Tutorial GiBUU
Part B: Hands-On (neutrino init)

K. Gallmeister for the GiBUU group
Goethe-Universität, Frankfurt

GiBUU implementation

Hands On: Final state with neutrino init

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Bundesministerium für Bildung und Forschung

Sept 2019

BUU: Testparticle ansatz

$$\left[\partial_t + (\nabla_p H_i)\nabla_r - (\nabla_r H_i)\nabla_p\right] f_i(\vec{r}, t, \vec{p}) = C\left[f_i, f_j, \ldots\right]$$

- *idea:*
  approximate full phase-space density distribution by a sum of delta-functions

$$f\left(\vec{r}, t, \vec{p}\right) \sim \sum_{i=1}^{N_{\text{test}}} \delta\left(\vec{r} - \vec{r}_i(t)\right) \delta\left(\vec{p} - \vec{p}_i(t)\right)$$

- each delta-function represents one (test-)particle
  with a sharp position and momentum
- large number of test particles needed

Nuclear ground state

- density distribution: Woods-Saxon (or harm. Oscillator)
- particle momenta: ‘Local Thomas-Fermi approximation’

$$f_{(\mathrm{n,p})}(\vec{r},\vec{p}) = \Theta\left[p_{F(\mathrm{n,p})}(\vec{r}) - |\vec{p}|\right]$$

- Fermi-momentum:

$$p_{F(\mathrm{n,p})}(\vec{r}) = \left(3\pi^2\rho_{(\mathrm{n,p})}(\vec{r})\right)^{1/3}$$

- Fermi-energy:

$$E_{F(\mathrm{n,p})} = \sqrt{p^2_{F(\mathrm{n,p})} + m_N^2} + U_{(\mathrm{n,p})}(\vec{r},p_F)$$

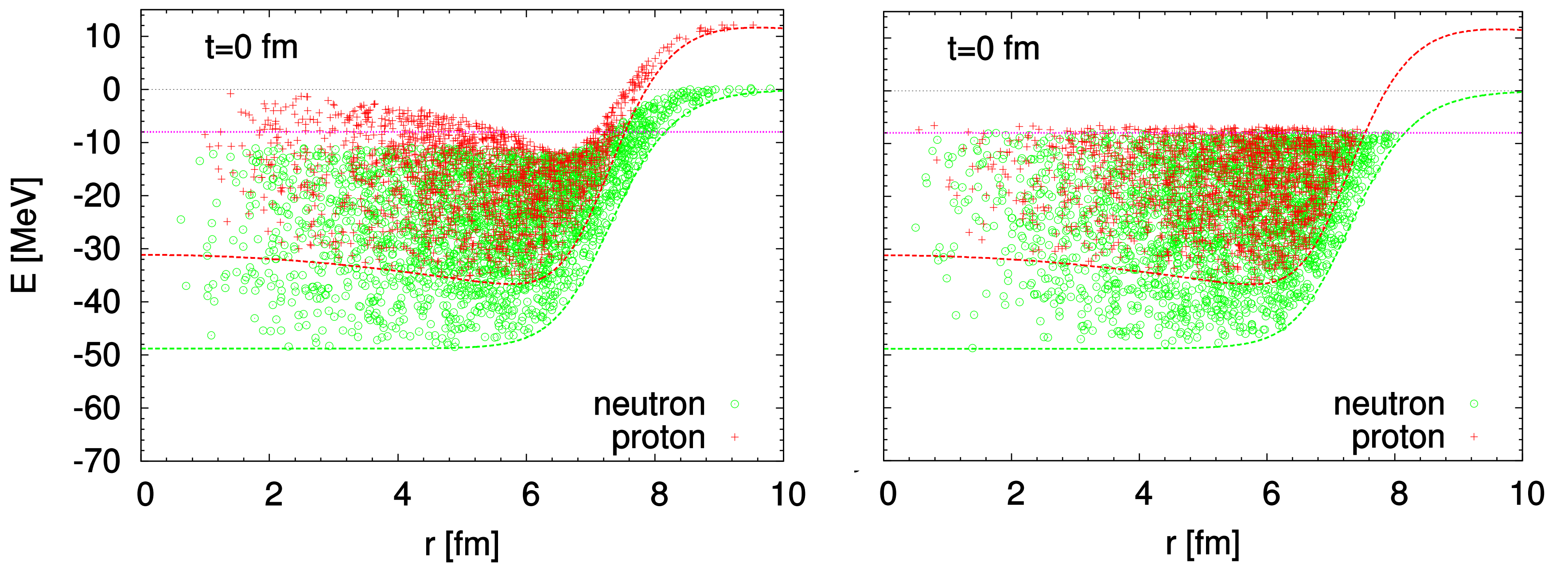
- no nuclear excitations
- neutrino energy > 50 MeV

