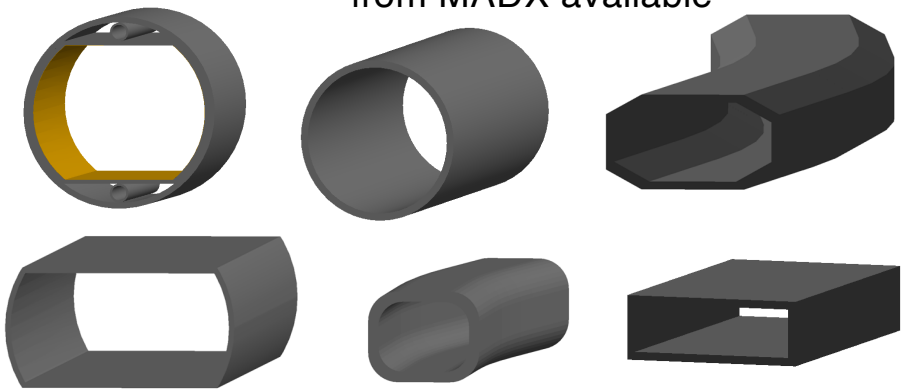
- Generic scalable geometry for each accelerator component
- 8 styles for each magnet
  - coils included correctly even if magnet split
  - proportions can be adjusted
  - C, H and LHC style dipoles
- 8 aperture types included
- All work together safely without overlaps



