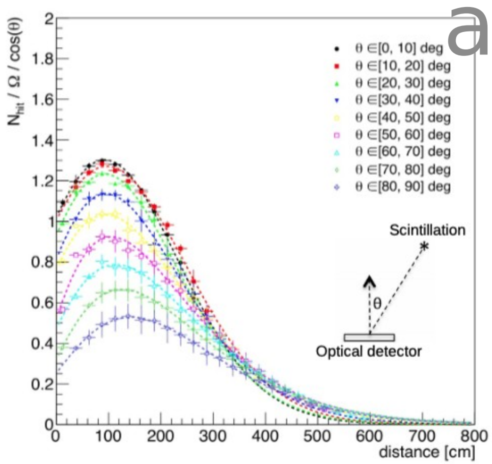
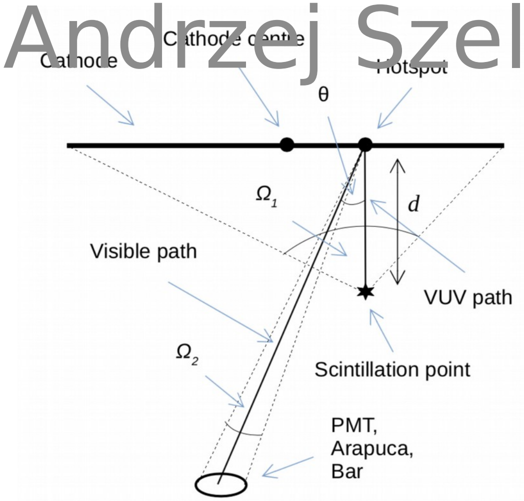
UNIVERSIDAD DE GRANADA

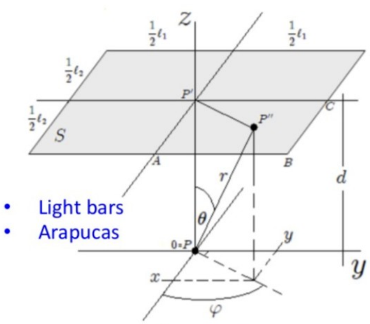
The University of Manchester

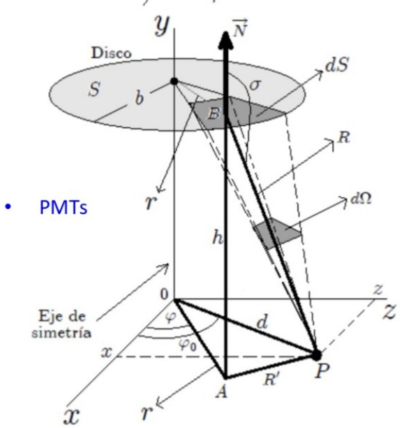
A semi-analytic way of Simulating light

Diego Garcia Gamez,
Patrick Green,
and Andrzej Szelc









Introduction

- Why we need a semi-analytic light simulation model (and why we need it now).
- How it works and performs.
  - Timing
  - Number of hits
- Changes to the code.
- Future development.
- Other things we're trying to slip under the radar.

Optical Libraries

- Up to now we've been mostly using optical libraries to simulate in LArSoft. This has worked reasonably well, but it's not an ideal solution:
  - libraries required are very large: loading library causes severe memory issues + large file size causes issues for grid jobs. Current libraries for SBND and DUNE are >1GB requiring the use of Stash cache.
  - This is with limiting the size of voxels to several cm a side (some dimensions even more).
  - Does not provide timing information (this is solved in LArSoft).
  - Need to run a campaign of grid jobs every time a detector parameter changes.
  - In case of DUNE 1x2x6 segmenting the bars into subdetectors makes it impossible to generate the library due to memory issues. Needed for the TDR.
- DUNE needs a realistic X-Arapuca supercell geometry for physics studies very quickly, SBND libraries are also becoming cumbersome.
- We have developed an alternative method for simulation of light collection to make things work.
- This consists of:
  - Improved parametrization of photon arrival times for both VUV and Visible light
  - A semi-analytic model for predicting the number of hits based on position in the detector for direct VUV light and reflected light coming off the cathode.

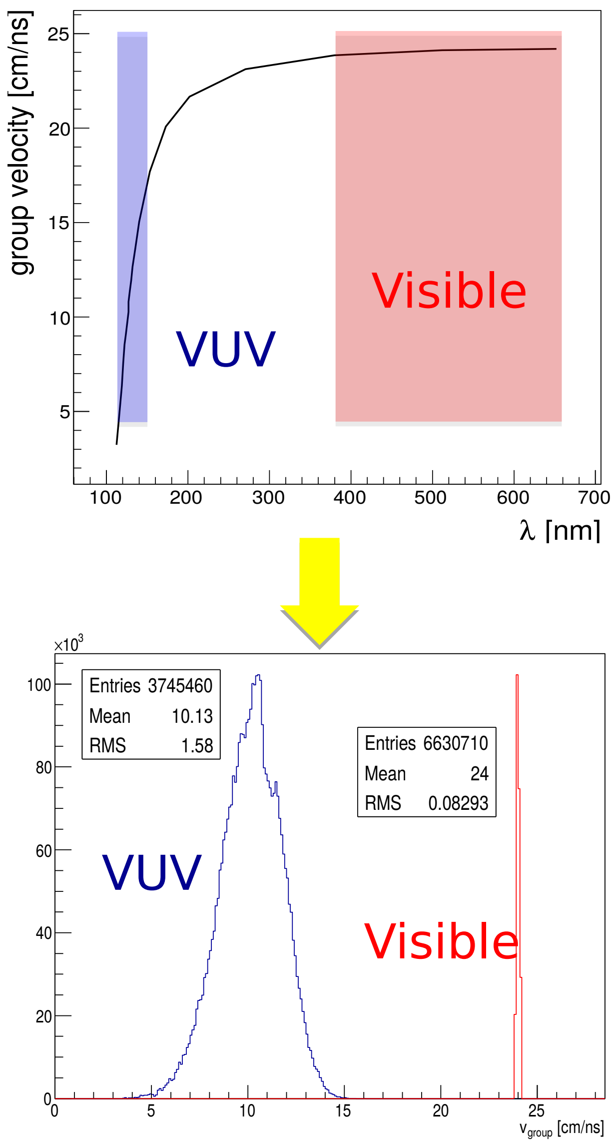
Scintillation Light in Argon (2)

Liquid argon is mostly transparent to its own scintillation.

At longer distances:
- Rayleigh scattering ~55cm $f(\lambda)$
- absorption, e.g. on nitrogen ~30 m @2ppm N2

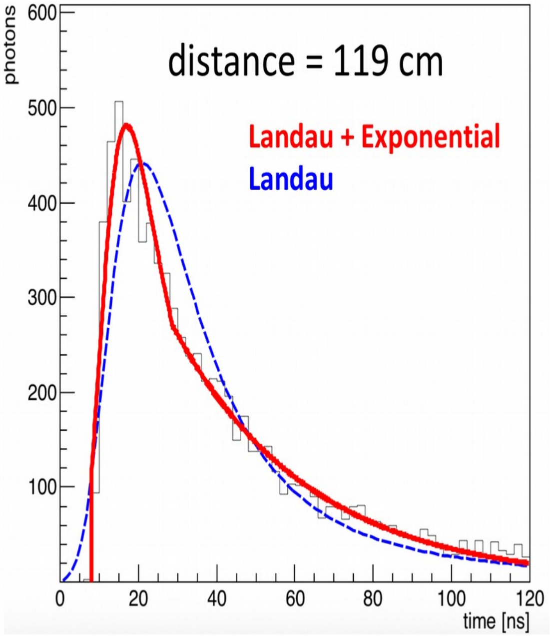
begin to play a role.

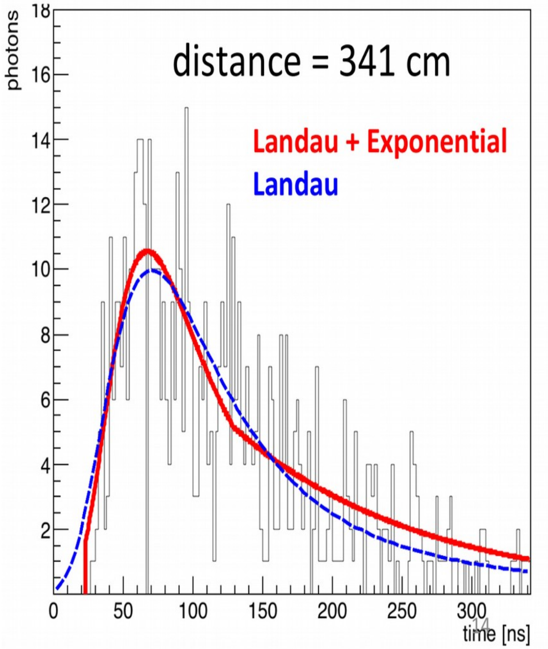
Note high refractive index ~1.5 and gradient of for VUV → relatively slow light.



VUV arrival times

- A previous version, using polynomials to predict the arrival timing distribution already exists in LArSoft. Good to about 300-350 cm.
- Landau + Exponential parameterisation of transport time distribution for the VUV (direct) component of light:
  - Landau + Exponential for distances < 300 cm
  - Landau for distances > 300 cm
- Parameterised in terms of the distance between scintillation point and optical detector.
- Predicts earliest arrival time and arrival time distribution accurately.
- Scales to size of DUNE and realistic X-Arapuca geometry without issues, no requirement for parameters saved in extended library.



