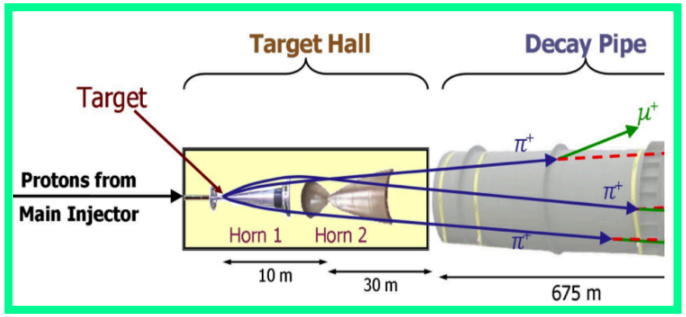
Integrating PPFX reweighting into larsim

Katrina Miller

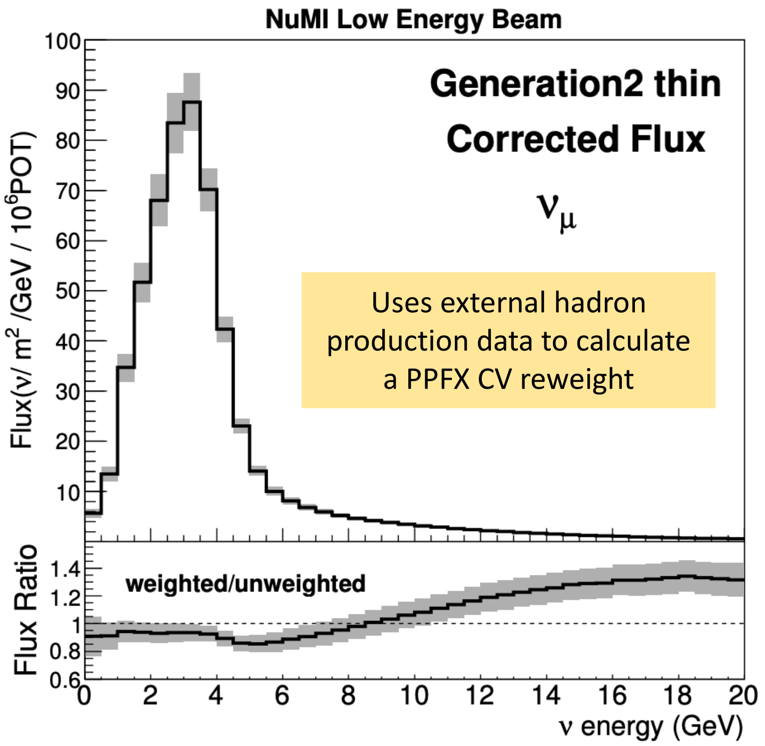
June 2, 2020

LArSoft Coordination Meeting

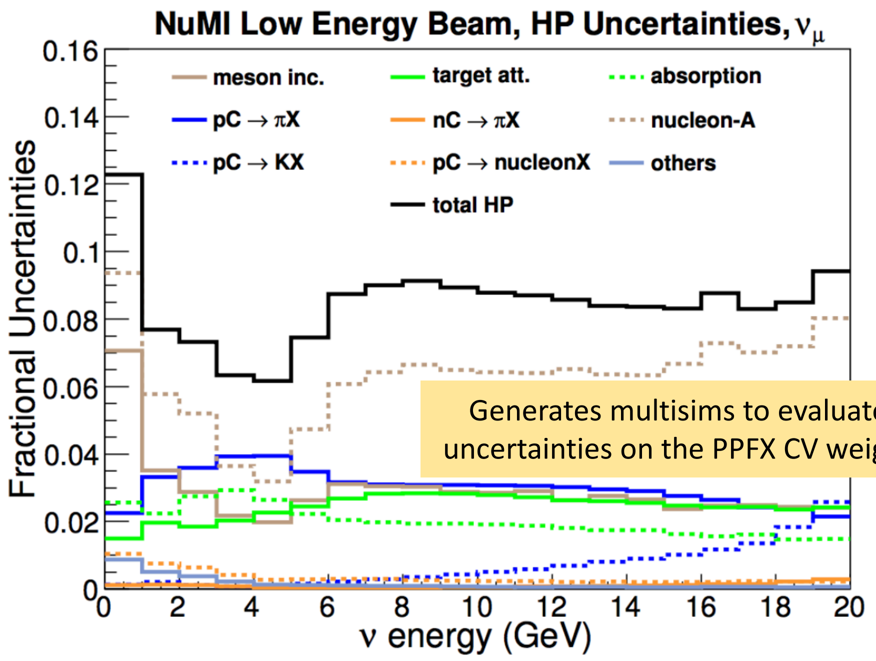
Introduction to PPFX



- PPFX is an **experiment-agnostic** reweight package developed by the MINERvA collaboration to correct the NuMI GEANT4 flux simulation
- Requires **dk2nu** file format to access neutrino ancestry (hadron production) information