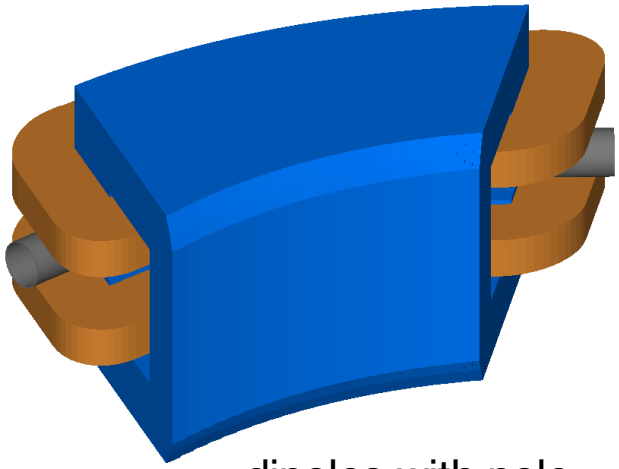


different yoke styles

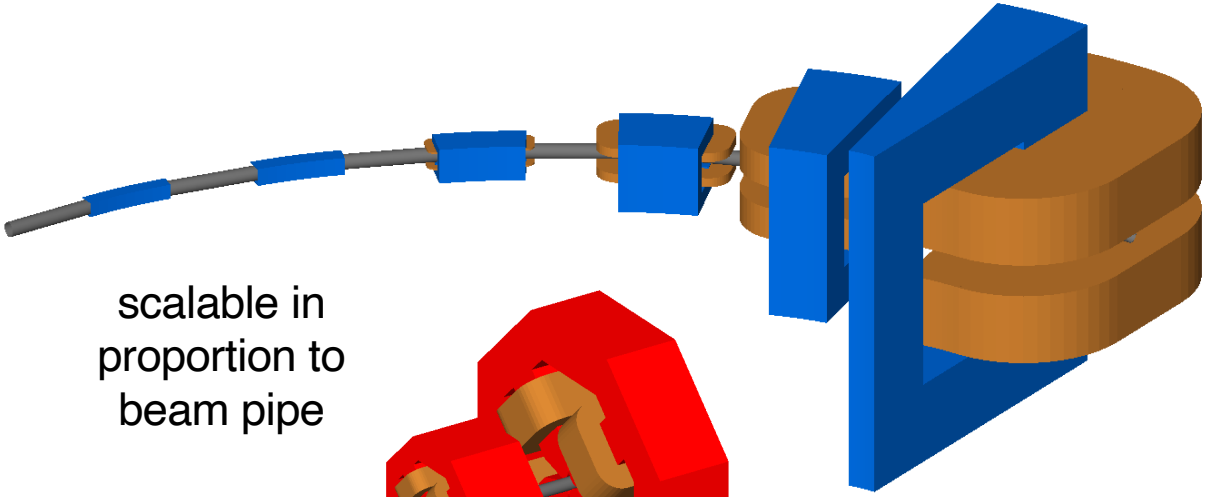
all aperture types
from MADX available



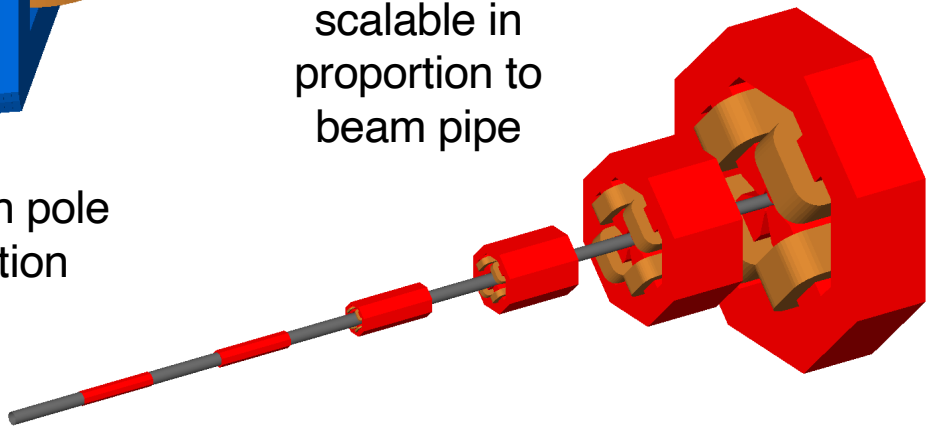
Scalable Geometry

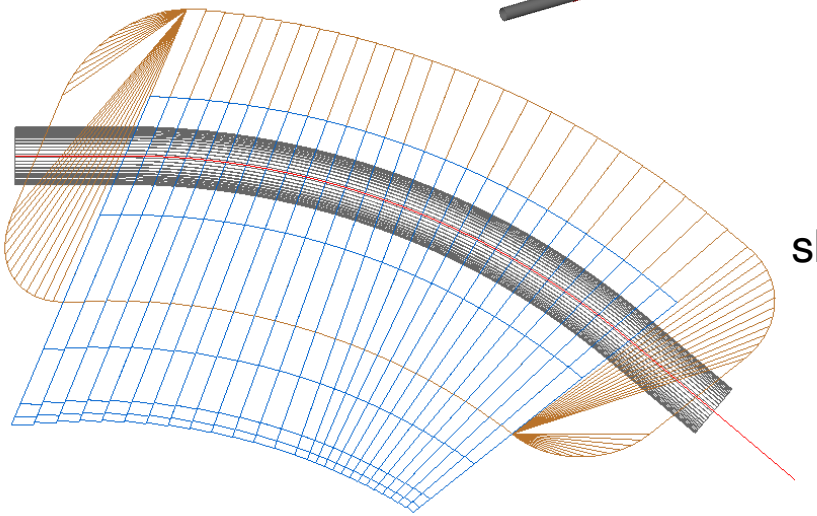


dipoles with pole face rotation

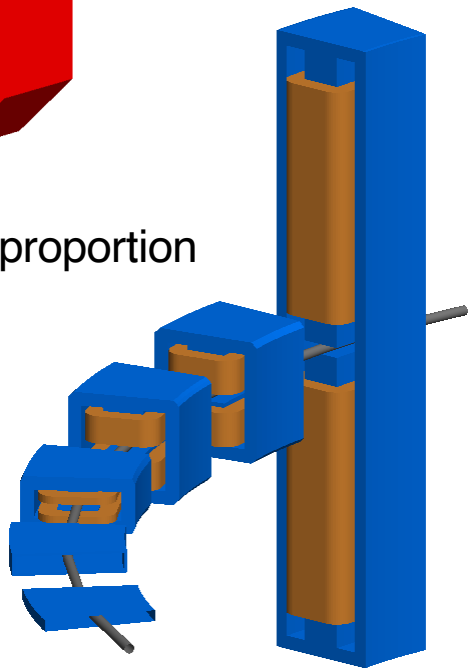


scalable in proportion to beam pipe





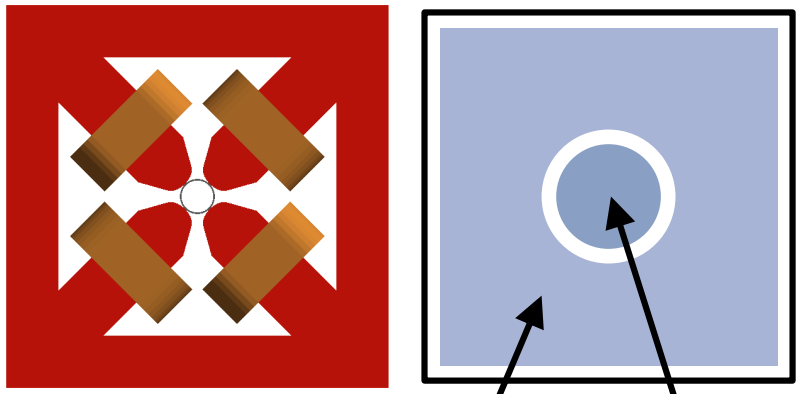
dipoles built in slices to allow any shape required

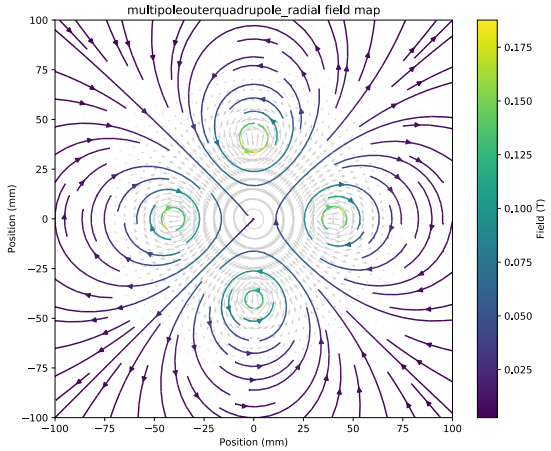


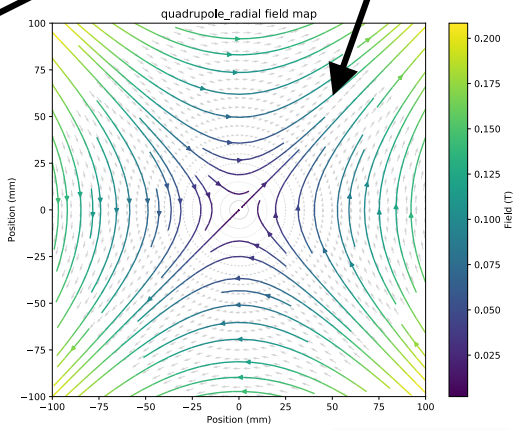
scalable in proportion

Tracking Implementation

- Custom numerical integrators
  - for 1st order matrix transport maps
- These ignore the field and are constructed with a strength
  - like "k1" for quadrupole
  - scaled with rigidity of particle
- Fall back to RK4 if…
  - non-paraxial
  - low radius of curvature (spiralling)
- Provide suitable fields
  - pure field in vacuum
  - current source yoke field
- Requires curvilinear coordinate system



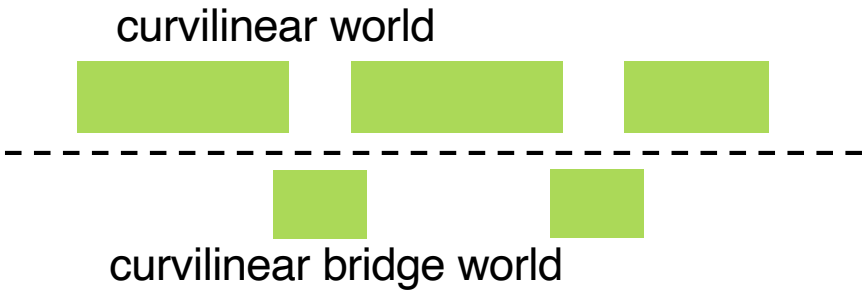


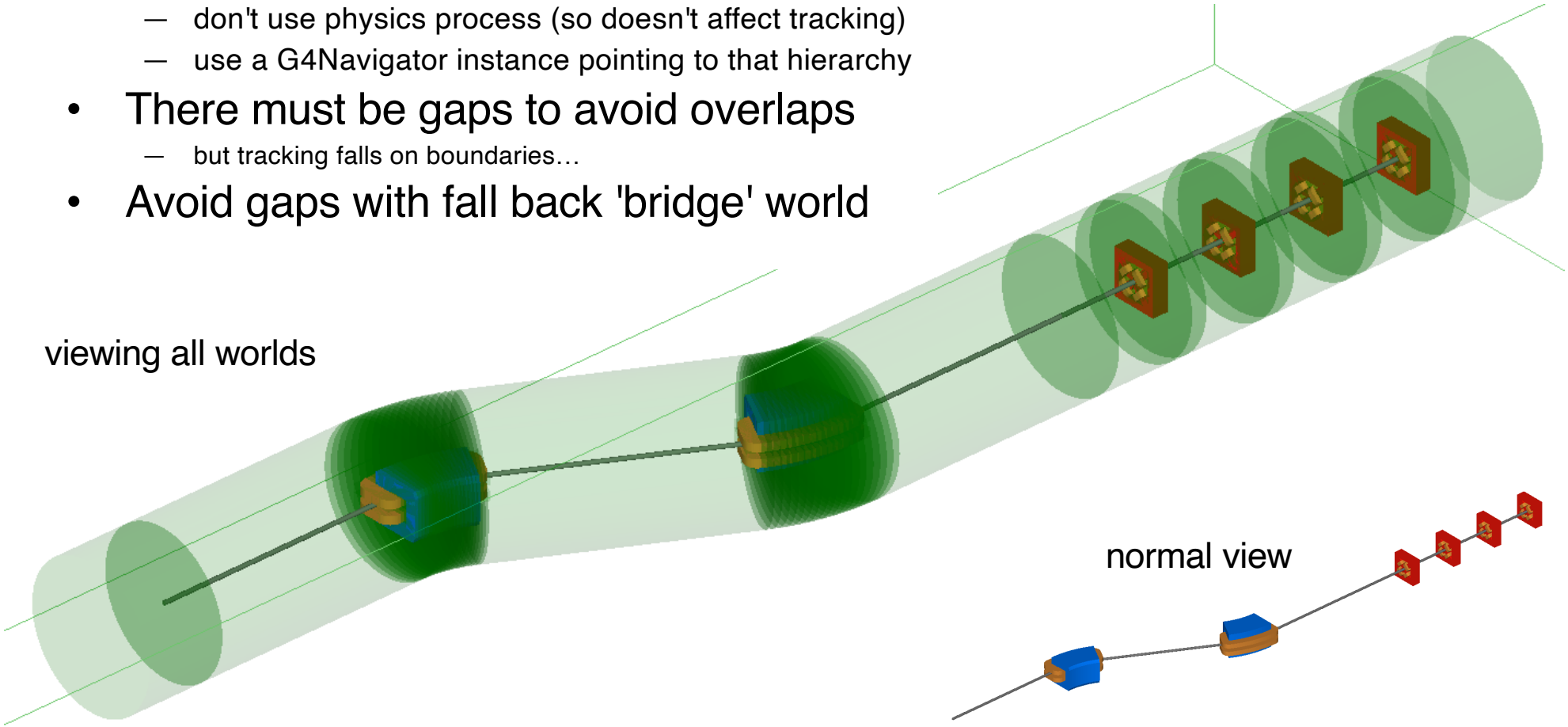


$$B_x = \frac{\partial B_y}{\partial x} y$$
$$B_y = \frac{\partial B_y}{\partial x} x$$
$$B_z = 0$$