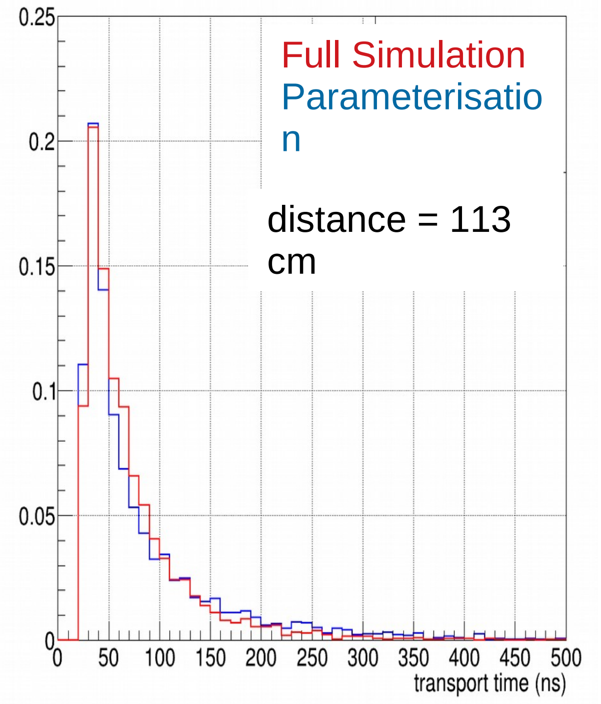


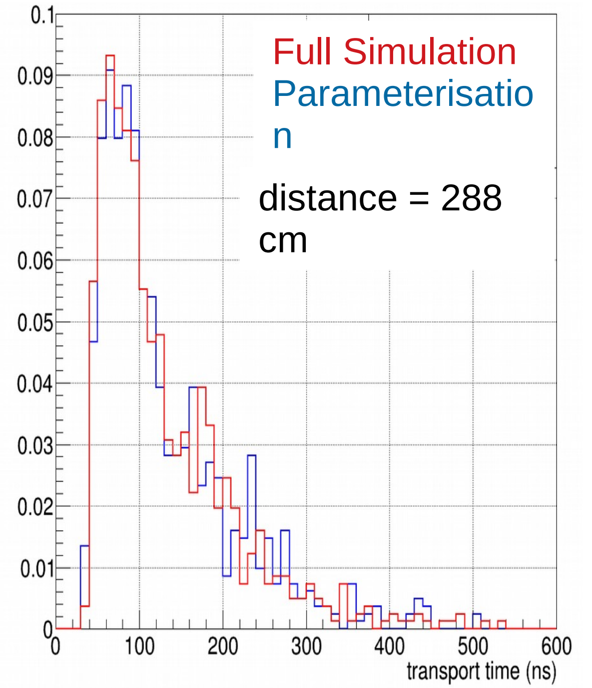
Visible arrival times

More challenging as light has to get to the cathode (as VUV) and then get to detectors (as visible). Many different paths possible.

1. Earliest arrival time:
   - fastest path light can take is calculated geometrically
   - VUV part of path given by Landau + Exponential
   - Visible part of path given by distance/velocity
2. Distribution approximated by smearing times along fastest path:
   - exponential smearing constructed such that earliest time unchanged but later times increasingly smeared
   - cut-off applied to avoid long tail from exponential
   - parameterised in terms of distance to cathode plane and angle along the fastest path

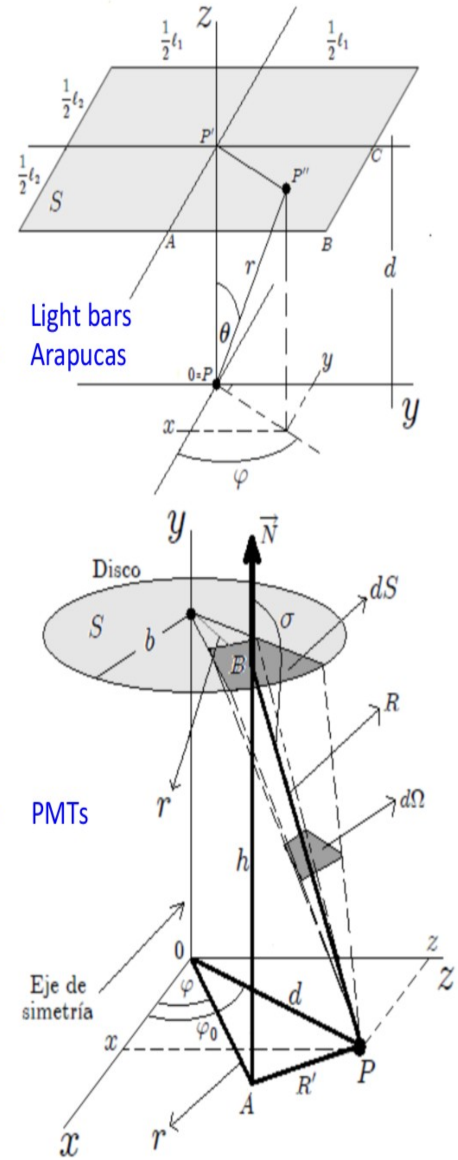
- Earliest arrival time predicted to +/- 0.5 ns and arrival time distribution well approximated by smearing.





Semi-analytic modelling of light

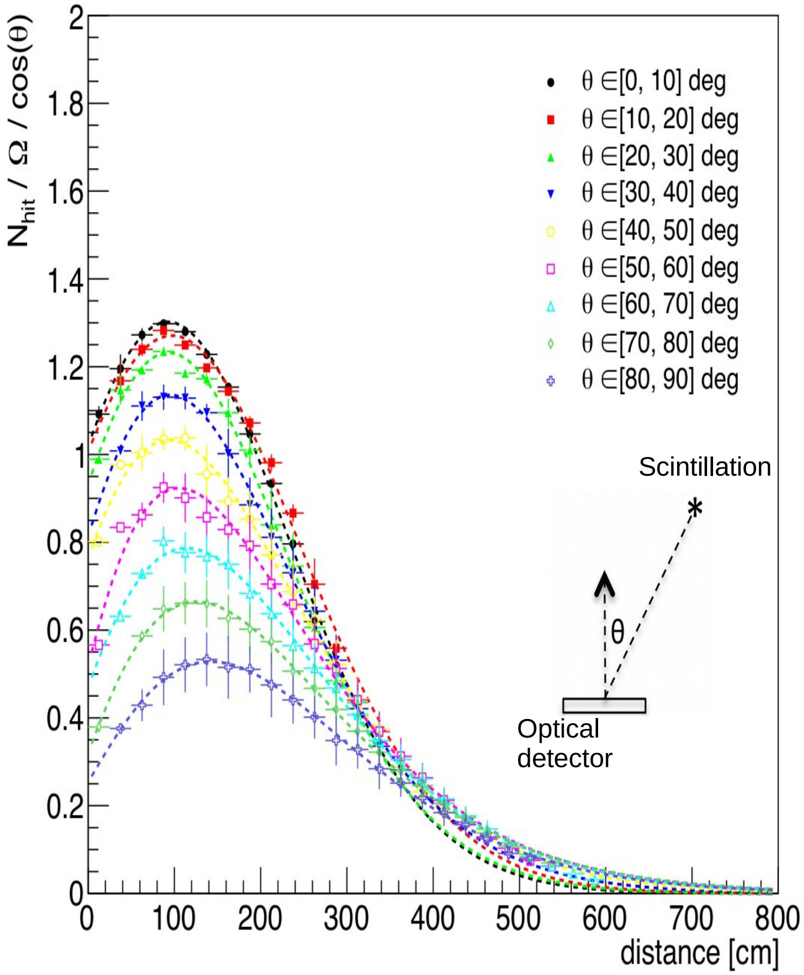
- Utilises the solid angle subtended by the optical detectors to predict the number of incident photons.
- Semi-analytic because corrections are required for effects that cannot be predicted via the solid angle:
  - Rayleigh scattering
  - Reflections from border walls / field cage
- Provides alternative to optical library for fiducial volume:
  - avoids large memory requirement of libraries
  - no issues from segmentation of bars into X-Arapuca supercells or even individual windows
  - can scale to full size of DUNE without issues (not just 1x2x6 region)



- Light bars
- Arapucas
- PMTs

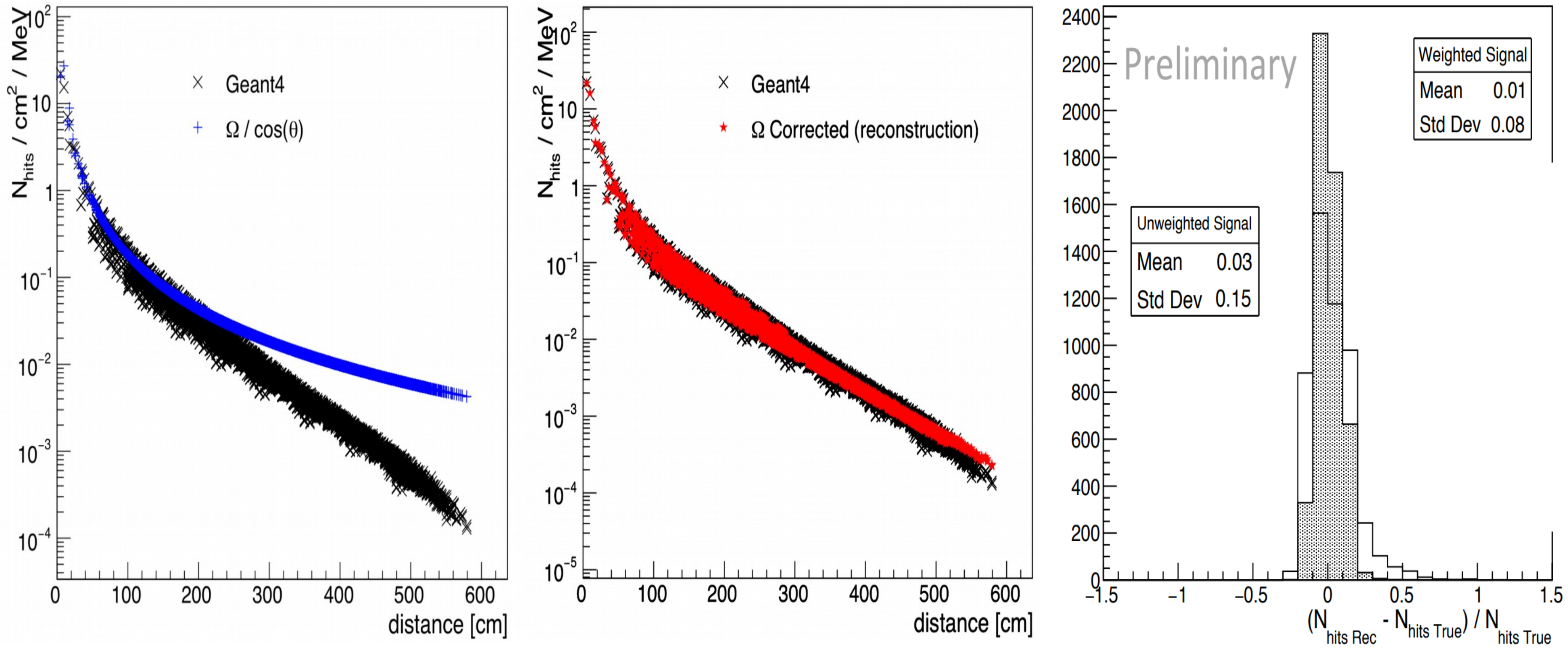
VUV (direct) light

- Semi-analytic model for the VUV (direct) component of the light:
  - solid angle of arapucas used to predict incident photons
  - Gaisser-Hillas corrections applied to account for Rayleigh scattering
- Effect of reflections from border walls small:
  - VUV photons predominantly absorbed
  - propagation of VUV photons heavily suppressed by scattering
- Detailed study of border effects is on-going.

