# *Tutorial GiBUU*
# *Part B: Hands-On (HiLepton init)*

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

GiBUU implementation

Hands On: Final state with HiLepton init

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Bundesministerium für Bildung und Forschung

Sept 2019

# BUU: Testparticle ansatz

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

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

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

- 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:

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

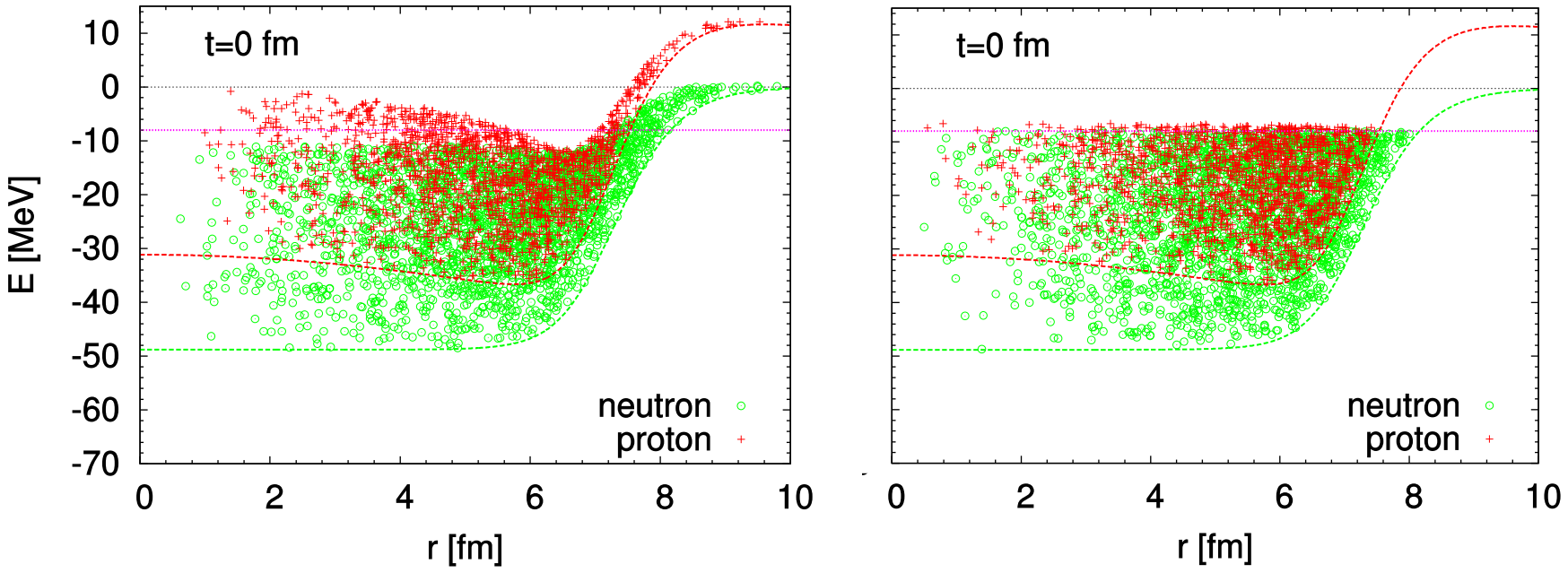
"GiBUU is semiclassical"

