# *How to use NuWro*

Tomasz Golan

NuSTEC, 14-17.05.2014

Uniwersytet Wrocławski

## Plan

## Installing NuWro

## Requirements

**1. ROOT**

`http://root.cern.ch/drupal/content/downloading-root`

**2. PYTHIA6**

`http://neutrino.ift.uni.wroc.pl/files/pythia6.tar.gz`

**3. NUWRO**

`http://borg.ift.uni.wroc.pl/gitweb/?p=nuwro`

## Installing ROOT

### 1. Check ROOT dependencies

```
http://root.cern.ch/drupal/content/build-prerequisites
```

### 2. Prepare pythia6

```
tar -xzvf pythia6.tar.gz
```

```
cd pythia6 && ./makePythia6.linux
```

### 3. Extract ROOT and put libPythia6.so to lib folder

```
tar -zxvf root_v*.source.tar.gz
```

```
mkdir root/lib
```

```
cp pythia6/libPythia6.so root/lib
```

## Installing ROOT

### 4. Configure and install ROOT

```
cd root && ./configure --with-pythia6-libdir=lib
```

*If it goes well, you will see: Enabled support for ..., **pythia6**, ...*

```
make
```

*Note, it will take some time.*

### 5. Add the following paths to your PATH

```
export ROOTSYS= (path to root directory)
```

```
export PATH=$PATH:$ROOTSYS/bin
```

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
```

*for bash shell add above lines into the .bashrc or .bash_profile file*

## Installing NuWro

**To install NuWro type**

```
tar -zxvf nuwro-*.tar.gz
```

```
cd nuwro && make
```

## Running NuWro

## Params.txt

**To run NuWro use the following command:**

```
./bin/nuwro   [-i  input parameters file] \
              [-o  output root file] \
              [-p  ''parameter name 1 = value 1''] \
              [-p  ''parameter name 2 = value 2''] ...
```

NuWro uses by default the params.txt file located in “nuwro” directory. If the file does not exist, the one from “nuwro/data” folder is loaded. If both files are missing or some of the parameters are not set in the file, default values are used (see attached table).

NuWro saves by default the event tree into the eventsout.root file. Cross sections are saved by default into the eventsout.root.txt file (it will be discussed later).

## Events vs test events

Test events are used to calculate cross section.
They are not saved! It is very fast. Usually, $10^6$ test events is enough.

```
number_of_test_events = unsigned int
```

A number of events saved in the output file is set by the parameter:

```
number_of_events = unsigned int
```

## Setting up a beam

### Single neutrino flavor beam

| | |
|---|---|
| `beam_type = 0` | |
| `beam_particle = PDG` $(\pm 12, \pm 14, \pm 16)$ | |
| `beam_energy =` $E$ | $\rightarrow$ `mono-energetic beam` |
| `beam_energy =` $E_{min}\ \ E_{max}$ | $\rightarrow$ `uniform beam` |
| `beam_energy =` $E_{min}\ \ E_{max}\ \ a_0\ \ a_1\ \ \ldots\ \ a_n$ | |

beam with energy range from $E_{min}$ to $E_{max}$, $a_i/\sum_j^n a_j$ gives a probability the energy will be drawn from $(i * \varepsilon,\ (i+1) * \varepsilon)$ interval, where $\varepsilon = (E_{max} - E_{min})/n$

*Example: beam_energy = 1000 2000 1 2 3 4*

*10%* $\rightarrow E_\nu$ *from 1000 1250, 20%* $\rightarrow E_\nu$ *from 1250 1500 ...*

## Setting up a beam

### Mixed neutrino flavor beam

```
beam_type = 1
```

beam_content = $n_1$ $x_1$% $be_1$

beam_content += $n_2$ $x_2$% $be_2$ ...

*$n_i$ $\rightarrow$ PDG, $x_i$ $\rightarrow$ fraction of this kind of neutrino*

*$be_i$ $\rightarrow$ like beam_energy*

*Example:*

```
beam_content = 12 75% 1000
beam_content += -12 20% 1000 2000
beam_content += 14 5% 1000 1500 1 5 10 15 5 1
```

*75% of mono-energetic electron neutrinos*

*20% of electron anti-neutrinos with uniformly distributed energy ...*

## Setting up a beam

**Predefined beams**

```
@beam/beamfile.txt
```

*Predefined beams are located in "nuwro/data/beam" directory.*

## Setting up a target

**Single nucleus**

| | |
|---|---|
| `target_type = 0` | |
| `nucleus_p =` *unsigned int* | $\longrightarrow$ `a number of protons` |
| `nucleus_n =` *unsigned int* | $\longrightarrow$ `a number of neutrons` |
| `nucleus_E_b =` *double* | $\longrightarrow$ `a binding potential` |
| `nucleus_kf =` *double* | $\longrightarrow$ `Fermi momentum` |
| `nucleus_target = 0 - 5` | $\longrightarrow$ `nucleus model` |

*0 - free nucleon, 1 - Fermi gas, 2 - local Fermi gas*

*Note, in local Fermi gas $k_F$ and $E_B$ are calculated from the density profile.*

## Setting up a target

### Composed target

```
target_type = 1
```

`target_content` = $p_1$ $n_1$ $f_1$x [$E_{B1}$ $k_{F1}$ $NT_1$]

`target_content` += $p_2$ $n_2$ $f_2$x [$E_{B2}$ $k_{F2}$ $NT_2$] ...

$p_i$ $\rightarrow$ *number of protons*, $n_i$ $\rightarrow$ *number of neutrons*

$f_i$ $\rightarrow$ *number of $i$-th kind of nucleus in the target*

$E_{Bi}$, $k_{Fi}$, $NT_i$ $\rightarrow$ *binding energy, Fermi momentum, nucleus_target*

*Example ($C_2H_6O$):*

```
target_content = 6 6 2x
target_content += 1 0 6x
target_content += 8 8 1x
```

## Setting up a target

**Predefined targets**

