Tutorial 5

Systematics & Event Re-weighting

NuSTEC Generator School

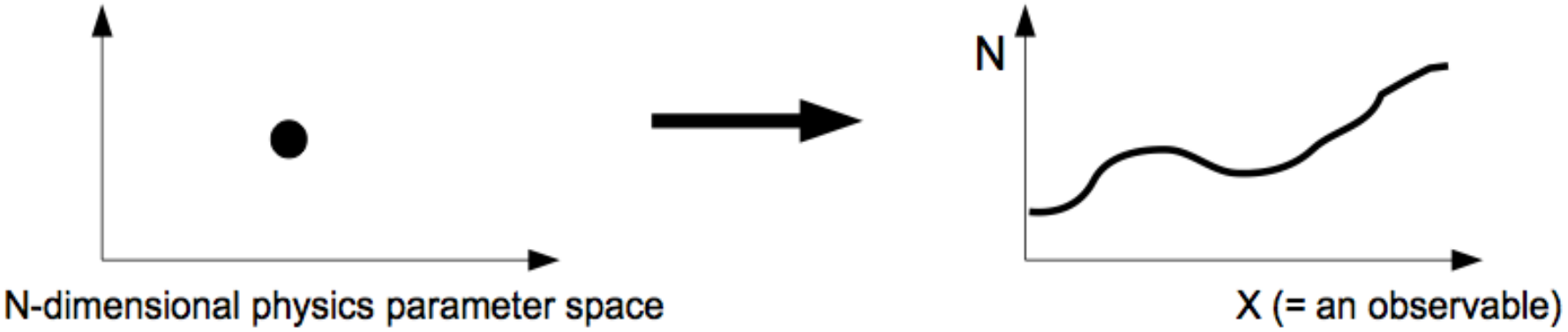
Liverpool, May 14-16, 2014

Costas Andreopoulos

University of Liverpool & STFC Rutherford Appleton Laboratory

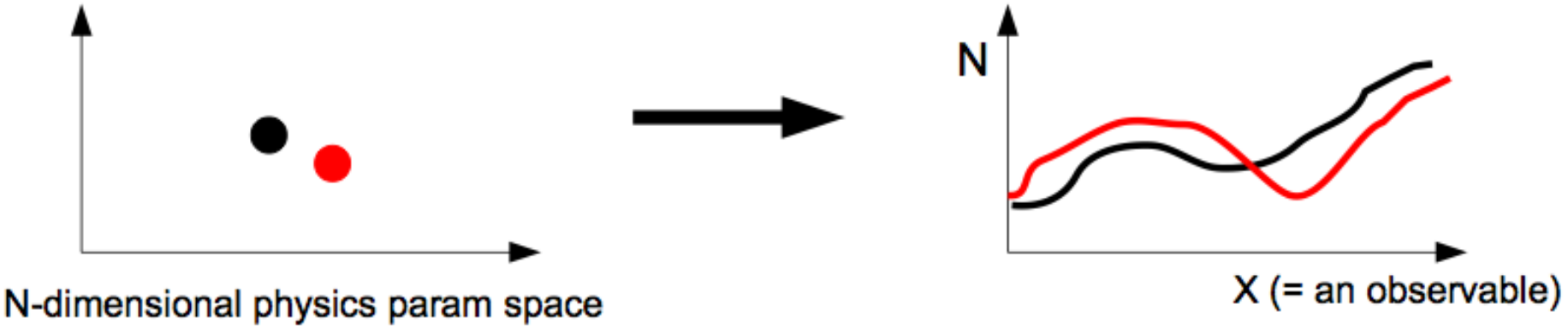
What we have discussed so far in the tutorials allows you to connect a vast amount of inputs (physics models, assumptions, external data and model parameter tunes) to distributions of observable quantities

(e.g. muon momentum - angle distributions for ν_μCC events, pion momentum distributions for ν_μCC1π^+ events, etc).



However, our inputs, for a specif c choice of models, have uncertainties. And, of course, there are many different model choices.

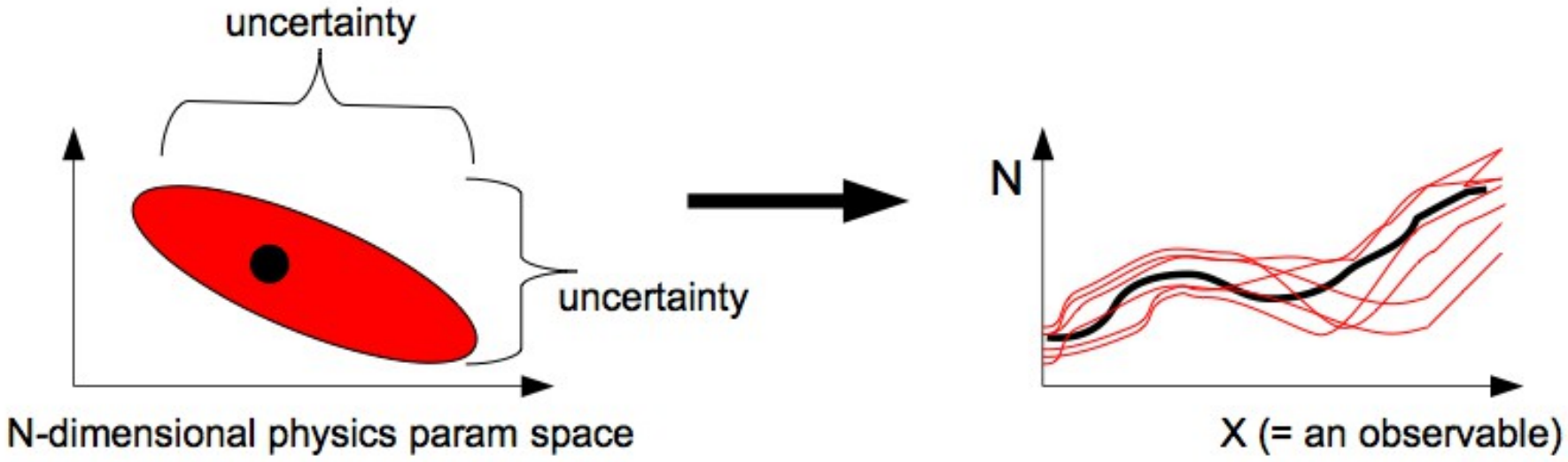
Experiments need to investigate the effect that simulation choices have on physics observables (model uncertainties oscillation systematics).



