Introduction to NEUT

Nguyen Thi Hong-Van

Institute of Physics (IOP),
Vietnam Academy of Science and Technology, Ha Noi
&
Institute for Interdisciplinary Research in Science and Education
(IFIRSE), Quy Nhon

Vietnam Neutrino Meeting
September 28, 2017

What is NEUT?

- NEUT - *A neutrino interaction simulation program library* is a program library to simulate neutrino interactions with nucleon and nucleus.
- NEUT has been developed to study the atmospheric neutrino and the accelerator neutrinos.
- The main application of NEUT is to simulate the interactions of atmospheric neutrino in the water Cherenkov detector
- Neutrino interactions are considered in NEUT:
  - Charged/neutral current (quasi-)elastic scattering ($\nu N \to \ell N'$)
  - Charged/neutral current single π production ($\nu N \to \ell N' \pi$)
  - Charged/neutral current single γ production ($\nu N \to \ell N' \gamma$)
  - Charged/neutral current single K production
  - Charged/neutral current single η production
  - Charged/neutral current deep inelastic scattering ($\nu N \to \ell N' hadrons$)

How to install NEUT and how it works?

- **How to install NEUT?**
  https://www.dropbox.com/sh/hnxgw4f0b5w3fdo/AAA6h3Ncx5Sp1D1n2IiNjlRoa?dl=0
- **Documentations:**
  https://www.dropbox.com/sh/yexakk03qt75ym3/AAB5NWIwpUmMl3W7TsJ_f-Ata?dl=0

How to use the

Neutrino Event Generators

"It's a kind of magic"

T2K Masterclass

Paul Martins
QMUL

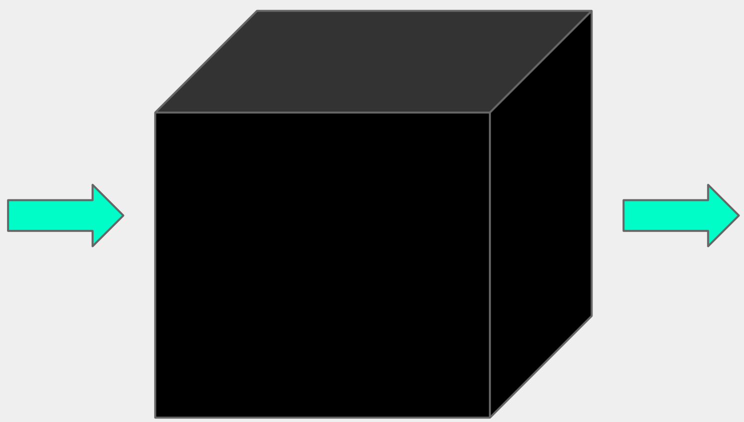
What are we talking about ?

Monte Carlo simulation of a neutrino interaction.

Simulation software

Inputs

- Neutrino type
- Target
- Energy (or flux)
- Interaction mode
- Extra parameters (Ma, …)



Outputs

- ROOT file
- kinematics of outgoing particles for each event
- Extra variables (x, y, …)

Softwares: NEUT, GENIE, Neugen, NuWro, GiBUU…

Overall Comparison

NEUT

- Developed by and for Super-K and T2K people. Use Assembla (~svn)
- Mainly written in fortran. Rather old but still well maintained.
- Used for T2K MC official production.

Contacts:
Hayato-san, Ryan Terri, Callum Wilkinson, Patrick Stowell, Clarence Wret, Pierre Lasorak

GENIE

- Universal neutrino events generator.
- Developed by an international collaboration.
- Written in C++ and well maintained, open source. Use svn

Contacts:
Costas Andreopoulos, Steve Sytman, Ryan Terri, Callum Wilkinson, Martti Nirkko, Teppei Katori

NuWro

- More theory oriented. Developed by people from Wroclaw University.
- Written in C++.
- Available from a GIT repository.