```
@target/targetfile.txt
```

*Predefined beams are located in "nuwro/data/target" directory.*

## Turn on/off channels

**Channels**

| | |
|---|---|
| `dyn_qel_cc = 0,1` | $\longrightarrow$ quasi-elastic charge current |
| `dyn_qel_nc = 0,1` | $\longrightarrow$ elastic neutral current |
| `dyn_res_cc = 0,1` | $\longrightarrow$ resonance pion production CC |
| `dyn_res_nc = 0,1` | $\longrightarrow$ RES NC |
| `dyn_dis_cc = 0,1` | $\longrightarrow$ deep inelastic scattering CC |
| `dyn_dis_nc = 0,1` | $\longrightarrow$ DIS NC |
| `dyn_coh_cc = 0,1` | $\longrightarrow$ coherent pion production CC |
| `dyn_coh_nc = 0,1` | $\longrightarrow$ COH NC |
| `dyn_mec_cc = 0,1` | $\longrightarrow$ meson exchange current CC |
| `dyn_mec_nc = 0,1` | $\longrightarrow$ MEC NC |

## Some other parameters

**Beam direction - default (0,0,1)**

```
beam_direction = x y x
```

**Electromagnetic form factors parameterizations**

```
qel_vector_ff_set = 1 - 6
```

**Axial form factors parameterizations**

```
qel_axial_ff_set = 1 - 4
```

**Axial mass (CC)**

qel_cc_axial_mass = $M_A$

**Spectral function**

```
sf_method = 0 - 2
```

**The model for meson exchange current**

```
mec_kind = 1 - 4
```

*see the attached table for details and the full list of parameters*

## Analyzing the output

# event1.h

## class event : public TObject

| | |
|---|---|
| `flags flag;` | qel, res, nc, cc ... |
| `vector <particle> in;` | incoming particles |
| `vector <particle> out;` | particles before FSI |
| `vector <particle> post;` | particles after FSI |

## Predefined functions

| | |
|---|---|
| `vect q();` | four-momentum transfer |
| `double q2();` | four-momentum transfer squared |
| `double W();` | invariant mass |
| `int nof (int PDG);` | #particles with PDG before FSI |
| `int fof (int PDG);` | #particles with PDG after FSI |

*and many more... see src/event1.h for details*

## particle.h

### CLASS PARTICLE : PUBLIC VECT

| | |
|---|---|
| `double E();` | total energy |
| `double Ek();` | kinetic energy |
| `double mass();` | mass |
| `double momentum();` | momentum (value) |
| `vec p();` | momentum as a vector |
| `vect& p4();` | four-momentum |

*and many more... see src/particle.h for details*

### CLASS VEC

| | |
|---|---|
| `double x, y, z;` | coordinates |
| `double length();` | vector length |
| `vec operator+ (vec a, vec b);` | and other operations |

### CLASS VECT

```
double t, x, y, z; ...
```

*and many more... see src/vec.h and src/vect.h for details*

## First simulation

*Consider charge current scattering of a mono-energetic muon neutrino beam ($E_\nu = 1$ GeV) on carbon.*

1. Create an empty file in nuwro directory (*run1.txt*)

2. Set up the parameters (in *run1.txt*):

| | |
|---|---|
| *beam_type = 0* | *mono-energetic beam* |
| *beam_particle = 14* | *muon neutrino* |
| *beam_energy = 1000* | $E_\nu = 1000$ *MeV* |
| *@target/C.txt* | *predefined carbon* |
| *dyn_qel_cc = 1* | *QEL CC* |
| *dyn_qel_nc = 0* | *EL NC* |
| *dyn_res_cc = 1* | *RES CC* |
| *dyn_res_nc = 0* | *RES NC* |
| *dyn_dis_cc = 1* | *DIS CC* |
| *dyn_dis_nc = 0* | *DIS NC* |
| *dyn_coh_cc = 1* | *COH CC* |
| *dyn_coh_nc = 0* | *COH NC* |
| *dyn_mec_cc = 1* | *MEC CC* |
| *dyn_mec_nc = 0* | *MEC NC* |

## First simulation

3. Run NuWro:

*./bin/nuwro -i run1.txt -o run1.root*

4. You will get two files:

*a) run1.root with the events tree*

*b) run1.root.txt with total cross sections in* $cm^2$

5. To analyze the ROOT file use:

*./bin/myroot*

## Command line

1. Load a ROOT file:

*TFile* f = new TFile (“run1.root”)*

2. Set up an pointer to event tree:

*TTree* t = (TTree*)f→Get(“treeout”)*

3. Draw some simple distributions:

*t→Draw(“in[0].E()”)* — *neutrino energy*

*t→Draw(“in[1].Ek()”)* — *primary nucleon kinetic energy*

*t→Draw(“out[0].p().z”)* — $p_z$ *of the outgoing lepton*

4. Add extra conditions:

4a. $\pi^+$ momentum distribution after FSI

*t→Draw(“post.momentum()”, “post.pdg == 211”)*

4b. $Q^2$ distributions for events with single $\pi^0$:

*t→Draw(“-q2()”, “fof(111) == 1 && fof(211)+fof(-211) == 0“)*

## Script example

1. Create the *script1.C* file:

```
TFile *f;
TTree *t;

void setFile (const char* input){
  f = new TFile(input);
  t = (TTree*)f->Get("treeout");
}

