Electromagnetic physics II Optical Photons

Version 11.1

Alexander Howard (Imperial)

Geant4 Advanced Course

Optical Photons

1. Optical Processes in Geant4
2. Photon Production Mechanisms
   - Cerenkov
   - Scintillation
3. Transportation of photons
   - Bulk Processes
   - Boundary Processes
4. Code implementation of optical processes
5. “Idiosyncrasies”
   - Birk’s Effect
   - Photon arrival times
   - Some “tricks”
6. Examples
7. Recent features

Alexander Howard, Imperial College London
(Documentation Co-ordinator)
with input from Daren Sawkey, Varian

Optical Processes in Geant4

- More info:

  Geant4 User's Guide for Application Developers
  (chap. 5: Physics Processes)

- Processes:
  - Cerenkov
  - Scintillation
  - Transition Radiation
- Classes in: /processes/electromagnetic/xrays!

Optical Processes in Geant4

- More info:

  Geant4 User's Guide for Application Developers
  (chap. 5: Physics Processes)

- Processes:
  - Cerenkov
  - Scintillation
  - Transition Radiation
- Classes in: /processes/electromagnetic/xrays!
- Warning:
  - Energy not conserved in the first 2 processes
    - Necessary in order to store energy deposits from excitation particle (i.e. monte carlo truth) **and** give the true detector observable (i.e. register the photon hit)

Optical Processes in Geant4

- More info:

  Geant4 User's Guide for Application Developers
  (chap. 5: Physics Processes)

- Processes:
  - Cerenkov
  - Scintillation
  - Transition Radiation
- Classes in: /processes/electromagnetic/xrays!
- Warning:
  - Energy not conserved in the first 2 processes
    - Necessary in order to store energy deposits from excitation particle (i.e. monte carlo truth) **and** give the true detector observable (i.e. register the photon hit)
  - **Strings** are used a lot in material property descriptions – **beware**!
    - Checking introduced since v11.0 (see slide 44)

Optical Photons in Geant4

- Photons are treated the same as any other particle in Geant4
  - Discrete steps
  - Random (Monte Carlo) generation, transportation and interaction
  - Physics processes applied as with any other particle
- Differences:
  - Secondary process (created after energy deposit)
    - `edep` and **photon creation** are recorded (**risk** of double counting)
  - Boundary processes significant
  - Skin surfaces can be shared between volumes (different to standard mass world)
- For a typical optical simulation (especially scintillation) the CPU load can become quite heavy
  - 1 primary particle could give **>10000** photons/MeV of energy deposit!

What is a photon (in Geant4)?

- Optical photon in Geant4: $\lambda \gg$ Atomic spacing
- Treated as waves: subject to reflection, refraction…
- Polarization plays a role
- G4OpticalPhoton ≠ G4Gamma, and there is no smooth transition between one and the other.

The G4Scintillation Process

- Number of photons generated is proportional to the energy lost during the step…,
  - unless a value for the Birks constant of a material is given
- Emission spectrum defined by the user. Random polarization!
- Isotropic emission, uniform along the step!
- Emission time with one or two exponential decay components (allows for fast and slow decay constants)!
  - See slides 38 and 42 for the extended (3 time constant) situation

The G4Scintillation Process

- A scintillation material has a characteristic light yield per unit energy (→**SCINTILLATIONYIELD**)
- For memory optimization and computing reasons, the track of the primary is suspended and scintillation photons are tracked first (optional, inherited from Geant3)!
- It allows for different scintillation yields depending on the particle type (→**YIELDFACTOR**)!
- The number of photons produced fluctuates around a mean number with a width given by its square root. This can be broadened due to impurities for doped crystals or narrowed due to a Fano factor < 1 (→**RESOLUTIONSCALE**).

Transport of Optical Photons

- Bulk processes:
  - **Rayleigh Scattering**
  - **MIE Scattering**
  - **Bulk Absorption**
- Boundary processes:
  - **Reflection**
  - **Refraction**
  - **Absorption**
- Classes in: /processes/optical

Bulk Processes

- Bulk Absorption: kills the particle after a certain distance travelled within a particular material (→ **ABSLENGTH**)

- **Rayleigh Scattering**:
  - cross section proportional to **$1 + \cos^2(\theta)$**,
    where **θ** is the angle between the initial and final photon polarization

- Rayleigh scattering attenuation coefficient provided by the user (→ **RAYLEIGH**)

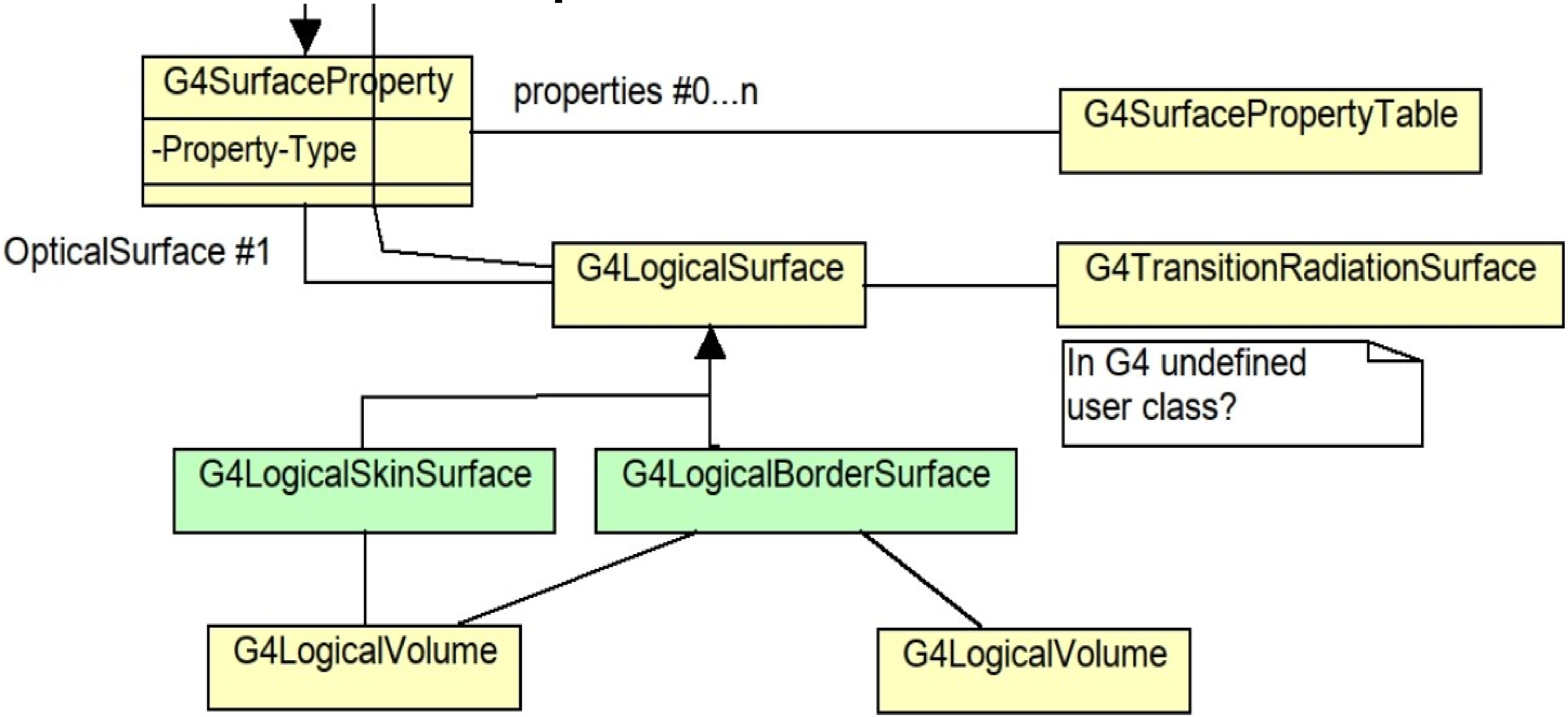
- **Mie Scattering**:
  - Analytical solution of Maxwell's equations for scattering of optical photons by spherical particles.
  - Significant only when the radius of the scattering object (spherical) is of order of the wave length.

Boundary Processes

- **Dielectric – Dielectric**:
  depending on photon's wave length, angle of incidence, polarization, and refractive index on both sides of the boundary:
  - Reflection
  - Refraction
  - Tot. Int. reflection!