Contacts:
Tomasz Golan, Krzysztof Graczyk, Cezary Juszczak, Jarosław Nowak, Jan Sobczyk, Jakub Żmuda,

NEUT (v.5.3.4)

Download: http://www.t2k.org/asg/xsec/niwgdocs/neut

Installation: http://www.t2k.org/asg/xsec/niwgdocs/neut/install_neut

Documentation: https://repo.nd280.org/viewvc/T2K/NIWG/neut_doc

To get the documentation:

```
$ export CVSROOT=:ext:username@repo.nd280.org:/home/trt2kmgr/T2KRepository/NIWG
$ cvs co neut_doc
```

(If you wish to contribute to the development of this software, you are free to ask Hayato -san to send you an Assembla invitation.)

To build the application, set the CERNLIB environment, source ROOT and compile NEUT:

```
NEUT/neut_trunk/tags/neut_5.3.4/src/neutsmpl$ source /opt/root/bin/thisroot.sh
NEUT/neut_trunk/tags/neut_5.3.4/src/neutsmpl$ ./Makeneutsmpl.csh
```

Then run the application:

```
NEUT/neut_trunk/tags/neut_5.3.4/src/neutsmpl$ ./neutroot2 neut.card output.root
```

You need to be here to run the command

executable **input** **output**

Can only be produced in neutsmpl
Move it once the job is finished

Analyse the output:

```
NEUT/neut_trunk/tags/neut_5.3.4/src/neutsmpl$ root -l make_histos.cc
```

NEUT structure

/src

/crsdat	XS tables (splines) for CCQE, CC1pi, 2p-2h, Nieves npnh,...
/gcalor	GEANT-CALOR Interface (Nucleon-meson transport code)
/kamflux	Kamioka flux
/mec	code for 2p-2h effects
/neutclass	definition of NEUT classes (what's in the output file, how is it filled)
/neutcore	**core of NEUT, where you implement the models of neutrino interaction (diff XS)**
/neutgeom	ND280 interface (optional) - compute events rate with ND280 geometry
/neutsmpl	**Build application to generate the events (call functions from neutcore)**
/neututils	Playground (macros to help you implement the model, do tests)
/nuccorspl	**???**
/nuceff	code for nuclear effects (pion absorptions, decays, gamma interactions,...)
/partnuck	Irvine-Michigan-Brookhaven (IMB) kinematics code (for proton decay **???**)
/pionsmpl	pion scattering and photo-production code
/radcorr	scripts to take radiative corrections into account
/reweight	T2KReWeight interface (optional) - Set up the dials for a reweight using NEUT
/skmcsvc	Set up common functions (masses, random numbers, rdm position, rdm direction,...)
/specfunc	Code for spectral functions model
/t2kflux_zbs	SK interface (optional)
/tauola	code for tau lepton decays
/zbsfns	zebra functions (used by SK to generate events, old file format)

NEUT - Inputs

All the parameters must be set in a .card file

```
C Number of events ; EVCT-NEVT
C
EVCT-NEVT  1000000
C
C------------------------------
C
C Particle Code ; EVCT-IDPT
C
 EVCT-IDPT 14
C
C------------------------------
C
C fixed  VERTEX ; EVCT-MPOS 1
C random VERTEX ; EVCT-MPOS 2
C
C EVCT-MPOS 1
C VECT-POS  100. 0. 0.
C
CEVCT-MPOS 2
EVCT-RAD  100.
```

Most of the parameters are described in the file.
You can find several examples of card files in src/neutsmpl/Cards/

The most important parameters are also described in the documentation.

When setting up your card file, check carefully that:

1. Your parameter starts at the first column
2. There is no “C” in front of the line (otherwise it will be considered as a comment)
3. the NEUT-MODE parameter: set the interaction mode (more information in neutcore/nemodsel.F)

```
C MODE : Interaction mode
C                       0 : normal ( default )
C                      -1 : input cross section by CRSNEUT
C                       n : sellect one mode ( n > 0 )   See nemodsel.F
C                           n =  1 : charged current Q.E.
C                           n = 11,12,13
C                                  : charged current Single pi production
C                           n = 16 : coherent Single pi production
C                           n = 21 : charged current Multi pi production
C                           n = 31,32,33,34
C                                  : neutral current Single pi production
C                           n = 36 : coherent Single pi production
C                           n = 41 : neutral current Multi pi production
C                           n = 51,52 : neutral current elastic
C                           n = 22,42,43 : single eta production
C                           n = 23,44,45 : single  K  production
C
NEUT-MODE    0
```

NEUT-MODE 0 : all interactions selected
NEUT-MODE > 0 : one interaction selected

In both cases, the Totcrs variable in the output file will correspond to the **TOTAL** XS ($x10^{-38}m^2$ per nucleon)

NEUT - Inputs

What if I want only coherent interactions and coherent XS ?