void leptonEnergy (){
  t->Draw("out[0].E()");
}

void pi0cosine (){
 t->Draw("post.p().z/post.momentum()", "post.pdg == 111");
}
```

## Script usage

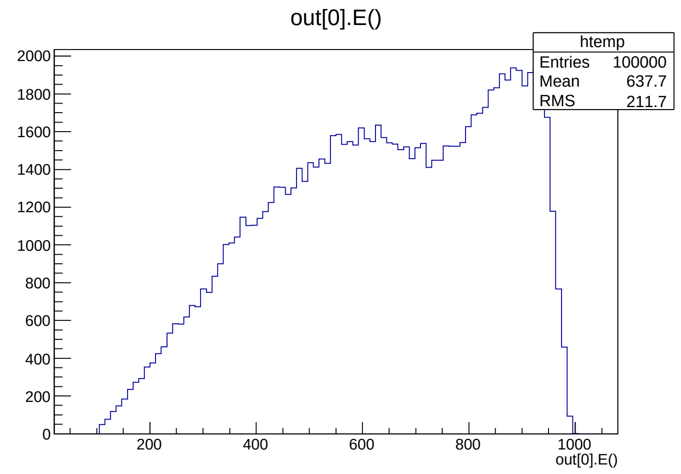
### 2. Usage:

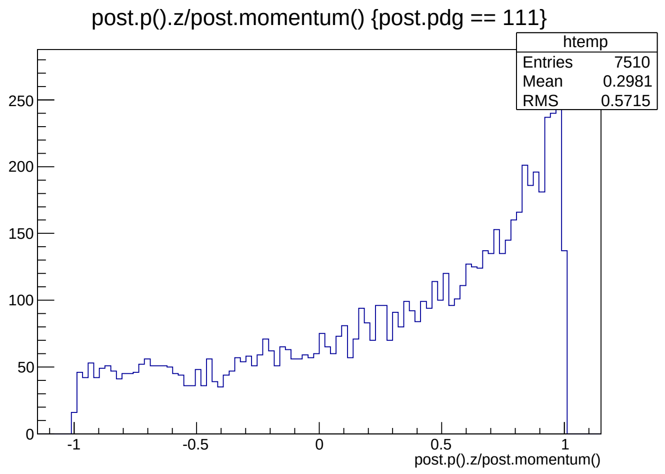
*.L script1.C*

*setFile("run1.root")*

*leptonEnergy()*

*pi0cosine()*

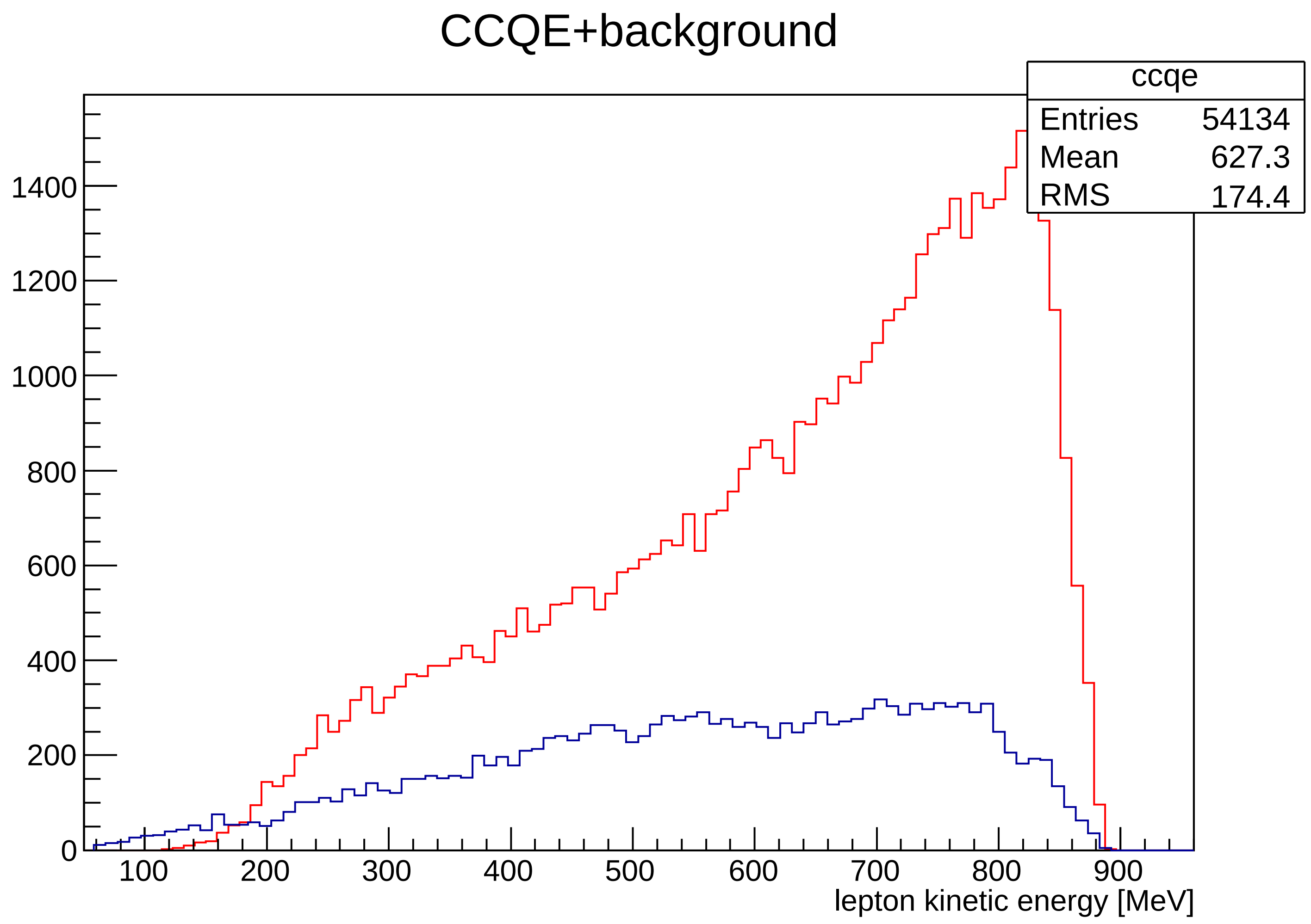




```
void firstPlot (const char* input){
  TFile *f = new TFile(input);
  TTree *t = (TTree*)f->Get("treeout");

  //create "ccqe" and "background" histograms with some cuts
  //goff -> do not create autocanvas

  t->Draw("out[0].Ek() >> ccqe", "flag->qel", "goff");
  t->Draw("out[0].Ek() >> background", "!flag->qel \
          && fof(211)+fof(111)+fof(-211)==0", "goff");

