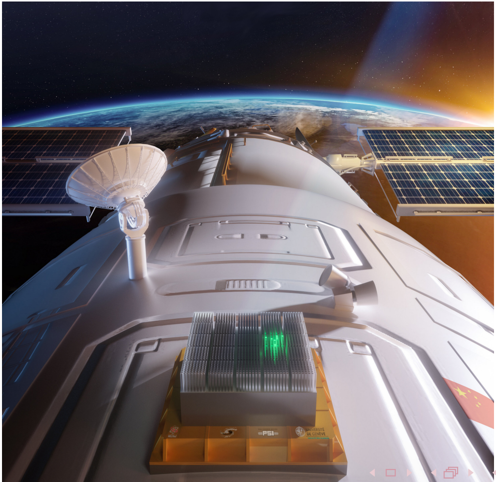
Simulating POLAR in Geant4

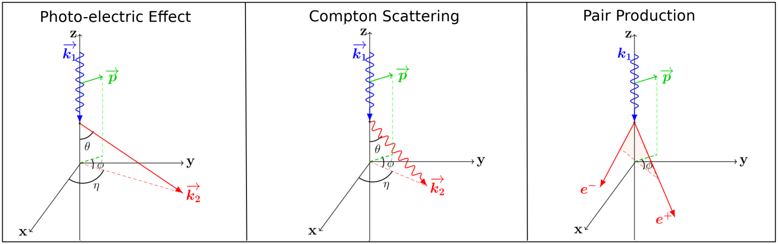
1st National Workshop on GEANT4

Merlin Kole



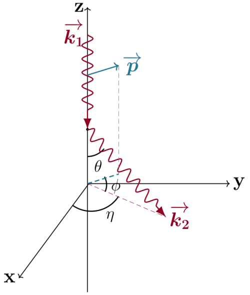
General Polarimetry

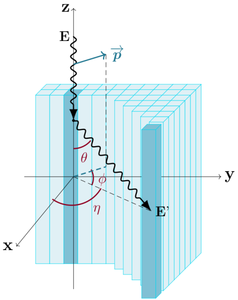
- POLAR is a dedicated Gamma-Ray Burst Polarmeter
- Measures polarization of photons in the 50-500 keV energy range
- Employs Compton scattering to measure polarization



- Azimuthal scattering angle (ϕ) depends on polarization vector
- Preferential scattering perpendicular to polarization vector
- Segmented detector allows to measure angle
- Scattering angle distribution $\rightarrow$ Modulation Curve

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} = \frac{\mathrm{r_o^2}}{2}\frac{\mathrm{E}'^2}{\mathrm{E}^2}\left(\frac{\mathrm{E}'}{\mathrm{E}} + \frac{\mathrm{E}}{\mathrm{E}'} - 2\sin^2\theta\cos^2\phi\right). \tag{1}$$





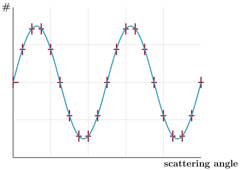
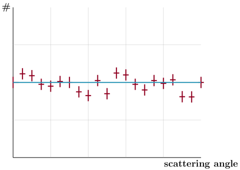
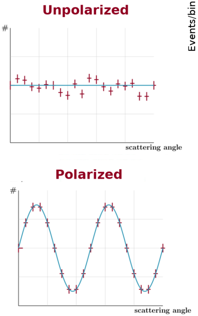
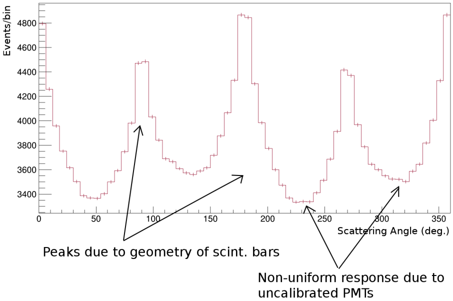