“GiBUU is semiclassical”

Nuclear ground state

- improvement: ensure constant Fermi-Energy

non-mom.dep potential, asymmetry-term, Coulomb



- needs iteration for mom.dep potential
- important for QE-peak (Gallmeister, Mosel, Weil, PRC94 (2016) 035502)

Init

- in principle:
  1) initialize nucleons
  2) perform **one** initial elementary event on **one** nucleon
  3) propagate nucleons and final state particles
- correct, but ‘waste of time’
- *idea:*
  final state particles do not really disturb the nucleus
- 2 particle classes:
  - ‘real particles’
  - ‘perturbative particles’

Particle classes

- ‘real particles’
  - nucleons
  - may interact among each other
  - interaction products are again ‘real particles’
- ‘perturbative particles’
  - final state particles of initial event
  - may only interact with ‘real particles’
  - interaction products are again ‘perturbative particles’
- ‘real particles’ behave as if other particles are not there

Init with perturbative particles

- init
  1) initialize nucleons
  2) perform **one** initial elementary event on **every** nucleon
  3) propagate nucleons and final state particles

  - final states particles are ‘perturbative particles’
  - different final states do not interfere

- every final state particle gets a ‘perturbative weight’:
  - value: cross section of initial event
  - is inherited in every FSI
  - *for final spectra the ‘perturbative weights’ have to be added, not only the particle numbers*

Init with perturbative particles

- *idea:*

  simple workaround against oscillating ground states:
  freeze nucleon testparticles

- since nucleons are real particles, their interactions among each other should not influence final state particles

- advantage: computational time
- disadvantage: ???