Uses external hadron production data to calculate a PPFX CV reweight

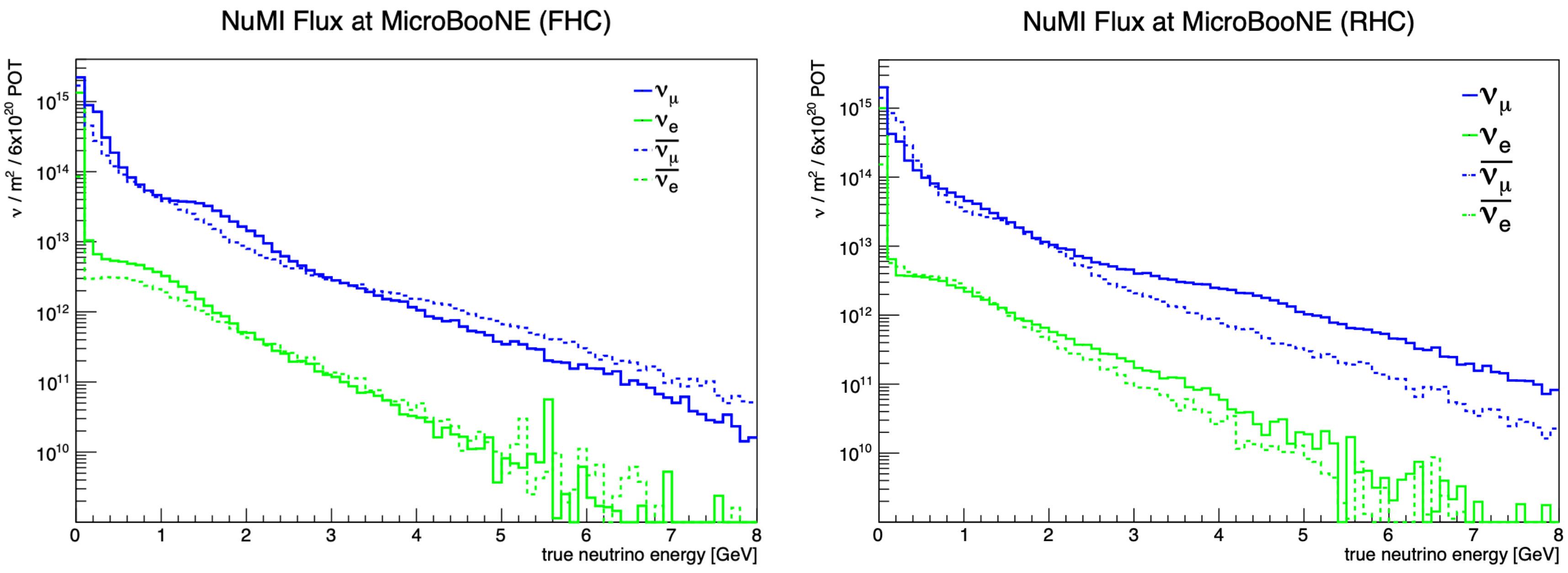


Generates multisims to evaluate uncertainties on the PPFX CV weight

source: L. Aliaga Soplin. Neutrino Flux Prediction for the NuMI Beamline. PhD thesis, William-Mary Coll., 2016.

PPFX in LArSoft – why?

- MicroBooNE has successfully implemented & validated PPFX within the LArSoft framework to produce CV flux predictions & uncertainties at MicroBooNE for NuMI cross section measurements



- Large scale production of NuMI analysis samples requires PPFX tools to be integrated into LArSoft
- There is interest to expand the utility of PPFX for NuMI analyses in ICARUS

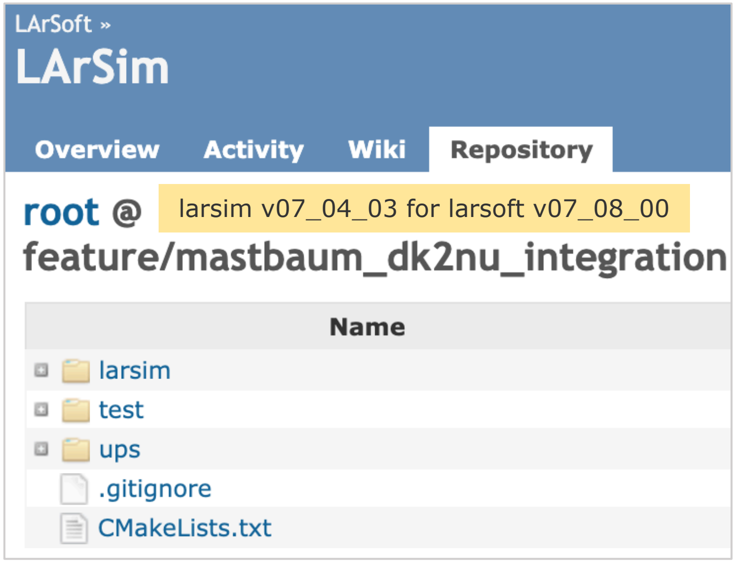
PPFX in LArSoft (larsim)

4 major pieces needed to interface PPFX with the LArSoft framework:

1. External **PPFX** package
2. Add **larsim/FluxReader** module
3. Add **larsim/EventWeight** PPFX calculators
4. Update **larsim/EventGenerator/GENIE/GENIEGen_module.cc** to store **dk2nu** information

A larsim feature branch with these changes is available here:
https://github.com/katrinamiller/larsim/tree/kmiller_dk2nu_integration

Versions & branches



root @ feature/mastbaum_dk2nu_integration

larsim v07_04_03 for larsoft v07_08_00

Name
larsim
test
ups
.gitignore
CMakeLists.txt

- An old feature branch was available with these changes, but was lost with the migration from **Redmine → GitHub**
- Goals:
  - **recreate** the feature branch with necessary additions ✅
  - **update** these additions for compatibility with a modern version of larsim ✅
  - **release** a version of LArSoft with PPFX capabilities for uboonecode & other experiments

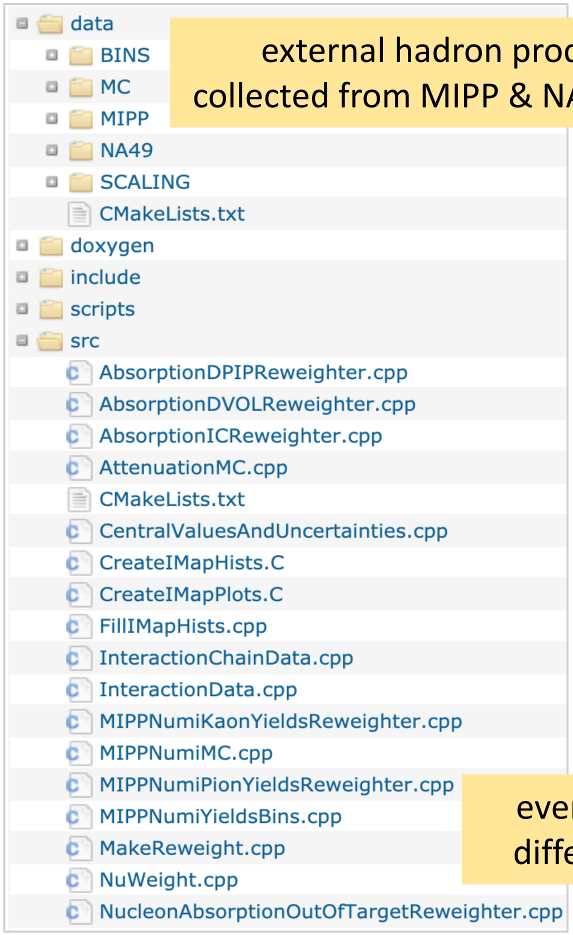
Results in this talk use:

- uboonecode v08_00_00_28
- larsim v08_02_00_07
- larsoft v08_05_00_10

PPFX in larsim

https://cdcvs.fnal.gov/redmine/projects/ppfx

1. **External PPFX package** – NuTools subproject (maintained by NoVA)



```
data
    BINS
    MC
    MIPP
    NA49
    SCALING
    CMakeLists.txt
doxygen
include
scripts
src
    AbsorptionDPIPReweighter.cpp
    AbsorptionDVOLReweighter.cpp
    AbsorptionICReweighter.cpp
    AttenuationMC.cpp
    CMakeLists.txt
    CentralValuesAndUncertainties.cpp
    CreateIMapHists.C
    CreateIMapPlots.C
    FillIMapHists.cpp
    InteractionChainData.cpp
    InteractionData.cpp
    MIPPNumiKaonYieldsReweighter.cpp
    MIPPNumiMC.cpp
    MIPPNumiPionYieldsReweighter.cpp
    MIPPNumiYieldsBins.cpp
    MakeReweight.cpp
    NuWeight.cpp
    NucleonAbsorptionOutOfTargetReweighter.cpp
```

external hadron production data
collected from MIPP & NA49 experiments

*Package to **P**redict the **FluX** (PPFX) implements the hadron production corrections & propagates uncertainties of the NuMI beam line. It is an experiment-independent **neutrino flux determination package for the NuMI beam that provides a correction for hadron production mis-modeling** using almost all relevant external data.*

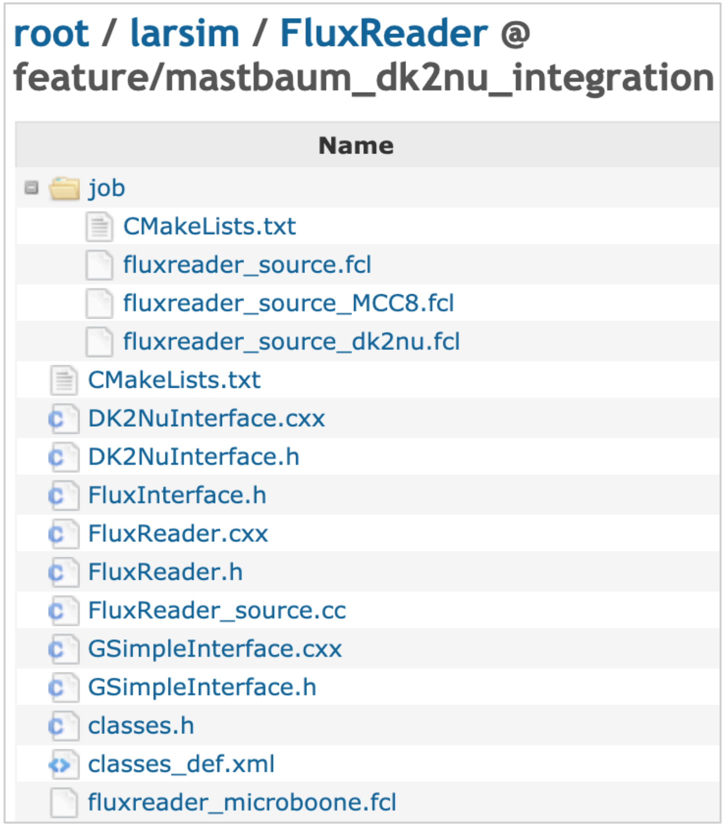
*The inputs are **dk2nu** and **dkmeta** objects for each neutrino event, and it **returns a set of correction values to be used as weights** to calculate the right neutrino yield.*

event-by-event reweighter classes for
different types of particle interactions

PPFX in larsim

https://cdcvs.fnal.gov/redmine/projects/fluxreader

1. External PPFX package
2. **Add FluxReader module** – developed internally by MicroBooNE



root / larsim / FluxReader @ feature/mastbaum_dk2nu_integration

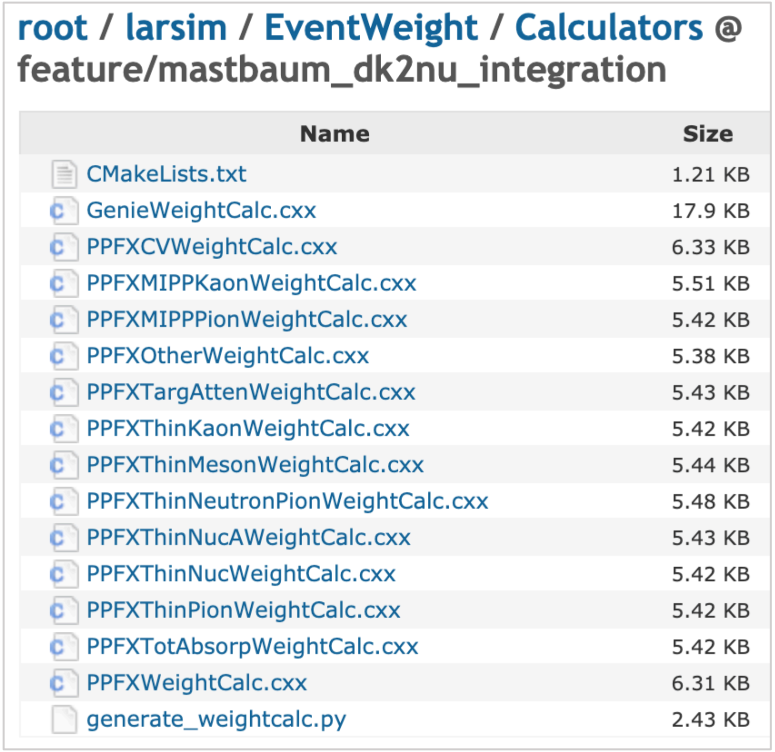
Name
job
CMakeLists.txt
fluxreader_source.fcl
fluxreader_source_MCC8.fcl
fluxreader_source_dk2nu.fcl
CMakeLists.txt
DK2NuInterface.cxx
DK2NuInterface.h
FluxInterface.h
FluxReader.cxx
FluxReader.h
FluxReader_source.cc
GSimpleInterface.cxx
GSimpleInterface.h
classes.h
classes_def.xml
fluxreader_microboone.fcl

- Quick & efficient way to convert **dk2nu** files into a large # of flux distributions at a given detector location
- Experiment-agnostic
- Output spectra stored in an art-root file is used as an input to PPFX reweighting

PPFX in larsim

1. External PPFX package
2. Add FluxReader module
3. **Add EventWeight PPFX calculators**

larsim v07_04_03:

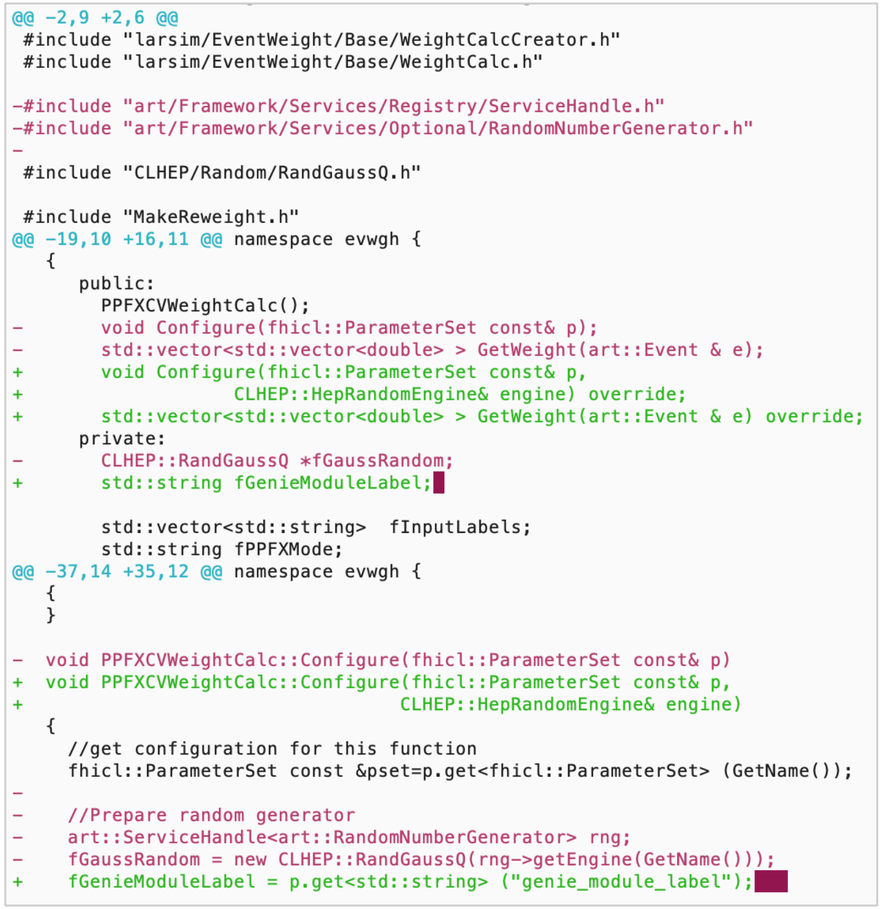


root / larsim / EventWeight / Calculators @ feature/mastbaum_dk2nu_integration

Name	Size
CMakeLists.txt	1.21 KB
GenieWeightCalc.cxx	17.9 KB
PPFXCVWeightCalc.cxx	6.33 KB
PPFXMIPPKaonWeightCalc.cxx	5.51 KB
PPFXMIPPPionWeightCalc.cxx	5.42 KB
PPFXOtherWeightCalc.cxx	5.38 KB
PPFXTargAttenWeightCalc.cxx	5.43 KB
PPFXThinKaonWeightCalc.cxx	5.42 KB
PPFXThinMesonWeightCalc.cxx	5.44 KB
PPFXThinNeutronPionWeightCalc.cxx	5.48 KB
PPFXThinNucAWeightCalc.cxx	5.43 KB
PPFXThinNucWeightCalc.cxx	5.42 KB
PPFXThinPionWeightCalc.cxx	5.42 KB
PPFXTotAbsorpWeightCalc.cxx	5.42 KB
PPFXWeightCalc.cxx	6.31 KB
generate_weightcalc.py	2.43 KB

PPFX in larsim

PPFXCVWeightCalc.cxx



```
@@ -2,9 +2,6 @@
 #include "larsim/EventWeight/Base/WeightCalcCreator.h"
 #include "larsim/EventWeight/Base/WeightCalc.h"

-#include "art/Framework/Services/Registry/ServiceHandle.h"
-#include "art/Framework/Services/Optional/RandomNumberGenerator.h"
-
 #include "CLHEP/Random/RandGaussQ.h"

 #include "MakeReweight.h"
@@ -19,10 +16,11 @@ namespace evwgh {
   {
      public:
        PPFXCVWeightCalc();
-       void Configure(fhicl::ParameterSet const& p);
-       std::vector<std::vector<double> > GetWeight(art::Event & e);
+       void Configure(fhicl::ParameterSet const& p,
+                   CLHEP::HepRandomEngine& engine) override;
+       std::vector<std::vector<double> > GetWeight(art::Event & e) override;
      private:
-       CLHEP::RandGaussQ *fGaussRandom;
+       std::string fGenieModuleLabel;

        std::vector<std::string>  fInputLabels;
        std::string fPPFXMode;
@@ -37,14 +35,12 @@ namespace evwgh {
   {
   }

-  void PPFXCVWeightCalc::Configure(fhicl::ParameterSet const& p)
+  void PPFXCVWeightCalc::Configure(fhicl::ParameterSet const& p,
+                                   CLHEP::HepRandomEngine& engine)
   {
    //get configuration for this function
    fhicl::ParameterSet const &pset=p.get<fhicl::ParameterSet> (GetName());
-
-   //Prepare random generator
-   art::ServiceHandle<art::RandomNumberGenerator> rng;
-   fGaussRandom = new CLHEP::RandGaussQ(rng->getEngine(GetName()));
+   fGenieModuleLabel = p.get<std::string> ("genie_module_label");
```

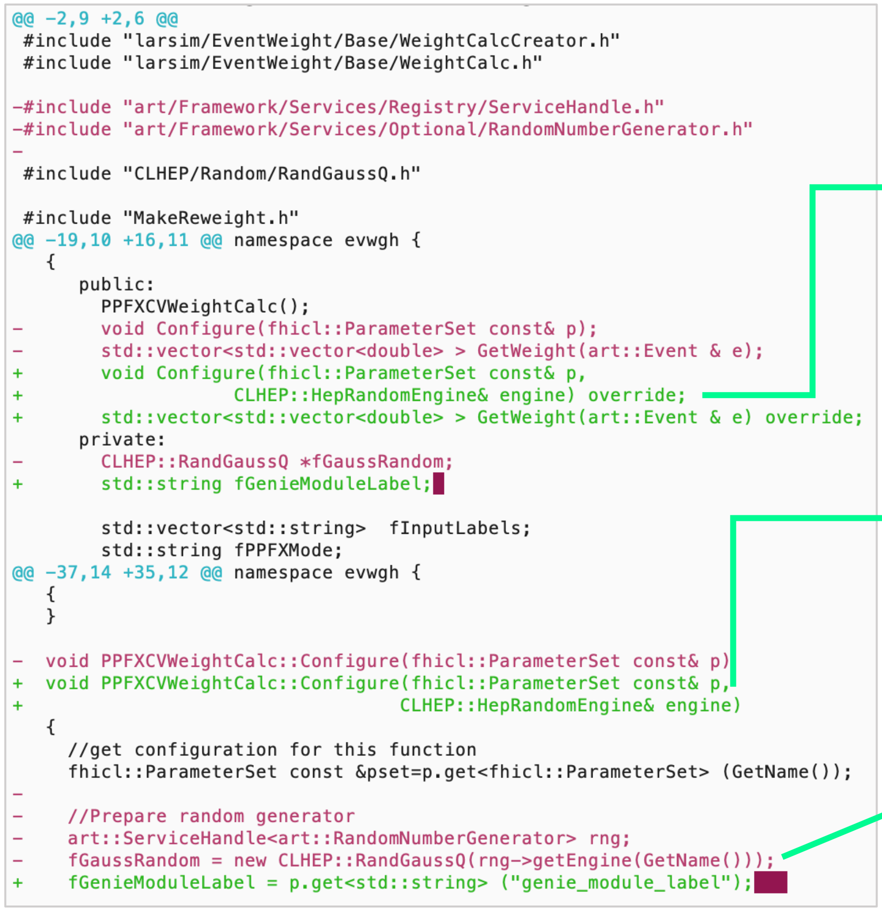
- PPFX Calculators were added, but as **larsim/EventWeight** migrated to art 3 after the original feature branch was created, the structure of these Calculators needed to be updated for art 3 compatibility