- **Dielectric – Metal**:
  - Reflection
  - Absorption

Optical Surfaces



http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/TrackingAndPhysics/physicsProcess.html#parameterization

Optical Surfaces

- Two methods to describe a surface:
  - **SkinSurface**
    - Useful if shared between multiple physical volumes
    - Disadvantage: only one property for the whole “skin”
  - **BorderSurface**
    - Pair ordered sequence of physical volumes
    - Useful if direction is important – i.e. passing from volume A to B rather than the opposite
    - Disadvantage: requires definition at every boundary – can get complicated if boundary is shared e.g. a liquid scintillator with gas above

Boundary Processes

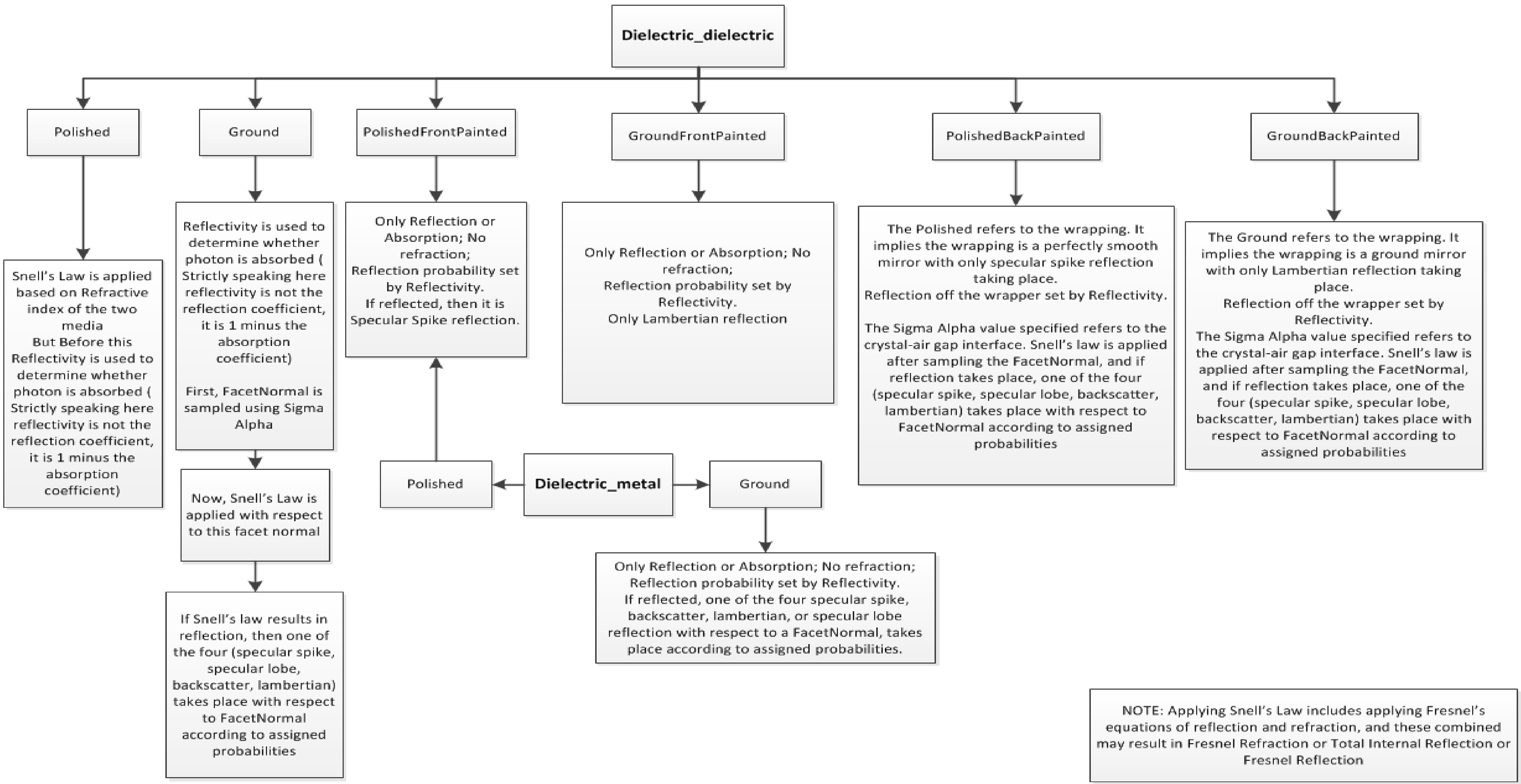
- Surface finish:
  - **Polished**: the normal used by the G4BoundaryProcess is the geometrical normal to the surface.
  - **Ground**: the normal is calculated based on microfacets that appear at angle α with the average surface!
- Two models:
  - **GLISUR** (Geant3)
  - **UNIFIED** (DETECT code @ TRIUMF)!
- GLISUR only allows **POLISH** and **GROUND**. UNIFIED has some other variations but depends on 4 parameters…

Dielectric Metal Surfaces

- **Dielectric-metal** surfaces are a bit special:
  - Specular reflectors
  - Photocathodes (sensitive detectors)
- Efficiency applied to represent the true **quantum efficiency**

```
G4LogicalVolume* volume_log;

G4OpticalSurface* OpSurface = new G4OpticalSurface("name");

G4LogicalSkinSurface* Surface = new
  G4LogicalSkinSurface("name",volume_log,OpSurface);

OpSurface -> SetType(dielectric_metal);
OpSurface -> SetFinish(ground);
OpSurface -> SetModel(glisur);

G4double polish = 0.8;

G4MaterialPropertiesTable* OpSurfaceProperty = new G4MaterialPropertiesTable();

OpSurfaceProperty -> AddProperty("REFLECTIVITY",pp,reflectivity,NUM);
OpSurfaceProperty -> AddProperty("EFFICIENCY",pp,efficiency,NUM);
OpSurface -> SetMaterialPropertiesTable(OpSurfaceProperty);
```

UNIFIED MODEL FOR OPTICAL SURFACES



LUT: Unified Model Surfaces

```
polishedlumirrorair,         // mechanically polished surface, with lumirror
polishedlumirrorglue,        // mechanically polished surface, with lumirror & meltmo
polishedair,                 // mechanically polished surface
polishedteflonair,           // mechanically polished surface, with teflon
polishedtioair,              // mechanically polished surface, with tio paint
polishedtyvekair,            // mechanically polished surface, with tyvek
polishedvm2000air,           // mechanically polished surface, with esr film
polishedvm2000glue,          // mechanically polished surface, with esr film & meltmo
etchedlumirrorair,           // chemically etched surface, with lumirror
etchedlumirrorglue,          // chemically etched surface, with lumirror & meltmount
etchedair,                   // chemically etched surface
etchedteflonair,             // chemically etched surface, with teflon
etchedtioair,                // chemically etched surface, with tio paint
etchedtyvekair,              // chemically etched surface, with tyvek
etchedvm2000air,             // chemically etched surface, with esr film
etchedvm2000glue,            // chemically etched surface, with esr film & meltmount
groundlumirrorair,           // rough-cut surface, with lumirror
groundlumirrorglue,          // rough-cut surface, with lumirror & meltmount
groundair,                   // rough-cut surface
groundteflonair,             // rough-cut surface, with teflon
groundtioair,                // rough-cut surface, with tio paint
groundtyvekair,              // rough-cut surface, with tyvek
groundvm2000air,             // rough-cut surface, with esr film
```

- To use a look-up-table all the user needs to specify for a G4OpticalSurface is:
  - **`SetType(dielectric_LUT)`**
  - **`SetModel(LUT)`**
  - And, for example, **`SetFinish(polishedtyvekair)`**

LUT: (Even) More precise data

- A recently implemented model is called the **LUT Davis model**
  - The model is based on measured surface data
  - Choose from a list of available surface finishes
  - Provided are a rough and a polished L(Y)SO surface that can be used without reflector, or in combination with a specular reflector (e.g. ESR) or a Lambertian reflector (e.g. Teflon)
  - Specular reflector can be coupled to the crystal with air or optical grease. Teflon tape is wrapped around the crystal with 4 layers.
- To enable the LUT Davis Model, the user needs to specify for a G4OpticalSurface:
  - `SetType(dielectric_LUTDAVIS)`, `SetModel(DAVIS)` and also, for example, `SetFinish(Rough_LUT)`
  - **`Note the underscores – they're real (!)`**

Environment Variables

- **`export G4REALSURFACEDATA=XXX/data/RealSurface2.2/`**
- A parametrised model for surface reflectivity, e.g. painted plastic scintillator
- For precise simulations be aware of your electromagnetic physics model selection
  - Need **`G4LEDATA`** set
  - But this will not affect the optical physics and ray-tracing which are considered secondarily and independently

In the code...

- DetectorConstruction class:
  - 1st we define a material

```
// LaBr3
density = 5.08*g/cm3;
G4Material* LaBr3 = new G4Material(name="LaBr3", density, ncomponents=2);
LaBr3->AddElement(Br, natoms=3);
LaBr3->AddElement(La, natoms=1);
```

In the code...

