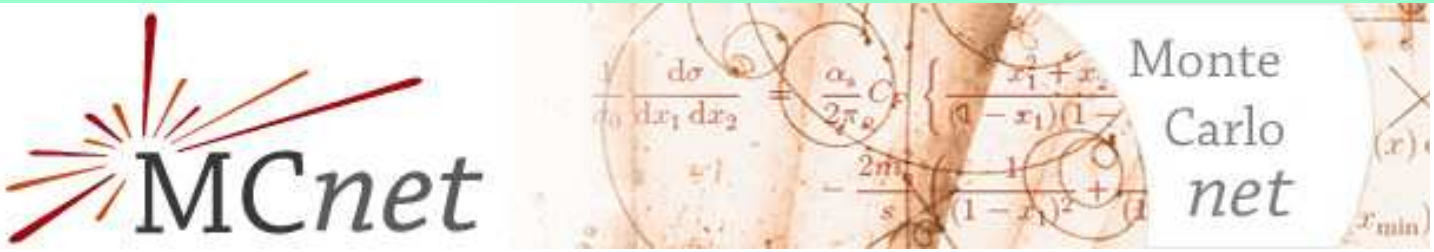
Rivet
Validation of Experiment and Theory



Lars Sonnenschein

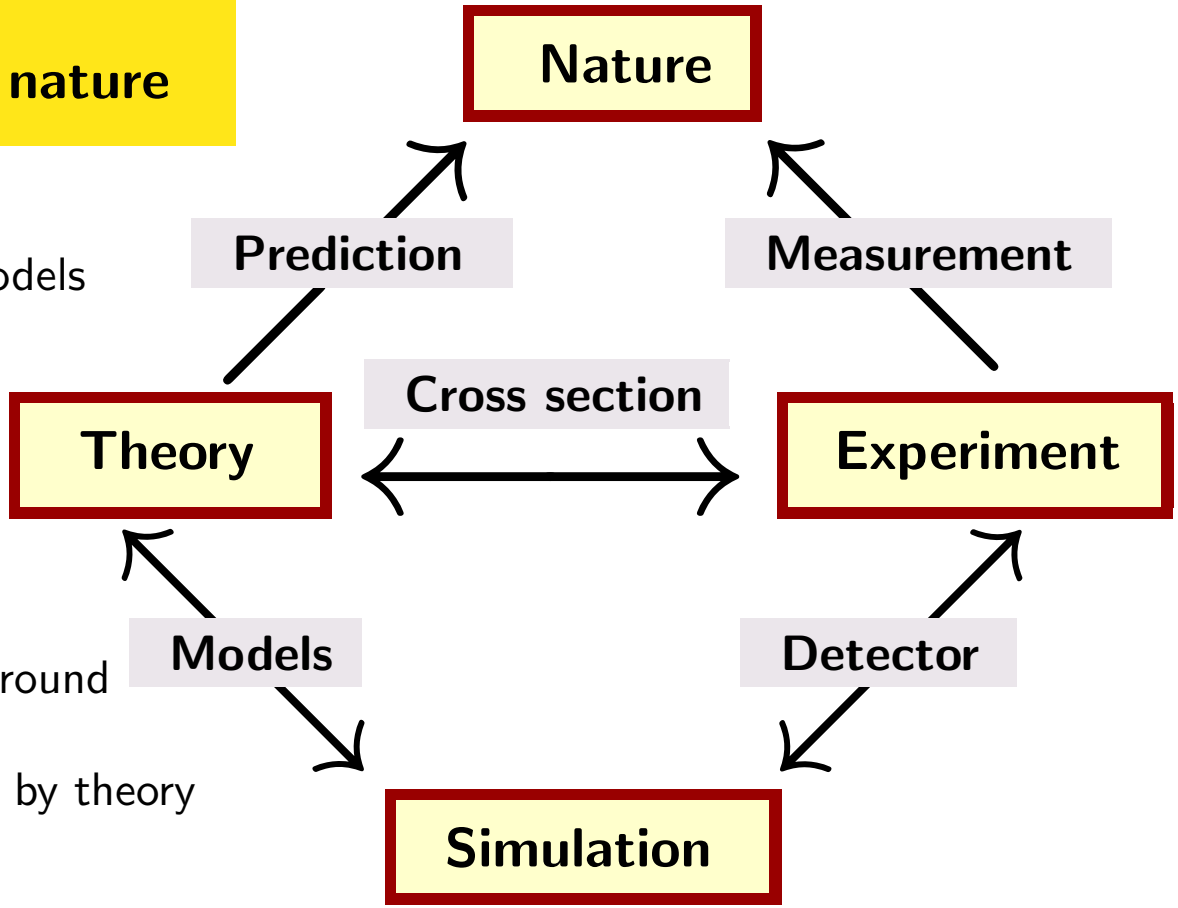
- Introduction
- Basics
- Conclusions

- Practical exercises
- Details
- The bigger picture

This work has been supported by a Marie Curie Early Stage Research Training Fellowship of the European Community's Sixth Framework Programme under contract number MRTN-CT-2006-035606, by LPNHE Paris and the *Commissariat à l'Energie Atomique* and CNRS/*Institut National de Physique Nucléaire et de Physique des Particules*, France
and by the HEPtools EU Marie Curie Research Training Network under the contract number MRTN-CT-2006-035505.