The GiBUU website

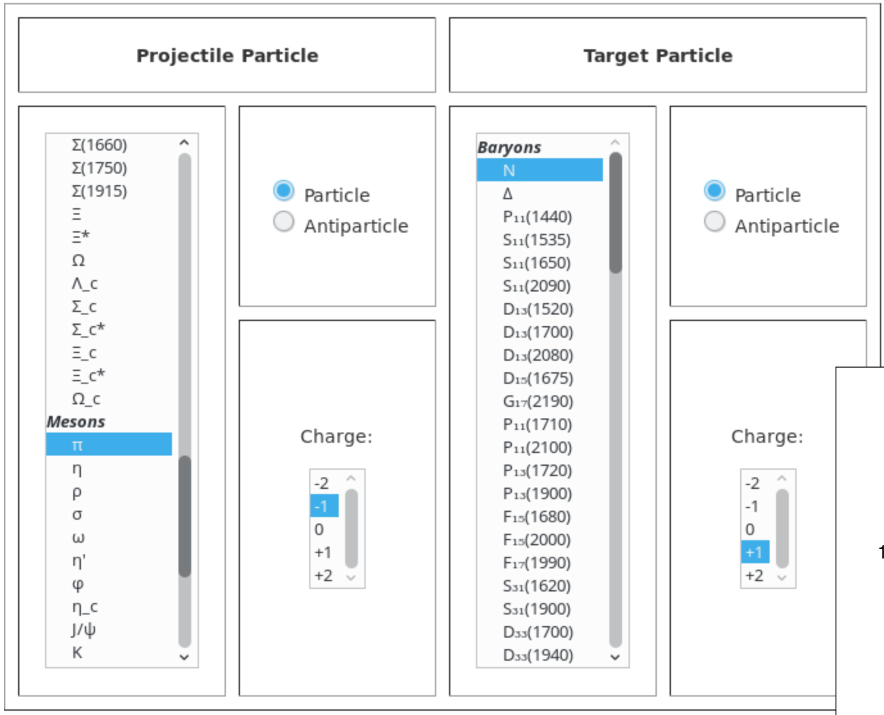
https://gibuu.hepforge.org

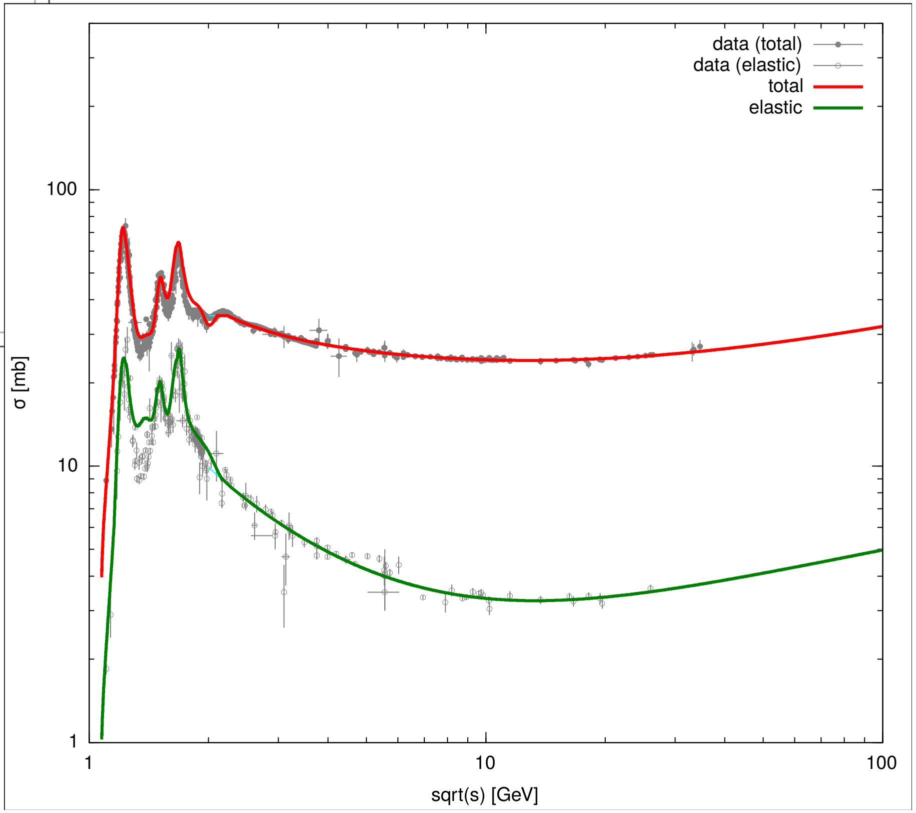
- central place for all information on GiBUU
- based on a wiki system (‘trac’)
- contains lots of information about the model and code
- documentation of input parameters, output files etc.
- source code viewer for svn repository
- timeline of news & changes

- cross section plotter (for hadronic interactions)

Cross section plotter

https://gibuu.hepforge.org/XSection/





Technical Prerequisites

- GiBUU runs on Linux, Mac, Windows
- Linux is preferred platform
- needed software tools:
  - ~~subversion (for code checkout)~~
  - GNU make
  - a Fortran compiler (e.g. gfortran 5.4)
  - perl
  - libbz2
  - (a running ROOT installation)

for output in ROOT format via RootTuple library: https://roottuple.hepforge.org/

  - see website for supported compilers
  - private observation: ifort generates fastest code