Figure 8: Modulation Curve

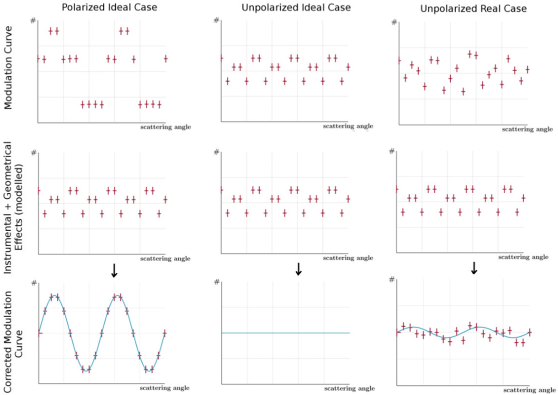






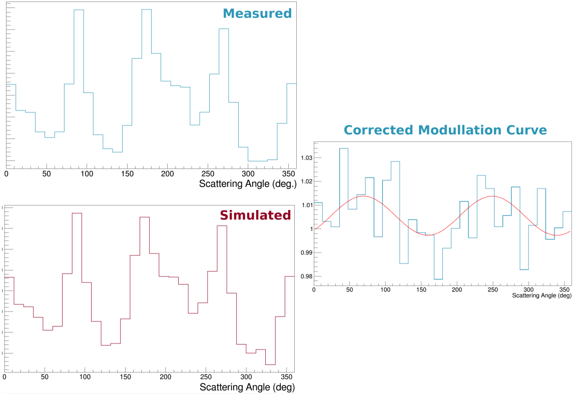
- Theory: plot the scattering angles → check amplitude of 180° modulation → convert to polarization → publish
- Reality: Modulation curves look complex
- Wide field of view results in off-axis effects → 360° modulation
- Instrument geometry and surrounding material add further complexities
- Instrument non-uniform response
- etc.
- etc.

Reconstructing PD & PA



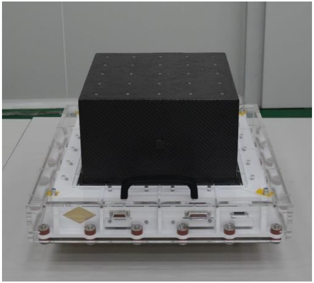
- Analysis proceeds typically by dividing the measured curve by simulated unpolarized curve
- Divide out all the extra effects and only leaves polarization induced effects
- Simulate scattering angle distribution for specific GRB (spectrum, angle, temperature etc.)
- Any error in the MC produces an positive PD

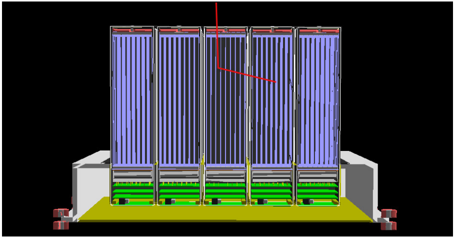
Systematic errors



- Modelling of all details complex
- Any deviation from reality leads to "polarization"
- Second law of thermodynamics $\rightarrow$ Making a line more flat (unpolarized) is unlikely
- Compare to spectral measurements where error can go either way

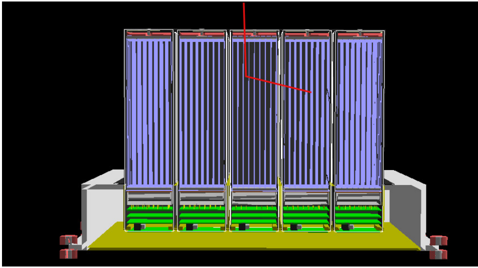
- Effective area of $\approx 300\mathrm{cm}^2$ at 400 keV
- Small pixels allows for high precision scattering angle measurements
- Field of View of half the sky