  TCanvas *c = new TCanvas;
  ccqe->SetLineColor(kRed); ccqe->SetTitle("CCQE+background");
  ccqe->SetXTitle("lepton kinetic energy [MeV]");

  ccqe->Draw();
  bkg->Draw("same"); //"same" -> on the same plot

  gSystem->ProcessEvents();
  TImage *img = TImage::Create();
  img->FromPad(c);
  img->WriteImage("first_plot.png");
}
```

## Plot example

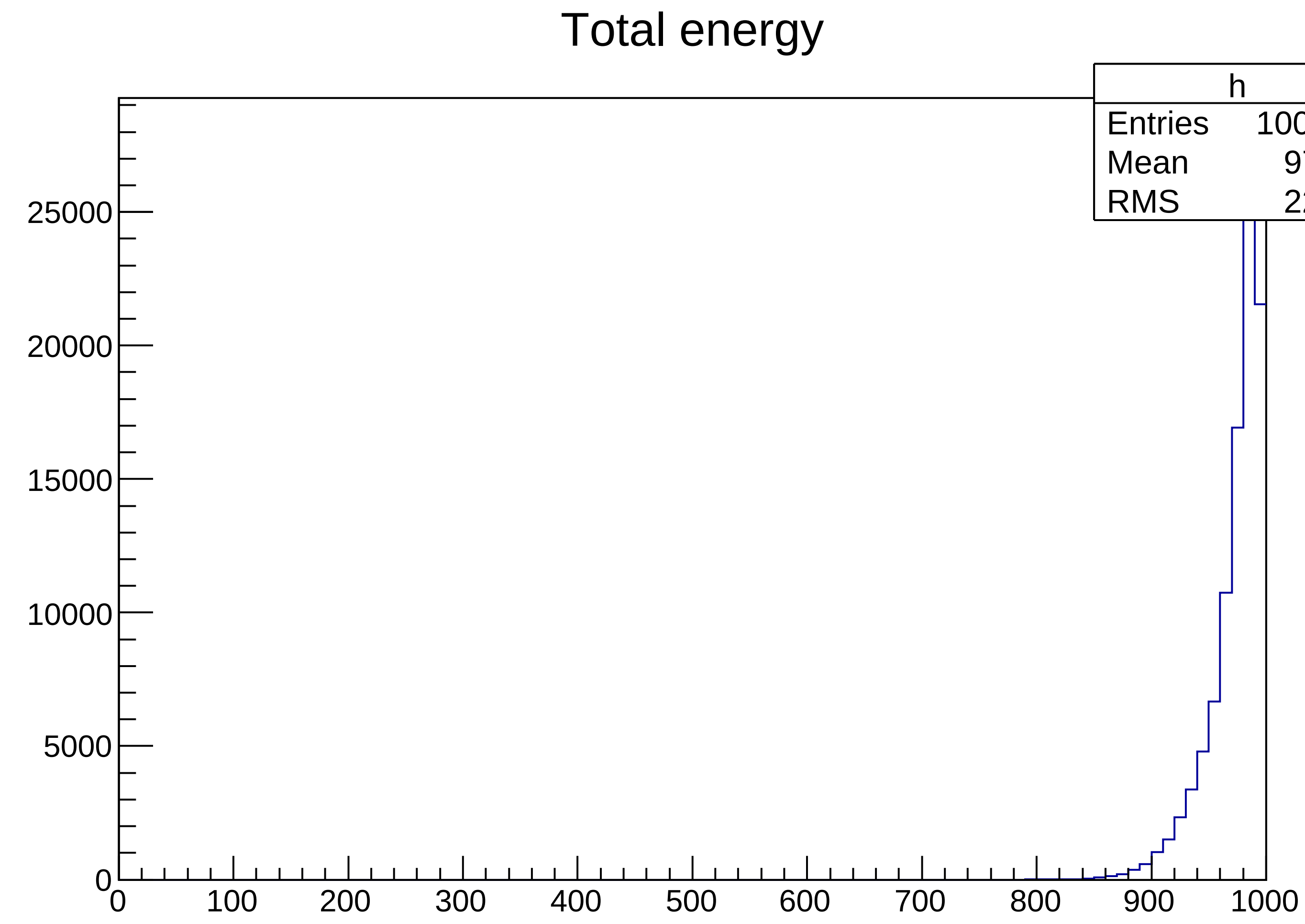
### The result of the above script:



```
void eventByEvent (const char* input){
  TFile *f = new TFile(input);
  TTree *t = (TTree*)f->Get("treeout");
  //create a pointer to event
  event *e   = new event();
  t->SetBranchAddress("e",&e);

  TH1D* h = new TH1D("h", "Total energy", 100, 0, 1000);

  for (int i = 0; i < t->GetEntries(); i++){
    t->GetEntry(i);

    double E = 0;
    for (int k = 0; k < e->post.size(); k++)
      if (e->post[k].nucleon())
        E += e->post[k].Ek();
      else
        E += e->post[k].E();

    h->Fill(E);
  }
  h->Draw();
}
```

## Event by event

### THE RESULT OF THE ABOVE SCRIPT:



## Cross section

In NuWro each event is accepted with the probability proportional to the cross section (in a tree each event is equally weighted).

$$\left.\frac{d\sigma}{dx}\right|_{x=x_0} \rightarrow \frac{N(x = x_0 \pm \Delta x/2)}{N_{total}} \frac{\sigma_{total}}{\Delta x}$$

The table with cross sections (per nucleon) is saved into the *eventout.root.txt* file:

| *Channel* | *#events* | *Fraction* | *Cross section [$cm^2$]* | |
|---|---|---|---|---|
| *0* | *54134* | *0.54134* | *5.71421e-39* | *(qel cc)* |
| *1* | *0* | *0* | *0* | *(qel nc)* |
| *2* | *33534* | *0.335339* | *3.53972e-39* | *(res cc)* |
| *3* | *0* | *0* | *0* | *(res nc)* |
| *4* | *48* | *0.000480844* | *5.07563e-42* | *(dis cc)* |
| *5* | *0* | *0* | *0* | *(dis nc)* |

...

Uniwersytet Wrocławski

## Cross section

To read cross section from the *txt* file you can use the following function:

```
double xsec (const char* input)
{
  double temp, res = 0;
  ifstream Input (input);

  getline (Input,string());
  while(Input)
  {
    for (int k = 0; k < 4; k++)
      Input>>temp;
    res+=temp;
  }
  Input.close();
  return res;
}
```

Uniwersytet Wrocławski

## Example

Now we try to figure it out how $M_A$ affects the shape and the normalization of the cross section

1. Prepare the *ccqe_par.txt* file (like *run1.txt* but only QEL CC is on)

2. Prepare a bash script (*ccqe.sh*):

```
#!/bin/sh
for i in $(seq 1000 100 1300)
do
  ./bin/nuwro -i ccqe_par.txt -o ccqe$i.root -p "qel_cc_axial_mass = $i"
done
```

You will get 4 ROOT files: ccqe1000.root, ccqe1100.root, ...

## Example

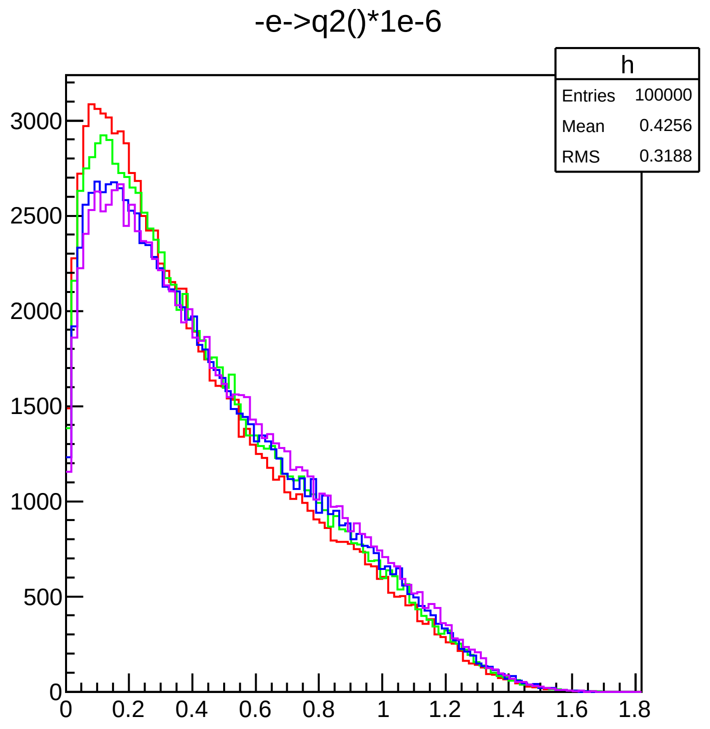
3. Prepare a function for the extraction of a histogram from a ROOT file, for example:

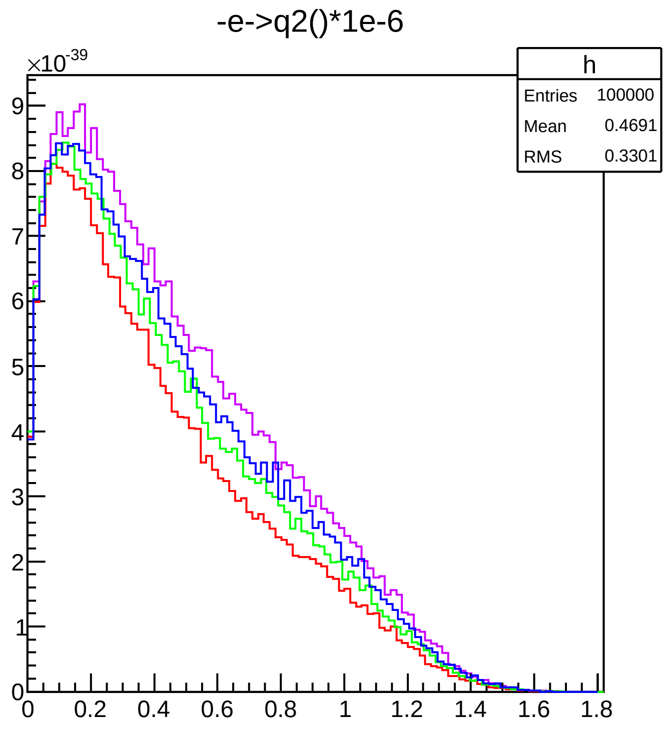
```
TH1F* ccqe_q2 (const char* input){
  TFile *f = new TFile(input);
  TTree *t = (TTree*)f->Get("treeout");
  t->Draw("-e->q2()*1e-6 >> h","","goff");
  TH1F *res = new TH1F(*h);
  return res;
}
```

```
void ccqe_run(){
  TH1F* h1000 = ccqe_q2("ccqe1000.root");
  TH1F* h1100 = ccqe_q2("ccqe1100.root");
  TH1F* h1200 = ccqe_q2("ccqe1200.root");
  TH1F* h1300 = ccqe_q2("ccqe1300.root");

  TCanvas *c = new TCanvas; c -> Divide(2,1); c -> cd(1);

  h1000->SetLineColor(kRed); h1000->Draw();
  h1100->SetLineColor(kGreen); h1100->Draw("same");
  h1200->SetLineColor(kBlue); h1200->Draw("same");
  h1300->SetLineColor(kViolet); h1300->Draw("same");

  c->cd(2);
  double factor = 1.0 / h1000->GetBinWidth(0) / h1000->GetEntries();

  TH1F* h1000n = new TH1F(*h1000 * xsec("ccqe1000.root.txt") * factor);
  TH1F* h1100n = new TH1F(*h1100 * xsec("ccqe1100.root.txt") * factor);
  TH1F* h1200n = new TH1F(*h1200 * xsec("ccqe1200.root.txt") * factor);
  TH1F* h1300n = new TH1F(*h1300 * xsec("ccqe1300.root.txt") * factor);