Getting the code

- ...via check-out from svn repository

- create a new dire

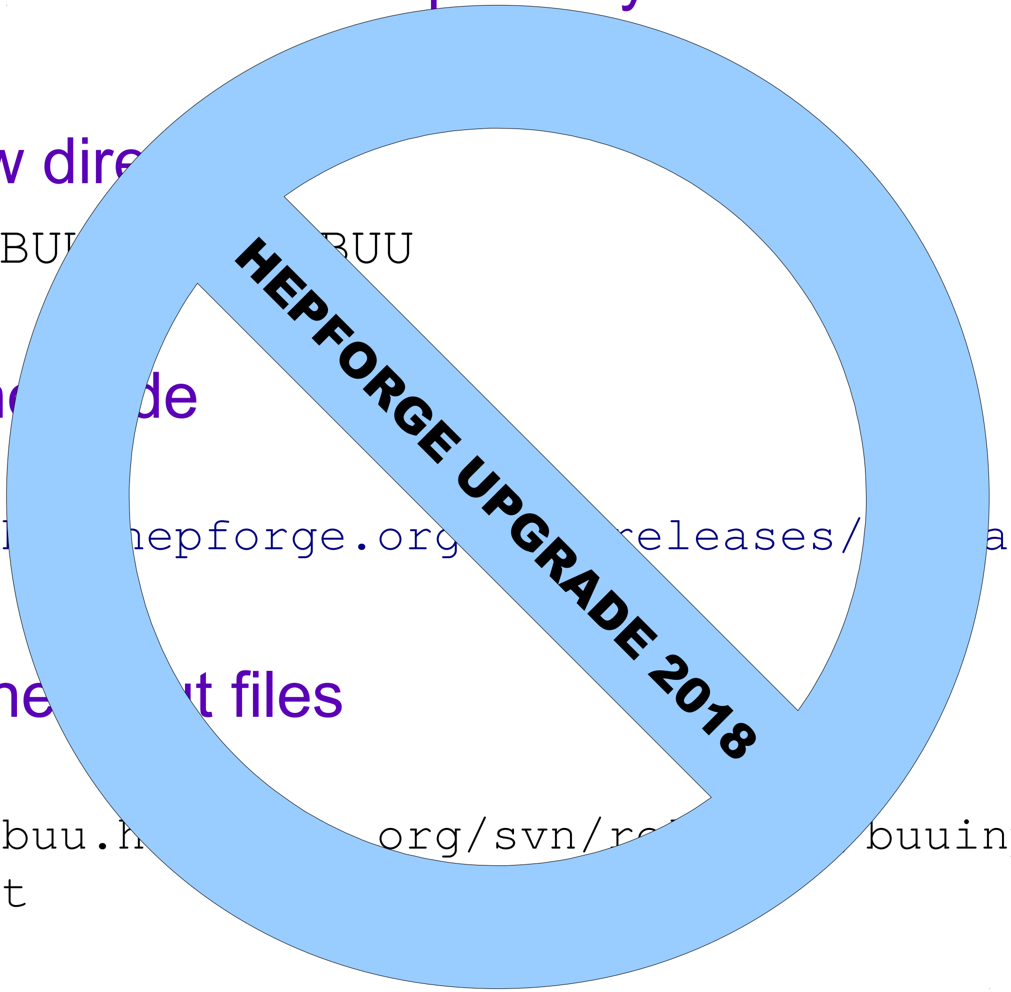
  mkdir GiBU BUU

- check-out th de

  svn co
  http://gi epforge.org releases/ ase2017

- check out the t files

  svn co
  http://gibuu.h org/svn/r buuinput2017
  ./buuinput



git access (GitHub) possible, but not really maintained

Getting the code

/trac/wiki/download

- ...via tar-balls

- create a new directory

```
mkdir GiBUU; cd GiBUU
```

- download the code

```
wget -content-disposition →
→https://gibuu.hepforge.org/downloads?f=release2019.tar.gz
tar -xzvf release2019.tar.gz
```

- download the input files

```
wget -content-disposition →
→https://gibuu.hepforge.org/downloads?f=buuinput2019.tar.gz
tar -xzvf buuinput2019.tar.gz
```

- (download RootTuple library)

```
wget -content-disposition →
→https://gibuu.hepforge.org/downloads?f=libraries2019_RootTuple.ta
tar -xzvf libraries2019_RootTuple.tar.gz
```

Getting the code

- ...via docker

- initiative by Luke Pickering:

  https://hub.docker.com/r/picker24/gibuu_2019

- … is this a way to go?

  (feedback/input very welcome!!!)

Compiling the code

- go to directory and make!

/trac/wiki/compiling

```
cd release2019; make
(cd release2019; make buildRootTuple; make withROOT=1)
```

- takes about 3 minutes on my laptop (one core)

- parallel make

```
make -j 4
```

- choosing a compiler

```
make FORT=gfortran-4.8
```

- no optimization

```
make MODE=opt0
```

- re-compile everything

```
make renew
```

if something went wrong...

SUCCESS: GiBUU.x generated.