- DetectorConstruction class:
  - 2^{nd} we give it optical properties

```
const G4int nEntries = 2;

G4double PhotonEnergy[nEntries] = {1.0*eV,7.0*eV};

// LaBr3

G4double LaBr3RefractionIndex[nEntries] = {1.9,1.9};
G4double LaBr3AbsorptionLength[nEntries] = {50.*cm,50.*cm};

G4MaterialPropertiesTable* LaBr3MPT = new G4MaterialPropertiesTable();

LaBr3MPT->AddProperty("RINDEX",PhotonEnergy,LaBr3RefractionIndex,nEntries);
LaBr3MPT->AddProperty("ABSLENGTH",PhotonEnergy,LaBr3AbsorptionLength,nEntries);

G4double ScintEnergy[nEntries] = {3.26*eV,3.44*eV};
G4double ScintFast[nEntries] = {1.0,1.0};

LaBr3MPT->AddProperty("FASTCOMPONENT",ScintEnergy,ScintFast,nEntries);

LaBr3MPT->AddConstProperty("SCINTILLATIONYIELD",63./keV);
LaBr3MPT->AddConstProperty("RESOLUTIONSCALE",1.);
LaBr3MPT->AddConstProperty("FASTTIMECONSTANT",20.*ns);
LaBr3MPT->AddConstProperty("YIELDRATIO",1.);

LaBr3->SetMaterialPropertiesTable(LaBr3MPT);
```

In the code...

- DetectorConstruction class:
  - 2nd we give it optical properties

```
const G4int nEntries = 2;

G4double PhotonEnergy[nEntries] = {1.0*eV,7.0*eV};

// LaBr3

G4double LaBr3RefractionIndex[nEntries] = {1.9,1.9};
G4double LaBr3AbsorptionLength[nEntries] = {50.*cm,50.*cm};

G4MaterialPropertiesTable* LaBr3MPT = new G4MaterialPropertiesTable();

LaBr3MPT->AddProperty("RINDEX",PhotonEnergy,LaBr3RefractionIndex,nEntries);
LaBr3MPT->AddProperty("ABSLENGTH",PhotonEnergy,LaBr3AbsorptionLength,nEntries);

G4double ScintEnergy[nEntries] = {3.26*eV,3.44*eV};
G4double ScintFast[nEntries] = {1.0,1.0};

LaBr3MPT->AddProperty("FASTCOMPONENT",ScintEnergy,ScintFast,nEntries);

LaBr3MPT->AddConstProperty("SCINTILLATIONYIELD",63./keV);
LaBr3MPT->AddConstProperty("RESOLUTIONSCALE",1.);
LaBr3MPT->AddConstProperty("FASTTIMECONSTANT",20.*ns);
LaBr3MPT->AddConstProperty("YIELDRATIO",1.);

LaBr3->SetMaterialPropertiesTable(LaBr3MPT);
```

In the code...

- DetectorConstruction class:
  - 3rd the material fills a volume

```
//Crystal

G4Tubs* solidCrystal = new G4Tubs("Crystal", 0.*cm, ScintRadius,
                                  ScintHalfLength,StartPhi,DeltaPhi);

G4LogicalVolume* logicCrystal = new G4LogicalVolume(solidCrystal,LaBr3,
                                                    "Crystal");

G4ThreeVector positionCrystal = G4ThreeVector(0.*cm,0.*cm,0.*cm);

G4VPhysicalVolume* physiCrystal = new G4PVPlacement(0,positionCrystal,
                                                    "Crystal",logicCrystal,
                                                    physiReflector,false,0);
```

In the code...

- DetectorConstruction class:
  - 4th we define boundaries

```
// Reflector - sintillator surface

G4OpticalSurface* OpRefCrySurface =
new G4OpticalSurface("RefCrySurface");

OpRefCrySurface->SetType(dielectric_metal);
OpRefCrySurface->SetModel(glisur);
OpRefCrySurface->SetFinish(polished);

G4LogicalBorderSurface* RefCrySurface =
new G4LogicalBorderSurface("RefCrySurface",physiCrystal,physiReflector,OpRefCrySurface);
```

Note the order of the physical volumes

In the code...

- PhysicsList class:
  - Just use the Optical Physics constructor…

```
// Optical Physics
G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
RegisterPhysics(opticalPhysics);

opticalPhysics->SetScintillationYieldFactor(1.);
opticalPhysics->SetScintillationExcitationRatio(0.);

opticalPhysics->SetTrackSecondariesFirst(kScintillation,true);
```

In the code...

- PhysicsList class:
  - Just use the Optical Physics constructor…
  - More sophisticated extension:

```
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->ReplacePhysics(new G4EmStandardPhysics_option4());
G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
opticalPhysics->SetWLSTimeProfile("delta");

opticalPhysics->SetScintillationYieldFactor(1.0);
opticalPhysics->SetScintillationExcitationRatio(0.0);

opticalPhysics->SetMaxNumPhotonsPerStep(100);
opticalPhysics->SetMaxBetaChangePerStep(10.0);

opticalPhysics->SetTrackSecondariesFirst(kCerenkov, true);
opticalPhysics->SetTrackSecondariesFirst(kScintillation, true);

physicsList->RegisterPhysics(opticalPhysics);
```

"Birks" Effect

- Most real scintillators suffer from a reduction in observable light yield
  - called "Birks Effect" (or "law")
- This is an empirical fit to the correlation between LET and scintillation yield

Scintillations from Organic Crystals: Specific Fluorescence and Relative Response to Different Radiations

By J. B. BIRKS
Department of Natural Philosophy, The University, Glasgow*

MS. received 12th April 1951

ABSTRACT. The scintillation response S of organic crystals depends on the nature and energy E of the incident ionizing particle, of residual range r. The specific fluorescence dS/dr is not in general proportional to the specific energy loss dE/dr. By considering the quenching effect of the molecules damaged by the particle on the 'excitons' produced by it, it is shown that $dS/dr = (A\,dE/dr)/(1+kB\,dE/dr)$. A and kB are constants, which have been evaluated for anthracene from observations of S and E, and the range–energy data. Curves are computed for the relative response S of anthracene to electrons, protons, deuterons and α-particles of E up to 15 MeV, and these are shown to agree closely with the available experimental results. The method used for evaluating the relative response is applicable to ionizing particles of any nature or energy, and also to the other organic scintillation crystals

§1. INTRODUCTION

IONIZING radiations impinging on a fluorescent material produce short individual light flashes, or scintillations. These scintillations can be detected with a photo-multiplier tube, and converted into electrical pulses, which can be counted and measured by standard electronic methods. This technique of scintillation counting is being widely applied for the detection and measurement of nuclear radiations. The fluorescent organic crystals are of particular interest for scintillation counting, since they combine a reasonable fluorescent efficiency with a high transparency and a very short luminescent decay time, of the order of 10^{-8} second. These properties make them suitable for the detection of the more penetrating nuclear radiations, and for studies of fast nuclear and meson decay processes, and much of the previous work in this field has been primarily concerned with such applications. The nature of the scintillation process has received rather less attention, and it is, therefore, proposed in this and subsequent papers to consider various fundamental aspects of the fluorescence produced in organic crystals by ionizing radiations.

§2. RESPONSE TO DIFFERENT RADIATIONS

The intensity of the scintillations produced in anthracene and other organic fluorescent crystals depends both on the energy and on the nature of the incident ionizing particle. The amplitude S (volts) of the voltage pulse from a photo-multiplier, operating under constant conditions and observing the crystal, is proportional to the number of fluorescent quanta produced, and hence S may be used as an arbitrary measure of the scintillation intensity. For electrons of energy greater than 125 kev., the scintillation intensity S from an anthracene crystal increases linearly with the energy E (Hopkins 1951), so that the fluorescent