Parallel Worlds

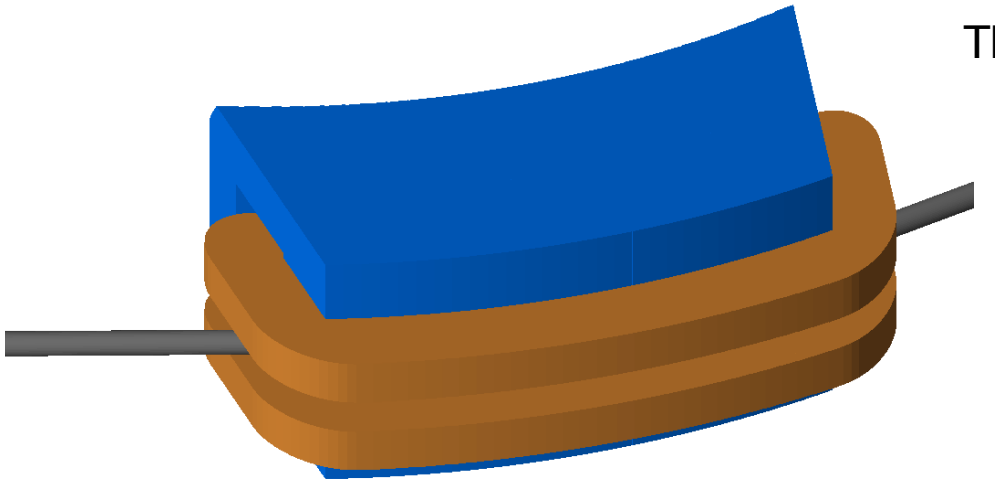
- Require curvilinear coordinate system
- Transform to current volume is not always the same
  - unknown level in hierarchy
- Use parallel world(s)
  - don't use physics process (so doesn't affect tracking)
  - use a G4Navigator instance pointing to that hierarchy
- There must be gaps to avoid overlaps
  - but tracking falls on boundaries…
- Avoid gaps with fall back 'bridge' world





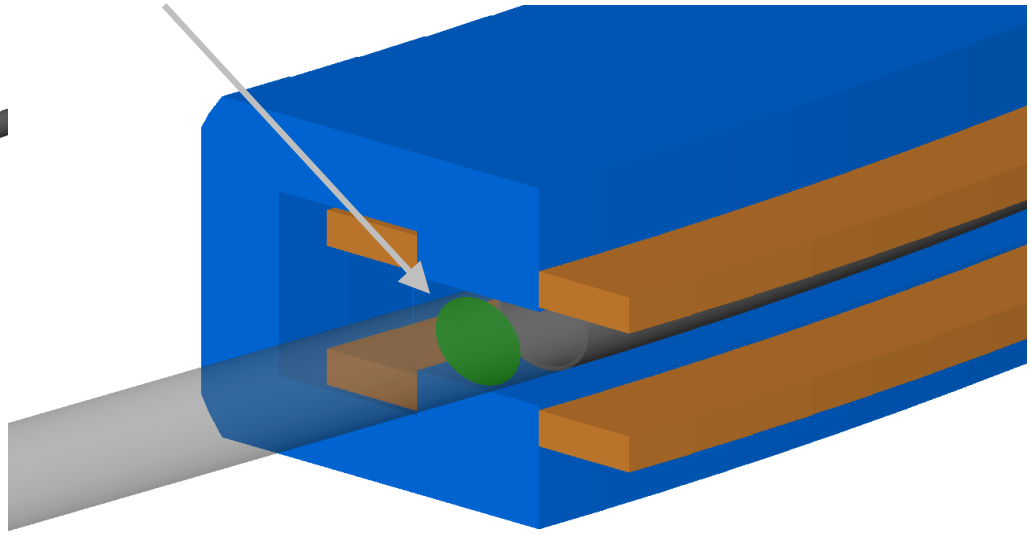
Thin Elements & Edge Effects

- Pole face angles cause focussing / defocussing that contributes significantly to optics
  - crucial for low energy applications
- Additional imperfections usually implemented via thin elements in tracking
  - entrance / exit or in the middle of magnet



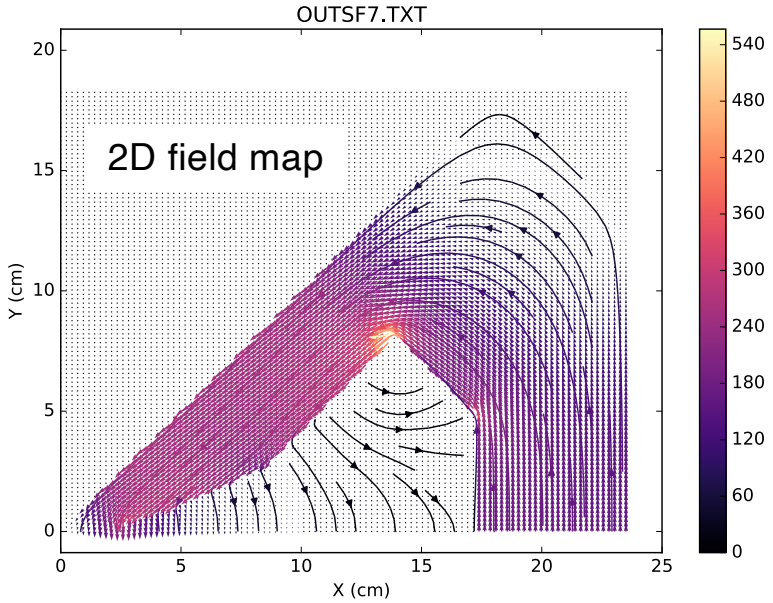
Angled beam pipe and yoke geometry as well as coils

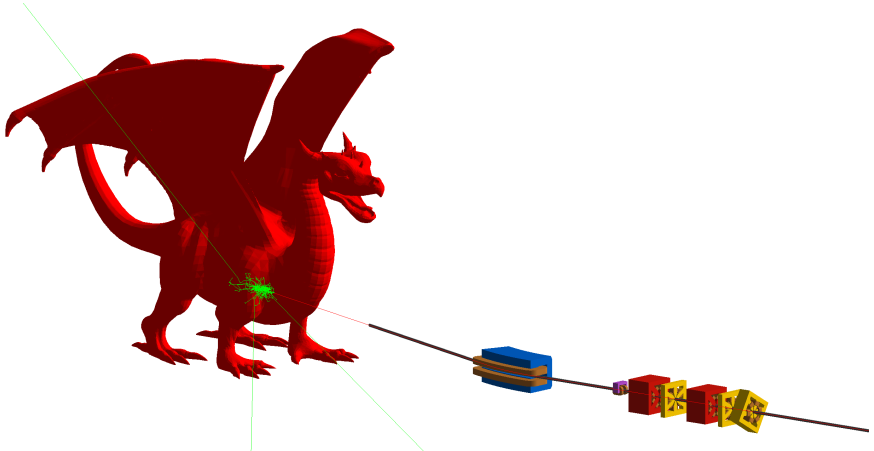
Thin element for fringe field



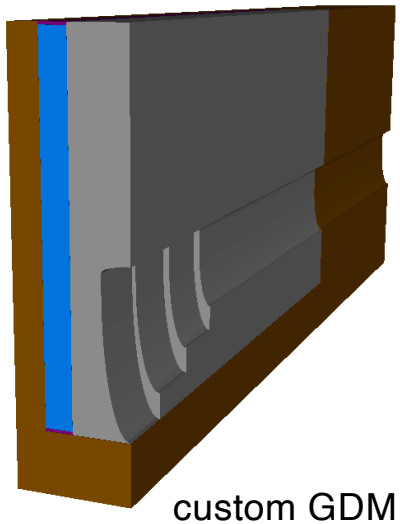
Customisation

- Can stick in custom pieces of geometry and overlay fields
- 1 - 4D field maps and interpolators provided
  - nearest neighbour, linear & cubic
- External geometry such as GDML can be placed in beam line
  - copes with degenerate names in GDML