- used **GenieWeightCalc.cxx** as a template

- 13 Calculators updated individually

PPFX in larsim

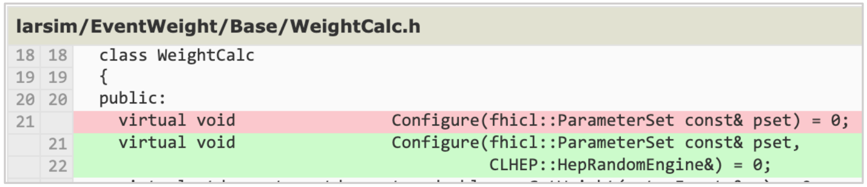
PPFXCVWeightCalc.cxx



```
@@ -2,9 +2,6 @@
 #include "larsim/EventWeight/Base/WeightCalcCreator.h"
 #include "larsim/EventWeight/Base/WeightCalc.h"

-#include "art/Framework/Services/Registry/ServiceHandle.h"
-#include "art/Framework/Services/Optional/RandomNumberGenerator.h"
-
 #include "CLHEP/Random/RandGaussQ.h"

 #include "MakeReweight.h"
@@ -19,10 +16,11 @@ namespace evwgh {
   {
      public:
        PPFXCVWeightCalc();
-       void Configure(fhicl::ParameterSet const& p);
-       std::vector<std::vector<double> > GetWeight(art::Event & e);
+       void Configure(fhicl::ParameterSet const& p,
+                      CLHEP::HepRandomEngine& engine) override;
+       std::vector<std::vector<double> > GetWeight(art::Event & e) override;
      private:
-       CLHEP::RandGaussQ *fGaussRandom;
+       std::string fGenieModuleLabel;

        std::vector<std::string>  fInputLabels;
        std::string fPPFXMode;
@@ -37,14 +35,12 @@ namespace evwgh {
   {
   }

-  void PPFXCVWeightCalc::Configure(fhicl::ParameterSet const& p)
+  void PPFXCVWeightCalc::Configure(fhicl::ParameterSet const& p,
+                                   CLHEP::HepRandomEngine& engine)
   {
     //get configuration for this function
     fhicl::ParameterSet const &pset=p.get<fhicl::ParameterSet> (GetName());
-
-    //Prepare random generator
-    art::ServiceHandle<art::RandomNumberGenerator> rng;
-    fGaussRandom = new CLHEP::RandGaussQ(rng->getEngine(GetName()));
+    fGenieModuleLabel = p.get<std::string> ("genie_module_label");
```

Updated the Calculator structure to interface with the art 3 EventWeight framework:

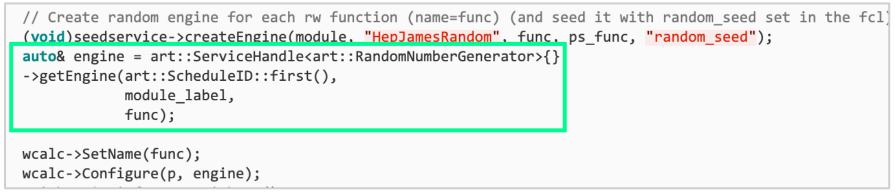
larsim/EventWeight/Base/WeightCalc.h:



larsim/EventWeight/Base/WeightCalc.h

```
   class WeightCalc
   {
   public:
-    virtual void                       Configure(fhicl::ParameterSet const& pset) = 0;
+    virtual void                       Configure(fhicl::ParameterSet const& pset,
+                                                 CLHEP::HepRandomEngine&) = 0;
```

larsim/EventWeight/Base/WeightManager.h:

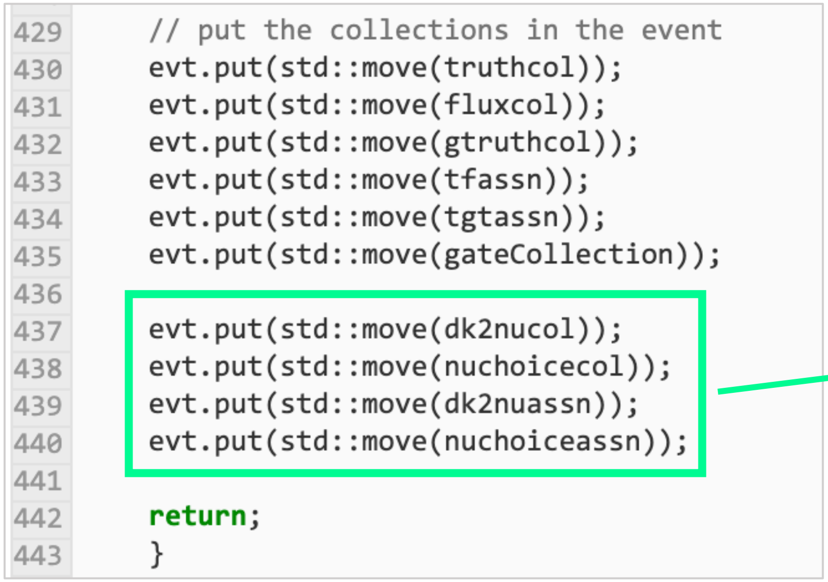


```
// Create random engine for each rw function (name=func) (and seed it with random_seed set in the fcl)
(void)seedservice->createEngine(module, "HepJamesRandom", func, ps_func, "random_seed");
auto& engine = art::ServiceHandle<art::RandomNumberGenerator>{}
->getEngine(art::ScheduleID::first(),
            module_label,
            func);

wcalc->SetName(func);
wcalc->Configure(p, engine);
```