- no nuclear excitations
- energy/momentum transfer > 50 MeV

# 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
- total energy, total baryon number, etc. not conserved!

# 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

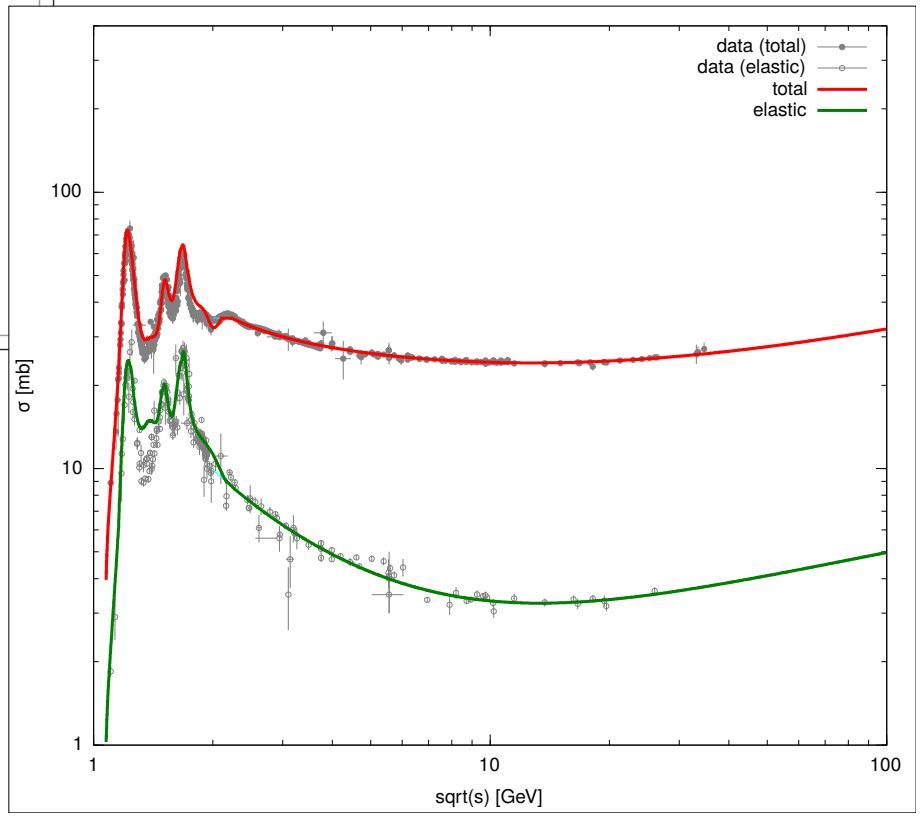
https://gibuu.hepforge.org

- 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/re              buuinput2017
./buuinput
```


git access (GitHub) possible, but not really maintained

# Getting the code

- ...via tar-balls

/trac/wiki/download

- 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
```

SUCCESS: GiBUU.x generated.

if something went wrong...