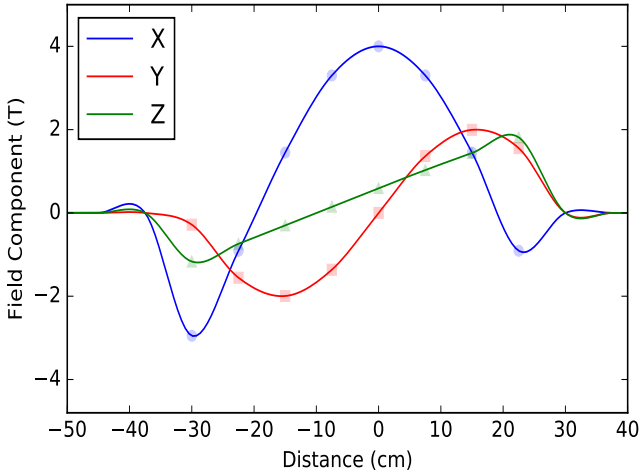




by a humorous PhD student…



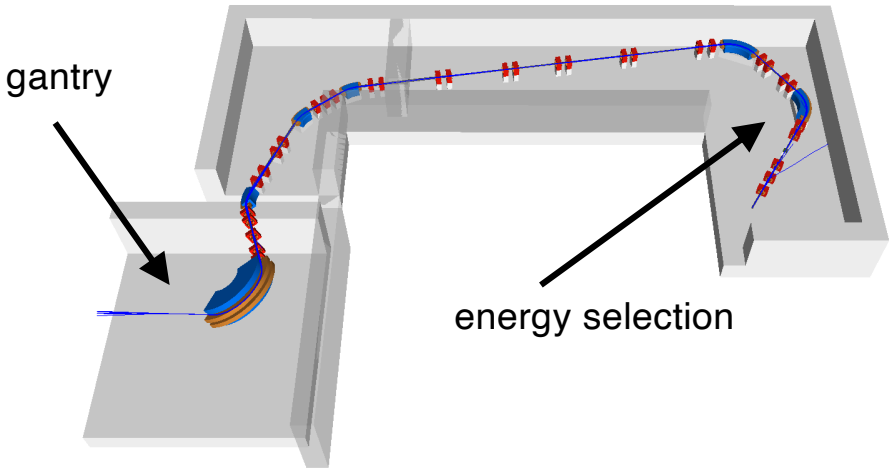
custom GDML collimator in pyg4ometry

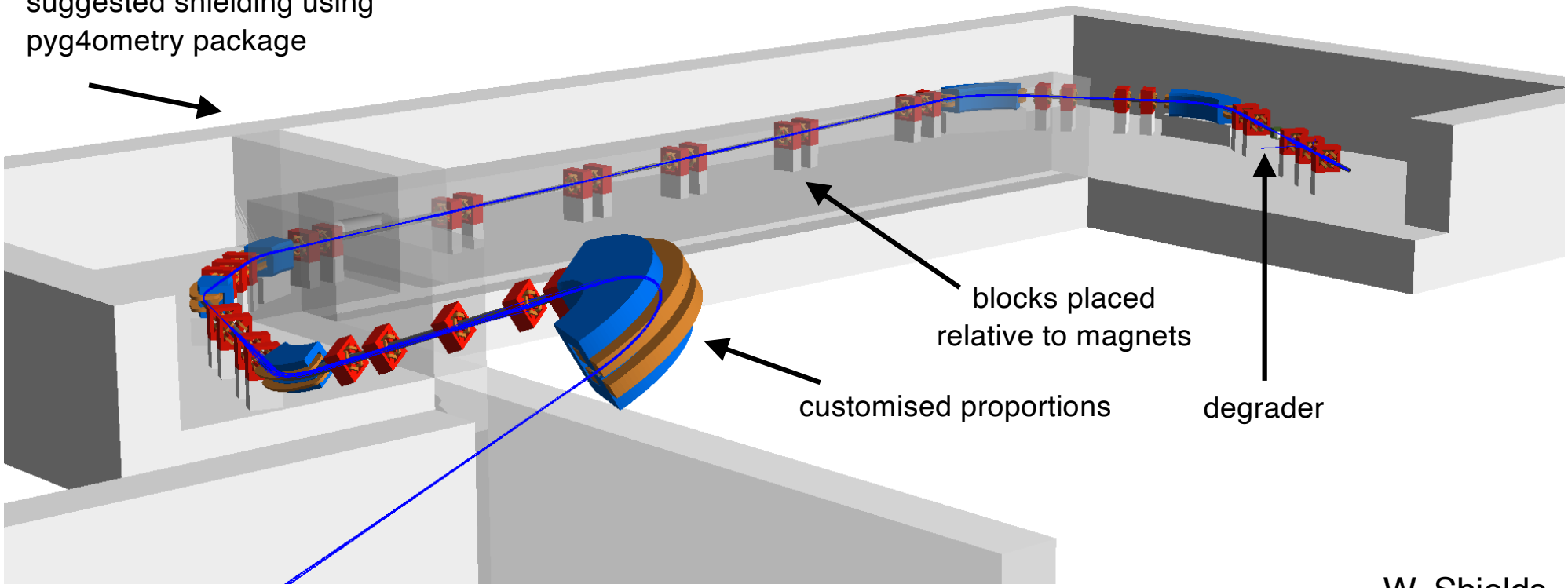


cubic interpolation

Example Model - PSI Gantry 2

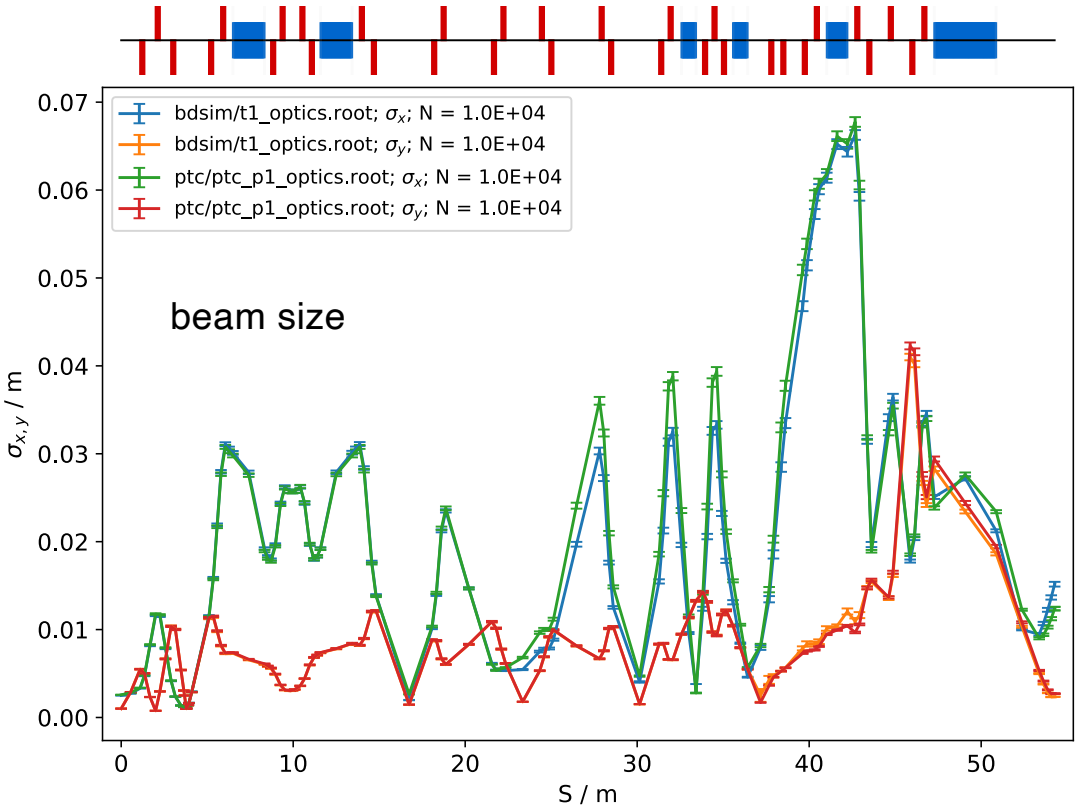
- Paul Scherrer Institute in Switzerland Gantry 2
  - lattice at publicly available http://aea.web.psi.ch/Urs_Rohrer/MyFtp
- Conceptual design - different machine build