PPFX in larsim

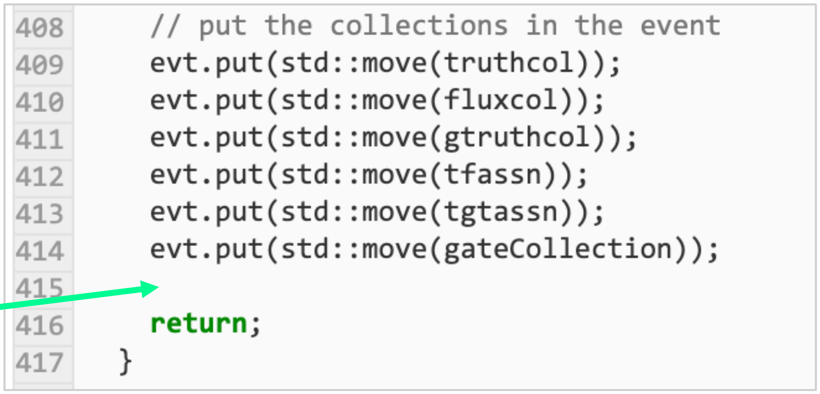
1. External PPFX package
2. Add FluxReader module
3. Add EventWeight PPFX calculators
4. **Update EventGenerator/GENIE/GENIEGen_module.cc to store dk2nu information**

larsim v07_04_03 (feature/mastbaum_dk2nu_integration):



```
    // put the collections in the event
    evt.put(std::move(truthcol));
    evt.put(std::move(fluxcol));
    evt.put(std::move(gtruthcol));
    evt.put(std::move(tfassn));
    evt.put(std::move(tgtassn));
    evt.put(std::move(gateCollection));

    evt.put(std::move(dk2nucol));
    evt.put(std::move(nuchoicecol));
    evt.put(std::move(dk2nuassn));
    evt.put(std::move(nuchoiceassn));

    return;
  }
```

larsim v08_02_00_07 (develop):

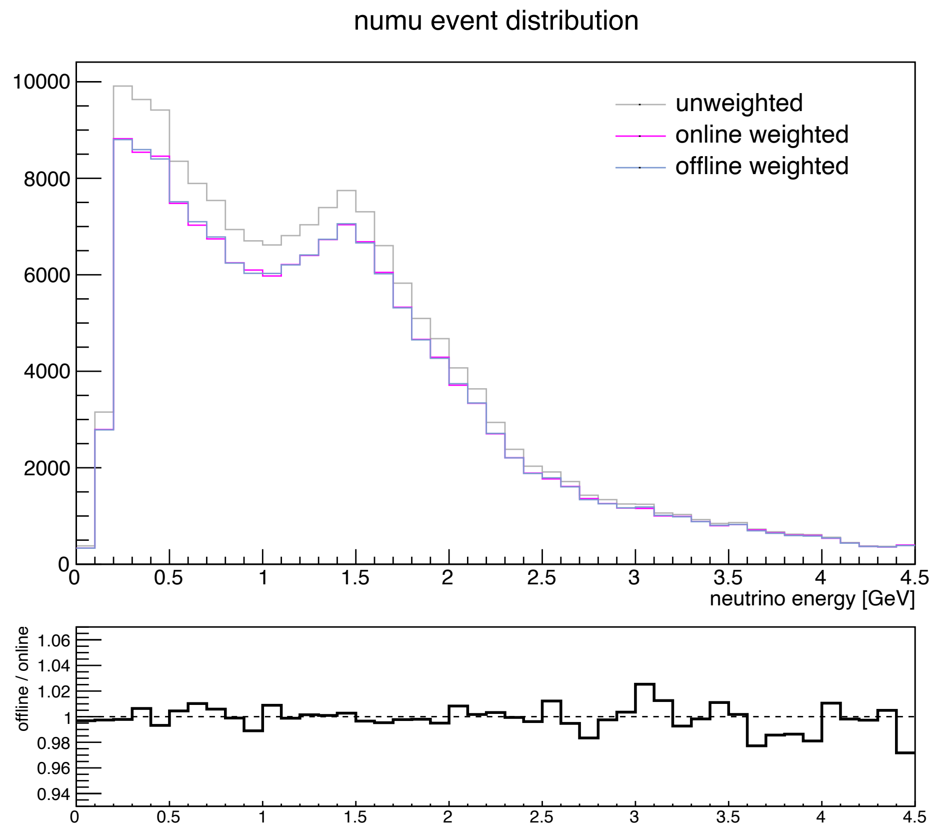


```
    // put the collections in the event
    evt.put(std::move(truthcol));
    evt.put(std::move(fluxcol));
    evt.put(std::move(gtruthcol));
    evt.put(std::move(tfassn));
    evt.put(std::move(tgtassn));
    evt.put(std::move(gateCollection));

    return;
  }
```

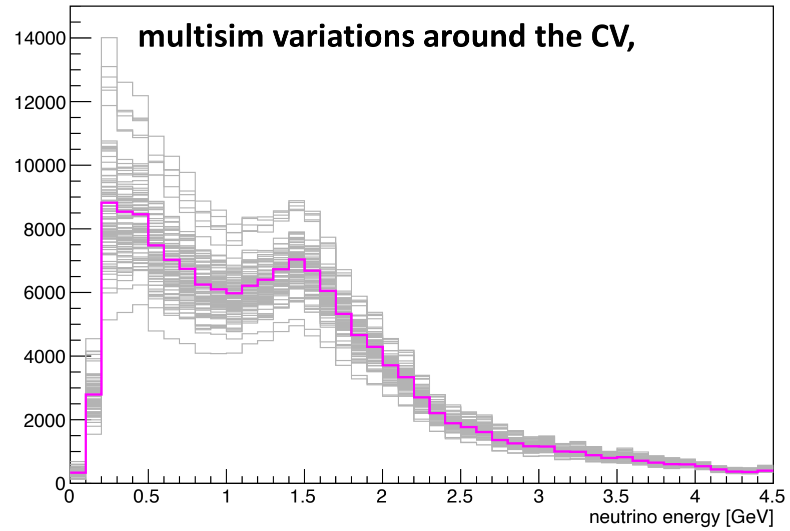
Validating PPFX integration

- i.e., does the code produce reasonable weights & uncertainties?