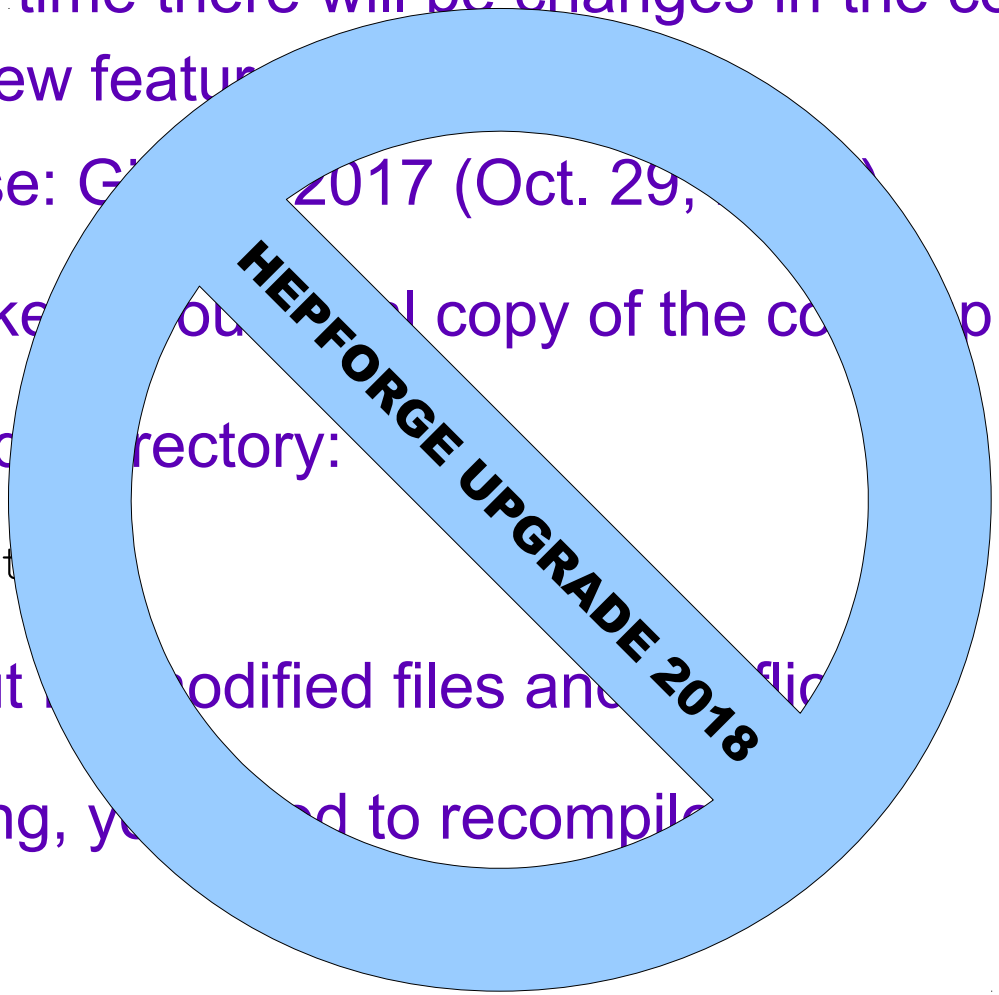
# Updating the code via svn

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

```
svn updat
```

- check output odified files an 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)
  - ‘HiLeptonNucleus’
  - ‘target’
  - ‘EventOutput’ (producing particle output)
  - ...

# The Namelist 'input'

014_HiLepton_A.job

- the basic settings that need to be supplied

```
&input
      eventtype        =          14 ! HiLepton
      numEnsembles     =        1000
      numTimeSteps     =         200
      delta_T          = 0.1          ! time step size

      length_perturbative = 2000 ! okay up to ..., Xe

      num_runs_SameEnergy=1  ! number of runs per energy

      localEnsemble = .TRUE.
      freezeRealParticles = .TRUE.

      path_To_Input    = '~/GiBUU/buuinput'
/
```

- 'path_to_input' must point to local path of buuinput directory

# The Namelist 'HiLeptonNucleus'

014_HiLepton_A.job

- infos about the elementary electron event

```
&HiLeptonNucleus
!       shadow   = F
!       DoStatistics = T ! additional output

       iExperiment= 5 ! JLAB,    5GeV
       iDetector = -1 ! use default

/
```

# The Namelist 'HiLeptonNucleus'

- iExperiment:

**0: no experiment/fixed kinematics**
1: Hermes, 27GeV, D,N,Kr
2: Hermes, 27GeV, Ne
3: Hermes, 27GeV, H
4: JLAB, 12GeV
5: JLAB, 5GeV
6: EMC, 100GeV
7: EMC, 120GeV
8: EMC, 200GeV
9: EMC, 280GeV
10: Hermes, 12GeV
11: Hermes, 27GeV, arXiv:0704.3270
12: Mainz, Yoon: Ebeam=1.5GeV
13: Hermes, 27GeV, arXiv:0704.3712 (pT-broadening)
14: JLAB, 5GeV, rho0 experiment
15: JLAB, 4GeV, rho0 experiment
16: EIC, E_e and E_A given explicit (3+30,11+30,4+100)
**17: no detector, total cross section, Ebeam**
18: E665, 470GeV
19: CLAS/JLAB, 12GeV RunGroupA optimized 10.6 GeV
20: CLAS/JLAB, 12GeV RunGroupA theoterical

defines the electron kinematics:
- acceptance cuts
- efficiency

hadronic cuts for online analysis

# The Namelist ‘HiLeptonNucleus’

- `iDetector:`

  **-1 : use default**
  0 : no detector
  1 : HERMES, full efficiency
  2 : EMC, full efficiency
  3 : CLAS, only cuts (th_e=12°..50°, th_hadron=6°..143°)
  4 : CLAS, full efficiency + cuts as for 5GeV
  5 : CLAS, electron: cuts (th_e=12°..50°),
  hadrons: efficiency+cuts as for 5GeV
  90 : full acceptance