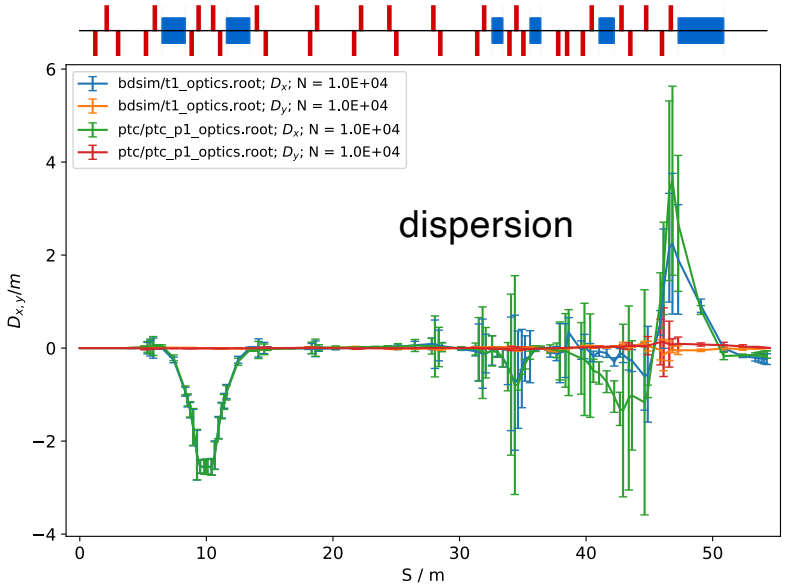




Optics Validation

- Record coordinates after every element
- Calculate various moments up to 4th order
- Calculate optical functions and sigma and variance
  - full statistical uncertainties calculated

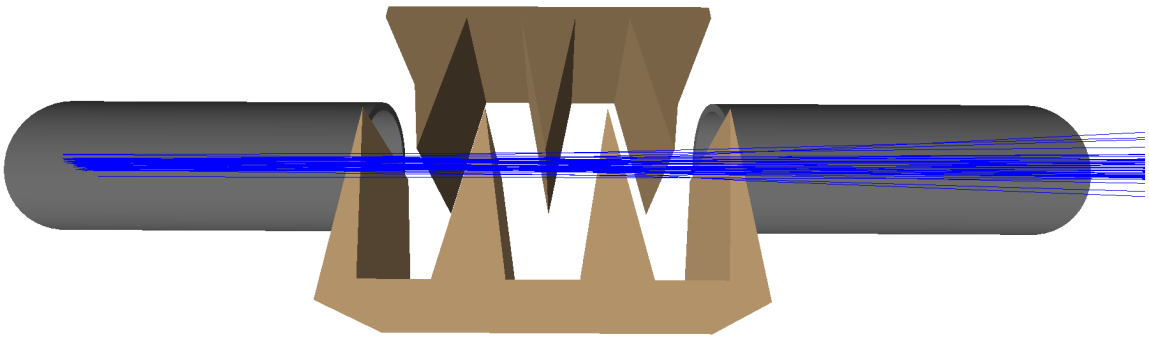


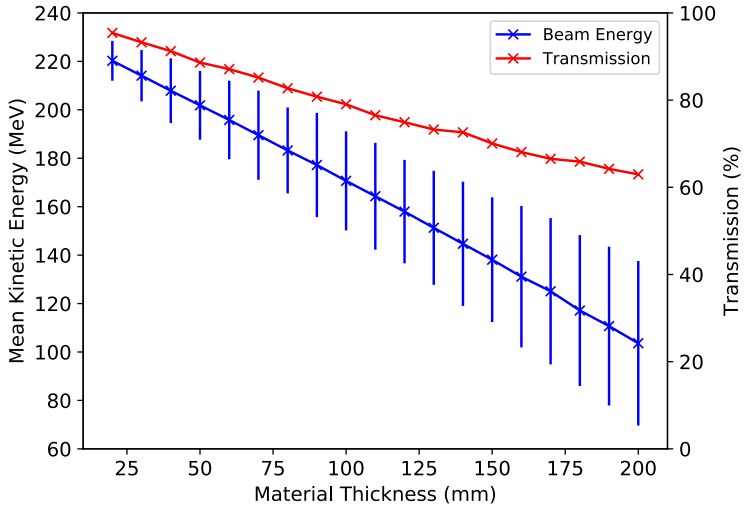


BDSIM vs PTC particle tracking

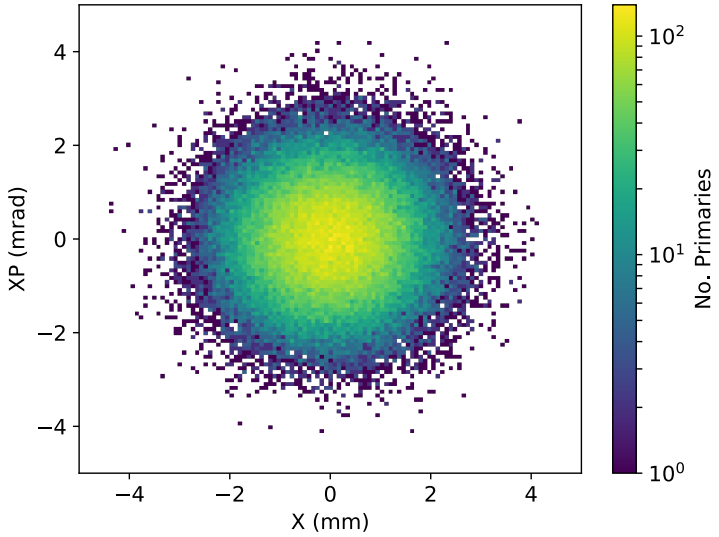
Hadron Therapy Degrader

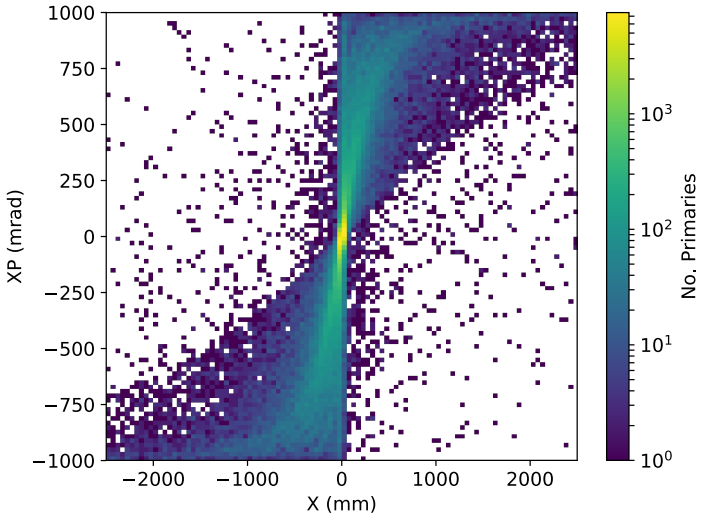
- Degrader component based on design of Center for Proton Therapy at PSI