  h1300n->Draw(); h1000n->Draw("same");
  h1100n->Draw("same"); h1200n->Draw("same");
}
```

# Example - result

## The result of the above script:





params.txt

# Params.txt

NuWro uses by default the *params.txt* file located in "nuwro" directory. If the file does not exist, the one from "nuwro/data" folder is loaded. If both files are missing or some of the parameters are not set in the file, default values are used. In the table below one can find a detailed description of all parameters.

| **Parameter name** | **Possible arguments** | **Default value** | **Description** |
|---|---|---|---|
| **General settings** | | | |
| number_of_events | any positive integer number | 100 000 | The number of equally weighted events to be saved in the output ROOT file *(eventsout.root)*. |
| number_of_test_events | any positive integer number | 1 000 000 | The number of events used to calculate cross sections (not saved by default). |
| user_events | 0, 1 | 0 | Used to turn on the fitting procedure:<br>0 - Run NuWro;<br>1 - Fit axial mass to MiniBooNE data for CCQE. |
| user_params *(use with user_events 1)* | x y z | - | Parameters for the axial mass extraction procedure:<br>x - the minimum axial mass value;<br>y - the maximum axial mass value;<br>z - the axial mass step. |
| random_seed | any positive integer number | 0 | Controls the random seed persistence:<br>0 - use time(NULL) as a seed for the random number generator;<br>1 - read state from "random_seed" file or use seed=time(NULL), if the file was not found;<br>$n$ - use $x$ as the seed for the random number generator. |
| mixed_order | 0, 1 | 1 | If 1, events are saved to the output file in random order. |

| | | | |
|---|---|---|---|
| save_test_events | 0 - 2 | 0 | Turn on to use test events in the analysis:<br>0 - test events are not saved;<br>1 - test events are finalized and stored in *weight.eventsout.root* file, the average weight is equal to the total cross section;<br>2 - test events of non-zero weights are finalized and stored in *weight.eventsout.root* file, the weights are respectively scaled, so the average weight is equal to the total cross section. |
| **Beam specification** | | | |
| beam_direction | $x\ y\ x$ | 0 0 1 | The direction of the neutrino momentum in $xyz$ coordinates. |
| beam_particle *(use with beam_type 0)* | ± 12, ± 14, ± 16 | 14 | PDG number of the incident neutrino. |
| beam_type | 0 - 4 | 0 | Types of beams:<br>0 - a single neutrino flavor beam;<br>1 - a mixed flavor beam;<br>2 - a beam loaded from a ROOT file;<br>3 - a beam loaded from the histogram *(histout.txt)*;<br>4 - create *histout.txt* file based on a ROOT file *(than use beam_type 3 to run NuWro)*. |
| beam_energy *(use with beam_type 0)* | (1) $E$<br>(2) $E_{min}\ E_{max}$<br>(3)<br>$E_{min}\ E_{max}\ a_0 \dots a_n$ | 1000 | The energy profile:<br>(1) set a mono energetic beam;<br>(2) set an uniform beam with energy range from $E_{min}$ to $E_{max}$;<br>(3) set a beam with energy range from $E_{min}$ to $E_{max}$, $a_i/\sum_j^n a_j$ gives a probability the energy will be drawn from $(i * \varepsilon,\ (i+1) * \varepsilon)$ interval, where $\varepsilon = (E_{max} - E_{min})/n$. |

| | | | |
|---|---|---|---|
| beam_content *(use with beam_type 1)* | *n x%* + beam_energy | *empty* | The mixed beam definition: beam_content = $BC_1$ beam_content += $BC_2$ ... $BC_i = n_i\ x_i\%\ BE_i$, $n_i$ is a PDG number of the incident neutrino, $x_i$ is a percent of this kind of neutrino in the beam, $BE_i$ is the definition of the energy range (like in beam_energy). |
| beam_folder *(with beam_type 2,4)* | *path* | ../flux | The path to the directory with ROOT files. |
| beam_file_first *(with beam_type 2,4)* | any positive integer number | 1 | The number of the first file in the folder to be read. |
| beam_file_limit *(with beam_type 2,4)* | any positive integer number | 0 | The number of files to be loaded (0 - read files to the last one in the directory). |
| beam_offset | *x y z* | 0 0 0 | The offset of the position of the interaction in *xyz* coordinates. |
| beam_placement *(in cascade mode only)* | 0 - 2 | 0 | The starting position of the particle: 0 - the propagation starts at the center of the nucleus; 1 - the propagation starts at a random place inside the nucleus; 2 - the propagation starts just under the surface of the nucleus. |
| One can also use predefined beam specifications instead of the above parameters. The list of beams can be found in "nuwro/data/beam" directory. To use one of those beams, one must use the following line: @beam/*beamfile.txt* where *beamfile.txt* is the name of the file from "nuwro/data/beam" directory. | | | |
| **Target specification** | | | |
| target_type | 0, 1, 2 | 0 | Types of targets: 0 - a single nucleus; 1 - a target composed from some nuclei; 2 - a detector geometry loaded from a ROOT file. |

| | | | |
|---|---|---|---|
| nucleus_p<br>*(use with target_type 0)* | any positive integer number | 6 | A number of protons in the target nucleus. |
| nucleus_n<br>*(use with target_type 0)* | any positive integer number | 6 | A number of neutrons in the target nucleus. |
| nucleus_E_b<br>*(use with target_type 0)* | any positive number | 34 | The binding potential (sum of binding and Fermi energies). |
| nucleus_kf<br>*(use with target_type 0)* | any positive number | 220 | The Fermi momentum. |
| nucleus_target | 0 - 5 | 2 | Nucleus models used in a primary interaction:<br>0 - free nucleon;<br>1 - Fermi gas;<br>2 - local Fermi gas;<br>3 - Bodek-Ritchie model;<br>4 - spectral function;<br>5 - deuterium. |