$$\frac{dS}{dr} = \frac{A\,dE/dr}{1 + kB\,dE/dr}.$$

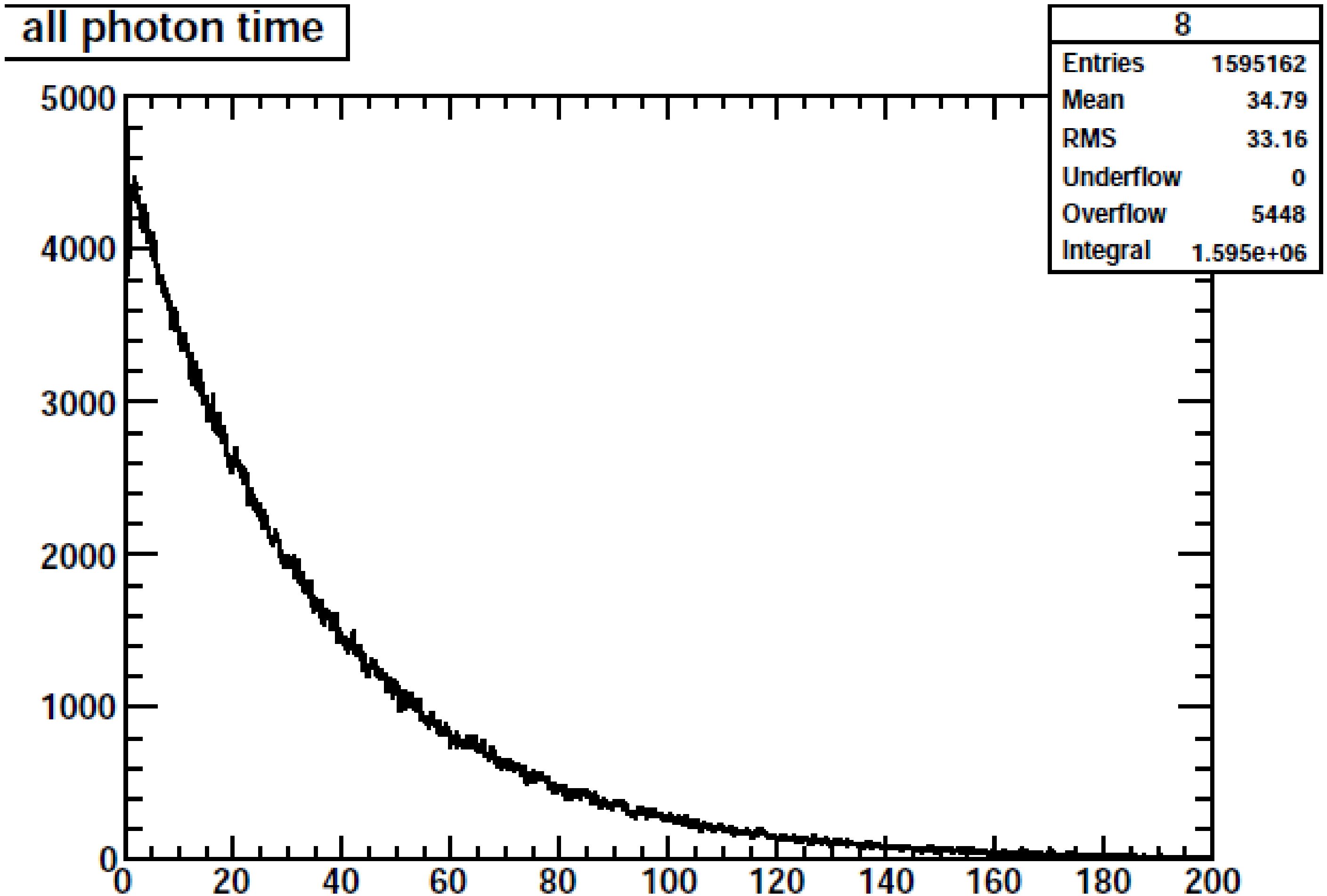
"Birks" Effect

- There is **no** fundamental law/model for Birks
- Geant4 allows the introduction of Birks corrections
  - energy loss corrections, or particle specific response yields – e.g. dark matter detectors
- Birks was originally fitted to organic liquid scintillators (anthracene) and plastics
- Typically called "**quenching**" in terms of signal reduction for either the energy deposit (vs. energy loss) → Lindhardt and Hartree-Fock
- Some materials (e.g. liquid xenon) have greater than unity coefficients for highly densely ionising particles
  - If normalised by electrons/gammas
- Be aware that this needs to be taken into account in any simulation
  - Effects yield, normalisation, poission fluctuation/energy resolution, timing, discrimination

$$\frac{dS}{dr} = \frac{A\, dE/dr}{1 + kB\, dE/dr}\,.$$

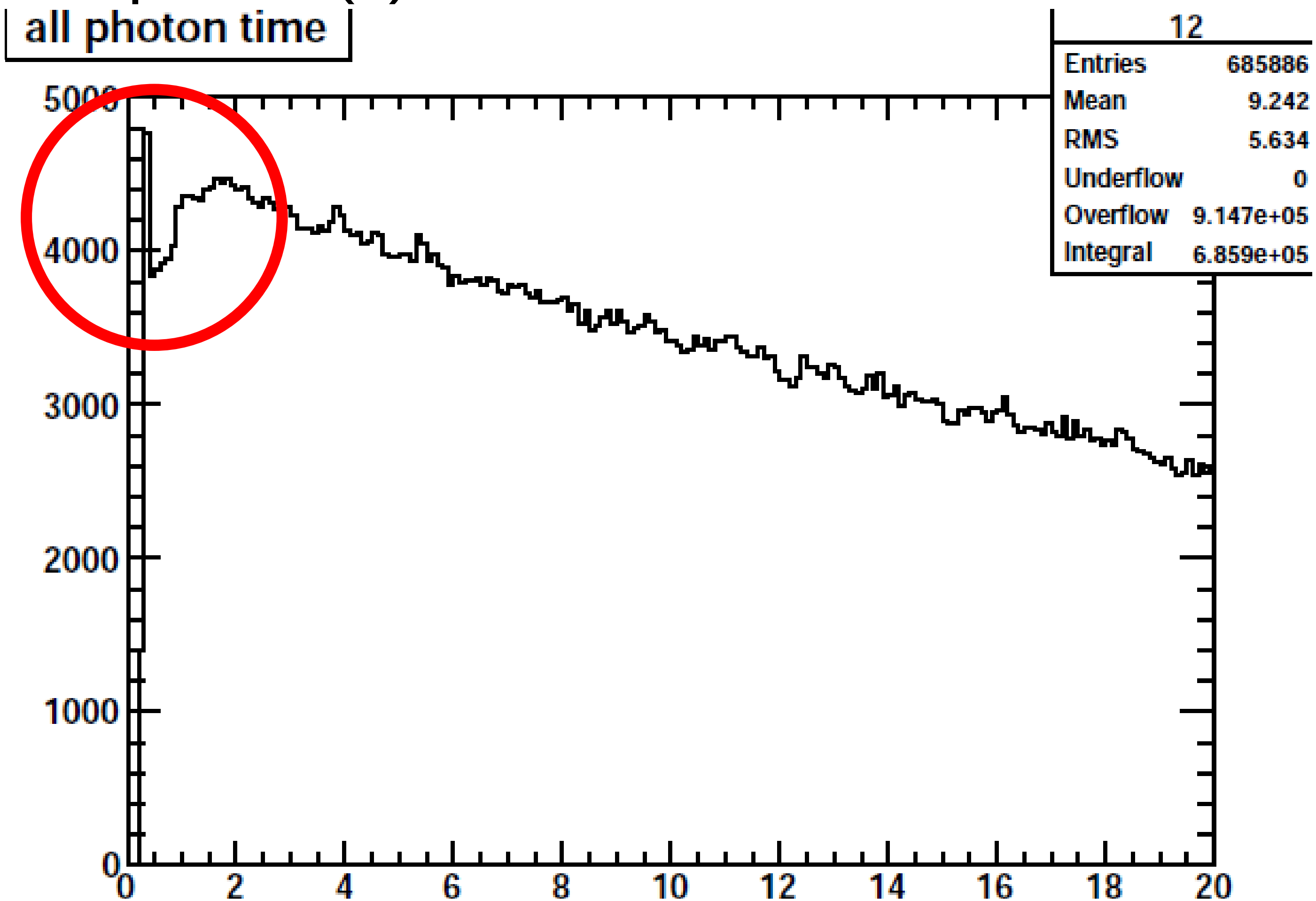
Photon Arrival Times

- Scintillation Photons arrive according to the exponential time component(s)