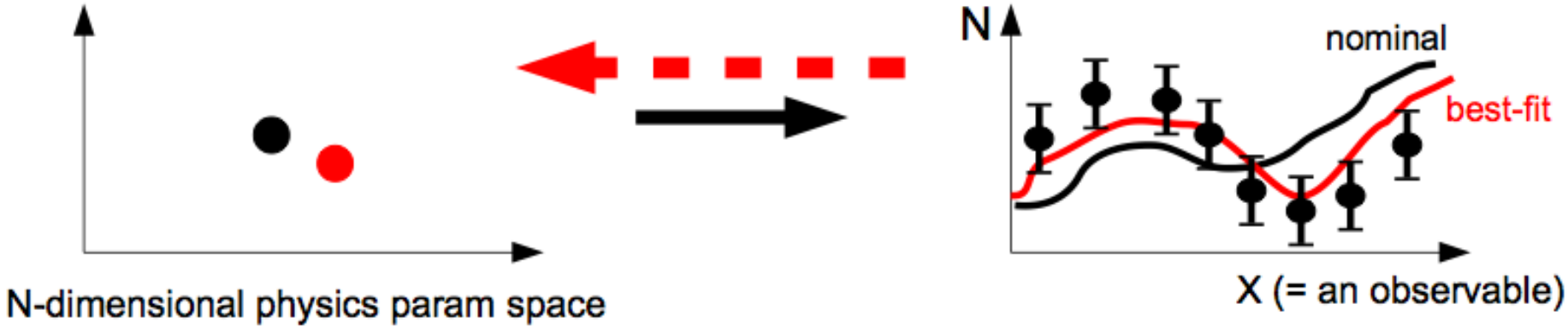
But can not run the full experiment simulation multiple times

T2K/ND280: Generating and reconstructing a sample corresponding to 5×10^{21} POT, takes 600 CPU × weeks (circa 2010).

We need a faster procedure to ”emulate” the simulation output, under different input physics assumptions, without rerunning the full simulation.



If this procedure we could inverted, we could also use it to fit observable distributions from our experiments, to improve the level of data/MC agreement, constrain uncertainties and obtain the correlations imposed by our experimental constraint



Event reweighting

Use one sample to emulate another...

Can be used to propagate uncertainties to analyses and bug-fix precious large statistics event samples

Event reweighting

GENIE provides a simple reweighting interface (**GReWeight**).
You can add several weight calculators as appropriate for your application.

```
GReWeight rw;

rw.AdoptWghtCalc("xsec_ccqe",         new GReWeightNuXSecCCQE       );
rw.AdoptWghtCalc("xsec_ccqe_vec",     new GReWeightNuXSecCCQEvec    );
rw.AdoptWghtCalc("xsec_ccres",        new GReWeightNuXSecCCRES      );
rw.AdoptWghtCalc("xsec_ncres",        new GReWeightNuXSecNCRES      );
rw.AdoptWghtCalc("xsec_nonresbkg",    new GReWeightNonResonanceBkg  );
rw.AdoptWghtCalc("xsec_dis",          new GReWeightNuXSecDIS        );
rw.AdoptWghtCalc("xsec_coh",          new GreWeightNuXSecCOH        );
rw.AdoptWghtCalc("nuclear_qe",        new GreWeightNuXSecFGM        );
rw.AdoptWghtCalc("nuclear_dis",       new GreWeightDISNuclMod       );
rw.AdoptWghtCalc("hadro_res_decay",   new GReWeightResonanceDecay   );
rw.AdoptWghtCalc("hadro_fzone",       new GReWeightFZone            );
rw.AdoptWghtCalc("hadro_intranuke",   new GReWeightINuke            );
rw.AdoptWghtCalc("hadro_agky",        new GReWeightAGKY             );

... more, as appropriate
```

Event reweighting

Each weight calculator was added with a name.

At any point, if needed, you can retrieve any weight calculator and fine-tune it calling methods specific to each calculator.

Example:

```
GReWeightNuXSecCCQE * rwccqe =
        dynamic_cast<GReWeightNuXSecCCQE *> (rw.WghtCalc("xsec_ccqe"));

rwccqe -> RewNue       (false);
rwccqe -> RewNuebar    (false);
rwccqe -> RewNumubar(false);
```

Event reweighting

A tweak factor **x** modifies a physics param **P** as: $P \rightarrow P' = P0 * (1 + x * dP/P)$