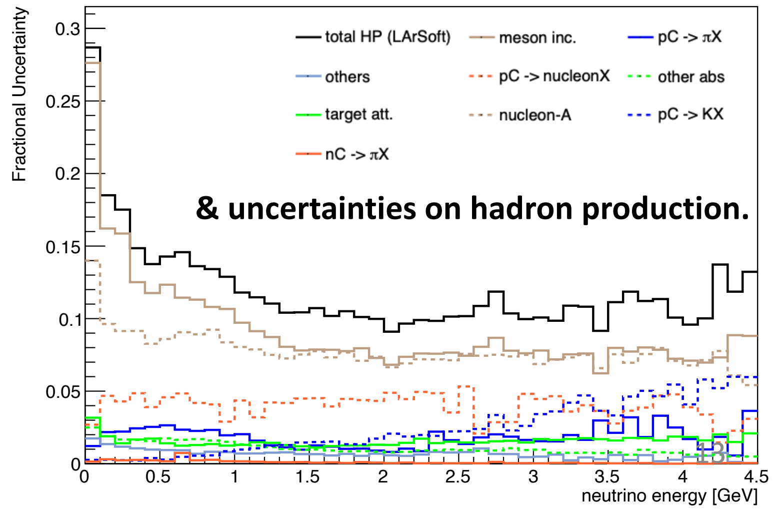
Test sample shows that these updates successfully produce CV weights for the event rate distribution,



multisim variations around the CV,



& uncertainties on hadron production.



Summary

Purpose of the changes:

- Integrating PPFX tools into larsim is necessary for NuMI production & cross section measurements in MicroBooNE
- These tools will also benefit future ICARUS analyses

What is being introduced:

- External PPFX package
- larsim/FluxReader module
- PPFX Calculators in larsim/EventWeight
- Minor updates to GENIEGen_module.cc

(See: https://github.com/katrinamiller/larsim/tree/kmiller_dk2nu_integration)

Impact to existing code:

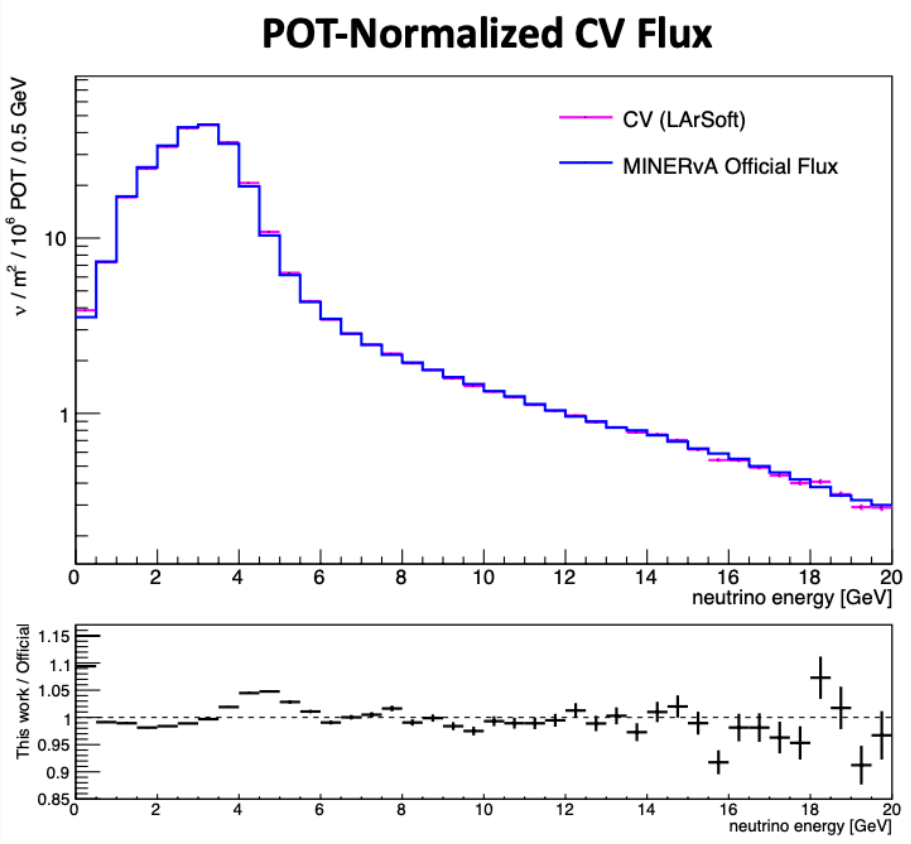
- The proposed changes do not impact other code

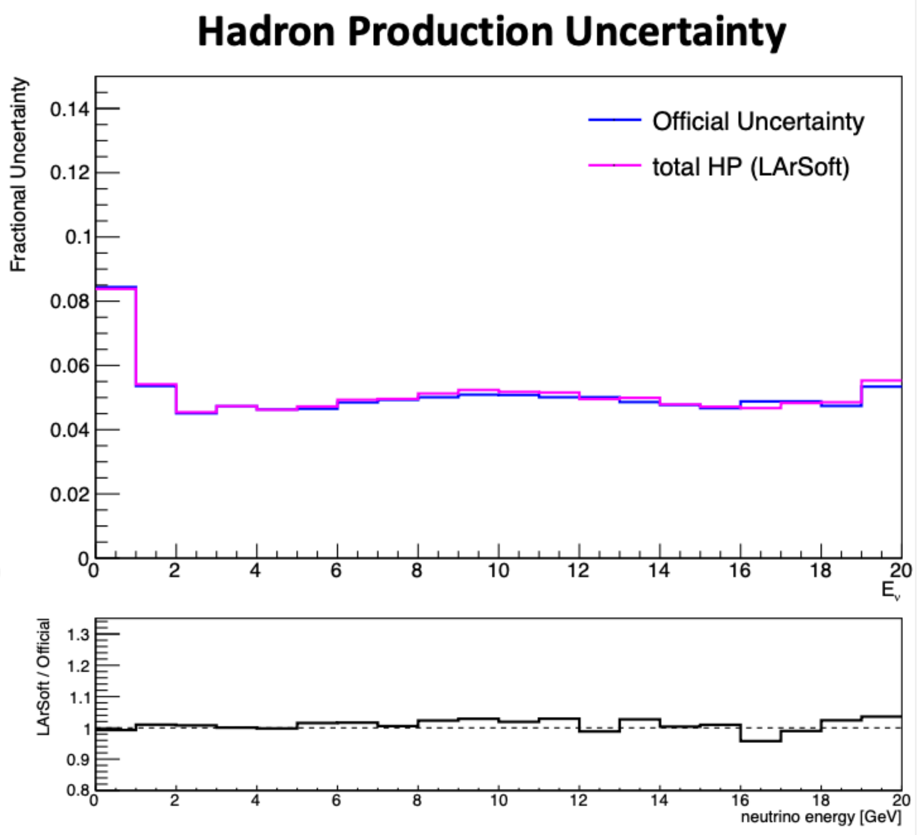
Planned work:

- Update PPFX ups product with the official version in NuTools (maintained by NoVA)

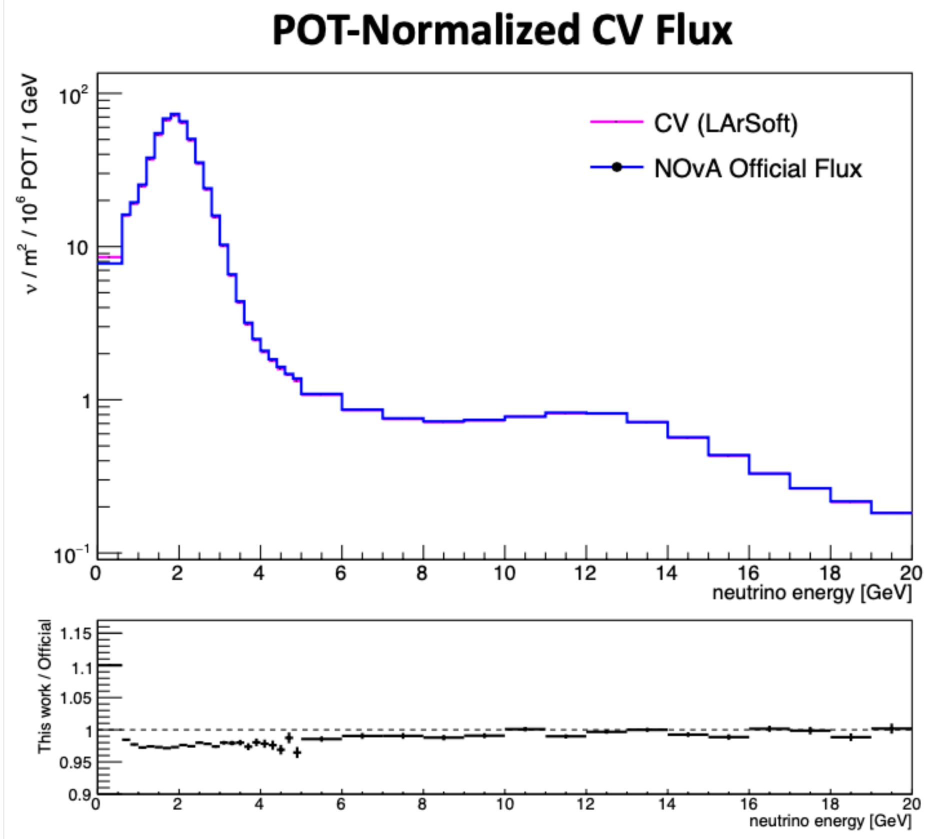
Extras

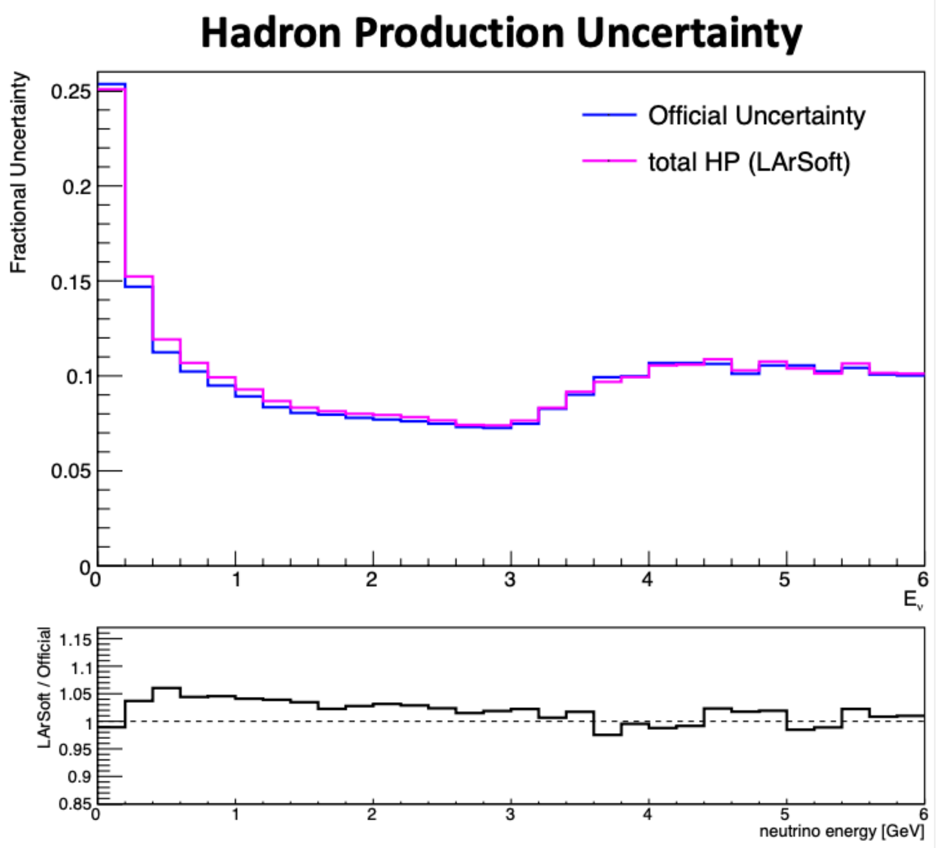
On-Axis PPFX Validation with MINERvA





Off-Axis PPFX Validation with NoVA





PPFX Reweighters

Reweighter	**Description**
MIPP NuMI π	pion production from NuMI thick target
MIPP NuMI K	kaon production from NuMI thick target
$pC \rightarrow \pi X$	pion production in proton-carbon interactions
$pC \rightarrow KX$	kaon production in proton-carbon interactions
$pC \rightarrow nucleonX$	nucleon production from proton-carbon interactions
$\mathrm{nC} \rightarrow \pi X$	pion production from neutron-carbon interactions
$nucleonA$	nucleon interactions on materials along the beamline
$meson\ inc.$	meson interactions on materials along the beamline
$others$	other interactions for which no external data is available
$targ\ att.$	attenuation of particles passing through target
$absorption$	attenuation of particles passing through materials on beamline