Introduction

The ultimate goal:
A better understanding of nature



- Theory needs input from experiment
  - Verification/Falsification of concurrent models
  - Description of nature
    being probed by experiment

- Experiment needs input from theory
  - Predictions for observables
  - Understanding of processes and rates
  - Discriminating instrumental effects/background
    from (new) physics
  - Improved understanding of nature, guided by theory

- Improving description of nature
  - Recursive interplay between experiment and theory
  - Focussing in this talk on the validation of MC event generators

Understanding (verification/validation/optimization/improvement) of **Monte Carlo** event generation/**Simulation** is **crucial!**

Introduction

- Rivet runs analyses on generated events
- Real data comes from HepData database (already existing)
- Simulated observables can be compared to published measurements

MC Event Generator Parameters

- Theory makes predictions to very few fixed orders (LO, NLO) plus resummation of radiation
- More or less phenomenological models are needed for comparison with measurements
- Models are implemented in MC event generators, they contain phenomenological parameters:
- Parton shower termination parameters: $p_{\perp\min}$, $m_{\min}$
- Hadronisation: Lund string and cluster fragmentation parameters: string function parameters, mass
- Underlying event: Primordial $k_\perp$, Color reconnection parameters
- Parton Distribution Functions (PDF's)
- Models need to be validated/adjusted using real data!

Real Data: HepData Database

- Archive of published HEP data from the last 30 years
  - Almost exclusively data which is corrected for detector effects
- Focus on cross sections and similar distributions
  $\longrightarrow$ Complementary to PDG
- Legacy database is being upgraded from FORTRAN accessed BDB to modern relational database by CEDAR
- Available at `http://projects.hepforge.org/hepdata`

HepData Database Web portal

Available at `http://durpdg.dur.ac.uk/hepdata`

Durham University — Science & Technology Facilities Council

HEPDATA: REACTION DATA Database

...containing numerical values of HEP scattering data such as total and differential cross sections, fragmentation functions, structure functions, and polarisation measurements, from a wide range of experiments. It is compiled by the **Durham Database Group (UK)** with help from the COMPAS group (Russia,) and is updated at regular intervals.

Reaction Database HELP · Full User Guide · Use register to tell us who you are, or feedback to send us any commments, suggestions or complaints.

Standard (Keyword) Search Method

Enter search command:

irn 5992206

then: Submit Search **or** Clear **or ask for** HELP

Search sysntax: keyword {op} value {boolean keyword {op} value} {...

where "op" is **=, >, <, >=, <=,** (the default is =).

and "boolean" is **AND, OR, and NOT.**

{} indicates optional elements.

Example searches:

```
reac = gamma gamma/
reac = p p --> p p and obs = dsig/dt
exp desy-hera/ and obs f2
auth smith or auth jones (records with authors either smith or jones)
de ua1 and year 1992
re e+ e- --> lambda/ and obs sig
re pbar p --> jet/ and plab > 100
```

Note(1): the "/" at the end of values performs a right truncated search.

Note(2): searches are case insensitive.

Note(3): even identical keywords must be repeated in complex searches.

Keywords to use (select for specific help):

[AUTH] [REF] [YEAR] [REAC] [FSP] [BEAM] [TARG]

Other Methods of Search the Database

- **Form Interface**
  A fill-in form for making simple queries of the data base.
- **Easy search method**
  Step-by-step through a search using a series of menus from initial states to final states and observables.

Data Reviews

Compilations of selected sub-sets of data organized in an easy to locate format.

- Structure Functions in DIS
- Single Photon Productionin Hadronic Interactions
- Two-Photon Reactions leading to Hadron Final States
- Drell-Yan cross sections
- Inclusive particle production data in e+e- Interactions
- Hadronic Total Cross Seciton (R) in e+e- Interactions
- Low Energy Neutrino Cross Sections

These Data Reviews are published in the IoP's, Journal of Physics G - Nuclear and Particle Physics. Electronic versions of the reviews can be obtained through the relevant IoP links above.