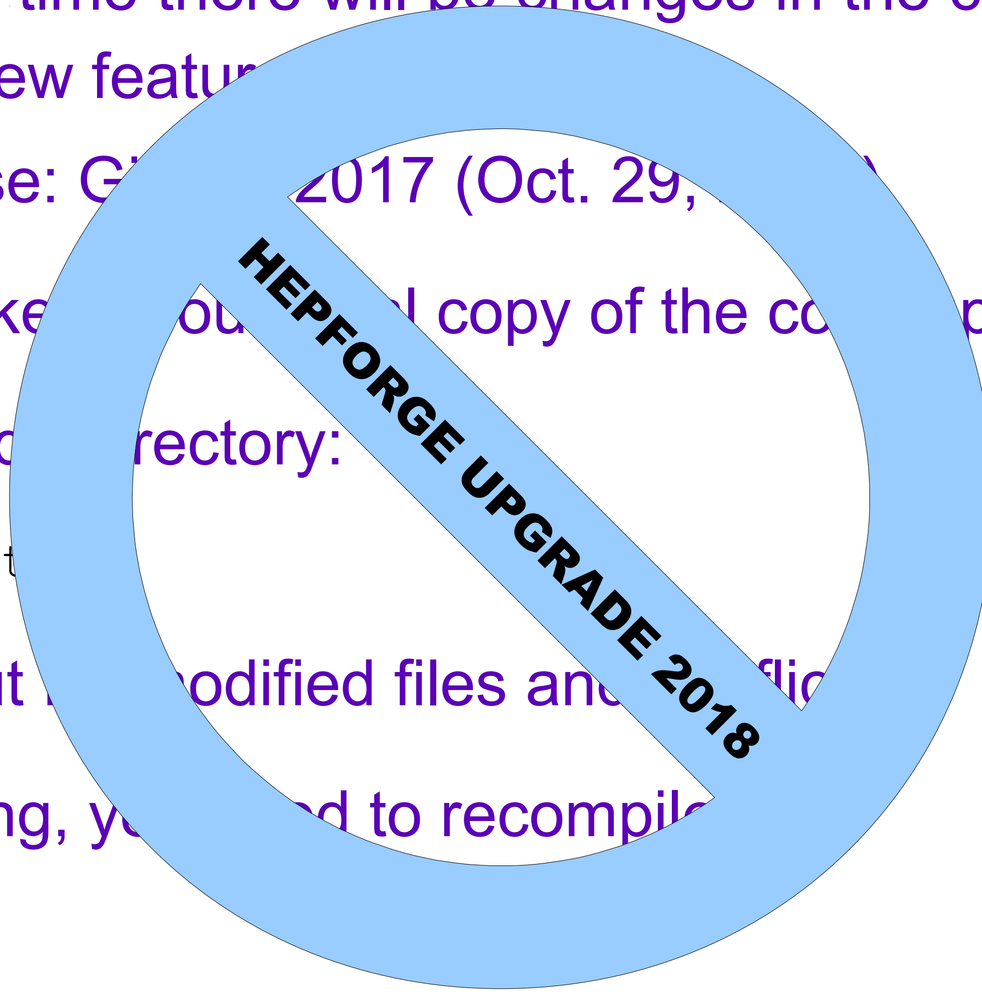
Updating the code via svn

- from time to time there will be changes in the code (bugfixes, new featur
- latest release: G 2017 (Oct. 29,)
- you should ke ou l copy of the co p to date
- do in the coc rectory:

```
svn updat
```

- check output f odified files anc flic
- after updating, y d to recompil

```
make
```



Running the code

- after successful compilation, there is the executable
  `./objects/GiBUU.x` (linked also `./testRun/GiBUU.x`)

- run the executable with input and output files

```
./GiBUU.x < input.job > log.txt
```

- either
  - run 'in-tree', i.e. in the directory testRun
    `cd testRun; ./GiBUU.x`
  - copy it somewhere else
  - use it from somewhere else with full path

  } recommended, since several output files are generated

- the file 'log.txt' will contain a log of GiBUU control & debug messages, physics output will be written to other files

Input parameters

- input via the Fortran way: **‘jobcard’**

  (= plain text file with data in some specific format)

  sample jobcards in `./testRun/jobCards`

- format: data in a ‘jobcard’ is grouped in **‘namelists’**

```
&namelist1
  switch1 = value1     ! some comment
  switch2 = value2     ! another comment
/

&namelist2
  switch3 = value3
  switch4 = value4
/
```

- capitalization (upper/lower case) does not matter

Input parameters

- there are a lot of input parameters!
- documented at website
  https://gibuu.hepforge.org/Documentation2019/code/robo_namelist.html
  https://gibuu.hepforge.org/Documentation2019/namelists.pdf

- most of them not relevant for beginners
- most of them have reasonable default values

- some relevant namelists for neutrino events:
  - 'input' (basics)
  - 'neutrino_induced'
  - 'target'
  - 'EventOutput' (producing particle output)
  - ...