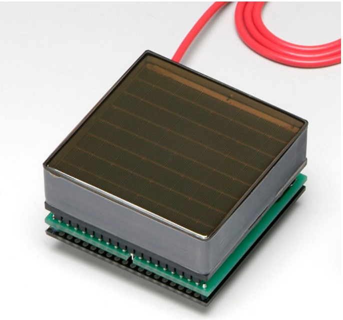


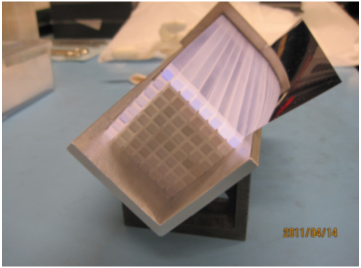


POLAR

- Plastic scintillator array readout using MAPMTs
- Scintillators of $5.9 \times 5.9 \times 172\,\mathrm{mm}^3$
- MAPMTs reads out 64 scintillators
- Carbon fibre shielding around the scintillators
- Aluminium frame housing the electronics
- Mounted on Tiangong-2 spacelab

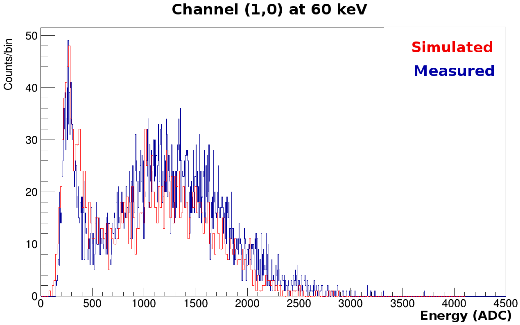






MC rules for POLAR collaboration

1. Only one MC simulation package
2. Keep it as simple as possible
3. Output data from MC framework is identical to real data
4. Geant4 handles only the physics, electronics is performed in a second stage
5. No data analysis until MC is fully understood



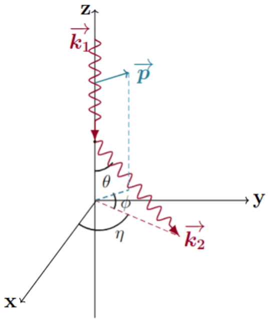
POLAR Geant4: Framework

Get code: `https://github.com/POLAR-2/POLAR-SIM`

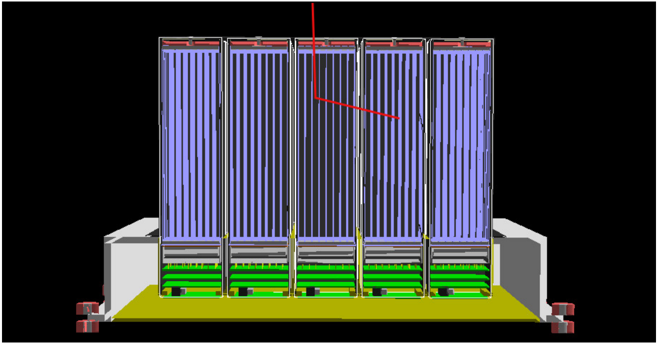
- DetectorConstruction.cc → contains all the detector details
- Physicslist.cc → contains links to physics lists to be used
- ActionInitialization.cc → Defines format of output data file (ROOT files in out case)
- Run.cc → extracts necessary info from each step
- ConfigMessenger.cc → passes configuration information to the simulations
- main.cc wraps everything together
- 2 or 3 other files might be needed but that is it
- Complexity of the code tends to increase when copying from another project

POLAR Geant4: Physicslists

- Use G4EmLivermorePolarizedPhysics.hh for normal science simulations
- Use different physics list for background simulations (e.g. G4HadronPhysicsQGSP_BERT.hh)
- Do not blindly trust physicslists
- Example: first version of G4LowEPPolarizedComptonModel did not work at all
- Always check for changes when updating Geant4 versions