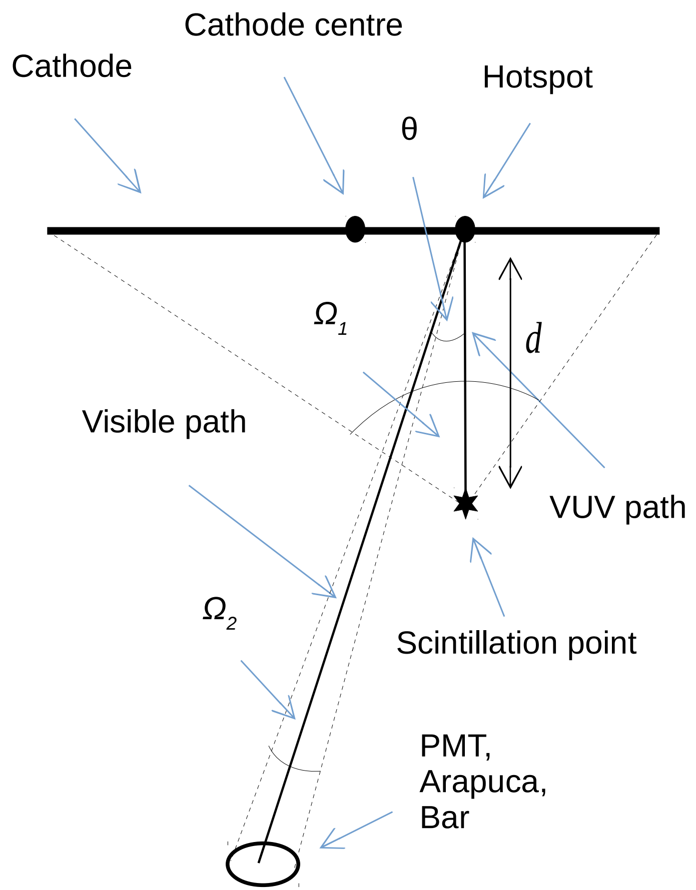


VUV (direct) light

- Results in idealised case without border effects very good: no bias and ~ 10 % resolution.
- Performs better than optical libraries: 15% underestimation and 23% resolution.



Visible (reflected) light



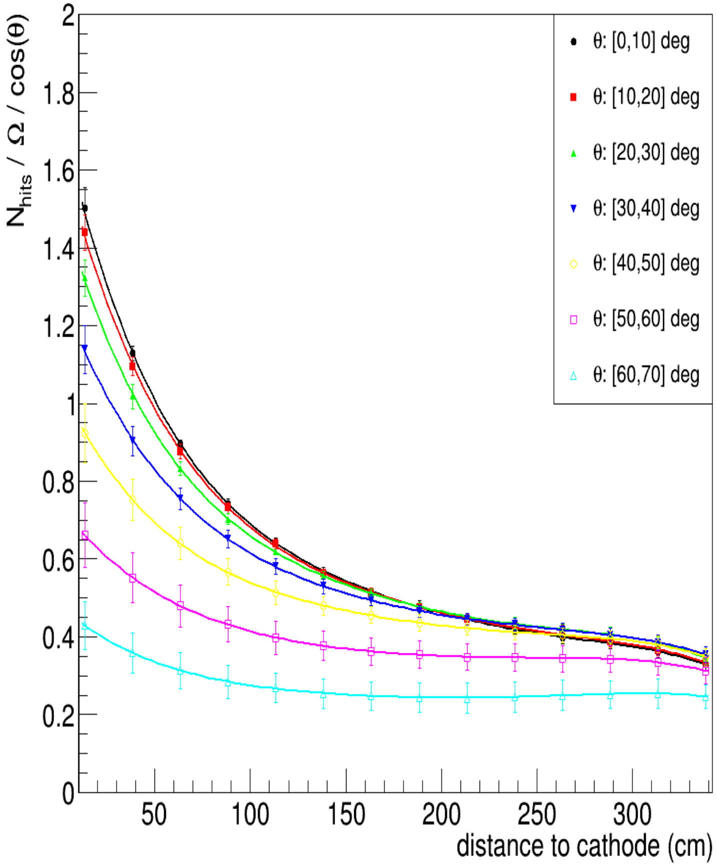
Semi-analytic model for visible light from TPB coated foils on the cathode:

- number of VUV photons incident on cathode calculated using solid angle Ω_1
- corrected for Rayleigh scattering using Gaisser-Hillas curves (direct VUV light)
- hotspot region assumed to dominate hits
- number of visible photons incident on optical detector calculated from solid angle Ω_2
- corrections applied to account for distribution of hits across reflective foils and for border effects

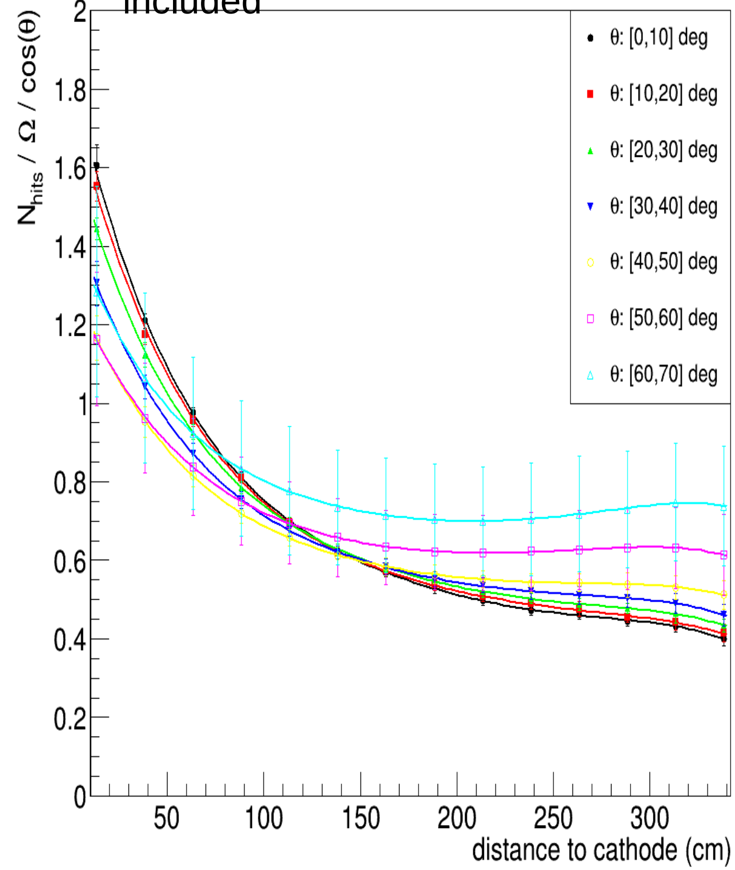
Visible (reflected) light

- Corrections for DUNE geometry with Arapuca window sized optical detectors.