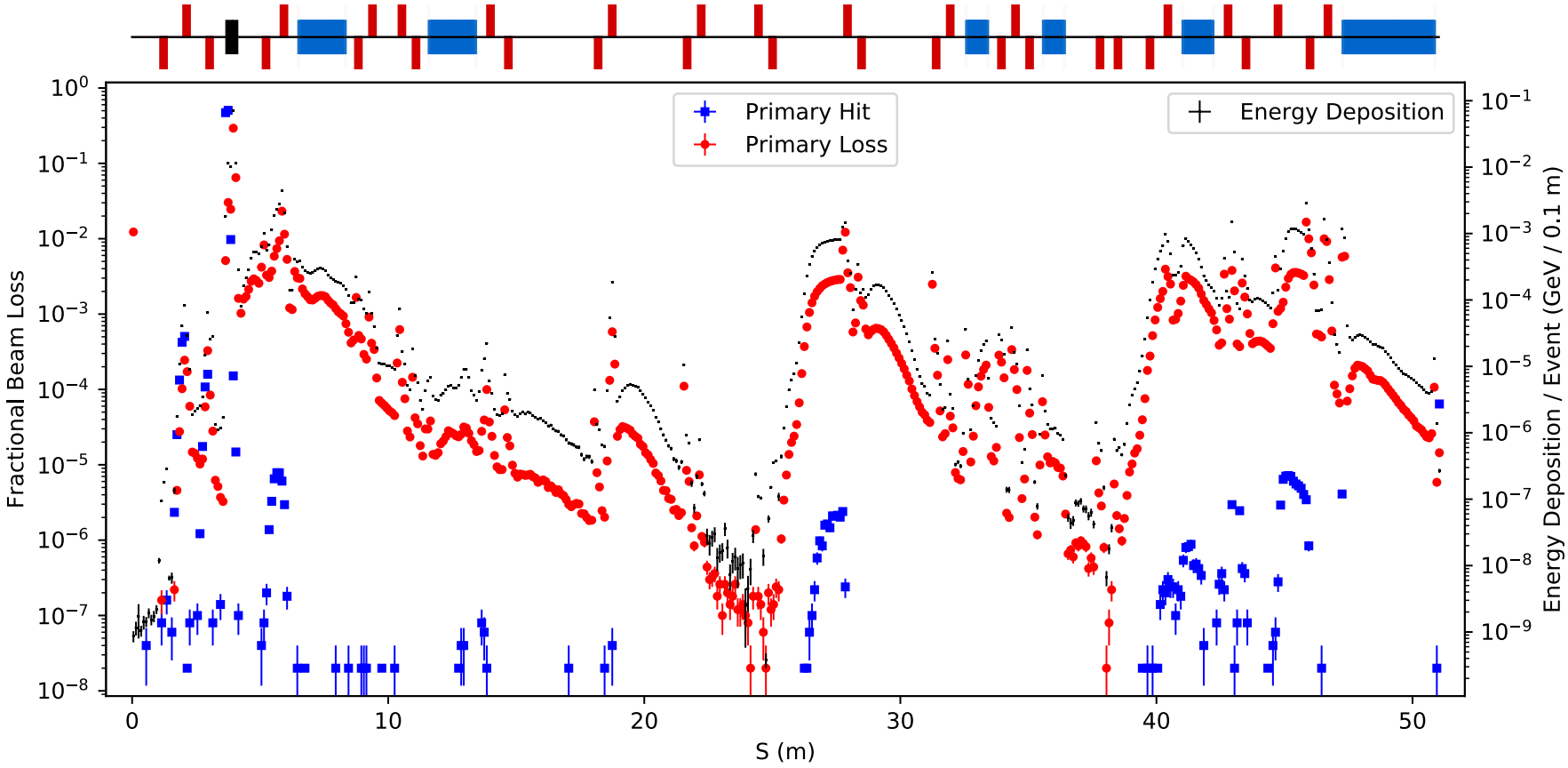
Horizontal phase space before (left) and after (right) a degrader.





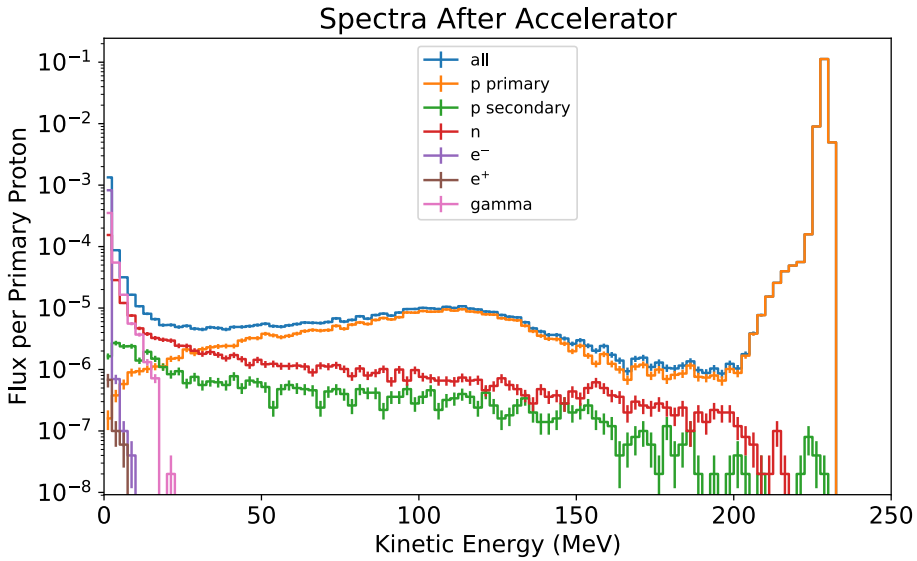
Beam Loss & Energy Deposition

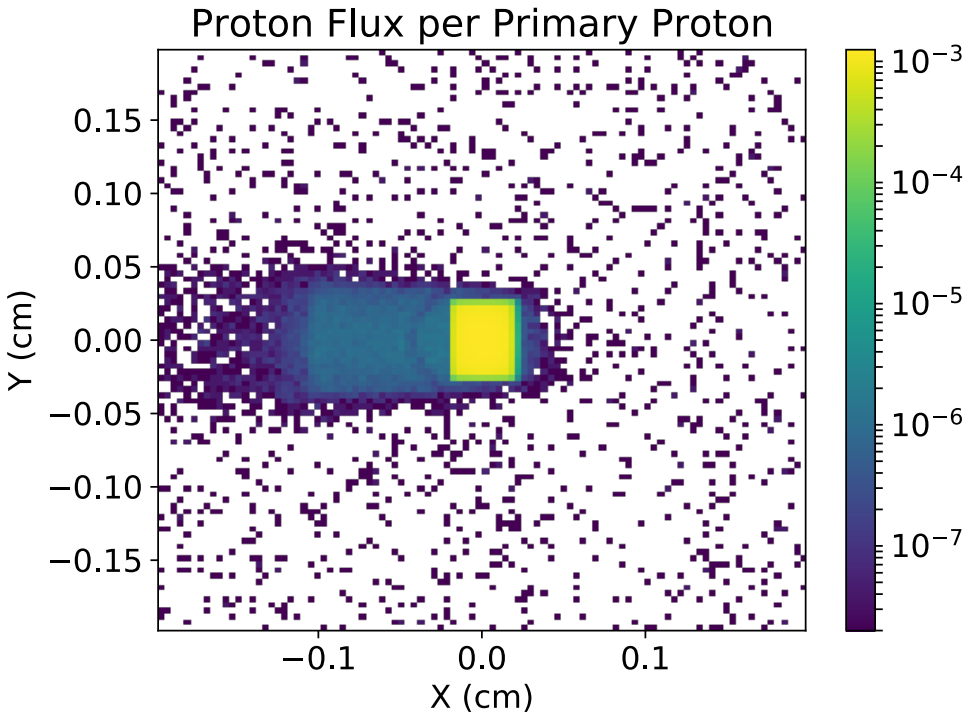
- 50M events with FTFP_BERT, standard EM, decay, hadronic elastic
- Energy deposition in machine along its length
- 250MeV protons degraded to 230MeV