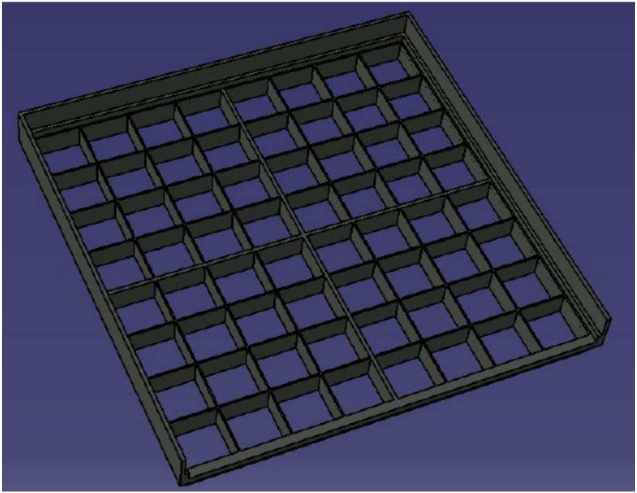
- GDML vs. coding by hand
- GDML is not easy to read
- Using CAD to gdml converters tends to make code slow
- writing by hand is a pain... but probably worth it



```
// Sylgard_184
Sylgard_184_ = new G4Material(name = "Sylgard_184", density = 0.965*g/cm3, ncomponents = 4);
Sylgard_184_->AddElement(elC,  natoms = 2);
Sylgard_184_->AddElement(elH,  natoms = 6);
Sylgard_184_->AddElement(elO,  natoms = 1);
Sylgard_184_->AddElement(elSi, natoms = 1);

// AlloyAl_7075
AlloyAl_7075_ = new G4Material(name = "AlloyAl_7075", density = 2.81*g/cm3, ncomponents = 9);
AlloyAl_7075_->AddElement(elSi, fractionmass = 0.4*perCent);
AlloyAl_7075_->AddElement(elFe, fractionmass = 0.5*perCent);
AlloyAl_7075_->AddElement(elCu, fractionmass = 1.6*perCent);
AlloyAl_7075_->AddElement(elMn, fractionmass = 0.3*perCent);
AlloyAl_7075_->AddElement(elMg, fractionmass = 2.5*perCent);
AlloyAl_7075_->AddElement(elCr, fractionmass = 0.23*perCent);
AlloyAl_7075_->AddElement(elZn, fractionmass = 5.6*perCent);
AlloyAl_7075_->AddElement(elTi, fractionmass = 0.2*perCent);
AlloyAl_7075_->AddElement(elAl, fractionmass = 88.67*perCent);

// AlloyAl_2219
AlloyAl_2219_ = new G4Material("AlloyAl_2219", 2.85*g/cm3, 7);
AlloyAl_2219_->AddElement(elV,  fractionmass = 0.1*perCent);
AlloyAl_2219_->AddElement(elSi, fractionmass = 0.1*perCent);
AlloyAl_2219_->AddElement(elFe, fractionmass = 0.15*perCent);
AlloyAl_2219_->AddElement(elZr, fractionmass = 0.15*perCent);
AlloyAl_2219_->AddElement(elMn, fractionmass = 0.3*perCent);
AlloyAl_2219_->AddElement(elCu, fractionmass = 6.3*perCent);
AlloyAl_2219_->AddElement(elAl, fractionmass = 92.90*perCent);

// AlloyAl_5083
//taken from https://www.smithmetal.com/pdf/aluminium/5xxx/5083.pdf
AlloyAl_5083_ = new G4Material("AlloyAl_5083", 2.65*g/cm3, 8);
AlloyAl_5083_->AddElement(elFe, fractionmass = 0.4*perCent);
AlloyAl_5083_->AddElement(elCu, fractionmass = 0.1*perCent);
AlloyAl_5083_->AddElement(elMg, fractionmass = 4.45*perCent);
AlloyAl_5083_->AddElement(elSi, fractionmass = 0.4*perCent);
AlloyAl_5083_->AddElement(elZn, fractionmass = 0.25*perCent);
AlloyAl_5083_->AddElement(elCr, fractionmass = 0.15*perCent);
AlloyAl_5083_->AddElement(elTi, fractionmass = 0.15*perCent);
AlloyAl_5083_->AddElement(elAl, fractionmass = 94.1*perCent);
```

- Include materials in as much detail as possible
- Example: Aluminium alloys used in POLAR result in measurement background