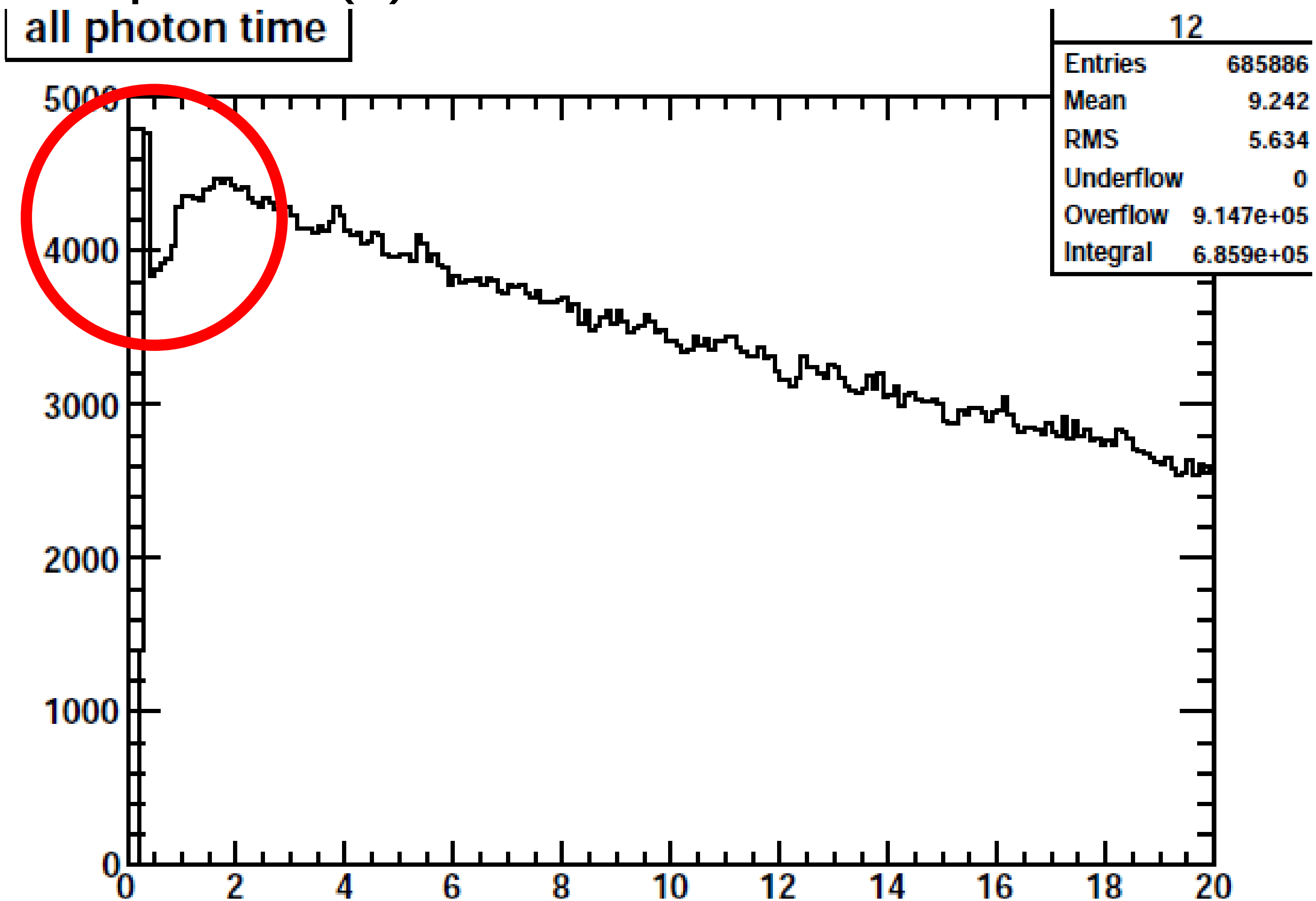
Photon Arrival Times

- Scintillation Photons arrive according to the exponential time component(s)



Photon Arrival Times

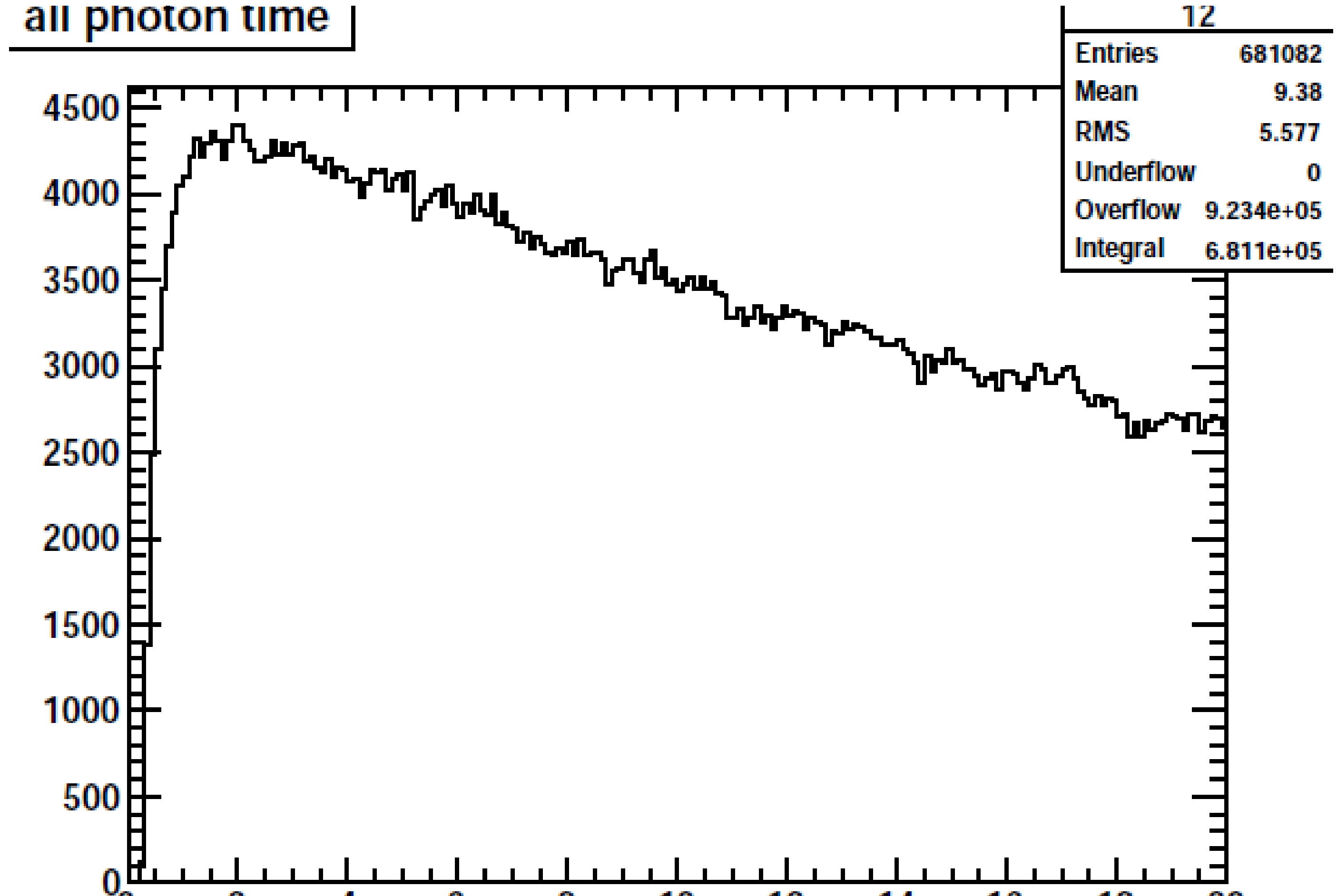
- Scintillation Photons arrive according to the exponential time component(s)



What about Cerenkov?

Photon Arrival Times

- To prove the hypothesis:
  Artificially remove cerenkov process (internal to Geant4)



Some “Tricks”

- Photons are CPU intensive (by nature)
  - Benefit from Multi-threading in Geant4!
- Can reduce input yield according to the Quantum Efficiency of your photodetector
  - NB: This is not direct, as QE is typically measured for normal incidence and you’re correcting for the geometrical/optical acceptance of your set-up
  - Poisson will also be increased in this case – depending upon observable this may be significant
- Optical properties are difficult to ascertain/define
  - Experimentally they are not precisely known
  - Wavelength dependence can also exacerbate the problem
  - There is no model to determine these properties (in general)
- Beware that MPT is filled by **string** value → name **very** important
  - If name mis-typed the property will not be loaded – defaults to vacuum or non-reflectivity or zero-efficiency (for example)
  - Should change in the future to check the type
    - internally properties are converted to an index, so the “string” association is somewhat historical
- Very short (non-physical) absorption lengths can cause problems if close to safety

Examples

- **$G4INSTALL/examples/extended/optical**
- **OpNovice**
  - Simulation of optical photons generation and transport.
    - Defines optical surfaces and exercises optical physics processes (Cerenkov, Scintillation, Absorption, Rayleigh, ...).
    - Uses stacking mechanism to count the secondary particles generated.
- **OpNovice2**
  - Investigate optical properties and parameters.
    - Details of optical photon boundary interactions on a surface are recorded.
    - Details of optical photon generation and transport are recorded.
- **LXe**
  - Multi-purpose detector setup implementing:

    (1) scintillation inside a bulk scintillator with PMTs

    (2) large wall of small PMTs opposite a Cerenkov slab to show the cone

    (3) plastic scintillator with wave-length-shifting fiber readout.
- **wls**
  - Simulates the propagation of photons inside a Wave Length Shifting (WLS) fiber.