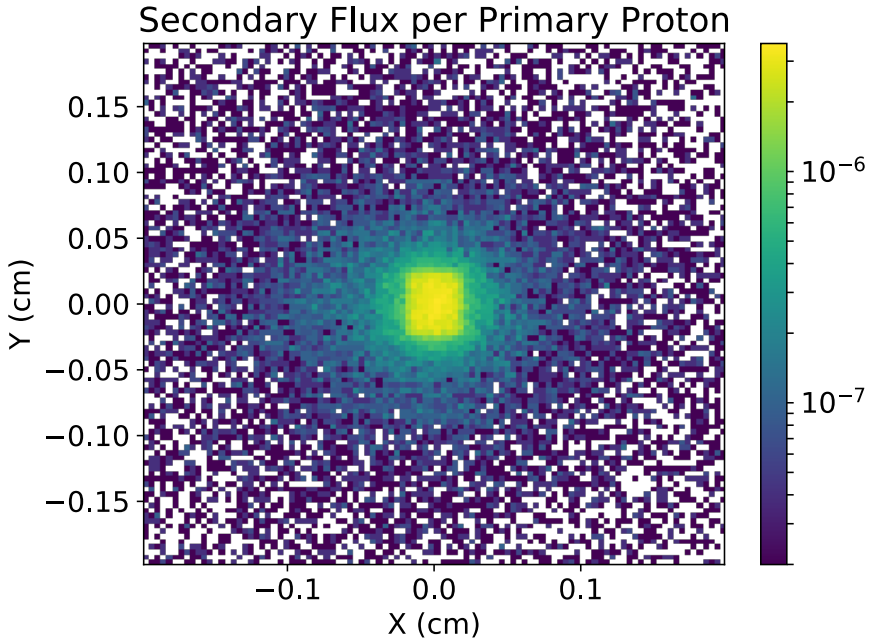


Beam Profiles

- Beam profiles after last magnet
  - before nozzle
- Flux of secondaries predicted
- Spectra of all particles
- Can include nozzle as external geometry







Summary & Outlook

- BDSIM is a tool to create Geant4 models of accelerators
- Flexible library of generic components
- Customisable with geometry and fields
- Ability to simulate complex environment of accelerator
- Exploit ongoing Geant4 physics development
- Ongoing active development
  - symplectic tracking being finished (separate tracker too)
  - possibility of collective effects from external library through tracker
  - introducing beam loss monitors for comparison to measured data
- Please get in contact for collaboration!


John Adams Institute
for Accelerator Science
ROYAL
HOLLOWAY
UNIVERSITY
OF LONDON
BDSIM
Beam Delivery Simulation
GEANT4
A SIMULATION TOOLKIT
Thank you
https://arxiv.org/abs/1808.10745
http://www.pp.rhul.ac.uk/bdsim

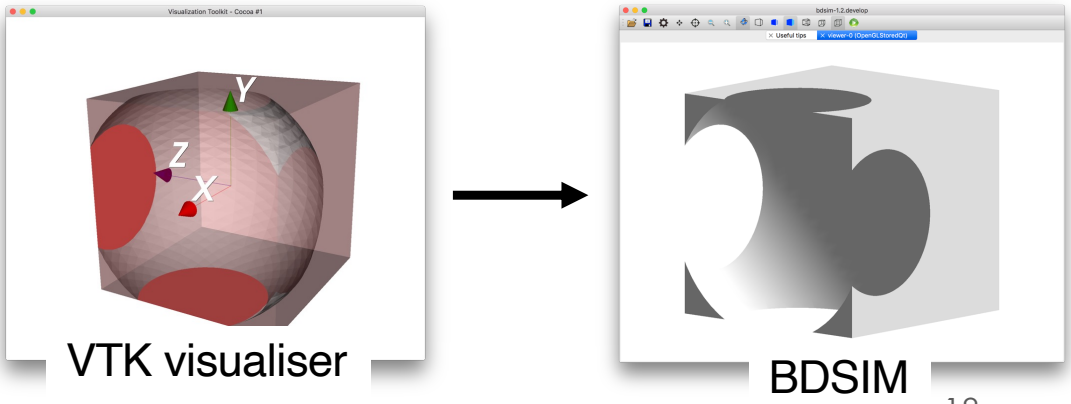
pyg4ometry

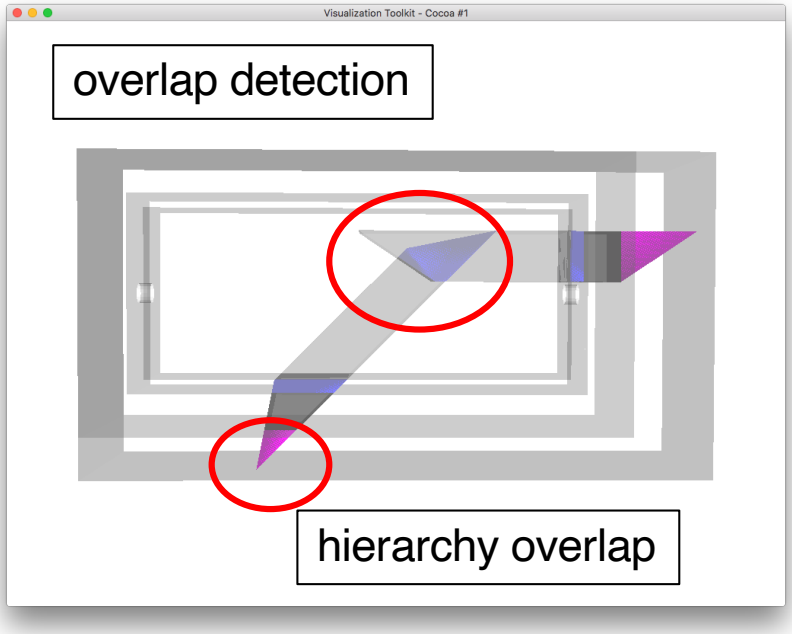
https://bitbucket.org/jairhul/pyg4ometry

- Python package to create Geant4 geometry in GDML
- Python class for each Geant4 primitive solid
- Combine with meshes from STL / STEP
- Generate meshes and uses VTK visualiser

```
boxSolid1    = _g4.solid.Box('box1',100,56,78)
boxLogical1  = _g4.LogicalVolume(boxSolid1,'G4_Cu','boxLogical1')
boxPhysical1 = _g4.PhysicalVolume([0,0,0],[0,0,0],boxLogical1,'boxPhysical1',worldLogical)
```