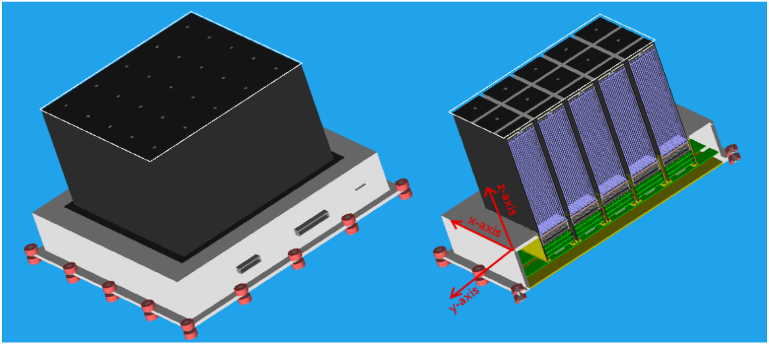
POLAR Geant4: Detector Construction



- Complex geometries can be done using things like G4SubtractionSolid
- Draw one G4Box and subtract another G4Box → clean code
- Downside: slows down simulations
- Ugly coding can be worth it at times

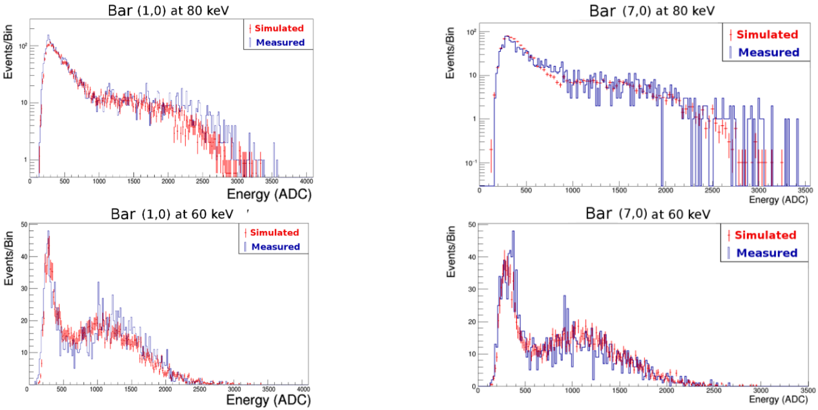
POLAR Geant4: Output

- Geant4 only handles the physics
- Information on primary particles is stored for debugging purposes
- For each physical interaction we store 5 parameters
  1) Detector channel number
  2) X,Y,Z interaction position in this channel
  3) Deposited energy in that location
- Deposited energy is corrected for Birks' effect

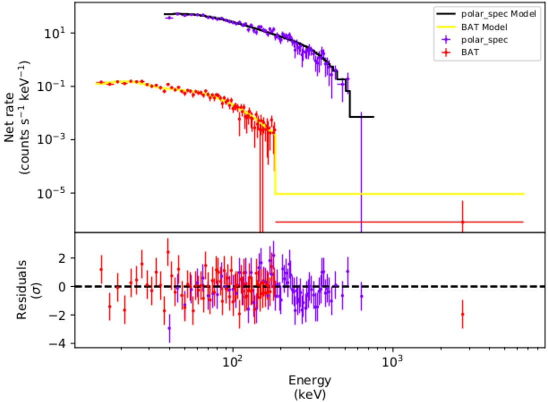


POLAR Geant4: Digitization

- Electronics response is handled in second MC C++ package
- Converts deposited energy to optical photons
- Converts optical photons to photo-electrons in PMT
- Conversion based on optical simulations (separate in G4)
- Converts PMT signal into electrical signal
- Handles trigger logic
- Output is equivalent to POLAR data



- Digitization contains all parameters and their dependence on environment
- e.g. channel gain vs HV, channel gain vs temp, threshold vs temp etc etc
- Splitting digitization from G4 speeds things up
- G4 part is relatively simple and can be run once
- G4 part is not dependent on instrument conditions (e.g. temperature, HV etc.)
- Cross calibration against other satellites



POLAR Geant4: Result

- Calibration shows ARF and RMF of each channel match measurement
- Once this works, polarization is easy