No border

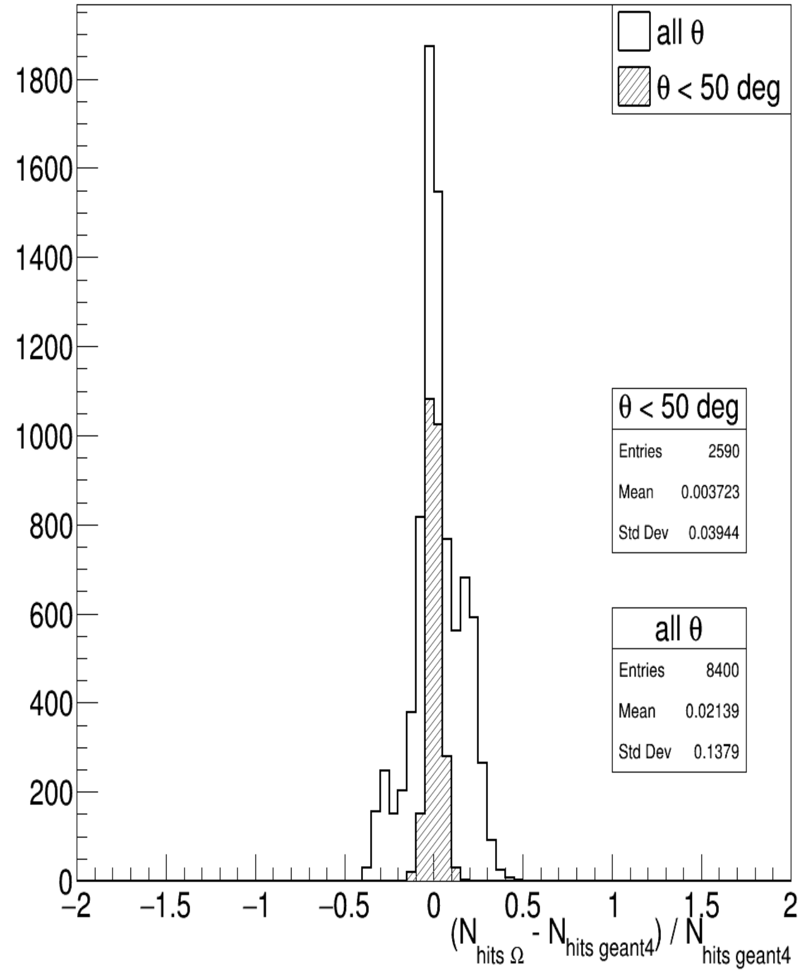


Reflective walls + field cage included



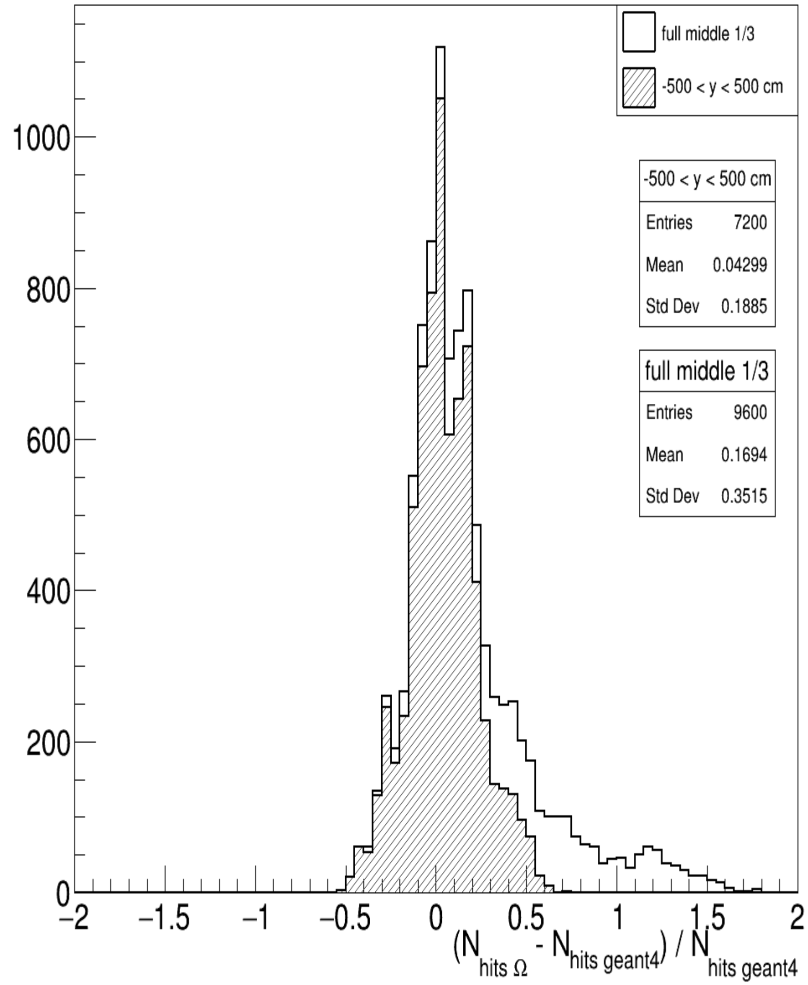
DUNE 1x2x6: centre

- Performance of visible (reflected) light semi-analytic model in DUNE very good:
  - No bias unlike optical libraries
  - Resolution ~ 15% across all angles and distances
  - Resolution ~ 5% for angles < 50 degrees
- Similar accuracy to VUV (direct) light semi-analytic model.
- Better performance than optical library (15% underestimation and 23% resolution).



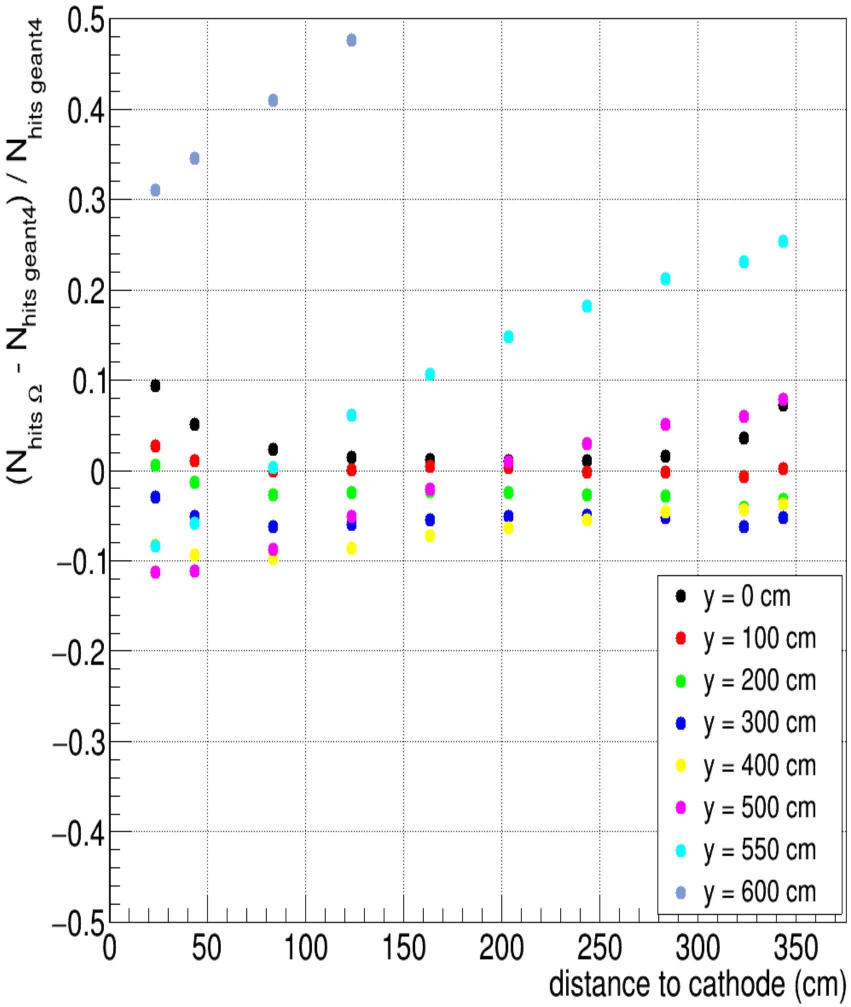
DUNE 1x2x6: middle 1/3

- In DUNE 1x2x6 middle 1/3 region:
  - always far from border walls in z-direction; no decrease in accuracy in number of hits compared with centre
  - but reflections from top and bottom (y-direction) have significant effect
- Semi-analytic model performs as well as optical library for range -500 < y < 500 cm:
  - resolution better than 20% across all angles and distances + no bias
  - covers ~80% of middle third volume
  - worse for y < -500 and y > 500 cm, further study of corrections for this region on-going



DUNE 1x2x6: middle 1/3

- Total hits summed across all optical channels from an energy deposition:
  - discrepancy within ~ +/- 10% in range – 500 < y < 500 cm, covering ~ 80% of middle third region
  - worse for y > 500 cm without further corrections being included, study of these effects on-going
- Performance similar for both individual arapuca window sized apertures and X-Arapuca supercell sized apertures.

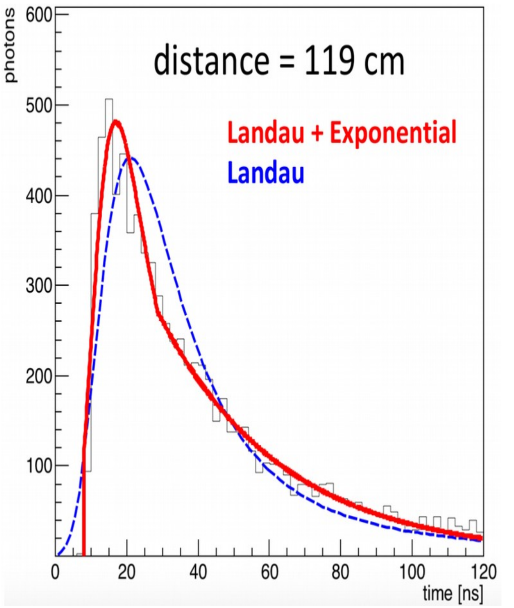


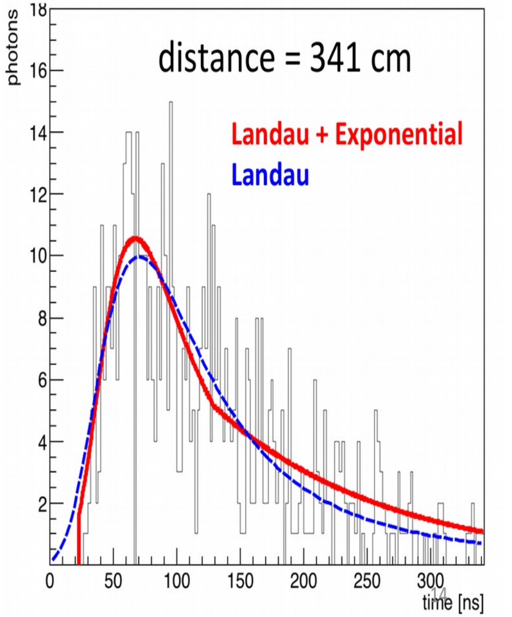
Implementation

- The main changes to the code are in larsim and larana, feature/lightprop_ugr_mcr
- List of files modified in larsim:
- **larsim/LarG4/OpDetPhotonTable.cxx** : added “fReflectedDetectedPhotons.clear();”
- **larsim/LarG4/OpFastScintillation.hh:**
  - added functions vuv & vis timings, vuv and vis hits + required parameters
  - added functions for interpolations, solid angle calculations, gaisser-hillas functions
- **larsim/LarG4/OpFastScintillation.cxx:** [bulk of added code]
  - edited constructor to read in paramters for timings and hits when flags set
  - added if statements to use timings / nhits model when flags set instead of library
  - implementation of all above functions
- **larsim/PhotonPropagation/PhotonVisibilityService.h:**
  - added required variables to store parameters loaded from fcl file
  - added functions to load to enable parameters to be loaded by reference to OpFastScintillation by constructor
- **larsim/PhotonPropagation/PhotonVisibilityService_service.cc**:
  - loads parameters from fcl file
  - faster calculation of timing.
- **larsim/PhotonPropagation/opticalsimparameterisations.fcl:**
  - contains all parameterisations
- **larsim/PhotonPropagation/photpropservices.fcl:**
  - now includes opticalsimparamterisations.fcl
  - new visibility services for using timings and nhits models

Optimising VUV calculation

- **larsim/PhotonPropagation/PhotonVisibilityService_service.cc**
- Previously the Landau + Exponential parameterisation was very slow, this now been fully resolved:
  - Original: parameterisations generated and sampled for exact distances, could not be saved between uses.
  - To allow random numbers to be drawn from distribution TF1::GetRandom() builds integral array – this is very slow.
  - Updated: discretisated in distance allowing the parameterisations to be generated once then stored.
- Huge efficiency improvement: ~ 100 times faster with 1cm steps (10000 20 MeV energy depositions randomly distributed)
- Discrepancy introduced by discretisation negligible: ~ 0.1 ns with 1cm steps





Design goals

- Default is for this to be turned off, this should minimize trouble for other experiments. If someone does want to use it, we will be happy to help out.
- We pre-define parameter sets in fHiCL which allow to choose in the parameter space:
  - Use/don't use analytic method
  - Use/don't use timing parametrization.
  - Use/don't use reflected light.
- The parameter sets are stored as lookup tables (1d or 2d arrays in FHiCl).