NEUT-MODE -1: customized interactions

```
C  nu                      nub
C  1:      CC Q.E.         CC Q.E.( Free )
C  2-4:    CC 1pi          CC 1pi
C  5:      CC DIS 1320     CC DIS 1.3 < W < 2.0
C  6-9:    NC 1pi          NC 1pi
C  10:     NC DIS 1320     NC DIS 1.3 < W < 2.0
C  11:     NC els          CC Q.E.( Bound )
C  12:     NC els          NC els
C  13:     NC els          NC els
C  14:     CC coherent     NC els
C  15:     NC coherent     CC coherent
C  16:     CC eta          NC coherent
C  17      NC eta          CC eta
C  18:     NC eta          NC eta
C  19:     CC K            NC eta
C  20      NC K            CC K
C  21:     NC K            NC K
C  22:     N/A             NC K
C  23:     CC DIS          CC DIS (W > 2.0)
C  24:     NC DIS          NC DIS (W > 2.0)
C  25:     CC 1 gamma      CC 1 gamma
C  26,27:  NC 1 gamma      NC 1 gamma
C
C
C CRS  :  Multiplied factor to cross section on each mode. ( neu )
C CSRB :  Multiplied factor to cross section on each mode. ( neu-bar )
C
C          1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
NEUT-CRS   0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
NEUT-CRSB  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

According to the sign of the EVCT-IDPT (ν or $\bar{\nu}$), we can choose a set of specific interactions.

For example, ν_μ NC 1π coherent corresponds to #15.

In that case, the variable Totcrs in the output tree will correspond to the coherent XS.

In contrast, setting NEUT-MODE 36 will also generate the same interaction but Totcrs won't be the coherent XS.

NEUT - Events generation