Various x's (systematic parameters) allowed (defined in $GENIE/src/ReWeight/GSyst.h)
You can modify them and propagate changes to GENIE.

```
GSystSet & syst = rw.Systematics();

syst.Set (kXSecTwkDial_NormCCQE,          +1.0);
syst.Set (kXSecTwkDial_MaCCQEshape,       -1.0);
syst.Set (kXSecTwkDial_NormCCRES,         +0.0);
syst.Set (kXSecTwkDial_MaCCRESshape,      -0.4);
syst.Set (kXSecTwkDial_MvCCRESshape,      +0.9);
syst.Set (kXSecTwkDial_NormNCRES,         -1.0);
syst.Set (kXSecTwkDial_MaNCRESshape,      +0.3);
syst.Set (kXSecTwkDial_MvNCRESshape,      +0.5);
syst.Set (kXSecTwkDial_RvpCC1pi,          -1.0);
syst.Set (kXSecTwkDial_RvnCC1pi,          +0.3);
syst.Set (kXSecTwkDial_MaCOHpi,           +0.7);
syst.Set (kINukeTwkDial_MFP_pi,           +0.4);
syst.Set (kINukeTwkDial_MFP_N,            -1.0);
syst.Set (kINukeTwkDial_FrPiProd_pi,      -0.7);
syst.Set (kHadrNuclTwkDial_FormZone,      +0.2);
syst.Set (kRDcyTwkDial_Theta_Delta2Npi,   +0.3);

... more, as appropriate

rw.Reconfigure();
```

Are you from T2K, and all this looks very familiar? We wrote the T2K reweighting framework too

Event reweighting

A tweak factor **x** modifies a physics param **P** as: $P \rightarrow P' = P0 * (1 + x * dP/P)$

Common question:
Where can I get the value of P0 and P'

Answer:
Depends on what P is, but **you do not need P**.
Formulate your problem entirely in terms of x, not P.
E.g let x's be the parameters you are constraining in your near detector fit, or the parameters you profile out in the far detector fit.

- In some cases, P is indeed a parameter (eg Ma)
- But, in other cases, it may be a function (eg pion-nucleon cross-section as function of the pion energy)
- And it in other cases that nominal physics parameter P might be the outcome of an MC procedure and not known analytically.

Event reweighting

Once tweaked parameters are propagated to GENIE,
you can calculate event weights invoking:
GReWeight::CalcWeight().

The return value is the calculated weight and is computed as the product of the weights computed by all included weight calculators for the current set of systematics stored in **GSystSet**.

Note: The function expects an **event record (GHEP format) as input.**
If you built your analysis using simpler formats, for convenience, you may not be able to reweight, e.g.
- you may not have kept enough information, or
- it may be ineff cient to translate between formats in order to reweight

You can also calculate a penalty factor, for the current set of systematics by invoking **GReWeight::CalcChisq().**

Event reweighting apps

You will need to embed the reweighing functionality into your own GENIE-based analysis code

There is a single generally useful reweighing application included in GENIE: the **grwght1scan** utility:

- source in $GENIE/src/support/rwght/gRwght1Scan.cxx
- built if you specify -enable-rwght at configuration

The application:

- Reads-in an input event file (GHEP format)
- Reads-in a systematic parameter, a range (eg -5, 5 for -5σ to +5σ scan) and the number of points to consider in that range
- Outputs a ROOT file containing a tree with event weights.
  - The tree has entry for every input event.
  - Each such tree entry contains a TArrayF of all computed weights and a TArrayF of all used tweak dial values.

The application can be easily extended to vary multiple parameters simultaneously.

Event reweighting apps

Common question:
So, now I have a tree with all the information about the event (GHEP event tree) or a tree with summary information about the event (GST format or other) **and** other tree(s) with event weights. How do I combine them?

Use the ROOT friend tree mechanism:

```
TFile * event_file =
  new TFile("./path/to/your/event_file.root", "read");
TTree * event_tree =
   (TTree *) event_file->Get("your_tree");

TFile * weight_file =
  new TFile ("./path/to/weight_file.root","read");
TTree * weight_tree =
 (TTree *) weight_file->Get("name_depends_on_systematic");
TArrayF *twkdials = 0;
TArrayF *weights  = 0;
weight_tree->SetBranchAddress("twkdials", &twkdials);
weight_tree->SetBranchAddress("weights",  &weights);

event_tree->AddFriend(weight_tree);
```

Which physics parameters are currently reweighted?

Event reweighting (cross-sections)

x_P	Description of P	$\delta P/P$
$x_{M_A^{NCEL}}$	Axial mass for NC elastic	±25%
$x_{\eta^{NCEL}}$	Strange axial form factor η for NC elastic	±30%
$x_{M_A^{CCQE}}$	Axial mass for CC quasi-elastic	-15% +25%
$x_{CCQE-Norm}$	Normalization factor for CCQE	
$x_{CCQE-PauliSup}$	CCQE Pauli suppression (via changes in Fermi level k_F)	±35%
$x_{CCQE-VecFF}$	Choice of CCQE vector form factors (BBA05 ↔ Dipole)	-
$x_{CCRES-Norm}$	Normalization factor for CC resonance neutrino production	
$x_{NCRES-Norm}$	Normalization factor for NC resonance neutrino production	
$x_{M_A^{CCRES}}$	Axial mass for CC resonance neutrino production	±20%
$x_{M_V^{CCRES}}$	Vector mass for CC resonance neutrino production	±10%
$x_{M_A^{NCRES}}$	Axial mass for NC resonance neutrino production	±20%
$x_{M_V^{NCRES}}$	Vector mass for NC resonance neutrino production	±10%
$x_{M_A^{COHpi}}$	Axial mass for CC and NC coherent pion production	±50%

Event reweighting (cross-sections)