OpFastScintillation.hh

```
    std::vector<double> getVUVTime(double, int);
    void generateparam(int index);
    // Functions for vuv component Landau + Exponential timing parameterisation, updated method

    std::vector<double> getVISTime(TVector3 ScintPoint, TVector3 OpDetPoint, int Nphotons);
    // Visible component timing parameterisation

    int VUVHits(int Nphotons_created, TVector3 ScintPoint, TVector3 OpDetPoint, int optical_detector_type);
    // Calculates semi-analytic model number of hits for vuv component

    int VISHits(int Nphotons_created, TVector3 ScintPoint, TVector3 OpDetPoint, int optical_detector_type);
    // Calculates semi-analytic model number of hits for visible component

double finter_d(double*, double*);
double LandauPlusExpoFinal(double*, double*);
//For new VUV time parametrization
double interpolate( std::vector<double> &xData, std::vector<double> &yData, double x, bool extrapolate );
double* interpolate( std::vector<double> &xData, std::vector<double> &yData1, std::vector<double> &yData2,
                     std::vector<double> &yData3, double x, bool extrapolate);
double model_close(double*, double*);
double model_far(double*, double*);
// gaisser-hillas function
double GaisserHillas(double *x, double *par);
// structure definition for solid angle of rectangle function
struct acc{
  // ax,ay,az = centre of rectangle; w = width; h = height
  double ax, ay, az, w, h;
};
// solid angle of rectangular aperture calculation functions
double Rectangle_SolidAngle(double a, double b, double d);
double Rectangle_SolidAngle(acc& out, TVector3 v);

// solid angle of circular aperture calculation functions
double Disk_SolidAngle(double *x, double *p);
double Disk_SolidAngle(double d, double h, double b);
```

The new functions included

OpFastScintillation.cxx

```
if(pvs->IncludePropTime()) {
  //New VUV time parapetrization
  pvs->LoadTimingsForVUVPar(fparameters, fstep_size, fmax_d, fvuv_vgroup_mean, fvuv_vgrou

  // VIS time parameterisation
  if (pvs->StoreReflected()) {
    // load parameters
    pvs->LoadTimingsForVISPar(fdistances_refl, fcut_off_pars, ftau_pars, fvis_vmean, fn_L
  }
```

Parameter loading

```
if(pvs->UseNhitsModel()) {
  std::cout << "Using semi-analytic model for number of hits:" << std::endl;

  // Load Gaisser-Hillas corrections for VUV semi-analytic hits
  std::cout<<"Loading the GH corrections"<<std::endl;
  pvs->LoadGHForVUVCorrection(fGHvuvpars, fheight, fwidth, fradius, foptical_detect

  if(pvs->StoreReflected()) {
    // Load corrections for VIS semi-anlytic hits
    std::cout << "Loading vis corrections"<<std::endl;
    pvs->LoadParsForVISCorrection(fvispars, fplane_depth, fcathode_width, fcathode_
```

OpFastScintillation.cxx

```
std::map<int, int> DetectedNum;

std::map<int, int> ReflDetectedNum;

for(size_t OpDet=0; OpDet!=NOpChannels; OpDet++)
{
  G4int DetThisPMT = 0.;
  if(Visibilities && !pvs->UseNhitsModel()){
    DetThisPMT = G4int(G4Poisson(Visibilities[OpDet] * Num));
  }
  else {
    TVector3 ScintPoint( xyz[0], xyz[1], xyz[2] );
    TVector3 OpDetPoint(fOpDetCenter.at(OpDet)[0], fOpDetCenter.at(OpDet)[1], fOpDetCenter.at(OpDet)[2]);
    DetThisPMT = VUVHits(Num, ScintPoint, OpDetPoint, foptical_detector_type);

  }

  if(DetThisPMT>0)
  {
    DetectedNum[OpDet]=DetThisPMT;

    //   mf::LogInfo("OpFastScintillation") << "FastScint: " <<
    //   //   it->second<<" " << Num << " " << DetThisPMT;

    //det_photon_ctr += DetThisPMT; // CASE-DEBUG DO NOT REMOVE THIS COMMENT
  }
  if(pvs->StoreReflected()) {
    G4int ReflDetThisPMT = 0;
    if (!pvs->UseNhitsModel()){
      ReflDetThisPMT = G4int(G4Poisson(ReflVisibilities[OpDet] * Num));
    }
    else {
      TVector3 ScintPoint( xyz[0], xyz[1], xyz[2] );
      TVector3 OpDetPoint(fOpDetCenter.at(OpDet)[0], fOpDetCenter.at(OpDet)[1], fOpDetCenter.at(OpDet)[2]);
      ReflDetThisPMT = VISHits(Num, ScintPoint, OpDetPoint, foptical_detector_type);
    }

    if(ReflDetThisPMT>0)
    {
      ReflDetectedNum[OpDet]=ReflDetThisPMT;

    }
  }
```

Calculating the number of photons

OpFastScintillation.cxx

```
else if (pvs->IncludePropTime()) {
  // Get VUV photons arrival time distribution from the parametrization
  G4ThreeVector OpDetPoint(fOpDetCenter.at(OpChannel)[0]*CLHEP::cm,fOpDetCenter.at(OpChannel)[1]*CLHEP::cm,fOpDetCenter.at(OpChannel)[2]*CLHEP::cm);

  if (!Reflected) {
    double distance_in_cm = (x0 - OpDetPoint).mag()/CLHEP::cm; // this must be in CENTIMETERS!
    arrival_time_dist = getVUVTime(distance_in_cm, NPhotons); // in ns


  }
  else {
    TVector3 ScintPoint( x0[0]/CLHEP::cm, x0[1]/CLHEP::cm, x0[2]/CLHEP::cm ); // in cm
    TVector3 OpDetPoint_tv3(fOpDetCenter.at(OpChannel)[0], fOpDetCenter.at(OpChannel)[1], fOpDetCenter.at(OpChannel)[2]); // in cm
    arrival_time_dist = getVISTime(ScintPoint, OpDetPoint_tv3, NPhotons); // in ns
  }
}
```

Calculating the arrival times

PhotonVisibilityService.h

Loading parameter sets from the service (originally declared in fcl)

```
void LoadTimingsForVUVPar(std::vector<double> v[9], double& step_size, double& max_d, double& vuv_vgroup_mean, double& vuv_vgroup_max, double& inflexion_point_distance) const;
void LoadTimingsForVISPar(std::vector<double>& distances, std::vector<std::vector<double>>& cut_off, std::vector<std::vector<double>>& tau, double& vis_vmean, double& n_vis, double& n_vuv, double& pla
ne_depth) const;
void LoadGHForVUVCorrection(std::vector<std::vector<double>>& v, double& w, double& h, double& r, int& op_det_type) const;
void LoadParsForVISCorrection(std::vector<std::vector<double>>& v, double& plane_depth, double& w_cathode, double& h_cathode, std::vector<double>& cntr_cathode, double& w, double& h, double& r, int& o
p_det_type) const;
```

PhotonVisibilityService.h

Timings

```
//--------------------------------------------------------
void PhotonVisibilityService::LoadTimingsForVUVPar(std::vector<double> v[9], double& step_size, double& max_d,
                                                   double& vuv_vgroup_mean, double& vuv_vgroup_max, double& inflexion_point_distance) const
{
  v[0] = fDistances_all;
  v[1] = fNorm_over_entries;
  v[2] = fMpv;
  v[3] = fWidth;
  v[4] = fDistances;
  v[5] = fSlope;
  v[6] = fExpo_over_Landau_norm[0];
  v[7] = fExpo_over_Landau_norm[1];
  v[8] = fExpo_over_Landau_norm[2];

  step_size = fstep_size;
  max_d = fmax_d;
  vuv_vgroup_mean = fvuv_vgroup_mean;
  vuv_vgroup_max = fvuv_vgroup_max;
  inflexion_point_distance = finflexion_point_distance;

}

void PhotonVisibilityService::LoadTimingsForVISPar(std::vector<double>& distances, std::vector<std::vector<double>>& cut_off, std::vector<std::vector<double>>& tau,
                                                   double& vis_vmean, double& n_vis, double& n_vuv, double& plane_depth) const
{
distances = fDistances_refl;
cut_off = fCut_off;
tau = fTau;

vis_vmean = fvis_vmean;
n_vis = fn_LAr_vis;
n_vuv = fn_LAr_VUV;
plane_depth = fPlane_Depth;

}
```

PhotonVisibilityService.h

Nhits

```
void PhotonVisibilityService::LoadGHForVUVCorrection(std::vector<std::vector<double>>& v, double& w, double& h, double& r, int& op_det_type) const
{
  v = fGH_PARS;

  op_det_type = fOptical_Detector_Type;
  h = fAPERTURE_height;
  w = fAPERTURE_width;
  r = fPMT_radius;

}

void PhotonVisibilityService::LoadParsForVISCorrection(std::vector<std::vector<double>>& v, double& plane_depth, double& w_cathode, double& h_cathode, std::vector<double>& cntr_cathode, double& w, double& h, double& r, int& op_det_type) const
{
  v = fVIS_PARS;
  plane_depth = fPlane_Depth;
  w_cathode = fCATHODE_width;
  h_cathode = fCATHODE_height;
  cntr_cathode = fCATHODE_centre;

  op_det_type = fOptical_Detector_Type;
  h = fAPERTURE_height;
  w = fAPERTURE_width;
  r = fPMT_radius;

}
```