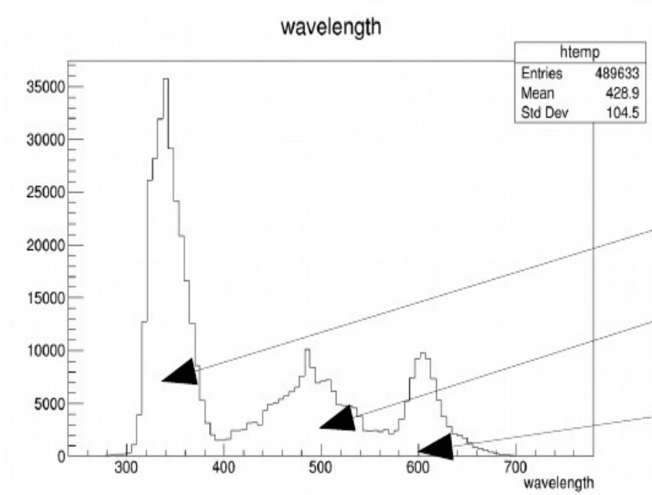
Recent Features

- 2 wavelength shifting processes in the same material
- 3 time constants for particle dependent scintillation decay

Two WaveLengthShifting Process

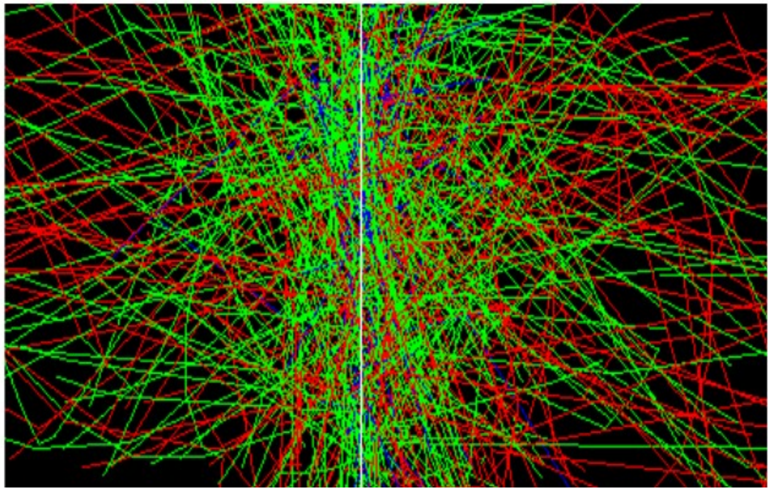
- Simple clone of the single WLS in G4OpticalPhysics constructor
- Cannot convolve the response function as it is a discrete mechanism
  - Either WLS-1 or WLS-2 and some transfer from WLS-1 to WLS-2
- Try it with OpNovice2/wls.mac

```
scintCoreMaterialProperties->AddProperty("WLSABSLENGTH",Wls1AbsEnergy,Wls1AbsLength,WLS1_ABS_ENTRIES);
scintCoreMaterialProperties->AddProperty("WLSABSLENGTH2",Wls2AbsEnergy,Wls2AbsLength,WLS2_ABS_ENTRIES);
scintCoreMaterialProperties->AddConstProperty("WLSTIMECONSTANT",Parameters::GetInstance()->WlsDecayTime()*ns);
scintCoreMaterialProperties->AddConstProperty("WLSTIMECONSTANT2",Parameters::GetInstance()->WlsDecayTime()*ns);
```



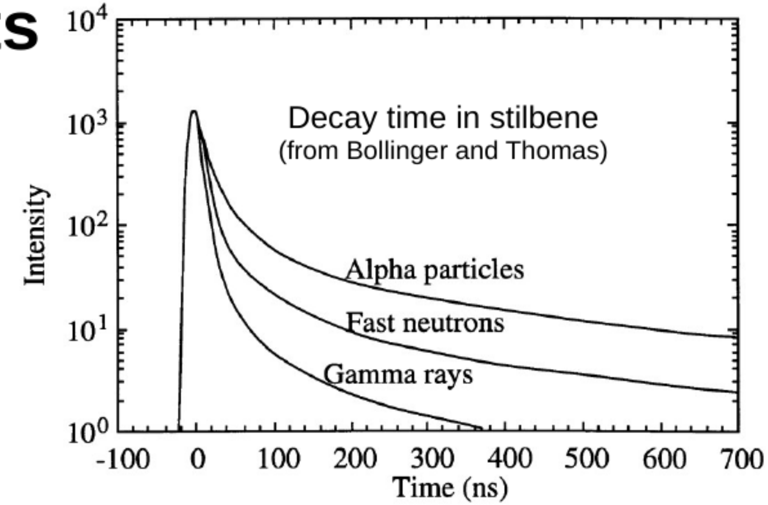
Generated Photons

Blue: Primary Scintillation
Green: WLS1
Red: WLS2

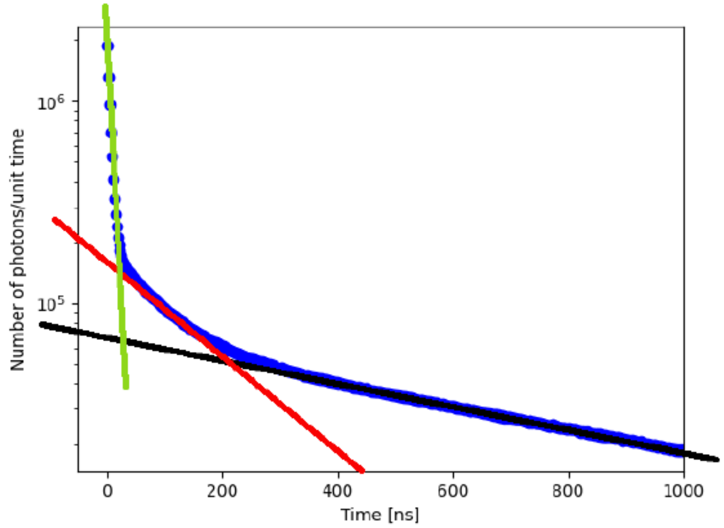


Since Geant4 10.7

New physics: Scintillation time constants



- In <= 10.7, have the choice of fast and slow time constants, with the same yield for all particles; OR particle specific yields, with one time constant
  - Could also write your own physics list!
- In >= 10.7, 3 time constants and particle-specific yields at the same time
  - by users requests
- **Both ways work now**, but the **old way to be deprecated in the next major release**
- In 10.6: material properties SCINTILLATIONYIELD and YIELDRATIO. New method uses SCINTILLATIONYIELD and SCINTILLATIONYIELD[1/2/3].
  - In both methods SCINTILLATIONYIELD gives number of photons (per unit energy).
- Fraction of photons in channel 1 is SCINTILLATIONYIELD1/(SCINTILLATIONYIELD1 + SCINTILLATIONYIELD2 + SCINTILLATIONYIELD3) etc.
- Change material property names from [FAST/SLOW]TIMECONSTANT etc. to SCINTILLATIONTIMECONSTANT[1/2/3] etc.
- Analogous names for particles: PROTONSCINTILLATIONYIELD1 etc.



New Features included since Geant4 v11.0

- Usability improved
- Specifying Material Properties
- Scintillation Material Properties
- Pre-defined Optical Material Parameters
- Creating new Material Property name

Usability:

Use optical physics like this:

```
auto physicsList = new FTFP_BERT;
auto opticalPhysics = new G4OpticalPhysics();

auto opticalParams =
    G4OpticalParameters::Instance();
opticalParams->SetBoundaryInvokeSD(true);

physicsList->RegisterPhysics(opticalPhysics);
runManager->SetUserInitialization(physicsList);
```

Use pre-packaged physics list in most cases

User-defined parameters live here. Modelled after G4EmParameters

See the examples

Specifying material properties

- **Use vectors**

  Run-time check that the vector of energies is the same length as the vector of values

- C arrays still work

  But no protection against out-of-bounds reads

```
std::vector<G4double> energy = {
    2.00 * eV, 2.03 * eV, 2.06 * eV, 2.09 * eV, 2.12 * eV, 2.15 * eV, 2.18 * eV,
    ...
    3.26 * eV, 3.29 * eV, 3.32 * eV, 3.35 * eV, 3.38 * eV, 3.41 * eV, 3.44 * eV
  };

  std::vector<G4double> emissionFib = {
    0.05, 0.10, 0.30, 0.50, 0.75, 1.00, 1.50, 1.85, 2.30, 2.75,
    ...
    0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
  };

  auto mptWLSfiber = new G4MaterialPropertiesTable();
  mptWLSfiber->AddProperty("WLSCOMPONENT", energy, emissionFib);
```

Scintillation material properties

- The “enhanced” time constant properties of 10.7 is now the only method of specifying scintillation properties

  Reduced need to build custom physics lists

  3 time constants, and particle-dependent yields with > 1 time constant

- Change to user code required if there is a material property with “FAST” or “SLOW” in its name

  In most cases simply rename the material properties

  - FAST/SLOW -> 1/2/3

  Documented in Book for Application Developers

Pre-defined optical material parameters

- Include MaterialProperties in the Geant4 distribution

  Ideally users shouldn't have to define their own properties for standard/uninteresting materials

- Use them as:

```
fiberProperty->AddProperty("RINDEX", "PMMA");
```

- So far, refractive indices for air, water, PMMA, fused silica
- Liquid argon and others to come
- Listed in: source/materials/include/G4OpticalMaterialProperties.hh
- Pre-defined Optical Properties

Creating new material property name

- Previously, users had the “RIDNEX” problem

- This compiles and runs fine but no Cerenkov photons are produced:

```
auto mpt = new G4MaterialPropertiesTable();
mpt->AddProperty("RIDNEX", energies, refractiveIndex);
```

- Now it is a run-time error

- If you do actually want to define a new property:

```
mpt->AddProperty("myProperty", energies, someValues, true);
```

Q:Track Secondaries First

- Is a track always sent to waiting stack and optical photons to the main stack?
- Is there an option flag, which can change logic of these processes DoIt()?
- If it exists, how such a flag may be enabled/disabled?