$x_{R_0^{COHpi}}$	Nuclear size param. controlling π absorption in RS model	$\pm 10\%$
$x_{R_{bkg}^{\nu p,CC1\pi}}$	Non-resonance bkg in νp $CC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu p,CC2\pi}}$	Non-resonance bkg in νp $CC2\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,CC1\pi}}$	Non-resonance bkg in νn $CC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,CC2\pi}}$	Non-resonance bkg in νn $CC2\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu p,NC1\pi}}$	Non-resonance bkg in νp $NC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu p,NC2\pi}}$	Non-resonance bkg in νp $NC2\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,NC1\pi}}$	Non-resonance bkg in νn $NC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,NC2\pi}}$	Non-resonance bkg in νn $NC2\pi$ reactions	$\pm 50\%$
$x_{A_{HT}^{BY}}$	A_{HT} higher-twist param in BY model scaling variable ξ_w	$\pm 25\%$
$x_{B_{HT}^{BY}}$	B_{HT} higher-twist param in BY model scaling variable ξ_w	$\pm 25\%$
$x_{C_{V1u}^{BY}}$	C_{V1u} u valence GRV98 PDF correction param in BY model	$\pm 30\%$
$x_{C_{V2u}^{BY}}$	C_{V2u} u valence GRV98 PDF correction param in BY model	$\pm 40\%$

Event reweighting (rescattering)

x_P	Description of P	$\delta P/P$
x^N_{mfp}	Nucleon mean free path (total rescattering probability)	$\pm 20\%$
x^N_{cex}	Nucleon charge exchange probability	$\pm 50\%$
x^N_{el}	Nucleon elastic reaction probability	$\pm 30\%$
x^N_{inel}	Nucleon inelastic reaction probability	$\pm 40\%$
x^N_{abs}	Nucleon absorption probability	$\pm 20\%$
x^N_{π}	Nucleon π-production probability	$\pm 20\%$
x^{π}_{mfp}	π mean free path (total rescattering probability)	$\pm 20\%$
x^{π}_{cex}	π charge exchange probability	$\pm 50\%$
x^{π}_{el}	π elastic reaction probability	$\pm 10\%$
x^{π}_{inel}	π inelastic reaction probability	$\pm 40\%$
x^{π}_{abs}	π absorption probability	$\pm 20\%$
x^{π}_{π}	π π-production probability	$\pm 20\%$

Event reweighting (hadronization & decays)

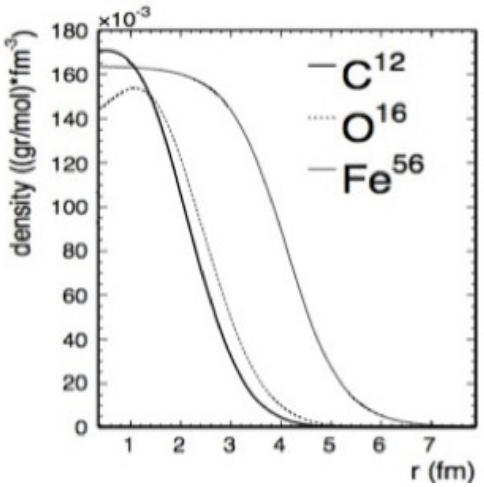
x_P	Description of P	$\delta P/P$
$x_{AGKY}^{pT1\pi}$	Pion transverse momentum (p_T) for $N\pi$ states in AGKY	-
$x_{AGKY}^{xF1\pi}$	Pion Feynman x (x_F) for $N\pi$ states in AGKY	-
x_{fz}	Hadron formation zone	$\pm 50\%$
$x_{\theta_\pi}^{\Delta\to\pi N}$	Pion angular distribution in $\Delta \to \pi N$ (isotropic $\leftrightarrow$ RS)	-
$x_{BR}^{R\to X+1\gamma}$	Branching ratio for radiative resonance decays	$\pm 50\%$
$x_{BR}^{R\to X+1\eta}$	Branching ratio for single-η resonance decays	$\pm 50\%$

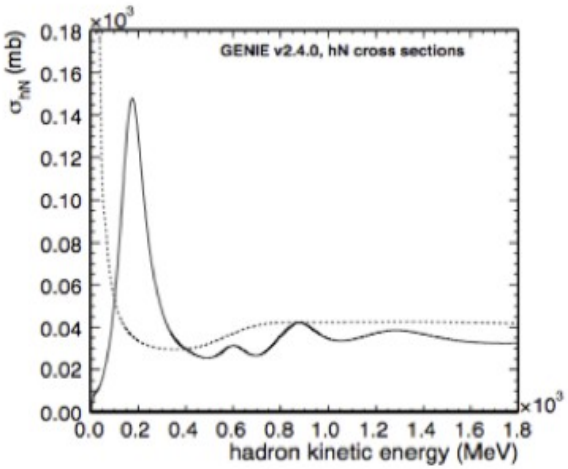
Event reweighting method – Examples

Mean free path for intranuclear rescattering

$$P^h_{rescat} = 1 - \int dr e^{-r/\lambda^h(\vec{r},E_h)}$$

$$\lambda^h(\vec{r},E_h) = 1/(\rho_{nucl}(r)\cdot\sigma^{hN}(E_h))$$





The reweighting re-traces the simulation steps

- Using hadron steps of 0.05 fm as in simulation
- Hadrons traced till they reach a distance of r = N R_{nucl} = N R_0 $A^{1/3}$
  - N = 3
  - e.g Fe^{56} : R_{max} = ~16 fm
- *… other details*
- Tweak mean free path

$$\lambda^h \rightarrow \lambda^{h\prime} = \lambda^h(1 + x^h_{mfp} * \delta\lambda^h/\lambda^h)$$

- Re-evaluate re-scattering probabilities

Event reweighting method – Examples

Mean free path for intranuclear rescattering

Say, that mean free path **increases**:

- the re-scattering probability **decreases**
  - A hadron was re-scattered in the original simulation
    - This is now less likely: weight < 1
  - A hadron was not re-scattered in the original simulation:
    - This is now more likely: weight > 1