larsim/PhotonPropagation/opticalsimparameterisations.fcl

New file with all the parameter definitions

```
# Contains information required for:
# Timing parameterisations for VUV and Visible photons
# Corrections for semi-analytic number of hits models for VUV and visible light

# ***********************************************************************************************
#          PARAMETERS SETS FOR SEMI-ANALYTIC SIMULATION ARE DEFINED HERE FOR LATER USE
# ***********************************************************************************************
BEGIN_PROLOG

# VUV/DIRECT LIGHT: TIMING PARAMETERISATION
# Parameters of the Landau + Exponential (<= 300 cm) and Landau (> 300 cm) models
  # Landau parameters
  Distances_landau_all: [62.5, 87.5, 112.5, 137.5, 162.5, 187.5, 212.5, 237.5, 262.5, 287.5, 312.5, 337.5, 362.5, 387.5, 412.5, 437.5,
                         462.5, 487.5, 512.5, 537.5, 562.5, 587.5, 612.5, 637.5, 662.5, 687.5, 712.5, 737.5, 762.5, 787.5, 812.5, 837.5,
                         862.5, 887.5, 912.5, 937.5, 962.5, 987.5, 1012.5]
  Norm_over_entries_all: [1.46739, 0.880294, 0.571107, 0.344403, 0.261388, 0.20993, 0.170956, 0.152152, 0.136094, 0.120939, 0.109945, 0.10561,
                         0.0958589, 0.0904341, 0.0839861, 0.0803693, 0.0761224, 0.0716448, 0.0689223, 0.0668696, 0.0635322, 0.0616896, 0.0600554,
                         0.0577425, 0.0561707, 0.0539962, 0.0517729, 0.050872, 0.0488119, 0.0469635, 0.0466058, 0.0453283, 0.04366, 0.0425648, 0.0411353,
                         0.0396493, 0.0388351, 0.0383864, 0.0385728]
  Mpv_all: [6.05668, 9.05062, 12.3438, 17.1553, 21.9225, 26.9042, 33.1595, 37.1543, 43.6424, 49.9016, 55.7976, 59.7093, 66.7296, 72.5434, 78.6925, 84.1388,
                         90.0175, 97.1053, 101.911, 107.799, 113.748, 119.527, 124.167, 130.438, 134.988, 141.886, 147.848, 153.056, 160.653, 168.234, 171.821,
                         178.68, 186.341, 193.222, 202.61, 212.357, 222.014, 232.012, 235.169]
  Width_all: [0.977654, 1.57296, 2.42236, 4.15371, 5.76189, 7.49866, 9.87714, 10.9349, 13.4434, 15.5028, 19.8005, 20.6927, 22.9557, 24.5475, 26.2879, 27.7879, 29.296,
                         31.1889, 32.5755, 33.5919, 35.4932, 36.7544, 37.8536, 39.5658, 40.8131, 42.6414, 44.6617, 45.9231, 48.2682, 50.799, 51.9537, 54.1583, 57.8934,
                         61.4115, 65.4056, 70.7218, 74.6922, 79.6439, 81.8717]
  # Exponential parameters
  Distances_exp_all: [55, 65, 75, 85, 95, 105, 115, 125, 135, 145, 155, 165, 175, 185, 195, 205, 215, 225, 235, 245, 255, 265, 275, 285, 295]
  Slope_all: [-0.0885849, -0.0688936, -0.0519349, -0.0481695, -0.0408145, -0.0359584, -0.0341716, -0.0302902, -0.0282722, -0.026348, -0.0240983, -0.023172, -0.0220861,
                         -0.0207058, -0.0197934, -0.0193794, -0.0182472, -0.0175204, -0.0173492, -0.0163832, -0.0164717, -0.0160823,-0.0153199, -0.0149124, -0.0142298]
  # Line fit to the profiles of the ratio between the normalization parameters of the exponential and the landau functions vs distance (<= 300 cm)
  # Fits made to profiles in three different offset angle bins [0, 30deg], [30deg, 60deg] and [60deg, 90 deg]
  Expo_over_Landau_norm_0_all:  [-4.44152e-02, 1.25321e-03]
  Expo_over_Landau_norm_30_all: [-2.74406e-02, 1.27240e-03]
  Expo_over_Landau_norm_60_all: [ 3.48944e-02, 1.27080e-03]

# VISIBLE/REFLECTED LIGHT: TIMING PARAMETERISATION
  # SBND
  Distances_refl_SBND: [25,40,60,80,100,120,140,160,180,200]
  Cut_off_SBND: [ [220.559,239.781,261.162,280.35,297.07,305.335,316.825,321.326,323.348,321.925],
                  [232.93,252.297,272.975,290.81,306.027,316.828,325.465,329.651,334.128,331.066],
                  [248.255,267.175,287.91,303.054,315.817,327.364,336.522,337.163,339.698,336.158],
                  [258.65,275.184,292.176,302.28,322.826,327.364,336.522,337.163,339.698,336.158] ]
  Tau_SBND: [ [9.65303,7.80435,4.45676,3.2075,2.40353,1.7977,1.337,0.799091,0.289916,0.00731707],
              [11.7346,9.58924,5.45275,3.68209,2.69315,2.08817,1.48642,0.95226,0.411622,0.0406915],
              [13.3461,11.8424,5.87778,4.09909,3.00104,2.22286,1.54286,1.05221,0.457658,0.0435644],
              [14.2143,13.7812,6.58852,3.65,3.2,2.22286,1.54286,1.05221,0.457658,0.0435644] ]

 # DUNE-SP
  Distances_refl_SP: [38.3841,63.3841,88.3841,113.384,138.384,163.384,188.384,213.384,238.384,263.384,288.384,313.384,338.384]
  Cut_off_SP: [ [397.174,446.549,474.813,528.457,537.279,602.126,622.074,627.156,636.266,721.015,730.136,736.706,685.158],
                [435.09,483.062,515.06,559.063,591.479,593.486,617.949,681.81,730.478,702.538,686.021,664.066,654.974],
                [465.746,530.772,580.235,611.371,642.964,654.374,677.029,668.885,656.286,635.362,635.65,605.085,582.609],
                [519.167,569.909,578.288,596.77,610.101,624.591,601.545,591.347,605.464,578.32,551.695,549.043,514.396],
                [480.531,503.503,541.092,551.809,542.019,531.692,549.968,564.952,517.862,432.633,523.463,511.93,511.93] ]
  Tau_SP: [ [5.62562,4.76375,3.33937,2.6725,2.16875,1.79792,1.5,1.28917,1.14875,1.015,0.789286,0.519643,0.553571],
            [8.1424,4.63065,3.54856,2.77372,2.24805,1.95161,1.62822,1.38248,1.20475,1.07895,0.802715,0.653406,0.63129],
            [7.27332,4.67257,3.54638,2.81115,2.36643,2.01963,1.7605,1.5292,1.38233,1.09352,0.852451,0.823318,0.771707],
            [7.24211,4.46299,3.33076,2.65696,2.19649,1.83591,1.59777,1.43403,1.13786,1.15586,1.17439,1.39592,2.00126],
            [7.00268,3.95271,2.88257,2.26547,1.83216,1.54479,1.38363,1.63333,1.57875,2.97708,2.52083,5.475,5.475] ]
```

VUV time

Visible time

larsim/PhotonPropagation/opticalsimparameterisations.fcl

New file with all the parameter definitions

```
# VUV/DIRECT LIGHT: NUMBER OF HITS CORRECTIONS
#  SBN Gaisser-Hillas
 GH_RS60cm_SBN: [ [1.31326, 1.28293, 1.24304, 1.13739, 1.03286, 0.904908, 0.779762, 0.654461, 0.525548],
                  [87.5397, 95.0615, 89.6917, 94.9943, 93.9111, 111.708, 114.998, 132.535, 144.225],
                  [57.3686, 59.9412, 53.8011, 56.1887, 63.5782, 61.0104, 66.3173, 63.379, 64.4428],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
 GH_RS120cm_SBN: [ [1.16866, 1.13776, 1.08677, 0.993735, 0.885896, 0.760942, 0.628574, 0.498151, 0.37109],
                   [86.5078, 96.3383, 90.7074, 97.8305, 99.7487, 119.343, 129.554, 148.349, 174.775],
                   [88.0653, 89.4535, 82.9928, 84.9811, 93.5736, 89.2581, 93.4002, 91.709, 86.177],
                   [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
 GH_RS180cm_SBN: [ [1.11436, 1.08245, 1.02718, 0.939834, 0.83028, 0.704374, 0.566687, 0.430942, 0.299605],
                   [85.1942, 95.5351, 90.6834, 98.2426, 102.794, 123.284, 135.601, 153.886, 189.781],
                   [118.062, 119.348, 111.604, 112.464, 121.064, 114.846, 120.385, 119.249, 106.702],
                   [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
#  DUNE-SP Gaisser-Hillas
 GH_RS60cm_SP: [ [1.37378, 1.3634, 1.31054, 1.23488, 1.14697, 1.01977, 0.886863, 0.751005, 0.592496],
                  [113.764, 128.753, 122.512, 141.309, 140.16, 153.797, 170.915, 184.999, 199.248],
                  [81.3747, 78.791, 87.2706, 81.9593, 92.3303, 102.592, 110.304, 112.577, 107.575],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
 GH_RS120cm_SP: [ [1.22881, 1.20776, 1.15355, 1.08087, 0.988751, 0.868487, 0.736578, 0.604445, 0.465248],
                  [120.126, 137.211, 129.695, 150.215, 151.926, 168.741, 199.556, 223.586, 260.437],
                  [120.445, 115.844, 127.995, 114.96, 130.093, 141.39, 147.55, 154.139, 136.948],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
 GH_RS180cm_SP: [ [1.16447, 1.14188, 1.08141, 1.00912, 0.911832, 0.793711, 0.656118, 0.517022, 0.38575],
                  [120.862, 138.321, 129.506, 152.468, 154.87, 171.04, 210.579, 240.266, 297.42],
                  [156.572, 146.229, 173.181, 147.513, 165.223, 175.133, 182.79, 211.805, 173.369],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
```

N_{hits} SBN

N_{hits} DUNE-SP

larsim/PhotonPropagation/opticalsimparameterisations.fcl

w file with all the parameter definitions

```
#  DUNE-DP Gaisser-Hillas
 GH_RS60cm_DP: [ [1.2378, 1.24291, 1.20084, 1.13647, 1.04805, 0.928209, 0.81468, 0.687154, 0.538787],
                  [95.9886, 105.046, 114.902, 121.08, 126.533, 142.666, 143.314, 156.796, 159.649],
                  [170.762, 161.485, 146.444, 136.313, 128.357, 112.543, 106.582, 88.2847, 81.1439],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
 GH_RS120cm_DP: [ [1.15393, 1.13664, 1.09137, 1.02059, 0.927886, 0.800404, 0.675405, 0.544569, 0.410451],
                  [120.77, 131.444, 136.192, 143.061, 153.195, 168.244, 176.448, 189.741, 209.645],
                  [250.962, 234.726, 217.918, 202.817, 185.121, 167.992, 156.201, 133.598, 111.581],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
 GH_RS180cm_DP: [ [1.11356, 1.09376, 1.04515, 0.969378, 0.873762, 0.741592, 0.613477, 0.476078, 0.344799],
                  [129.151, 146.954, 147.726, 155.173, 166.108, 180.721, 193.948, 206.119, 240.13],
                  [321.246, 291.538, 278.321, 258.852, 234.141, 219.926, 197.511, 175.507, 139.981],
                  [-200, -200, -200, -200, -200, -200, -200, -200, -200] ]
```

N_{hits} DUNE-DP

```
# VIS/REFLECTED LIGHT: NUMBER OF HITS CORRECTIONS & PARAMETERS
#  DUNE-SP Parameters
VIS_RS60cm_SP: [ [1.8492,1.77663,1.66362,1.48639,1.31516,1.29986,1.4445,1.4445,1.4445],
                 [-0.0218717,-0.0198687,-0.0183486,-0.0152432,-0.012402,-0.0114963,-0.0133689,-0.0133689,-0.0133689],
                 [0.000166354,0.000141254,0.000138214,0.000114116,9.25045e-05,8.60675e-05,0.000107824,0.000107824,0.000107824],
                 [-7.23103e-07,-5.65755e-07,-5.96314e-07,-4.9295e-07,-3.96589e-07,-3.69486e-07,-4.937e-07,-4.937e-07,-4.937e-07],
                 [1.66888e-09,1.21352e-09,1.37176e-09,1.14986e-09,9.26552e-10,8.85755e-10,1.22458e-09,1.22458e-09,1.22458e-09],
                 [-1.5687e-12,-1.08276e-12,-1.29085e-12,-1.1068e-12,-8.97237e-13,-8.89813e-13,-1.22944e-12,-1.22944e-12,-1.22944e-12] ]
Plane_Depth_SP: 363.38405                 # foils/cathode x-coordinate
CATHODE_height_SP: 1209.466               # height of full cathode plane (in cm)
CATHODE_width_SP: 1394.34                 # width of full cathode plane (in cm)
CATHODE_centre_SP: [363.38405, 0, 696.294]       # cathode centre coordinates

# SBND Parameters
Plane_Depth_SBND: 0.0
```

