Light travel time distribution in protoDUNE DP

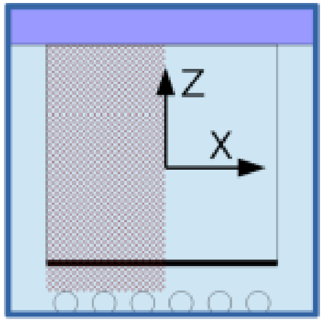
Update in LArSoft Dual Phase Light simulations

J. Soto

2. Where do we come from?
Light Maps

- **QScan** uses the *Light Maps* to compute the number of collected photons and their arrival time distribution.
- They are generated with LightSim by the **LAPP team (Annecy)** using a complete geometry.
- One map for LAr, another one for GAr.
- It contains 4 parameters per VoxelxPMT:
  - Probability to reach the PMT [visibility].
  - Travel time distribution (landau fit) [t0,MPV,σ].
- Every voxel is a cube of 25x25x25 cm^3.
- LAr absorption process is not included in the map generation. To be parametrized when using the map.
- Parameters used in the generation of the maps:

Rayleigh scattering length	55cm 350cm	(128nm) (435nm)
Absorption on stainless-steel and copper	100% 50%	(128nm) (435nm)
LAr refractive index	1.38 1.25	($\lambda < 130$nm) ($\lambda > 130$nm)



24x24x29 voxels for protoDUNE DP (see above that the volume included in the light maps is larger than the active volume in the drift direction).
4x4x12 voxels for WA105 3x1x1m³.

More information in Anne Chappuis' presentation:
https://indico.fnal.gov/event/15325/

31/01/18 DUNE Collaboration Meeting

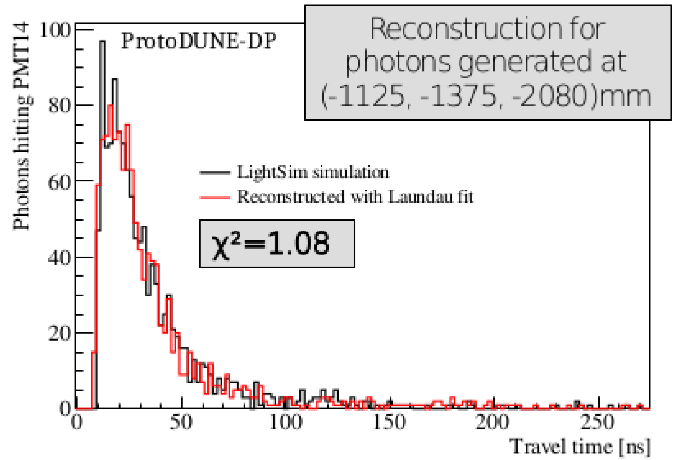
Travel time distribution characteristics

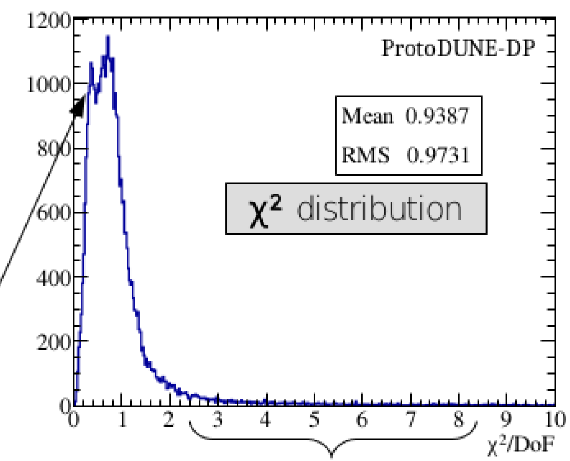
DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT — WA105

The time distribution is reconstructed using a landau fit and 3 parameters:

- t_0 (the first bin $N_{entries} > 0$)
- The MPV and σ of the landau fit
- Voxels with $N_{entries} < 50$ are not taken into account

For LAr light maps





Peak at $\chi^2 < 0.4$: voxels with a **small number** of collected photons

6.2% of voxels with $\chi^2 > 2.5$: voxels close to the PMT

→ Satisfactory in most cases

→ Has to be optimized for the voxels close to the PMT.