```
NEUT/neut_trunk/tags/neut_5.3.4/src/neutsmpl$ ./neutroot2 neut.card output.root
```

Most important subroutines are written `like that.`

When running ./neutroot2, the first function to be called is `/neutsmpl/neutroot.F`

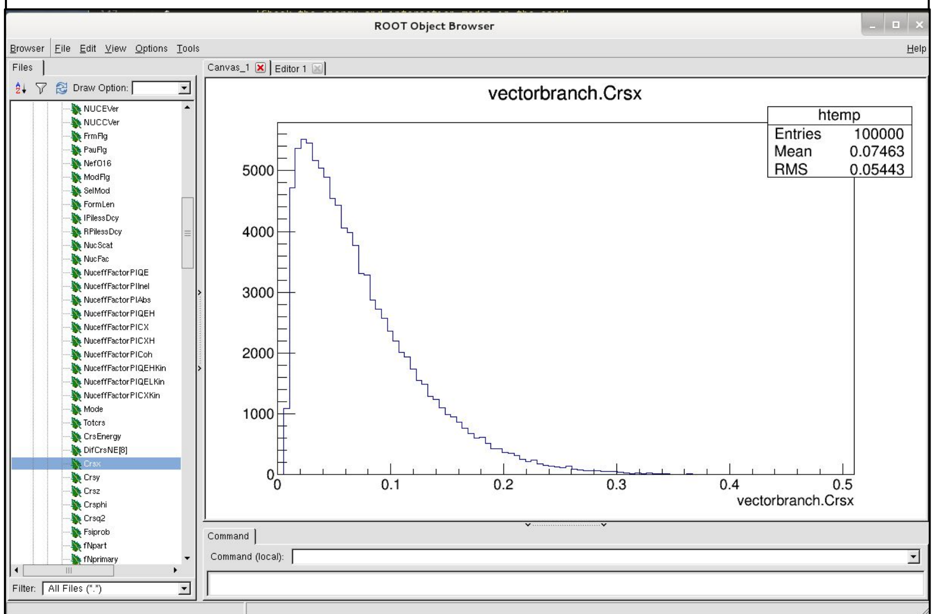
1. it reads the card file with `/neutcore/necard.F`
2. it fills the interaction model with `/neutcore/nefillmodel.F` (if coherent, should it be Rein-Sehgal or Berger-Sehgal?)
3. it creates the root output file
4. it reads input flux histogram (if there is one)
5. it starts the loop over the generated events
    a. set the vertex position
    b. set the neutrino direction
    c. set the neutrino energy
    d. draw the events rate (if there is an input flux histogram)
    e. generate events with `/neutsmpl/nevecgen.F`
    f. call `/neutcore/nevent.F` to compute the kinematics for each event

       … (can't go deeper as it becomes very specific to each model)

    g. consider other effects (radiative corrections, nucleon rescattering, ...)
6. it saves all the info in the ROOT output file.

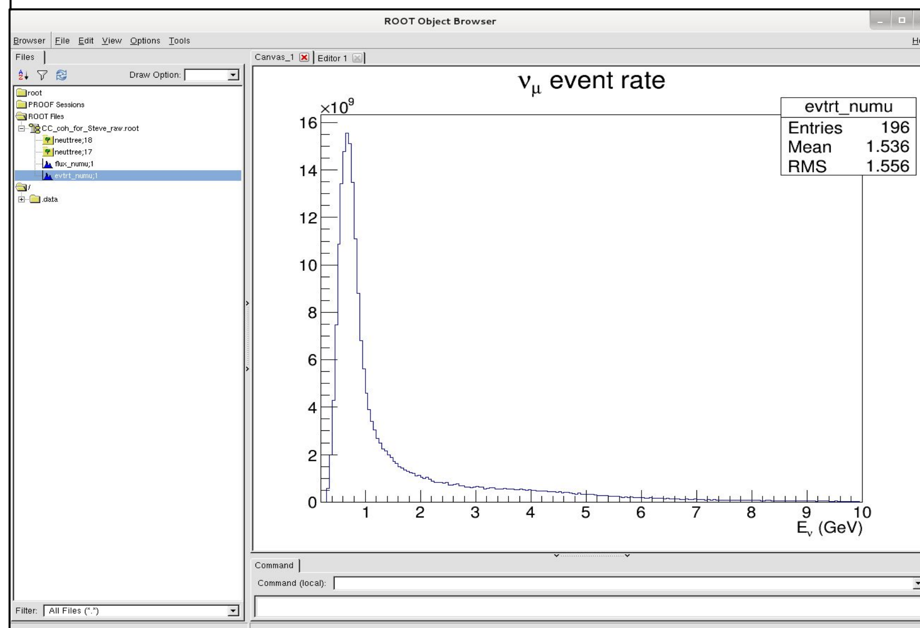
NEUT - Analysing the output file

The output file given by NEUT contains few values of the input parameters, few kinematics parameters such as x, y, and the total cross section Totcrs discussed earlier.



This file must be read by a simple ROOT macro where we will be able to extract all the information event by event.

If a flux file is given in input, you also have the event rate and the flux as a function of E_ν.



Dividing the event rate by the flux histogram will give you the XS as a function of the neutrino energy.

NEUT - Analysing the output file

However, it is more convenient to build your own root file from the NEUT output, where you can loop over all the generated events and add more specific variables. An example is given in /neutsmpl/chkreadneutroot.cc. I'll show here a modified version of the macros that should be available in the talk folder.

```
#include <iostream>
using namespace std;

void fill_histos(char *, char *, Double_t );

void make_histos()
{
    // Load NEUT libraries
    gSystem->Load("/data/pmartins/NEUT/neut_trunk/tags/neut_5.3.3/src/neutclass/neutvtx.so");
    gSystem->Load("/data/pmartins/NEUT/neut_trunk/tags/neut_5.3.3/src/neutclass/neutpart.so");
    gSystem->Load("/data/pmartins/NEUT/neut_trunk/tags/neut_5.3.3/src/neutclass/neutfsipart.so");
    gSystem->Load("/data/pmartins/NEUT/neut_trunk/tags/neut_5.3.3/src/neutclass/neutfsivert.so");
    gSystem->Load("/data/pmartins/NEUT/neut_trunk/tags/neut_5.3.3/src/neutclass/neutvect.so");
    // Fill your ROOT file based on the neutroot output
    fill_histos("output.root","my_new_file.root");
}