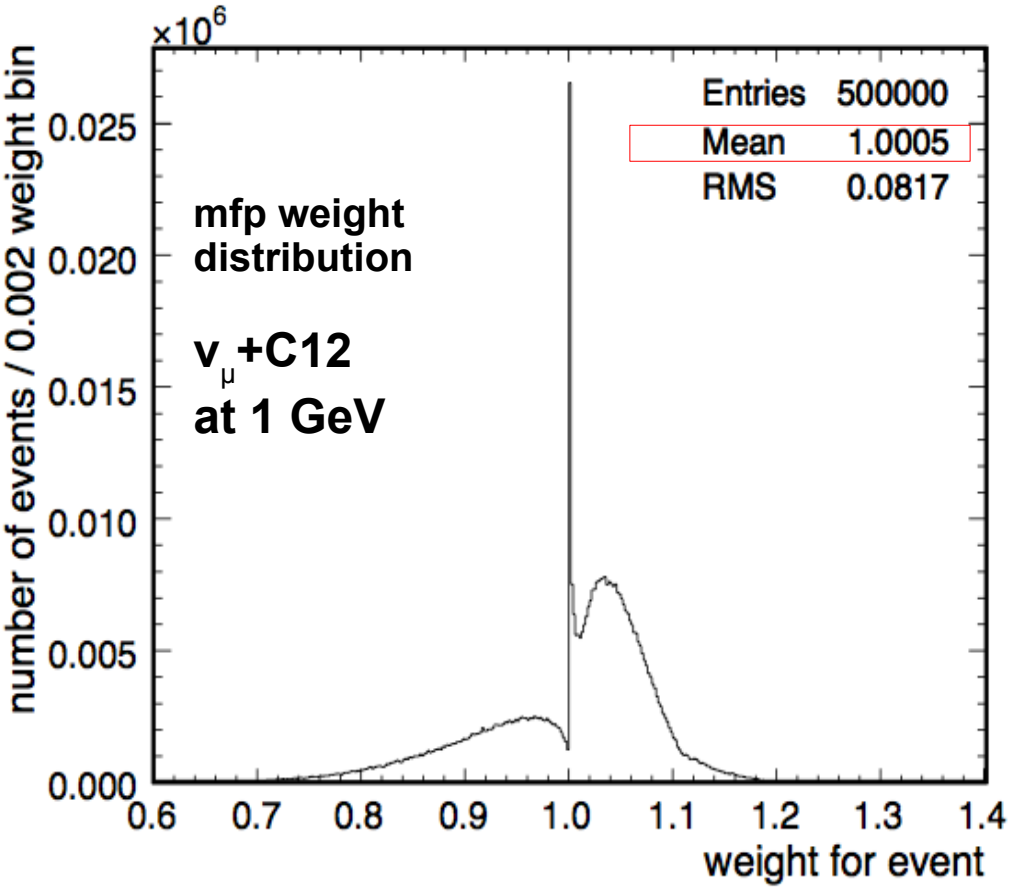
$$w^h_{mfp} = \begin{cases} \frac{1-P^{h\prime}_{surv}}{1-P^h_{surv}} & \text{if } h\text{re-interacts} \\ \frac{P^{h\prime}_{surv}}{P^h_{surv}} & \text{if } h\text{escapes} \end{cases}$$

The **event weight** is calculated as the product of **particle weights** (shown on the left)

Event reweighting method – Examples

Mean free path for intranuclear rescattering

We saw previously that, for every mfp tweak, some events will be weighted up while other events will be weighted down.



Tweaking the hadronic system should have no effect whatsoever if, for example, you are only looking at the leptonic system using a fully inclusive *(at MC truth level, before experimental cuts that bring in selection inefficiencies)* sample.

So the mean free path weights should have special properties **by construction**.

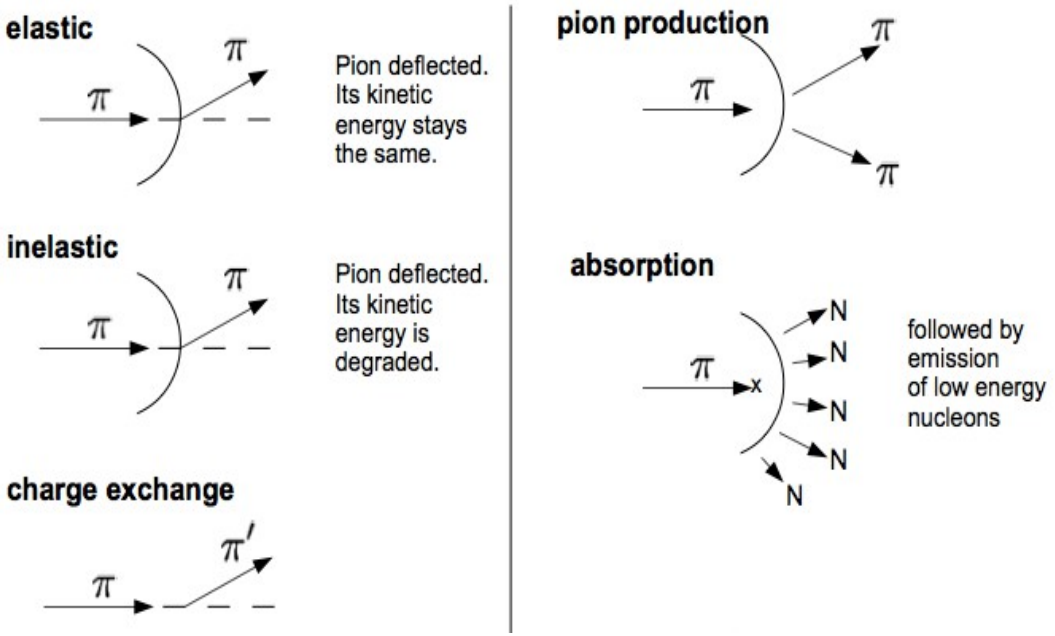
The average weight over an event sample should be 1 (i.e. the sum of weights, which basically is the number of events, should be constant.
We say that weights "**maintain unitarity**."