ONLY DUNE-SP available set available for the visible light number of hits model at this time

larsim/PhotonPropagation/photpropmodules.fcl

- Enable optical library + time correction for the VUV/direct component

```
#
# Just enable direct timing parameterization on top of a photon library.
# Should work in all geometries
#
standard_library_vuv_prop_timing_photonvisibilityservice:
{
   # Start from the standard visibility service
   @table::standard_photonvisibilityservice

   # Flag to enable time parameterizations
   IncludePropTime: true

   # Generic VUV timing parameterization
   @table::common_vuv_timing_parameterization

}
```

larsim/PhotonPropagation/photpropmodules.fcl

- Enable optical library + time correction for the VUV/direct & VIS/reflected components

```
#
# Enable direct and reflected timing parameterization on top of a photon library.
# Works only for DUNE SP.
#
dunesp_library_vuv_vis_prop_timing_photonvisibilityservice:
{
   # This will need to be repalced in dunetpc with
   # dunesp-specific library settings
   @table::standard_library_vuv_prop_timing_photonvisibilityservice

   [redacted]
   StoreReflected: true

   # DUNE-specific VIS parameterization
   @table::dunesp_vis_timing_parameterization

}
```

larsim/PhotonPropagation/photpropmodules.fcl

- Enable Nhits model + time correction for the VUV/direct component

```
#
# Enable direct timing parameterization and Nhits model estimation.
# Works only for DUNE SP.
#
dunesp_Nhits_vuv_prop_timing_photonvisibilityservice:
{
   # Flags to enable parameterizations, disable library
   IncludePropTime: true
   UseNhitsModel: true
   DoNotLoadLibrary: true

   # Generic VUV timing parameterization
   @table::common_vuv_timing_parameterization

   # Semi-analytic VUV Nhits parameters
   GH_PARS: @local::GH_RS60cm_SP

   # Optical Detector information - to be replaced with using geometry service subsequently
   Optical_Detector_Type: 0     # 0 = rectangular, 1 = disk
   APERTURE_height: 9.3         # sensitive window heigh (in cm)
   APERTURE_width: 46.8          # sensitive window width supercell (in cm)
   PMT_radius: 10.16            # 8 inch diameter PMT (in cm)
}
```

The idea is to get this information directly from the gdml file in future iterations

larsim/PhotonPropagation/photpropmodules.fcl

- Enable Nhits model + time correction for the VUV/direct & VIS/reflected components

```
#
# Enable direct and reflected timing parameterization and Nhits model estimation.
# Works only for DUNE SP.
#
dunesp_Nhits_vuv_vis_prop_timing_photonvisibilityservice:
{
   # DUNE-SP VUV timing and Nhits settings
   @table::dunesp_Nhits_vuv_prop_timing_photonvisibilityservice

   # Flag to enable visible light simulation & load TPB properties
[redacted]
   StoreReflected: true

   # DUNE-specific VIS parameterization
   @table::dunesp_vis_timing_parameterization

   # Semi-analytic VIS Nhits parameters
   VIS_PARS: @local::VIS_RS60cm_SP
   @table::dunesp_cathode_info_for_nhits_vis

}
```

Other things we're fixing: SimPhotonCounter

- **larana/OpDet/SimPhotonCounter_module.cxx:**
  - Fix to read in Reflected light collections.

```
//Get *ALL* SimPhotonsCollection from Event
std::vector< art::Handle< std::vector< sim::SimPhotons > > > photon_handles;
evt.getManyByType(photon_handles);
if (photon_handles.size() == 0)
  throw art::Exception(art::errors::ProductNotFound)<<"sim SimPhotons retrieved and you requested them.";

for(auto mod : fInputModule){
// sim::SimPhotonsCollection TheHitCollection = sim::SimListUtils::GetSimPhotonsCollection(evt,mod);
//switching off to add reading in of labelled collections: Andrzej, 02/26/19

  for (auto ph_handle: photon_handles) {
   // Do some checking before we proceed
   if (!ph_handle.isValid()) continue;
   if (ph_handle.provenance()->moduleLabel() != mod) continue;   //not the most efficient way of doing this, but preserves the logic of the module. Andrzej

  bool Reflected = (ph_handle.provenance()->productInstanceName() == "Reflected");
```

SimPhotonCounter cont'd

```
  else {
    // store in appropriate trees using "Reflected" handle and pvs->StoreReflected() flag
    // Increment per OpDet counters and fill per phot trees
    fCountOpDetAll++;
    if(fMakeAllPhotonsTree){
      if (!Reflected || (pvs->StoreReflected() && Reflected)) {
        fThePhotonTreeAll->Fill();
      }
     }

    if(odresponse->detected(fOpChannel, Phot))
    {
      if(fMakeDetectedPhotonsTree) fThePhotonTreeDetected->Fill();
      //only store direct direct light
      if(!Reflected)
        fCountOpDetDetected++;
      // reflected and shifted light is in visible range
      else if(pvs->StoreReflected() && Reflected ) {
        fCountOpDetReflDetected++;
        // find the first visible arrival time
        if(pvs->StoreReflT0() && fTime < fT0_vis)
          fT0_vis = fTime;
      }
      if(fVerbosity > 3)
        std::cout<<"OpDetResponseInterface PerPhoton : Event "<<fEventID<<" OpChannel " <<fOpChannel << " Wavelength " << fWavelength << " Detected 1 "<<std::endl;
    }
    else {
      if(fVerbosity > 3)
        std::cout<<"OpDetResponseInterface PerPhoton : Event "<<fEventID<<" OpChannel " <<fOpChannel << " Wavelength " << fWavelength << " Detected 0 "<<std::endl;
    }
  }
}
```

Remove duplicate code
and add selection on
Collection label rather than
On wavelength
(new LArG4 logic)

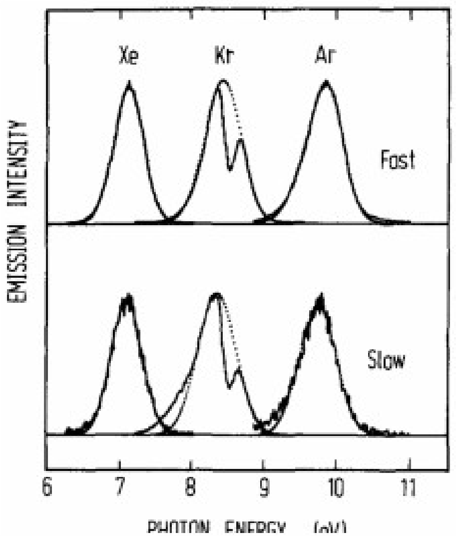
SimPhotonCounter Cont'd

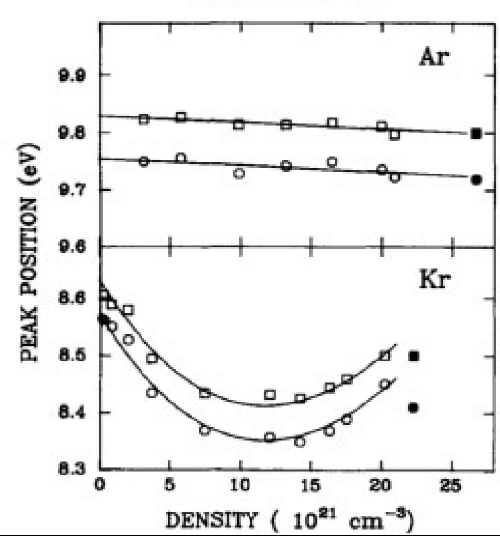
```
// If this is a library building job, fill relevant entry
art::ServiceHandle<phot::PhotonVisibilityService> pvs;
if(pvs->IsBuildJob() && !Reflected) // for library build job, both componenents stored in first object with Reflected = false
{
  int VoxID; double NProd;
  pvs->RetrieveLightProd(VoxID, NProd);
  pvs->SetLibraryEntry(VoxID, fOpChannel, double(fCountOpDetDetected)/NProd);

  //store reflected light
  if(pvs->StoreReflected())
    pvs->SetLibraryEntry(VoxID, fOpChannel, double(fCountOpDetReflDetected)/NProd,true);
  //store reflected first arrival time
  if(pvs->StoreReflected() && pvs->StoreReflT0())
    pvs->SetLibraryReflT0Entry(VoxID, fOpChannel, fT0_vis);
}
```

Full sim used to generate library does not fill the “Reflected” collection. Need to catch for that to maintain library building working.

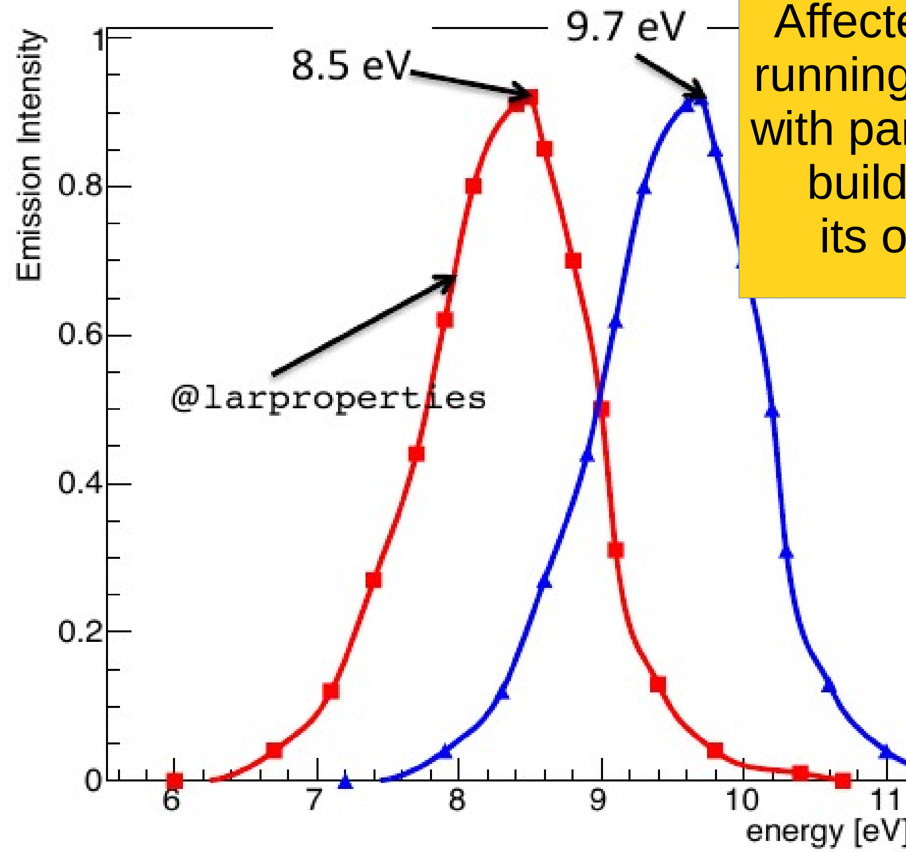
Fast and slow scintillation emission spectra in `larproperties.fcl`

J. Chem. Phys. 91, 1469 (1989)





The scintillation spectrums present in `larproperties.fcl` seem to be for Kr instear of Ar!