The Namelist ‘input’

005_NeutrinoClean_T2K-numu.job

- the basic settings that need to be supplied

```
&input
      eventtype          =          5 ! neutrino interactions
      numEnsembles       =       1000
      numTimeSteps       =        100
      delta_T            =        0.2 ! time step size [fm]
      freezeRealParticles  = T
      localEnsemble      = T

      path_To_Input      = '/some/path/to/buuinput'
/
```

- ‘path_to_input’ must point to local path of buuinput directory

The Namelist ‘neutrino_induced’

005_NeutrinoClean_T2K-numu.job

- infos about the elementary neutrino event

```
&neutrino_induced
       process_ID      =  2 ! 2:CC, 3:NC, -2:antiCC, -3:antiNC
       flavor_ID       =  2 ! 1:electron, 2:muon, 3:tau

       nuXsectionMode  = 16 ! 16: EXP_dSigmaMC
       nuExp           =  9 !  9: T2K-2.5kA-ND280

!   subprocesses to take into account:
       includeQE       = T
       includeDELTA    = T
       includeRES      = T
       include1pi      = T
       includeDIS      = T
       include2p2hQE   = T
       include2p2hDelta= F
       include2pi      = F
/
```

The Namelist 'neutrino_induced'

- nuXsectionMode: (required input)

$0 =$ integratedSigma: E_ν

$1 =$ dSigmadCosThetadElepton: E_ν, $\cos\theta$, E_{lepton}

$2 =$ dSigmadQsdElepton: E_ν, Q^2, E_{lepton}

$3 =$ dSigmadQs: E_ν, Q^2

$4 =$ dSigmadCosTheta: E_ν, $\cos\theta$

$5 =$ dSigmadElepton: E_ν, E_{lepton}

$6 =$ **dSigmaMC**: E_ν

$7 =$ dSigmadW: E_ν, W

$+10$ for taking experimental flux into account

The Namelist ‘neutrino_induced’

- nuExp:

1 MiniBooNE neutrino flux (in neutrino mode] positive polarity)

2 ANL

3 K2K

4 BNL

5 MiniBooNE anti-neutrino flux (in antineutrino mode] negative polarity)

6 MINOS muon-neutrino in neutrino mode

7 MINOS muon-antineutrino in neutrino mode

8 NOVA neutrino (medium energy NuMI, 14 mrad off-axis), FD

9 T2K neutrino off-axix 2.5 degrees (at ND280 detector)

10 *uniform distribution* from $E_{\mathrm{flux,min}}$ to $E_{\mathrm{flux,max}}$

11 MINOS muon-neutrino in antineutrino mode

12 MINOS muon-antineutrino in antineutrino mode

The Namelist 'neutrino_induced'

- nuExp: (cnt'd)

13 MINERvA muon neutrino, old flux

14 MINERvA muon antineutrino, old flux

15 LBNF/DUNE neutrino in neutrino mode

16 LBNF/DUNE antineutrino in antineutrino mode

17 LBNO neutrino in neutrino mode

18 NOMAD

19 BNB nue BNB= Booster Neutrino Beam

20 BNB nuebar

21 BNB numu

22 BNB numubar

23 NOvA ND

24 T2K on axis

25 MINERvA, 2016 flux

99 user defined flux

The Namelist ‘target’ etc.

005_NeutrinoClean_T2K-numu.job

- infos about the nucleus as target

```
&target
      Target_A =  12
      Target_Z =   6
!       ReAdjustForConstBinding = T
/
```

- analytic density treatment

```
&initDensity
      densitySwitch = 2           ! 2=analytic
/

&initPauli
      pauliSwitch = 2             ! 2=analytic
/
```

Analysis strategies

- **'on-line'** analysis directly inside GiBUU
  - direct analysis of desired quantity during the simulation
  - directly produce histograms etc.
  - no intermediate particle output
  - advantage: access to all internal information
  - disadvantage: needs recompilation for changes
  - ***mainly only for developers***

- **'off-line'** analysis
  - output all particles/events
  - LesHouches/ROOT format, proprietary format
  - analysis may be changed after simulation run
  - disadvantage: may produce large amount of data