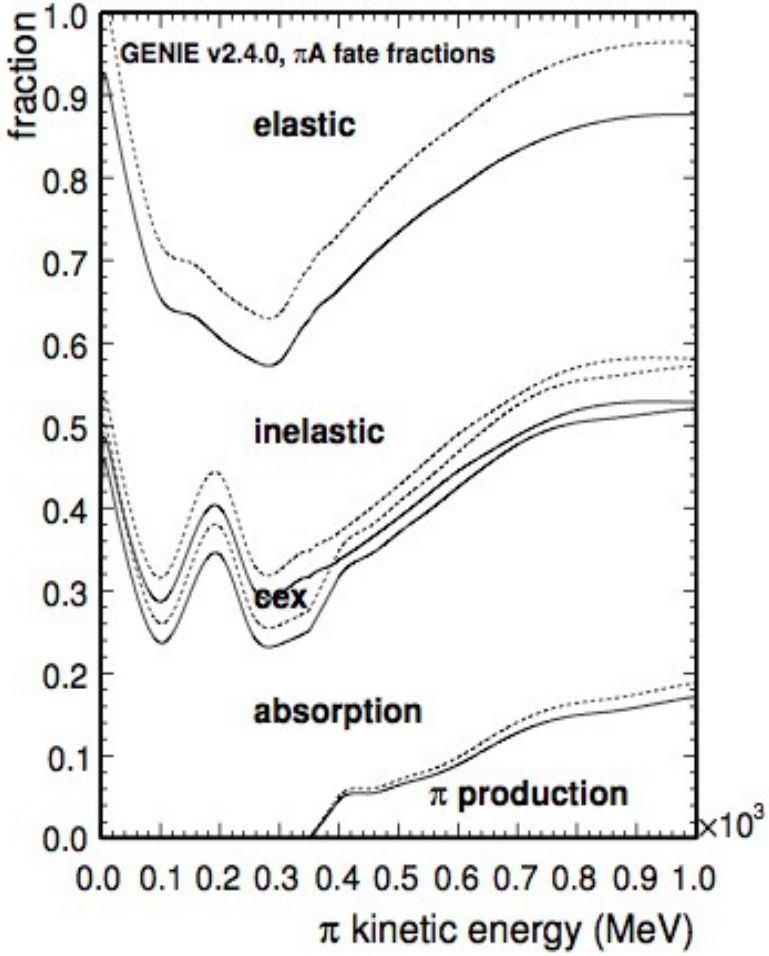
Be extra careful with forming weights using ratios of tweaked and nominal quantities, especially if all that you really have is conditional probabilities.
These are not proper weights.

Think about the properties your weights should have by construction and verify that, indeed, they have those properties.

Event reweighting method – Examples

Pion rescattering "fates" for intranuclear rescattering

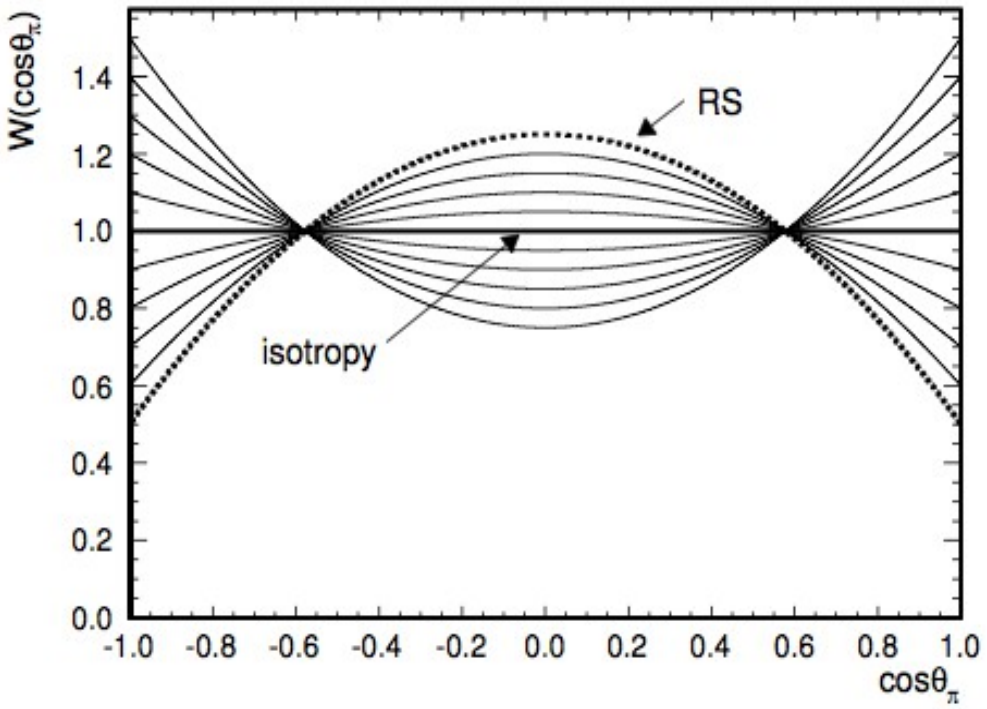




- Tweaking the default "fate" fractions shown on the left. A weight is calculates as the ratio of new/old fraction.
- The **event weight** is calculated as the product of **particle weights**
- Unitarity considerations again: Fractions have to always add-up to 1. Can not tweak all "fates" simultaneously.
    - Various schemes employed: Eg choosing "cushion".