A:Track Secondaries First

- By default, after each step in which there is Cerenkov or scintillation, the primary is suspended and the optical photons are tracked first. This behaviour may be configured with macros or code.
- Specify whether to track secondaries produced in the step before continuing with primary:
  - Macro command:

    ```
    /process/optical/cerenkov/setTrackSecondariesFirst(true)
    ```
  - C++ Command:

    ```
    G4OpticalParameters::Instance()→SetCerenkovTrackSecondariesFirst(G4bool);
    ```
  - Same for scintillation
  - Default is **TRUE**
- See documentation in Application Developers Guide Optical Processes
- Note: if you want to try with TrackSecondariesFirst = false, run with vmstat in another window. Memory may fill up quickly!

Track Secondaries First

- It is possible to disable individual processes:

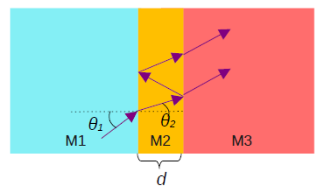
  `/process/optical/processActivation name bool`

  is used to deactivate individual processes

  - `name` may be one of Cerenkov, Scintillation, OpAbsorption, OpRayleigh, OpMieHG, OpBoundary, OpWLS, OWLS2
  - By default, all the processes are activated
  - Optical Physics Process Documentation
- Why you might want to switch secondary tracking:
  - If you have user actions with MC truth then when you suspend tracks this has to be modified, otherwise too many MC truth objects will be created
  - Suspend and restart a track does some overhead even inside Geant4 without user actions
  - However, there is a huge benefit for optical photons.

New Features?

- Thin Optical Coatings (user contribution)
- **Since v11.1**



Coefficient of transmission and reflection through thin layer
(used by CoatedDielectricDielectric())

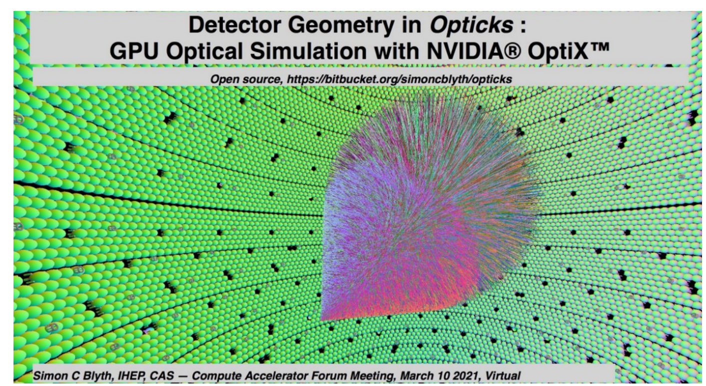
$$t = \frac{t_{12} t_{23} e^{i\beta}}{1 + r_{12} r_{23} e^{2i\beta}} \qquad r = \frac{r_{12} + r_{23} . e^{2i\beta}}{1 + r_{12} . r_{23} . e^{2i\beta}}$$

Not possible in Geant4 yet

$\theta_1 < \theta_L$:	$\theta_1 \geq \theta_L$:
θ_2 existed $\beta = k_2 d \cos(\theta_2)$	θ_2 doesn't existed → remplacing $n_2 \cos\theta_2$ by $i\gamma$ in Fresnel coefficient $\gamma = \sqrt{n_1^2 \sin(\theta_1)^2 - n_2^2}$ $\beta = i k_0 d \gamma$

$$k_2 = n_2 \frac{2\pi}{\lambda} = k_0 x n_2$$

- Opticks GPU acceleration
- Advanced example CaTS
- Can be useful optimisation, but geometrical limitations



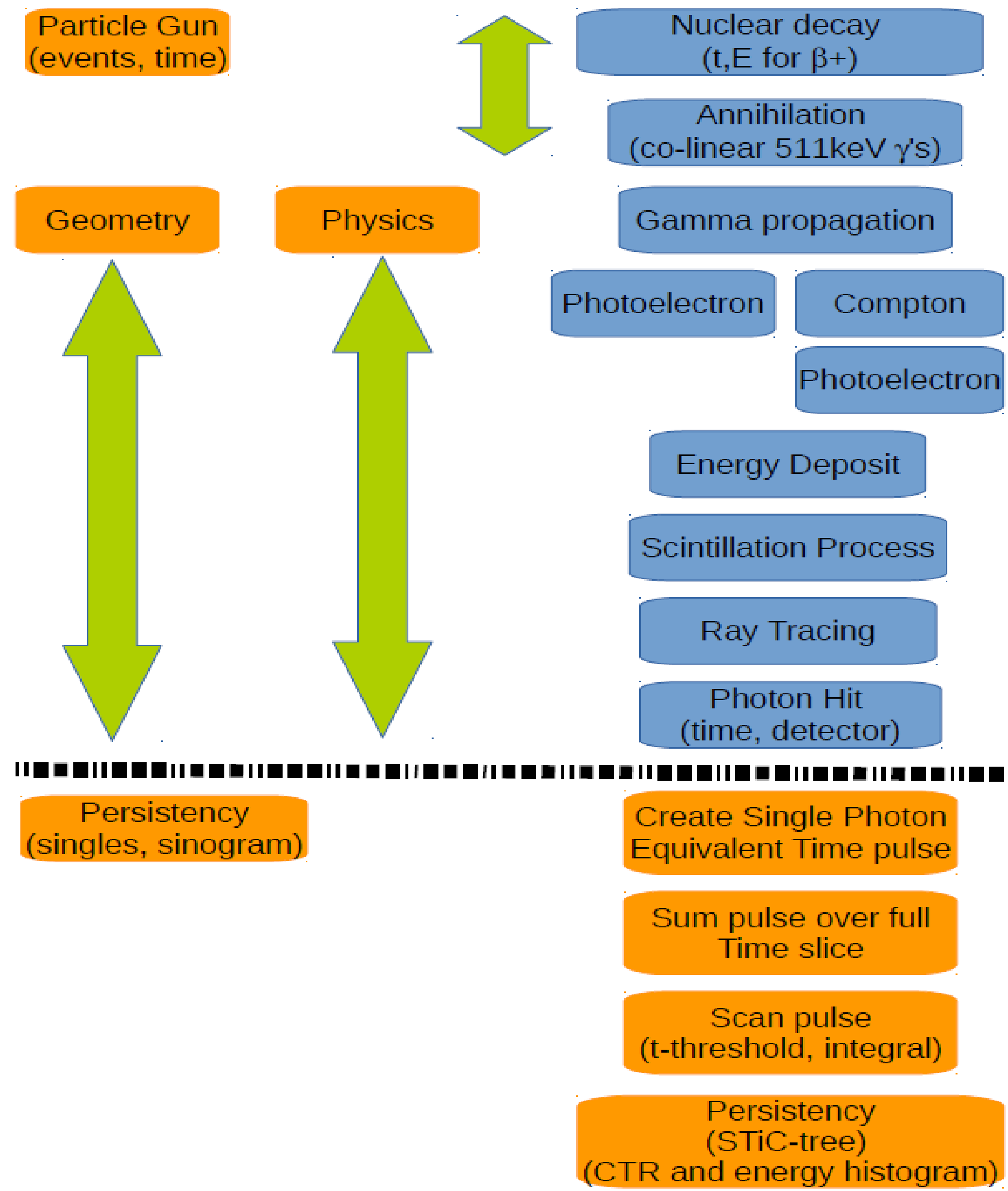
Summary

- Optical processes can be quite complicated
- Geant4 has a number of options to simulate photon production and transportation
- Performance needs to be considered
  - Can be quite CPU intensive
  - Many optical parameters are unknown (physically)
    - Some tuning with experimental data often required
- Hopefully we provide the tools you need!