void fill_histos(char *in_fname, char *out_fname)
{
    // Open the file created by the ./neutroot neut.card command.
    TFile *f = new TFile(in_fname,"READ");
    if ( f == NULL ){
        cout << "Failed to open " << in_fname << endl;
        return;
    }
    // Link the branches of the file.
    TTree *tn = (TTree *)(f.Get("neuttree"));
    NeutVtx *nvtx = new NeutVtx();
    tn->SetBranchAddress("vertexbranch",&nvtx);
    NeutVect *nvect = new NeutVect();
    tn->SetBranchAddress("vectorbranch",&nvect);

    // Define the histos that will show amazing results (kinematic variables)
    TH1D *NEUT_pmu         = new TH1D("NEUT_pmu","muon momentum",   100,0.,2.);
    TH1D *NEUT_anglemu     = new TH1D("NEUT_anglemu","muon angle", 180,0.,180.);
    TH1D *NEUT_cosanglemu = new TH1D("NEUT_cosanglemu","cos muon angle", 50,-1.,1.);
```

Load NEUT libraries

Open output.root

Link the branches

Define your histos

NEUT - Analysing the output file

```
    // ************************************** //
    // ********* Loop over the entries ********* //
    // ************************************** //

    Double_t nevents = tn->GetEntries();
    std::cout<<"nevents = "<<nevents<<std::endl;
    for ( Int_t j = 0 ; j < nevents ; j++ ){
        tn->GetEntry(j);
        if (nvect->Mode == 16){ // select only CC coherent interaction
            for ( Int_t i = 0 ; i < nvect->Npart() ; i++ ){
                if((nvect->PartInfo(i))->fPID == 13){ // Oh Look ! There is a muon
                    double muon_nbr = i;
                }
                else if((nvect->PartInfo(i))->fPID == 211){ // And here, it's a pion, this is awsome !
                    double pion_nbr = i;
                }
            }
            // ----------------------
            // - Fill Muon variables -
            // ----------------------
            double e_mu = (nvect->PartInfo(muon_nbr))->fP.E()/1000.;    // Muon energy in GeV
            double p_mu = (nvect->PartInfo(muon_nbr))->fP.P()/1000.;    // Muon momentum in GeV
            double angle_mu = (nvect->PartInfo(0))->fP.Angle((nvect->PartInfo(muon_nbr))->fP.Vect()); // Muon angle
            double angle_mu = angle_mu / 3.1415926535 * 180.;
            double cos_mu = cos((nvect->PartInfo(0))->fP.Angle((nvect->PartInfo(muon_nbr))->fP.Vect()));
            NEUT_pmu->Fill(p_mu);
            NEUT_anglemu->Fill(angle_mu);
            NEUT_cosanglemu->Fill(cos_mu);
        }
    } // End of Loop over the entries