| nucleus_model | 0, 1 | 1 | Nucleus density profiles for FSI:<br>0 - constant density;<br>1 - realistic density profile. |
| target_content<br>*(use with target_type 1)* | $a\ b\ c$x $d\ e\ f$ | - | The composed target definition:<br>target_content = TC$_1$<br>target_content += TC$_2$<br>...<br>TC$_i$ = $a_i$ $b_i$ $c_i$x $d_i$ $e_i$ $f_i$, $a_i$ is the number of protons, $b_i$ is the number of neutrons, $c_i$ is the number of $i$-th kind of nucleus in the target, $d_i$ (optional) is the binding energy, $e_i$ (optional) is the Fermi momentum, $f_i$ (optional) is the nucleus model (like in nucleus_target). |
| geo_file<br>*(use with target_type 2)* | *filename* | see description | The path to the file with the detector geometry (default *target/ND280_v9r7p5.root*). |
| geo_name<br>*(use with target_type 2)* | *geometry name* | see description | The name of the geometry in the file (default *ND280Geometry_v9r7p5*). |
| geo_o<br>*(use with target_type 2)* | $x\ y\ z$ | 0 0 0 | The coordinates of the center of the box. |
| geo_d<br>*(use with target_type 2)* | $x\ y\ z$ | see description | The half dimension of the box (default 2000 5000 5000). |

| | | | |
|---|---|---|---|
| geo_volume<br>*(use with target_type 2)* | *master volume name* | - | The name of the *master volume* in the detector file. |
| One can also use predefined target specifications instead of the above parameters.<br>The list of targets can be found in "nuwro/data/target" directory.<br>To use one of those beams, one must use the following line:<br>@target/*targetfile.txt*<br>where *targetfile.txt* is the name of the file from "nuwro/data/target" directory. | | | |
| **Interaction settings** | | | |
| dyn_qel_cc | 0, 1 | 1 | Turn on/off charge current quasi-elastic process. |
| dyn_qel_nc | 0, 1 | 1 | Turn on/off neutral current elastic process. |
| dyn_res_cc | 0, 1 | 1 | Turn on/off charge current resonance pion production.. |
| dyn_res_nc | 0, 1 | 1 | Turn on/off neutral current resonance pion production. |
| dyn_dis_cc | 0, 1 | 1 | Turn on/off charge current deep inelastic scattering. |
| dyn_dis_nc | 0, 1 | 1 | Turn on/off neutral current deep inelastic scattering. |
| dyn_coh_cc | 0, 1 | 1 | Turn on/off charge current coherent pion production. |
| dyn_coh_nc | 0, 1 | 1 | Turn on/off neutral current coherent pion production. |
| dyn_mec_cc | 0, 1 | 1 | Turn on/off charge current meson exchange current process. |
| dyn_mec_nc | 0, 1 | 1 | Turn on/off neutral current meson exchange current process. |
| **Quasi-elastic** | | | |
| qel_vector_ff_set | 1 - 6 | 2 | Electromagnetic form factors parametrization:<br>1 - dipole form;<br>2 - BBBA05 (Ref. [1]);<br>3 - BBA03 (Ref. [2]);<br>4 - JLab (Ref. [3]);<br>5 - NN10 with two photon exchange effect (Ref. [4]). |

| | | | |
|---|---|---|---|
| qel_axial_ff_set | 1 - 4 | 1 | Axial form factors parametrization:<br>1 - dipole form;<br>2 - 2-fold parabolic modification;<br>3 - 3-fold parabolic modification;<br>4 - 4-fold parabolic modification. |
| qel_strange | 0, 1 | 0 | Turn on/off the strange quark contribution to the NC axial form factors. |
| qel_strangeEM | 0, 1 | 0 | Turn on/off the strange quark contribution to the NC vector form factors. |
| delta_s | any number | -0.15 | $g_A^s$ (see Subsec. **??**). |
| qel_cc_axial_mass | any positive number | 1200 | The axial mass value for charge current form factors. |
| qel_nc_axial_mass | any positive number | 1350 | The axial mass value for neutral current form factors. |
| qel_s_axial_mass | any positive number | 1200 | The axial mass value used in the dipole strange form factor. |
| qel_rpa | 0 - 3 | 0 | RPA settings:<br>0 - do not use RPA;<br>1 - use RPA without effective mass of nucleon;<br>2 - use effective mass of nucleon without RPA (test only);<br>3 - use RPA with effective mass of nucleon (test only). |
| flux_correction | 0, 1 | 1 | Turn on/off flux correction. |
| sf_method | 0 - 3 | 0 | Spectral function settings (for CCQE):<br>0 - do not use spectral function;<br>1 - use grid spectral function (for $^{12}C$, $^{16}O$, $^{40}Ar$, $^{40}Ca$, $^{56}Fe$);<br>2 - use factorized spectral function (for $^{16}O$, $^{40}Ar$, $^{40}Ca$). |

| | | | |
|---|---|---|---|
| cc_smoothing | 0, 1 | 1 | If 1, the impossible quasi-elastic reaction (like CC $\nu$ scattering off proton) are skipped. |
| **Pion production** | | | |
| delta_FF_set | 1 - 7 | 1 | $\Delta$ production form factors:<br>1 - dipole form;<br>2 - Paschos and Lalakulich, 2.12 $M_A = 1.05 GeV$ BNL fit (Ref. [5]);<br>3 - Paschos and Lalakulich, 2.12 $M_A = 0.84 GeV$ ANL fit (Ref. [5]);<br>4 - Paschos and Lalakulich, page 4, bottom right (Ref. [5]);<br>5 - Paschos and Lalakulich, page 5, top left (Ref. [5]);<br>6 - Eq. 13 from Ref. [6];<br>7 - based on chiral quark model from Ref. [7]. |
| pion_axial_mass<br>*(for delta_FF_set 1)* | any positive number | 0.94 | The axial mass value used in dipole parametrization of the resonance pion production form factor. |
| pion_C5A<br>*(for delta_FF_set 1)* | any positive number | 1.19 | The $C_A^5$ value used in dipole parametrization of the resonance pion production form factor. |
| spp_precision | any positive number | 500 | Controls the precision in RES-DIS boundary region. Should not be changed. |
| red_dis_cut | any positive number | 1600 | Boundary of RES-DIS transition. Should not be changed. |
| coh_mass_correction | 0, 1 | 1 | Turn on/off Rein Sehgal correction to charge current coherent pion production. |