This could have only Affected someone running Full sim jobs with particles – library building defines its own energy

Future Improvements

- The way LArG4 modules saves the collections.
  - Full sim library jobs do not differentiate between visible and reflected. This makes sense, but perhaps shouldn't save the second collection then?
- The way SimPhotonCounter does too many things at once.
  - Might be worth splitting into two modules: tree building and library building.
- The way we get the detector sizes (should be from gdml)
  - Have code from Alex, need to implement (short term).
- Include border effects.
  - Could be just via .fcl parametrization.
- Split off into its own Physics Module

Testing

- DUNE CI tests ran to completion.
- ArgoNeuT and LAriAT as well.
- ICARUS and MicroBooNE have some problems (will check up with the relevant people)
- SBND has previous version tagged. Will recheck shortly.

Conclusions

- We have a semi-analytical model of predicting the light detected at a given PMT/light detector.
- DUNE urgently needs this for the next MC production.
- The code should be transparent to anyone who doesn't need it.

Backup

SBND new optical library

- ✓ 5x5x5cm3 voxel size
- ✓ cover all cryostat (active + non instrumented argon)
- ✓ 500000 photons/voxel

```
services.PhotonVisibilityService.XMin: -260.1
services.PhotonVisibilityService.XMax: 260.1
services.PhotonVisibilityService.YMin: -271.15
services.PhotonVisibilityService.YMax: 271.15
services.PhotonVisibilityService.ZMin: -143.1
services.PhotonVisibilityService.ZMax: 559.6

services.PhotonVisibilityService.NX: 104
services.PhotonVisibilityService.NY: 109
services.PhotonVisibilityService.NZ: 141
```

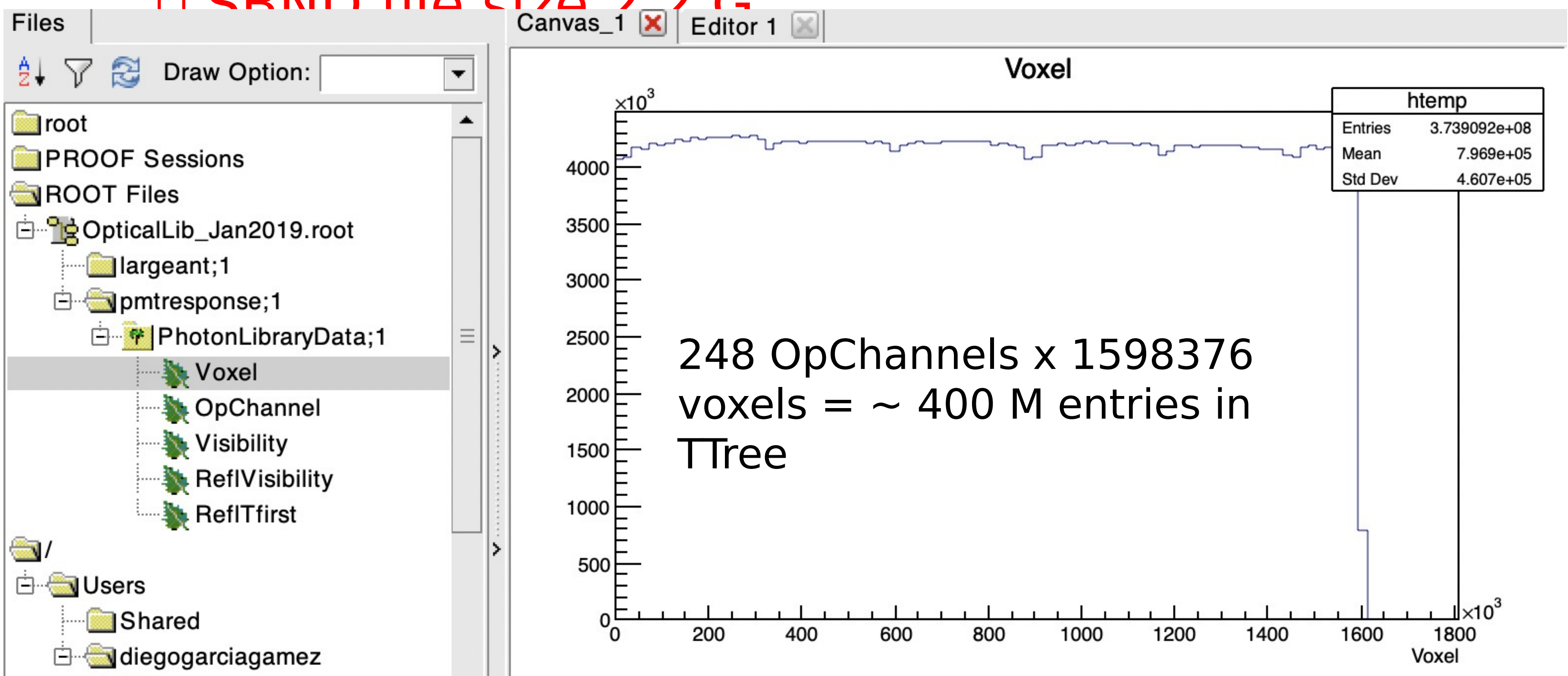
- Full LAr volume sampled: 1st time in SBND
- 104 x 109 x 141 = 1598376 voxels
- 104 voxels/job → 15369 grid jobs

```
#setting the optical properties of the materials in the geometry:
services.LArPropertiesService.ReflectiveSurfaceEnergies:       [ 1.77, 2.
services.LArPropertiesService.ReflectiveSurfaceNames:          [ "STEEL_STAINLESS_Fe7Cr2Ni", "Copper_Beryllium_alloy25", "G10", "vm2000", "ALUMINUM_Al" ]
services.LArPropertiesService.ReflectiveSurfaceReflectances:   [ [ 0.66, 0.64, 0.62, 0.60, 0.59, 0.57, 0.53, 0.47, 0.39, 0.36, 0.27, 0.25 ],
                         [ 0.902, 0.841, 0.464, 0.379, 0.345, 0.299, 0.287, 0.264, 0.337, 0.3, 0.0, 0.0 ],
                         [ 0.393, 0.405, 0.404, 0.352, 0.323, 0.243, 0.127, 0.065, 0.068, 0.068, 0.0, 0.0 ],
                         [ 0.93, 0.93, 0.93, 0.93, 0.93, 0.93, 0.1, 0.1, 0.7, 0.3, 0.0, 0.0 ],
                         [ 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.47, 0.39, 0.36, 0.27, 0.25 ] ]
```

Optical library issues (size)

SBND:

- ✓ 500000 photons/voxel
- ✓ 5x5x5 cm^3 voxel size
- ✓ cover all cryostat (active + non instrumented argon)
  - SBND file size 2.2 G



248 OpChannels x 1598376 voxels = ~ 400 M entries in TTree

Optical library issues (location)

Files in cvmfs must be less than 1 GB enable StashCache (persistent dCache areas) for SBND (already used by DUNE and to store flux files).

More details about StashCache are available at this link:
https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Introduction_to_FIFE_and_Component_Services#OASISCVMFS-process-for-handling-partially-reused-data-files-StashCache

- 1st step already done (two week ago)

```
dgamez@fnal:/pnfs/sbnd/persistent/stash
> ls -l
total 2342937
-rw-r--r-- 1 sbnd sbnd 2398118839 Feb  5 04:59 OpLib_SBN_ND_v0.root
```

- Fermilab computer support started the other tickets that need to be done with dCache and networking

Optical library issues (Memory)

SBND file size 2.2 G jobs loading the library will use about 4.5 GB of memory

- Loading both components: Direct and Reflected light

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
31121	dgamez	20	0	4393m	3.3g	12m	R	93.2	28.5	0:35.96	lar
17891	netdata	20	0	18236	3580	1128	S	2.0	0.0	1:26.39	apps.plugin
18254	jtenavid	20	0	724m	11m	3412	S	2.0	0.1	553:11.49	knotify4
24170	djbarker	20	0	669m	2272	1284	S	2.0	0.0	915:39.77	knotify4
30552	aezeribe	20	0	724m	10m	3464	S	2.0	0.1	570:13.43	knotify4

- Loading only the Direct light component (1/2 of memory used, as expected)

```
Mem:   12196144k total, 11900004k used,   296140k free,    15984k buffers
Swap:  2097148k total,  1656724k used,   440424k free,  3481936k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
30319	dgamez	20	0	2883m	1.9g	116m	R	100.0	16.7	0:31.46	lar
17891	netdata	20	0	18236	3596	1144	S	2.0	0.0	1:21.58	apps.plugin
30766	dbrailsf	20	0	669m	18m	12m	S	2.0	0.2	1:52.12	knotify4
18254	jtenavid	20	0	724m	12m	3616	S	1.7	0.1	553:07.25	knotify4

Can/should we make our library smaller?

ROOT::ECompressionAlgorithm::kZLIB
vs
ROOT::ECompressionAlgorithm::kLZMA

Done by Gianluca

Compression level	kZLIB	kLZMA	kLZ4
1	2689581002 (+12%)	2197283298 (-8%)	3242441570 (+35%)
2	2560402244 (+6%)	2158760978 (-9%)	3242440477 (+35%)
3	2488312464 (+3%)	2142154946 (-10%)	3242440250 (+35%)
4	2473930140 (+3%)	1931374682 (-19%)	2809094396 (+17%)
5	2450944666 (+2%)	1895073526 (-20%)	2780549548 (+15%)
6	2392740105 (+0%)	1891282882 (-21%)	2760065585 (+15%)
7	2373894946 (-1%)	1891283050 (-21%)	2746452834 (+14%)
8	2353355522 (-1%)	1891282866 (-21%)	2738858081 (+14%)
9	2348525797 (-2%)	1891282866 (-21%)	2734283988 (+14%)
10	2348525043 (-2%)	1891282928 (-21%)	2734283743 (+14%)

But new tree reads in 250 seconds, while the original one takes 160 seconds