Event reweighting method – Examples

Angular distribution of pions in resonance decays



example: $\Delta^{++}(1232) \rightarrow p + \pi^+$

Angular distribution for π^+, given 2 possible sub-states m_i=1/2, 3/2:

$$W(\theta) = 1 - p(3/2) * P_2(\cos\theta) + p(1/2) * P_2(\cos\theta)$$

m_i is the projection of Δ^{++} angular momentum along quantization axis and $P_2(\cos\theta)$ is the 2nd order Legendre polynomial.

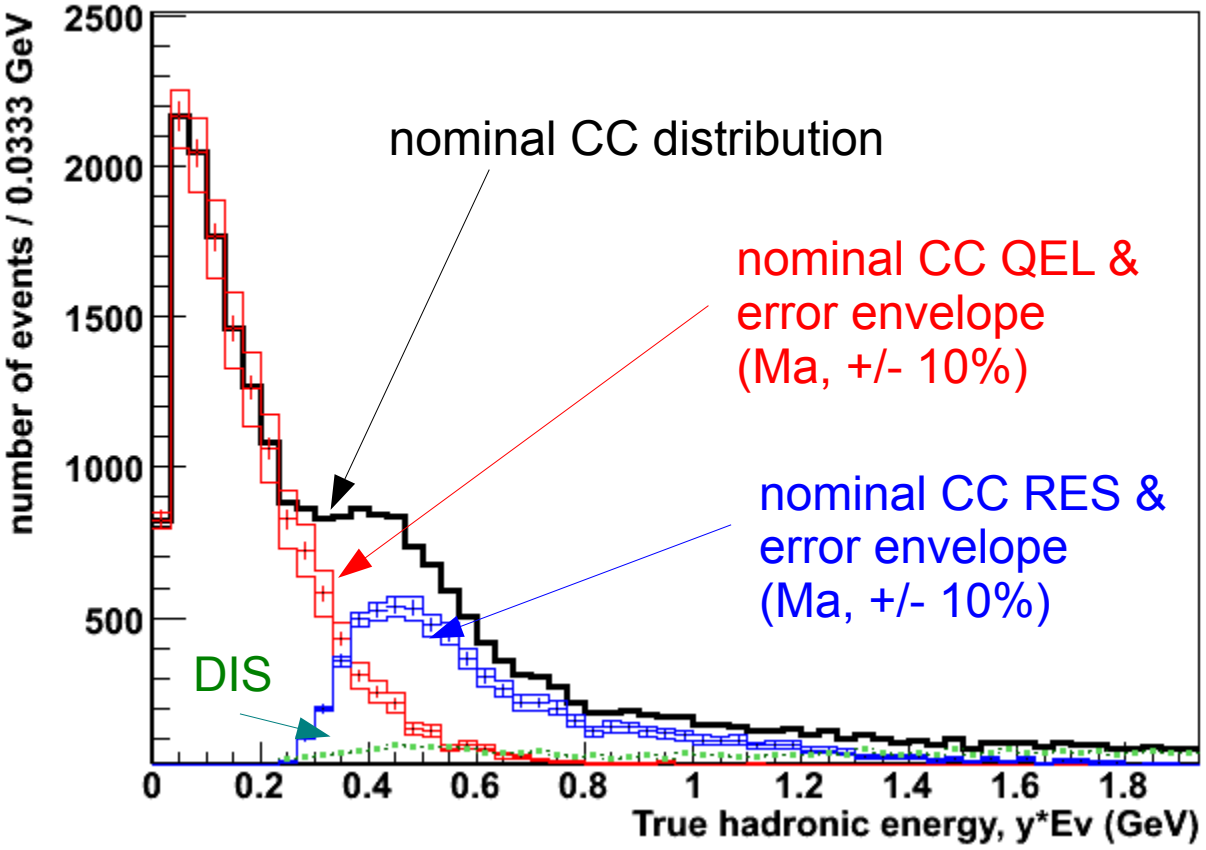
- Isotropy implies equal populations: p(3/2)=p(1/2)=0.5 (currently used in simulation)
- R/S actually predicts: p(3/2)=0.75, p(1/2)=0.25

Can trivially reweight from isotropic → RS (and consider intermediate configurations)

Example reweighing example

GENIE numu+O16 sample with T2K flux

Tweaked Ma-QEL and Ma-RES by +/- 10%



Adding a new reweighting method

bool IsHandled(genie::GSyst_t syst)
Declare whether the weight calculator handles the input systematic parameter.

void SetSystematic(genie::GSyst_t syst, double val)
Update the current value for the specified systematic parameter.

void Reset(void)
Set all handled systematic parameters to default values.

void Reconfigure(void)
Propagate updated systematic parameter values to actual GENIE MC code, if needed.

double CalcWeight(const genie::EventRecord & event)
Calculate a weight for the input event using the current values of all handled systematic params.

double CalcChisq(void)
Calculate a penalty factor for the current deviation of all handled systematic params from their default values.