| coh_new | 0, 1 | 1 | Change between old (0) and improved (1) implementation of coherent pion production. |

| | | | |
|---|---|---|---|
| **Two-body current** | | | |
| mec_kind | 1 - 4 | 1 | Two-body current models:<br>1 - Transverse Enhancement model (Ref. [8]);<br>2 - based on Marteau model (Ref. [9]);<br>3 - Nieves et al. model (Ref. [10]);<br>4 - Martini et al. model (Ref. [9, 11]). |
| mec_ratio_pp | any positive number from [0,1] | 0.6 | The fraction of mixed initial nucleon pairs for charge current interaction. For neutral current the fraction is calculated as 1/(2*mec_ratio_pp + 1). |
| **Final state interactions settings** | | | |
| kaskada_on | 0, 1 | 1 | Turn on (1) / off (0) final state interactions. |
| kaskada_w | any positive number | 7 | The value of the effective potential subtracted from the nucleons energy leaving the nucleus. |
| kaskada_redo | 0, 1 | 0 | If on, given output file (*eventsout.root* by default) is loaded, the primary vertex is copied and only final state interactions are simulated. New output file with *".fsi.root"* suffix is created. |
| kaskada_writeall | 0, 1 | 0 | If on, all particles created during final state interactions are saved in *all* vector. |
| step | any positive number | 0.2 | Length of max step in the cascade in fm. |
| xsec | 0, 1 | 1 | Cross section models for pion-nucleon interactions:<br>0 - based on Ref. [12];<br>1 - based on Ref. [13]. |
| pauli_blocking | 0, 1 | 1 | Turn on/off Pauli blocking. |
| formation_length<br>*(with formation_zone 7)* | any positive number | 1 | Formation length in fm. |

|  |  |  |  |
|---|---|---|---|
| tau | any positive number | 8 | The parameter control the formation length for *ranft* and *rl* models. |
| first_step | 0, 1 | 0 | If off, the formation zone is applied only for the particles created during final state interactions. |
| formation_zone | (0) nofz<br>(1) skat8<br>(2) cosyn<br>(3) cohl<br>(4) ranft<br>(5) rl<br>(6) delta<br>(7) const<br>(8) fz<br>(9) trans | fz | Formation zone models:<br>(0) formation zone is off;<br>(1) SKAT parametrization (Ref. [14]);<br>(2) parametrization based on Color Transparency measurements (Ref. [15]);<br>(3) coherence length (Ref. [16]);<br>(4) parametrization based on hadron-hadron and hadron-nucleus collision (Ref. [17]);<br>(5) as (4) but with fixed transverse momentum equal zero.<br>(6) for resonance pion production. Based on $\Delta$ lifetime (Ref. [18]);<br>(7) constant value;<br>(8) default model: (3) for quasi-elastic scattering, (6) for resonance pion production, (4) for deep inelastic scattering and (0) for meson exchange current.<br>(9) only for nuclear transparency analysis. |

# References

[1] R. Bradford et al. "A New parameterization of the nucleon elastic form-factors". Nucl.Phys.Proc.Suppl. 159 (2006), pp. 127–132.

[2] Howard Scott Budd, A. Bodek, and J. Arrington. "Modeling quasielastic form-factors for electron and neutrino scattering" . arXiv: hep-ex/0308005 [hep-ex] (2003).

[3] E.J. Brash et al. "New empirical fits to the proton electromagnetic form-factors". Phys.Rev. C65 (2002), p. 051001.

[4] Krzysztof M. Graczyk, Piotr Plonski, and Robert Sulej. "Neural Network Parameterizations of Electromagnetic Nucleon Form Factors". JHEP. 1009 (2010), p. 053.

[5] Olga Lalakulich and Emmanuel A. Paschos. "Resonance production by neutrinos. I. J = 3/2 resonances". Phys.Rev. D71 (2005), p. 074003.

[6] L. Alvarez-Ruso, S. K. Singh, and M. J. Vicente Vacas. "Charged current weak electroproduction of the $\Delta$ resonance". Phys. Rev. C. 57 (5 1998), pp. 2693–2699.

[7] D. Barquilla-Cano, A.J. Buchmann, and E. Hernandez. "Axial N-¿Delta(1232) and N-¿N*(1440) transition form factors". Phys.Rev. C75 (2007), p. 065203.

[8] A. Bodek, H.S. Budd, and M.E. Christy. "Neutrino Quasielastic Scattering on Nuclear Targets: Parametrizing Transverse Enhancement (Meson Exchange Currents)". Eur.Phys.J. C71 (2011), p. 1726.

[9] Jan T. Sobczyk. "Modeling nuclear effects in neutrino interactions in 1-GeV region" . arXiv: nucl-th/0307047 [nucl-th] (2003).

[10] J. Nieves, I. Ruiz Simo, and M. J. Vicente Vacas. "Inclusive charged-current neutrino-nucleus reactions". Phys. Rev. C. 83 (4 2011), p. 045501.

[11] M. Martini et al. "Unified approach for nucleon knock-out and coherent and incoherent pion production in neutrino interactions with nuclei". Phys. Rev. C. 80 (6 2009), p. 065501.

[12] N. Metropolis et al. "Monte Carlo Calculations on Intranuclear Cascades. I. Low-Energy Studies". Phys.Rev. 110 (1958), pp. 185–203.

[13] L.L. Salcedo et al. "Computer simulation of inclusive pion nuclear reactions". Nucl.Phys. A484 (1988), p. 557.

[14] D.S. Baranov et al. "An estimate for the formation length of hadrons in neutrino interactions" (1984).

[15] Wim Cosyn. "Exploring the limits of a hadronic picture of nuclei through pion and nucleon removal reactions". PhD thesis. Ghent University, 2009. URL: `http://lib.ugent.be/fulltxt/RUG01/001/350/817/RUG01-001350817\_2010\_0001\_AC.pdf`.

[16] A. Rubbia G. Battistoni A. Ferrari and P.R. Sala. *The FLUKA nuclear cascade model applied to neutrino interactions.* talk given at NuInt02. 2002.

[17] J. Ranft. “Hadron production in hadron-nucleus and nucleus-nucleus collisions in a dual parton model modified by a formation zone intranuclear cascade”. Zeitschrift für Physik C Particles and Fields. 43.3 (1989), pp. 439–446.

[18] Tomasz Golan, Cezary Juszczak, and Jan T. Sobczyk. “Final State Interactions Effects in Neutrino-Nucleus Interactions”. Phys.Rev. C86 (2012), p. 015505.