HepData Database Web portal

Available at `http://durpdg.dur.ac.uk/hepdata`

Back to the REACTION Database or HEPDATA top page

Database: **HEPDATA REACTION DATABASE**
Search Command: **irn 5992206**
Result: **1** documents found

1) **ABAZOV 05** - Experiment **FNAL-0823** - Detector/Collaboration: **D0**
Published in **PRL 94,221801** (2005)

Display the Full Data Record **Display the Table Index** **Show SLAC/HEP Entry**

Fermilab-Tevatron. Measurement of the normalized Delta PHI differential distributions of inclusive dijet production in PBAR P interactions at centre-of-mass energy 1960 GeV. Delta PHI is the azimuthal angle difference between the two jets of highest transverse momentum where in the tables below PT(P=3) > PT(P=4). The dijet definition uses the iterative midpoint cone algorithm with Rcone=0.7 and f=0.5 where f is the parameter for overlap treatment and the E-scheme for recombination of particles into jets. The data come from a sample of inclusive dijet events in the central rapidity region corresponding to an integrated luminsoity of 150pb-1.

Numerical values supplied by M Wobisch.
(RED 4678) (all kumacs) (all numbers)

- Rivet comes with the data,
  you don't need to access the HepData database for already implemented analyses
- If you are publishing author of a measurement,
  please **do** remember to send your data to the HepData database.
- Remember also that **if** your measurement is **corrected** for detector effects it will be **useful for ever!**

- Basics

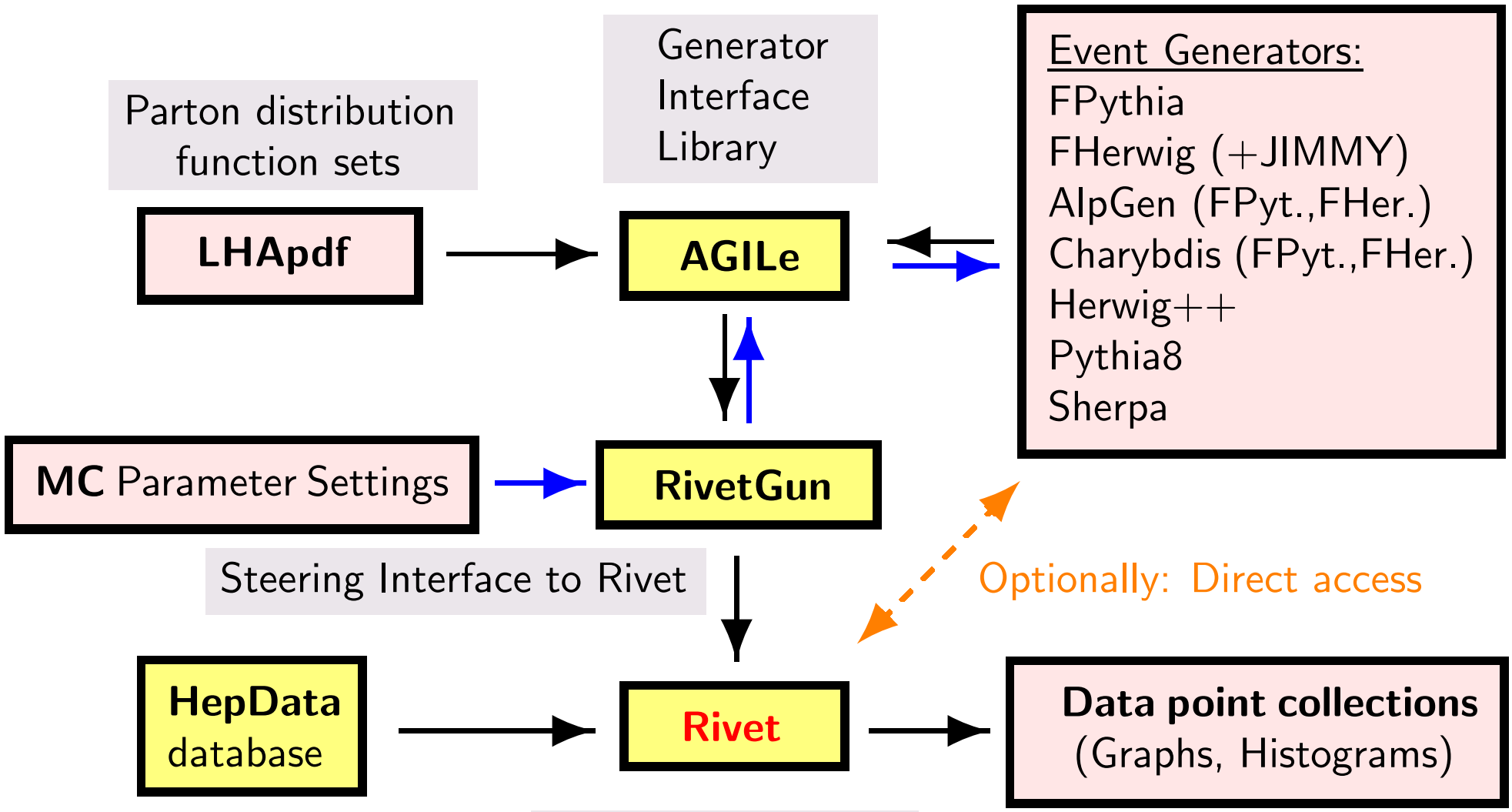
Validation of Experiment and Theory with Rivet