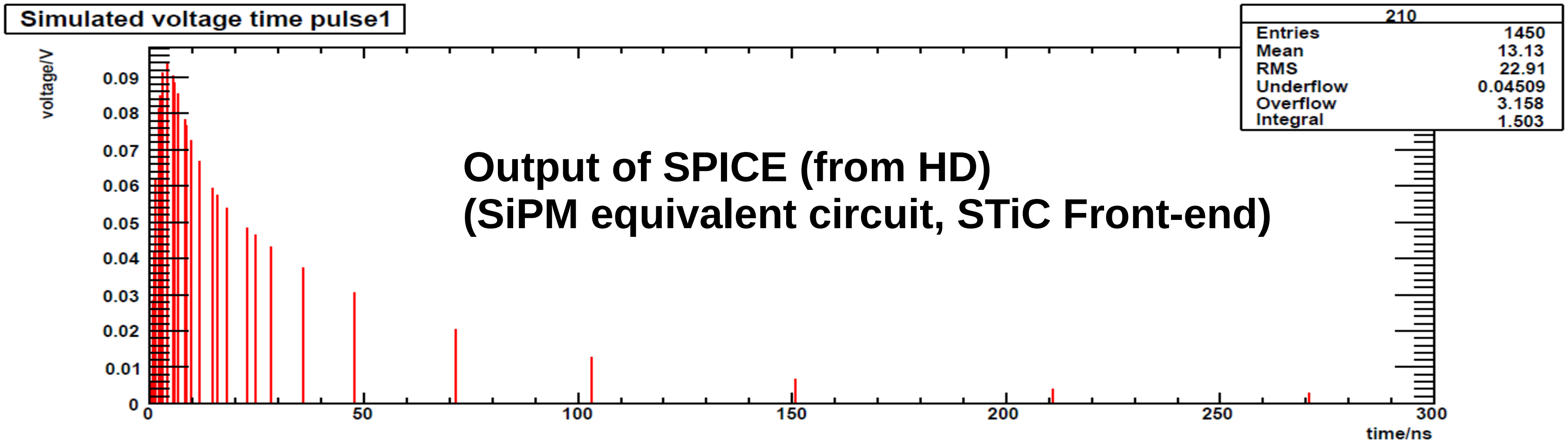
BACKUP – real example application

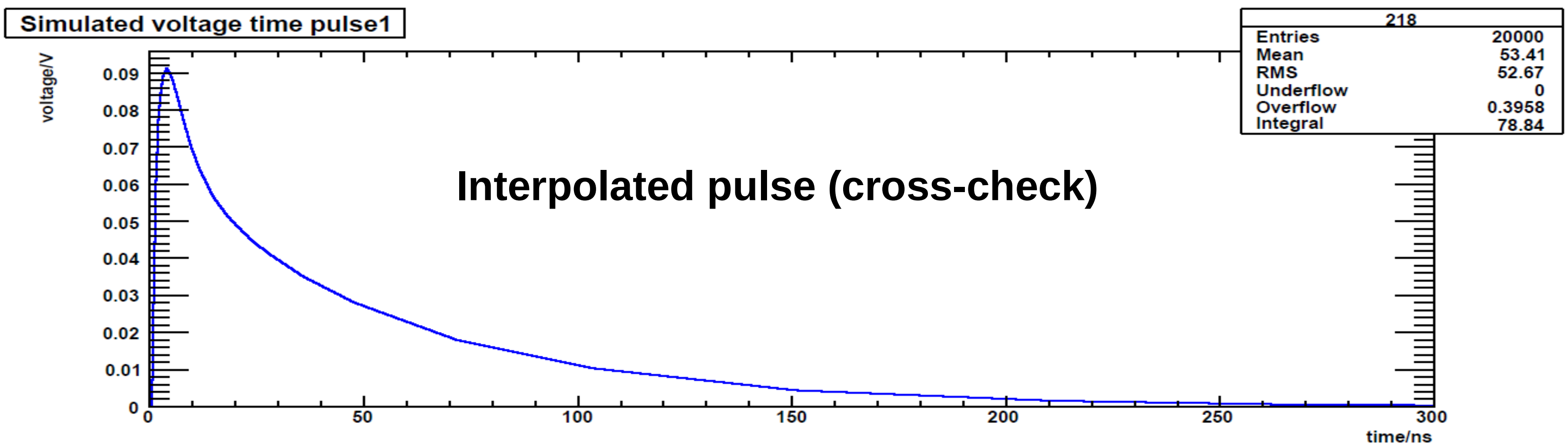
Simulation Program Flow

- **Blue** :– Geant4 toolkit
- **Orange** :– user defined/my code



Single Photon STiC Response

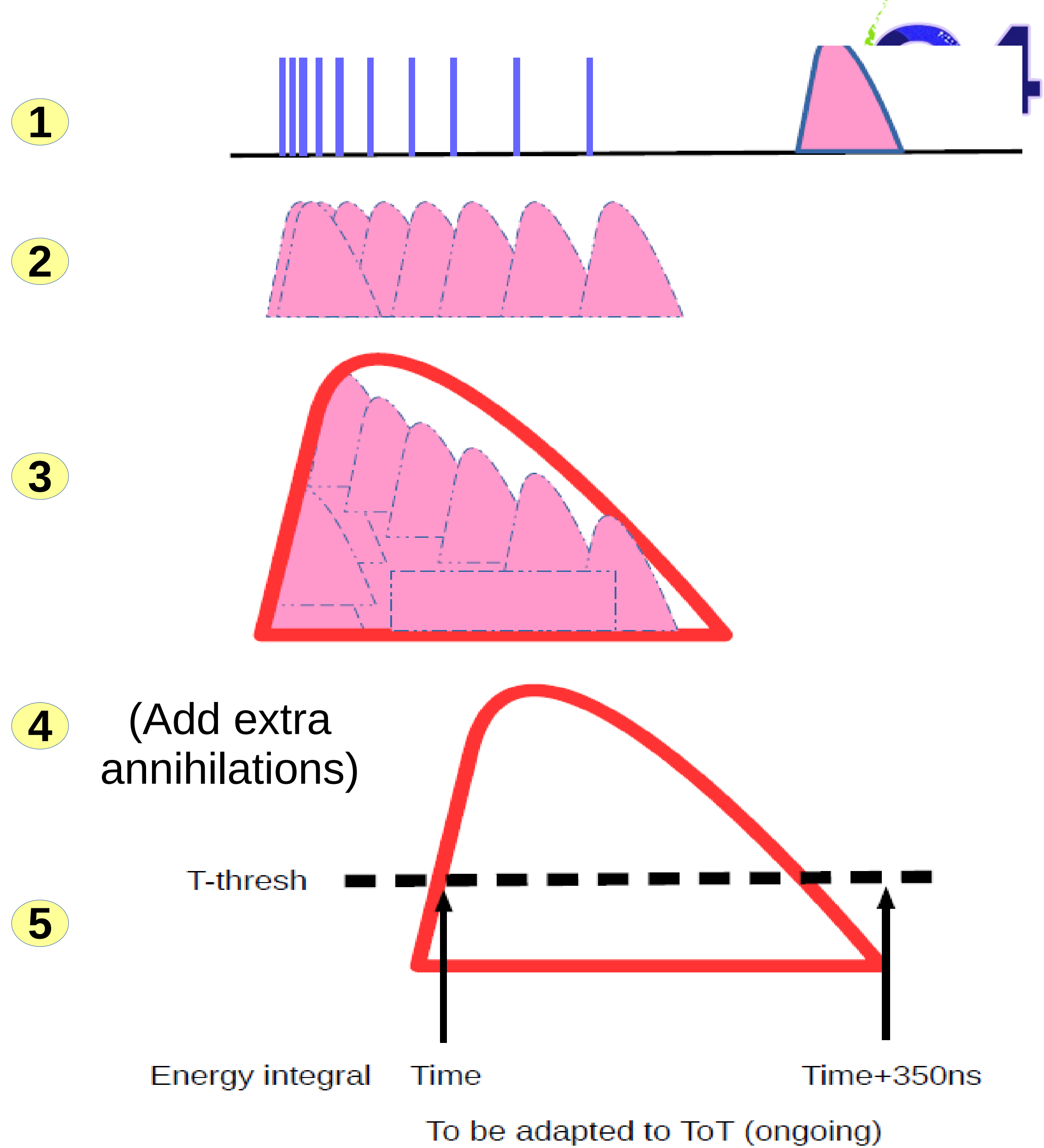




Building up the Pulse-Shape

1) Start with photon arrival times
2) At each time take the single photon response function
3) Linear sum to create the full pulse-shape
4) Events also summed across the 1 microsecond time-slice
   (reproduce pile-up effect)
5) Scan pulse for timestamp (t-thresh is crossed) and integrate t → t+350ns (will be true ToT soon)

Compared to data, Energy is 100% linear right down to zero, so no ToT “explosion” nor offset of minimum value



M/C CTR Timing Response - low

- 126ps at low activity
- (depends on cuts, TDC jitter and analysis)