    // Create the output file with all the histos that we need.
    // The useful ones
    TFile *f2 = new TFile(out_fname,"RECREATE");
    if ( f2 == NULL ){
        cout << "Failed to (re)create " << out_fname << endl;
        return;
    }
```

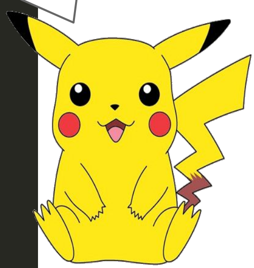
Loop over all entries

Fill your histos

Create your own file

NEUT - Analysing the output file

Instead of having your variable (muon momentum) as a function of the number of events, it would be better to normalise your histo in order to have the differential cross-section !

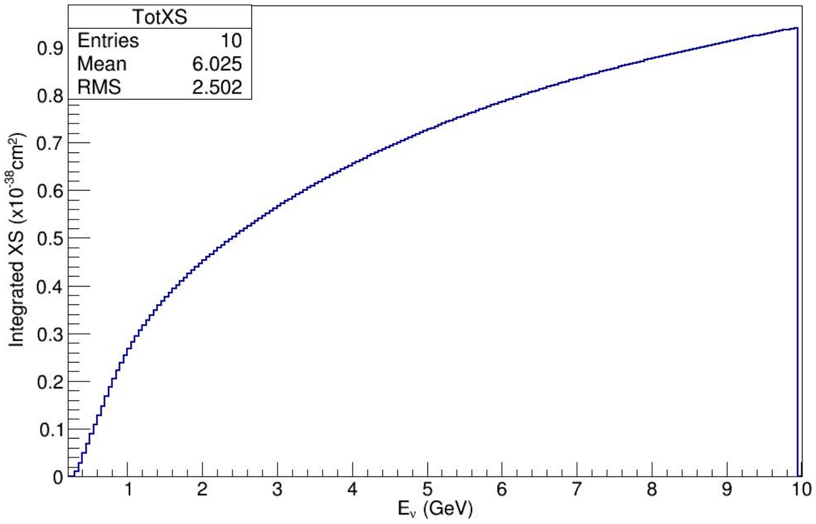


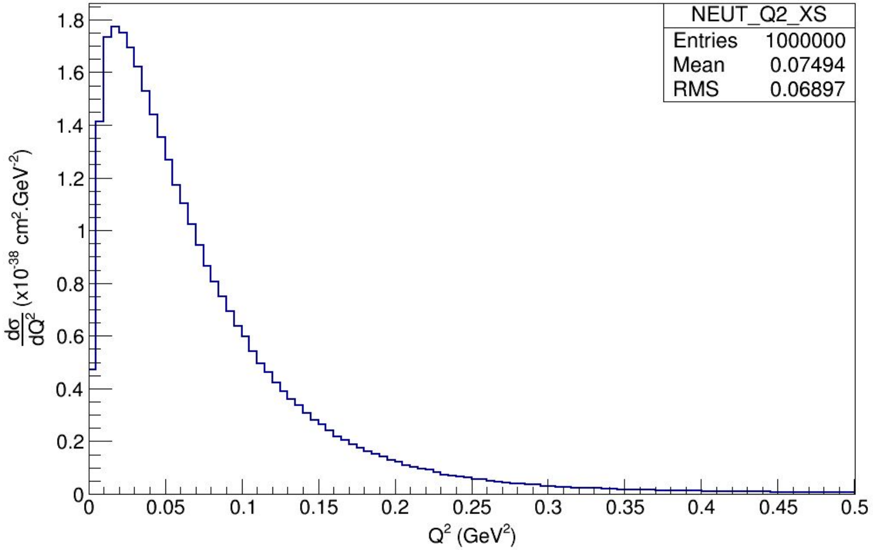
```
// ***************************
// ***** XS NORMALISATION *****
// ***************************

TH1D * h_flux  = (TH1D*)f->Get("flux_numu");
TH1D * h_evtrt = (TH1D*)f->Get("evtrt_numu");
double flux_numu   = h_flux->GetSumOfWeights();
double evtrt_numu  = h_evtrt->GetSumOfWeights();

// Calculate the normalisation factor.
double norm_rate = 12 * evtrt_numu/flux_numu;

h_flux->Write();
h_evtrt->SetName("TotXS");
h_evtrt->GetYaxis()->SetTitle("Integrated XS (x10^{-38}cm^{2})");
h_evtrt->GetYaxis()->CenterTitle();
h_evtrt->Divide(h_flux); // integrated XS per nucleons = event rate / flux
h_evtrt->Scale(12); // x12 because C has 12 nucleons
h_evtrt->Write();

// XS normalisation of NEUT for pion mom
TH1F * NEUT_pmu_XS  = (TH1F*)NEUT_pmu->Clone("NEUT_pmu_XS");
NEUT_pmu_XS->Scale(norm_rate*50/nevents); // x50 because of binning
NEUT_pmu_XS->GetYaxis()->SetTitle("#frac{d#sigma}{dp_{#mu^{-}}} (x10^{-38} cm^{2}.GeV^{-1})");
NEUT_pmu_XS->GetYaxis()->CenterTitle();
NEUT_pmu_XS->GetXaxis()->SetTitle("p_{#mu^{-}} (GeV)");
NEUT_pmu_XS->GetXaxis()->CenterTitle();
// Write everything ion the output file and close it
f2->Write();
f2->Close();
```

Get the total XS

Don't forget that the XS is per nucleon !

NEUT - Analysing the output file





Et Voila !