Anne CHAPPUIS, 13 June 2017 ProtoDUNE-DP 21/28

(Anne Chappuis)

LArSoft modifications

- New photonlibrary format: A new branch is added with the fit parameters.

```
root [2] PhotonLibraryData->Print()
******************************************************************************
*Tree    :PhotonLibraryData: PhotonLibraryData
*Entries :   601344 : Total =        14483488 bytes  File  Size =    7451392 *
*        :          : Tree compression factor =   1.94                       *
******************************************************************************
*Br    0 :Voxel     : Voxel/I                                                *
*Entries :   601344 : Total  Size=    2413650 bytes  File Size  =     849055 *
*Baskets :       76 : Basket Size=      32000 bytes  Compression=   2.84     *
*............................................................................*
*Br    1 :OpChannel : OpChannel/I                                            *
*Entries :   601344 : Total  Size=    2413970 bytes  File Size  =      21406 *
*Baskets :       76 : Basket Size=      32000 bytes  Compression= 112.69     *
*............................................................................*
*Br    2 :Visibility : Visibility/F                                          *
*Entries :   601344 : Total  Size=    2414050 bytes  File Size  =    1554259 *
*Baskets :       76 : Basket Size=      32000 bytes  Compression=   1.55     *
*............................................................................*
*Br    3 :timing_par : timing_par[3]/F                                       *
*Entries :   601344 : Total  Size=    7244418 bytes  File Size  =    5021755 *
*Baskets :      227 : Basket Size=      32000 bytes  Compression=   1.44     *
*............................................................................*
```

NEW

J. Soto

LArSoft modifications

- Feature branch in **larsim** with the modifications:

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   larsim/LArG4/OpFastScintillation.cxx
        modified:   larsim/PhotonPropagation/PhotonLibrary.cxx
        modified:   larsim/PhotonPropagation/PhotonLibrary.h
        modified:   larsim/PhotonPropagation/PhotonVisibilityService.h
        modified:   larsim/PhotonPropagation/PhotonVisibilityService_service.cc

no changes added to commit (use "git add" and/or "git commit -a")
```

  - Definitions of the corresponding functions attached to the new branches/variables.
  - To do: Creation of the parameters when creating a new photon library.

Changes in PhotonLibrary.h/cxx

- New functions and variables:

```
float GetTimingPar(size_t Voxel, size_t OpChannel, size_t parnum) const;
void SetTimingPar(size_t Voxel, size_t OpChannel, float Count, size_t parnum);

///Returns whether the current library deals with time propagation distributions.
int hasTiming() const { return fHasTiming; }

std::vector<std::vector<float>> fTimingParLookupTable;

/// Unchecked access to a parameter the time distribution
float const& uncheckedAccessTimingPar (size_t Voxel, size_t OpChannel, size_t parnum) const
{ return fTimingParLookupTable[uncheckedIndex(Voxel, OpChannel)][parnum];}

/// Unchecked access to a parameter of the time distribution
float& uncheckedAccessTimingPar(size_t Voxel, size_t OpChannel, size_t parnum)
{ return fTimingParLookupTable[uncheckedIndex(Voxel, OpChannel)][parnum]; }
```

- Edition of previous functions:

All new code must be activated with a fhicl parameter

```
if(fHasTiming!=0)
{
  fTimingParLookupTable.resize(LibrarySize());
  for(size_t k=0;k<LibrarySize();k++) fTimingParLookupTable[k].resize(getTiming,0);
}
```

Changes in PhotonVisibilityService.

- New functions and variables:

```
const std::vector<float>* GetTimingPar( double const* xyz ) const;
void SetLibraryTimingParEntry( int VoxID, int OpChannel, float value, size_t parnum );
const std::vector<float>* GetLibraryTimingParEntries( int VoxID ) const;
float GetLibraryTimingParEntry( int VoxID, int Channel, size_t npar ) const;

bool IncludeParPropTime() const { return fParPropTime; }
size_t ParPropTimeNpar() const { return fParPropTime_npar; }
std::string ParPropTimeFormula() const { return fParPropTime_formula; }

 bool                    fParPropTime;
 size_t                  fParPropTime_npar;
 std::string             fParPropTime_formula;
```

- Changes in existing functions:

```
fParPropTime          = p.get< bool        >("ParametrisedTimePropagation", false);
fParPropTime_npar     = p.get< size_t      >("ParametrisedTimePropagationNParameters", 0);
fParPropTime_formula  = p.get< std::string >("ParametrisedTimePropagationFittedFormula","");
if (!fParPropTime) fParPropTime_npar=0;
```

Fhicl parameters are inactivated by default.

LArSoft modifications

- Feature branch in **larsim** with the modifications:

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   larsim/LArG4/OpFastScintillation.cxx
        modified:   larsim/PhotonPropagation/PhotonLibrary.cxx
        modified:   larsim/PhotonPropagation/PhotonLibrary.h
        modified:   larsim/PhotonPropagation/PhotonVisibilityService.h
        modified:   larsim/PhotonPropagation/PhotonVisibilityService_service.cc

no changes added to commit (use "git add" and/or "git commit -a")
```

  – Introduction of the delay due to the propagation, with a function parametrized by the user:

```
TF1 TravelTimeFunction("Name","TMath::Landau(x, [0], [1])",par[0],range2);
for(int i=1; i<NPar; i++) TravelTimeFunction→SetParameter(i-1,par[i]);
deltaTime += TravelTimeFunction.GetRandom();
```

OpFastScintillation.cxx

- Modifications in **RecordPhotonsProduced**:
  - Reading the parameters:

```
const std::vector<float>* PropParameters = nullptr;
if(pvs->IncludeParPropTime()) PropParameters = pvs->GetTimingPar(xyz);

std::map<int, TF1> PropTimeFunction;

if(pvs->IncludeParPropTime()) {
  double range=0;
  for(size_t i=0; i<pvs->ParPropTimeNpar();i++) range+=100*PropParameters[OpDet][i];
  TF1 AuxFunction(Form("timingfunc%i",(int)((ssize_t)OpDet)),Form("%s",pvs->ParPropTimeFormula().c_str()),PropParameters[OpDet][0],range);
  for(size_t i=1; i<pvs->ParPropTimeNpar();i++) AuxFunction.SetParameter(i-1, PropParameters[OpDet][i]);
  PropTimeFunction[OpDet]=AuxFunction;
  //std::cout << "getrandom " << flandauu->GetRandom() << std::endl;
}
```

  - Introducing the delay:

```
if(pvs->IncludeParPropTime()) {
    deltaTime += PropTimeFunction[itdetphot->first].GetRandom();
}
```


J. Soto

How to use it

- Changes published in the feature branch:
  feature/jsoto_dphase_timing3x1x1
- Activate propagation time in your fhicl with:

  services.PhotonVisibilityService.ParametrisedTimePropagation: **true**

  services.PhotonVisibilityService.ParametrisedTimePropagationNParameters: **3**

  services.PhotonVisibilityService.ParametrisedTimePropagationFittedFormula: **"TMath::Landau(x,[0],[1])"**
- The first parameter is always the low range limit of the function, the second and third are placed inside the defined function according to the order given by the string.
- The extended photon library with the time parameters is needed (a tool to create it to be done).

Example

- A low energy alpha particle generated with a customized yield (to produced many photons).
- Left: Photons arrived to PMT 16 (around the center) with and without the propagation activated