- GiBUU tends to be 'silent' by default

1) on-line analysis

- inclusive output

```
&neutrino_induced
        ...
        printAbsorptionXS = T
        ...
/
```

- final state analysis

```
&neutrinoAnalysis
       XSection_analysis     = T ! for multiplicities
       detailed_diff_output = T ! differential cross sections
       ...
/
```

+ 4 other namelists

- ~80 parameters

produced output:
~ 2500 files / 650 MB

2a) off-line analysis

- neutrino events:
  due to historical reasons also proprietary event format

```
&neutrino_induced
        ...
        outputEvents = T
        ...
/
```

writes file '**FinalEvents.dat**':

1: Run
2: Event
3: ID 4: Charge
5: perweight
6-8: position(1:3)
9-12: momentum(0:3)
13: history
14: production_ID *(1=QE, 2=Delta, …, 34=2p2h)*
15: Enu

includes:
- outgoing lepton
- hit nucleon (for docu purpose)

sensitive to input parameters defining cuts !!!

2b) The Namelist 'EventOutput'

005_NeutrinoClean_T2K-numu.job

- generate particle output

```
&EventOutput
      WritePerturbativeParticles = T
      WriteRealParticles = F
!      EventFormat = 1 ! 1=LesHouches
/
```

- output only for perturbative particles
- file(s) generated 'EventOutput.Pert.*.lhe'

- possible formats:

  1 = LesHouches http://arxiv.org/abs/hep-ph/0609017

  2 = OSCAR 2013 http://phy.duke.edu/~jeb65/oscar2013

  3 = Shanghai 2014 http://www.physics.sjtu.edu.cn/hic2014/node/12

  4 = ROOT

Output format 'Les Houches'

- XML-like event format
- named after a town in France
- basic structure:

arXiv:hep-ph/0609017v1

```
<LesHouchesEvents version="1.0">
<header>
  ...
</header>
<init>
  ...
</init>
<event>
  ...
</event>
... (any number of <event> blocks can follow) ...
```

Output format 'Les Houches' (2)

```
<event>
     1      0   5.06E-07   0.00E+00   0.00E+00   0.00E+00
    2212 0 0 0 0 0   0.024   0.028   0.308   1.010   0.958E-01 0. 9.
# 5 1 5.06E-07 0.61 0. 0. 0.61 0.54 0.09 -8.03E-04 0.52 0.97 0.11 -3
</event>
```

- **line 1:** N=number of lines, 0, weight, boring zeros
- **following: N lines**, representing one particle each
  columns: 1 = ID (PDG code), 7-9 = $p_{x,y,z}$, 10 = E, 11 = mass
- **last line:** comment
  'magic number' 5 = special info for neutrino events
  eventtype, weight, momLepIn(0:3), momLepOut(0:3), momNuc(0:3)
- eventtype: 1 = QE, 2-31 = resonance, 32 = 1pi, ...

Analysis using 'Les Houches'

```
#!/usr/bin/env python3

from pylhef import *

data = read("EventOutput.Pert.00000001.lhe")

sigmatot = 0
sigmapi  = 0
for ev in data.events:
    sigmatot += ev.weight
    for part in ev.particles:
        if (part.id==211):
            sigmapi += ev.weight

print("sigmatot = ",sigmatot)
print("sigmapi  = ",sigmapi)
```

https://github.com/jrvidal/pylhef

sum up the weights!

```
sigmatot =  0.7087205200450297
sigmapi  =  0.17053788931443
```

$$\sigma(\nu A)/A = 0.71 \cdot 10^{-18}\mathrm{cm}^2$$

$$\sigma(\nu A \to \pi^+ X)/A = 0.17 \cdot 10^{-18}\mathrm{cm}^2$$

Output format ‘ROOT’

- same info as in LesHouches files
- needs:
  - working ROOT installation
  - building RootTuple library (included in GiBUU)
  - linking GiBUU with RootTuple
  - setting ‘EventFormat = 4’ in Jobcard
- … (I have no experience with ROOT; input/feedback welcome!!!)
- work in progress:
  patch to additionally write positional info

‘Tuning’

- modify cross sections of different channels/eventtypes:
  multiply perweights with a factor

- 2p2h on basis of structure functions:
  easy changeable

- implement own processes/particles:
  difficult

- … ?

- Event output
  please contact me!!!