- additional cuts:

```
&HiLeptonNucleus
...
!       user_numin = ... ! GeV
!       user_numax = ... ! GeV
!       user_costmin = ...
!       user_costmax = ...
!       user_smin = ... ! GeV^2
!       user_qsqmin = ... ! GeV^2
!       user_qsqmax = ... ! GeV^2
...
/
```

# Fixed electron kinematics

- fix kinematics by electronic variables:

```
&HiLeptonNucleus
...
      iExperiment=17 ! no detector, total cross section, Ebeam
!      iDetector = 0
      Ebeam = ... ! GeV
...
/
```

calculates total cross section

(cuts by `user_…` as above)

# Fixed photon kinematics

- fix the kinematics by photonic variables:

```
&HiLeptonNucleus
...
      iExperiment= 0 ! no experiment/fixed kinematics
!      iDetector = 0
...
/

&HiPhotonKinematics
     nu = ...
     Q2 = ...
     eps = ...
     srts = ...
     W = ...
     xBj = ...
     Ebeam = ...
/
```

exactly 3 of them have to be given

# EIC kinematics

- will be boosted to fixed target kinematics

```
&HiLeptonNucleus
...
      iExperiment=16 ! EIC
!      iDetector = 0
      EIC_Ee=3    ! GeV, energy electron
      EIC_EA=30   ! GeV, energy nucleus
...
/
```

# The Namelist ‘target’ etc.

014_HiLepton_A.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
/
```

# Potentials

014_HiLepton_A.job

- for high energies:

```
&baryonPotential
      noPerturbativePotential = T ! perturbative baryons feel no potential
/

&mesonPotential
      noPerturbativePotential = T ! perturbative mesons feel no potential
/
```

# 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

- mainly useful for developers
- see code/analysis/HiLeptonAnalysis.f90 for details

```
&HiLepton_Analysis
!       DoLeptonKinematics = T
!       DoHadronKinematics = T
!       DoTimes            = T
!       DoOutChannels = T
!       DoInvMasses   = T
!       DoFindRho0    = T
!       DoClasie      = T
!       DoMorrow      = T

!       DoClassifyFirst = T
/
```

# 2) off-line analysis: The Namelist 'EventOutput'

014_HiLepton_A.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>
     3      0  1.222E+1  0.000E+0  0.000E+0  0.000E+0
    2112     0     0     0     0     0  1.827E-1 -1.468E-1  2.975E+0  3.128E+0  9.380E-1 0. 9.
    -211     0     0     0     0     0 -1.045E-1  1.092E-2  1.015E+0  1.030E+0  1.380E-1 0. 9.
    -211     0     0     0     0     0  4.420E-2  8.971E-2  3.135E-2  1.732E-1  1.380E-1 0. 9.
# 14    3.520E+0   1.808E+0   4.946E-1   1.708E+0    2004
</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’ 14 = special info for HiLepton events
  nu, Q2, eps, phiLepton, eventtype

- eventtype: 2001 = Pythia VMD, …, 2004 = Pythia DIS, …

# 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 =  30628.30051
sigmapi  =  21461.13987
```

$$\sigma(e^- A)/A = 15.314\,\mu\mathrm{b}$$

$$\sigma(e^- A \to \pi^+ X)/A = 10.731\,\mu\mathrm{b}$$

$$\frac{1}{n_{\mathrm{Ens}} \cdot A}$$

# 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*
- implement own processes/particles:
  *difficult*
- … ?
- Event output
  *please contact me!!!*