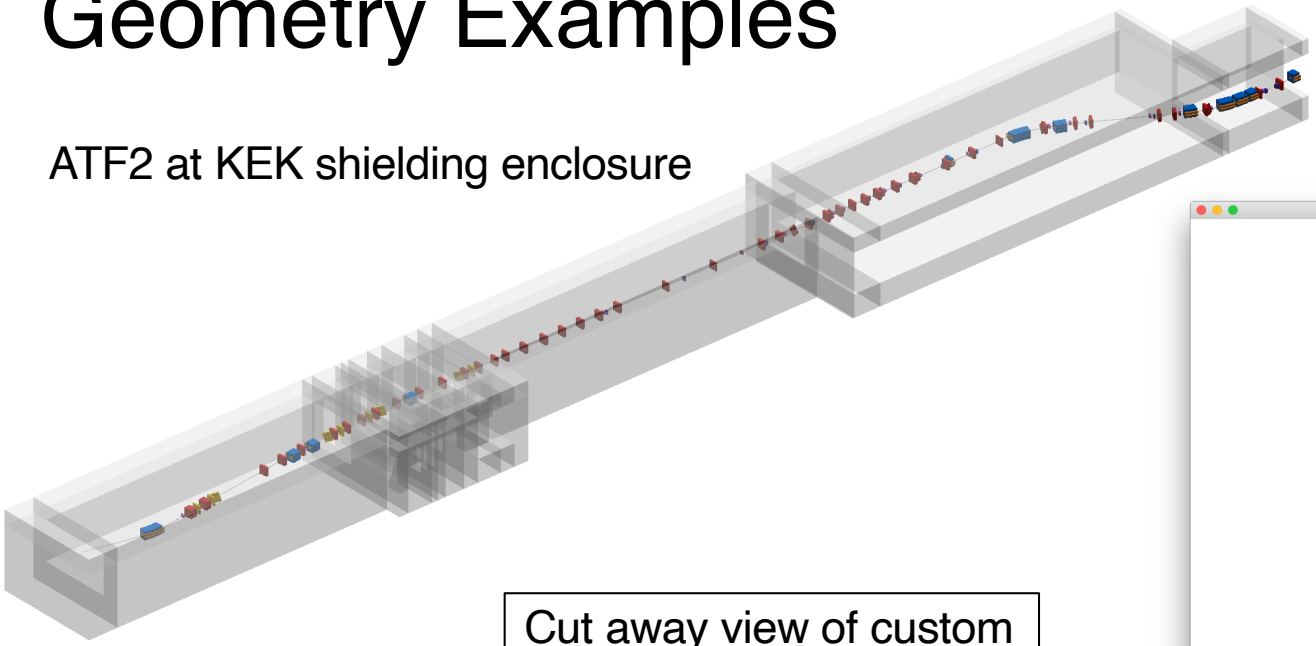
- Creates its own mesh
- Use mesh to identify overlaps
- Easy to create simplified pieces



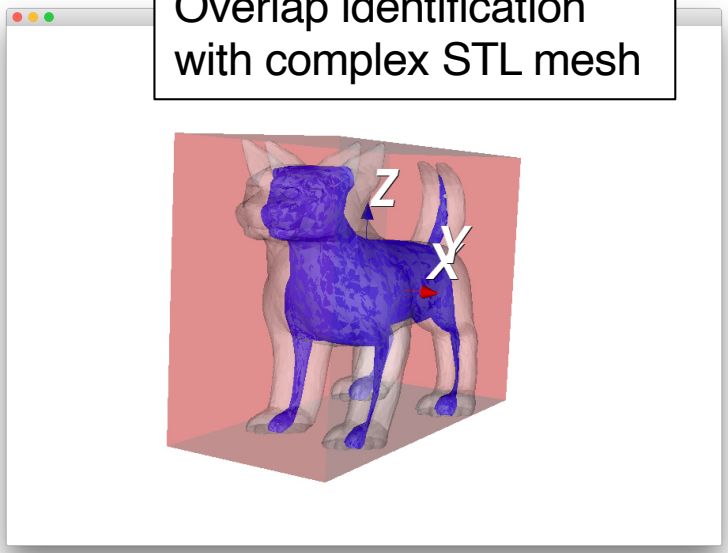


Geometry Examples

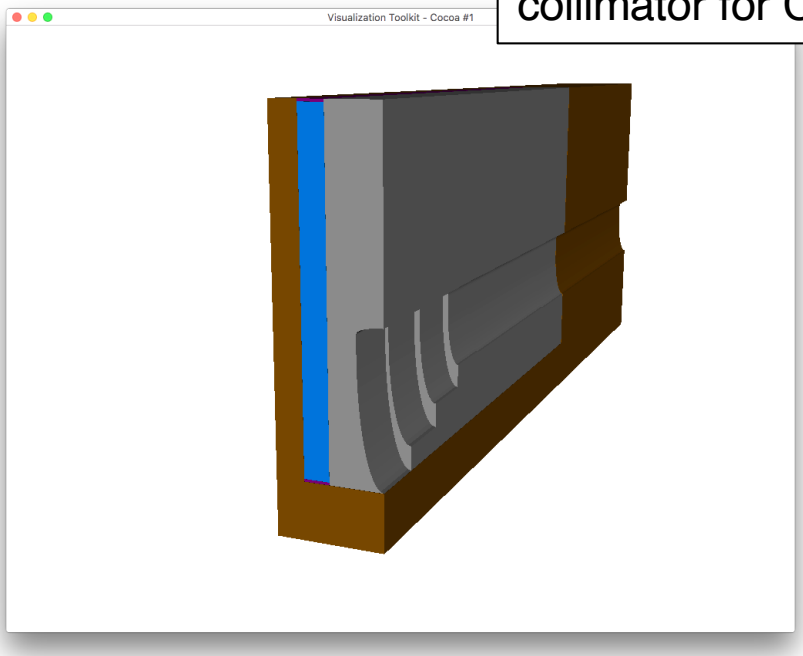
ATF2 at KEK shielding enclosure

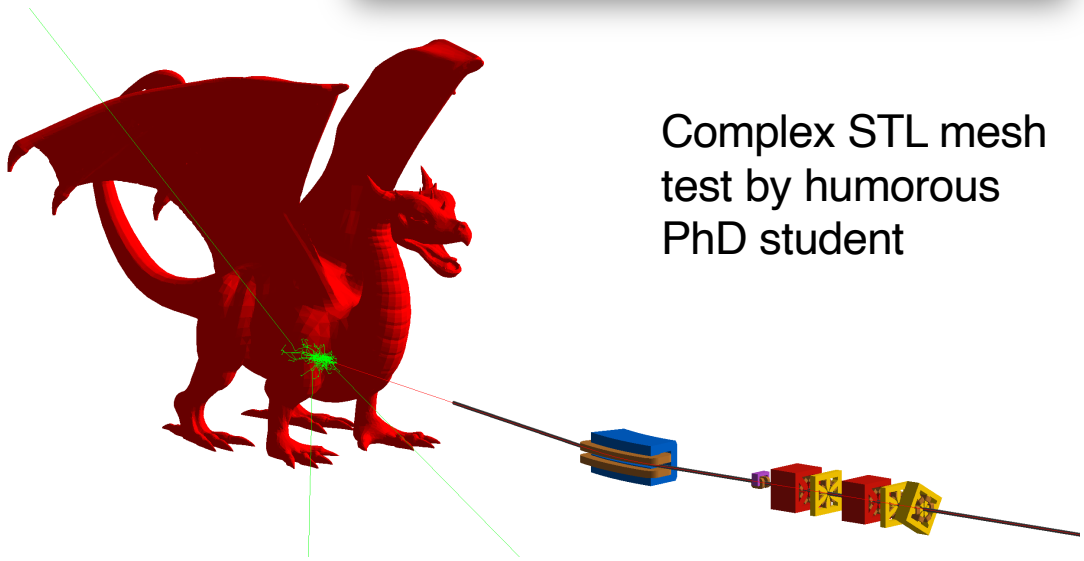


Overlap identification with complex STL mesh



Cut away view of custom collimator for CLIC





Complex STL mesh test by humorous PhD student