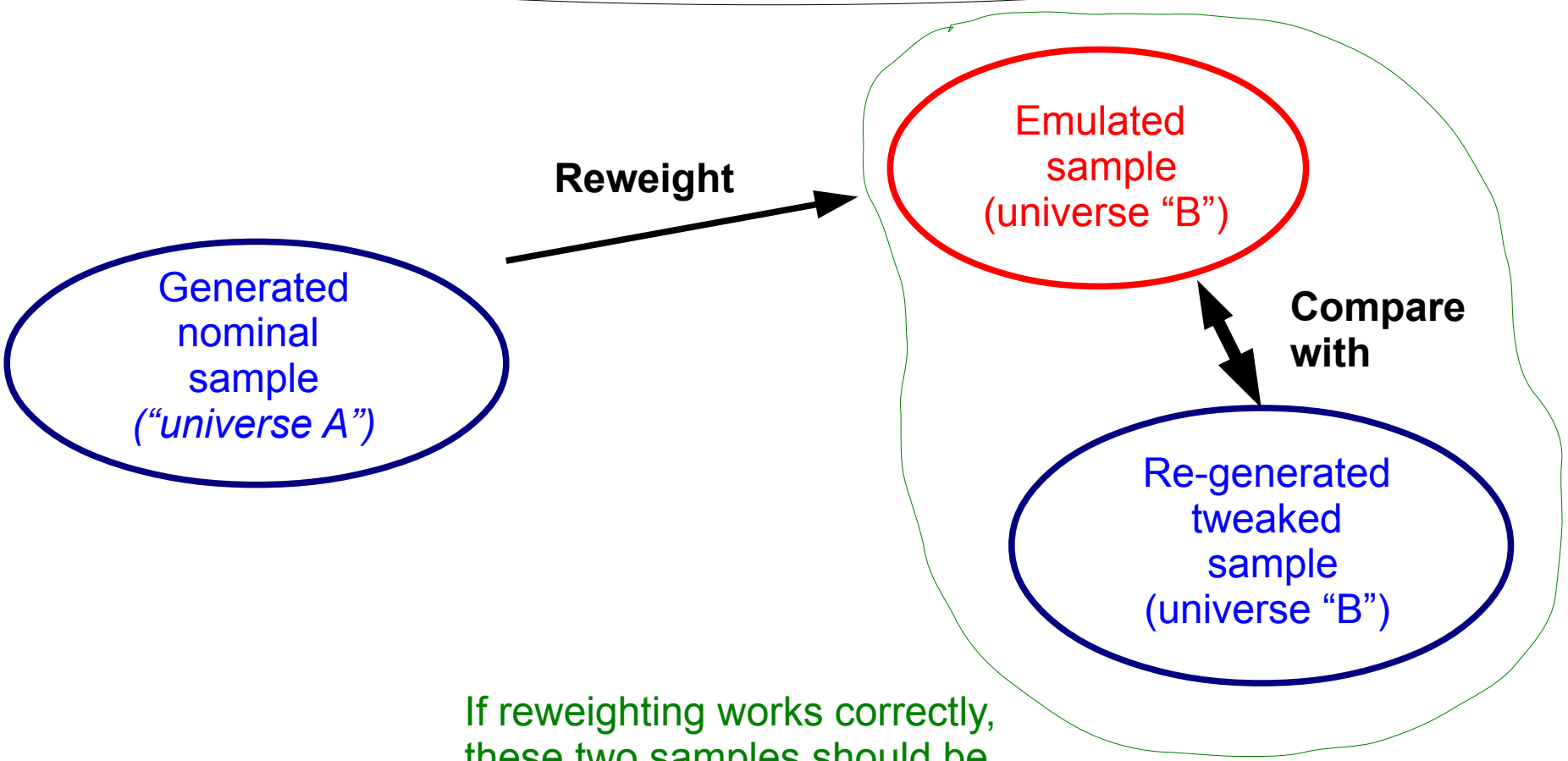
Add the new parameter in '$GENIE/src/ReWeight/GSyst.h'
Define a default 1 sigma error in $GENIE/src/ReWeight/GSystUncertainty.cxx'.

How do I know that my reweighting function works OK?

- Make sure the problem is really amenable to reweighting
  - You must have events in the full phase space to start with
    - You can not extend the phase space by reweighting: i.e. you can not reweight-up events that were not generated in the first place!
- Check properties of your event weights (eg unitarity)
- **Perform the "ultimate reweighting test"**

The ultimate reweighting test

Reweight a generated sample –
Compare with a sample generated with tweaked params



If reweighting works correctly, these two samples should be identical (within statistics)

Exercise

HyperK people seem to think that counting final state neutrons (having near detector detectors with Gd) can help constrain 1p-1h / 2p-2h.

Counter-argument: Neutrons are knocked out all the time. Whatever effect is there from the different 1p-1h and 2p-2h topologies, is largely washed away if you consider all processes (elastic and inelastic), CC and NC and typical systematics.

Do a quick study, based on the tools you already have, and give me your opinion.

Exercise

- **Generate numu + H2O samples with the T2K flux**
  - Two samples:
    - Default GENIE sample (all standard processes **but no** 2p-2h)
    - Non-default GENIE sample (all standard processes **and** 2p-2h)
- **Emulate 1-ring μ-like selection cuts** using the following simple truth-level cuts (cuts may not be entirely correct in detail, but give more or less a correct efficiency)
  - for true CCQE ask for p_μ > 200 MeV/c
  - allow CCnonQE contamination
    - true CcnonQE events with no π in the final state, or
    - true CcnonQE events with a single low energy charged π (E_π < 250 MeV)
  - allow NC contamination
    - events with a single punch-through π^+ (E_π > 200 MeV) which could be mis-IDed.
- **Count true final state neutrons**
- **Throw away (randomly) 1 in 10 neutrons (on average)**
  - Gd neutron capture efficiency 90%
- **Plot reconstructed the neutron multiplicity distribution for the two samples**
  - See any difference between having 2p-2h and not having 2p-2h?
  - Now consider a few systematics to see what happens.

Exercise

Some simple systematics to consider

- *pion and nucleon intranuclear mean free path (default GENIE errors)*
- *pion and nucleon charge exchange fraction (default GENIE errors)*
- *pion and nucleon absorption fraction (default GENIE errors)*
- *NC contamination (60% error)*

Hm! With only a handful of systematics, much of the sensitivity to even discriminate between extremes *(no 2p-2h at all → all the MB enhancement entirely due to 2p-2h)* goes away.

Neutrino generators are extremely useful in informing the experiment design. Use them!