- **Why Rivet?**
- **Goal:** To provide analyses which match exactly publications
- Phenomenologists spend an enormous amount of time to reproduce published data analyses in all details:
  - Jet algorithm details/how exactly applied
  - Publication might seem unambiguous at the time of writing, not so later on ...
- $\Rightarrow$ Authors of published (corrected) analyses should implement it into Rivet at time of publication
  - Only in this way exact reproduction is guaranteed

Validation of Experiment and Theory with Rivet

- Rivet is an Object Oriented C++ replacement for FORTRAN HZTool
- Combination of tools, analysis handler and analyses
- Structure is based on auto-cached Projections acting on HepMC events
- Analysis functions make use of projections to determine selection criteria and observables
- Histogramming etc. via AIDA interfaces
- AGILe/RivetGun replaces FORTRAN HZSteer
- Web portal at `http://projects.hepforge.org/rivet`

Rivet dependency flow chart



Getting Rivet

- Dependencies: HepMC, FastJet, LHAPDF, MC event generators, GSL
- Download via http://www.hepforge.org/downloads/
- Development in public SVN on HepForge
- Easy way of installation: use bootstrap script: http://projects.hepforge.org/rivet
- This downloads dependencies, builds and installs them in the right order.

(Rivet has been tested on Linux and MaC OS)

Using rivet-bootstrap

(bash shell example)

- `$ ./rivet-bootstrap $PWD/local`
- `$ export LD_LIBRARY_PATH=$PWD/local/lib:$LD_LIBRARY_PATH`
- `$ export PATH=$PWD/local/bin:$PATH`
- `$ which rivetgun`
  `/localscratch/lars/cedar/local/bin/rivetgun`
- `rivetgun -h`

Building and Installing Rivet

- Rivet (and other CEDAR packages) are built with GNU autotools and libtool
- (If downloaded from SVN, `autoreconf -i`)
- `./configure --prefix=/prefix/path`
- `make`
- `make install`
- That's it!

Rivet Projections and Analyses

- What is a Projection?
- Determination of particle subsets and of event/particle properties
- Basic selection of particles
  $\Rightarrow$ Standard operations, common to various analyses are cached
- Determination of more complex quantities, e.g. jets:
  HZTOOL did run the identical $k_\perp$ jet algorithm on same Final State particles in 15 different analyses $\longrightarrow$ inefficient
- Separated from Analyses
  - Efficiency
  - Modularisation
    (new analyses can profit from existing Projections)

Rivet Projections

Observables are computed through predefined projections:

- FinalState, VetoedFS, ChargedFS, HadronicFS
- Thrust, Sphericity, Hemisphere Masses and Broadenings
- FastJet $k_\perp$ and cone jets as plugin, D0 Run II cone
- Multiplicities, Track Jets, Jet Shapes, ...
- Beam paricles, Primary and Secondary vertices

Repository of projections will grow with number of implemented analyses

Rivet Analyses

Published analyses already implemented:

- ALEPH_1991_S2435284: Charged particle multiplicity
- DELPHI_1996_S3430090: Event shapes
- CDF_2001_S4751469: Field & Stuart, Underlying event
- D0_2001_S4674421: Differential W/Z boson cross section
- D0_2004_S5992206: Azimuthal dijet decorrelations
- OPAL_2004_S6132243: Event shapes
- CDF_2005_S6217184: Jet Shapes
- CDF_2006_S6653332: $Z + b$ jet production
- CDF_2007_S7057202: Inclusive $k_\perp$ jet cross section
- H1_1995_S3167097: Energy flow in DIS
- ZEUS_2001_S4815815: Dijet photoproduction (used for p.d.f. fit)
- ExampleAnalysis (demo), ExampleTree (ROOT demo)

- External user analysis plugging mechanism available
- Histogram autobooking (Simulation inherits from data format)

RivetGun

Rivet steering interface:

- Dynamic loading $\Rightarrow$ multiple generators of same brand but different versions can be swapped at runtime
- Uniform object oriented C++ interface to various event generators:
  - FHerwig(+Jimmy), FPythia
  - (+ AlpGen and Charybdis)
  - Pythia8, (Herwig++, Sherpa)

Rivet/Gun at work

`rivetgun` steers event generators and runs Rivet

- Generating events and running Rivet analyses:

```
rivetgun -g FPythia:6411 -n 10000
-a D0_2001_S4674421
--beam1 PROTON --mom1 980.
--beam2 ANTIPROTON --mom2 980.
-P fpythia.params
```

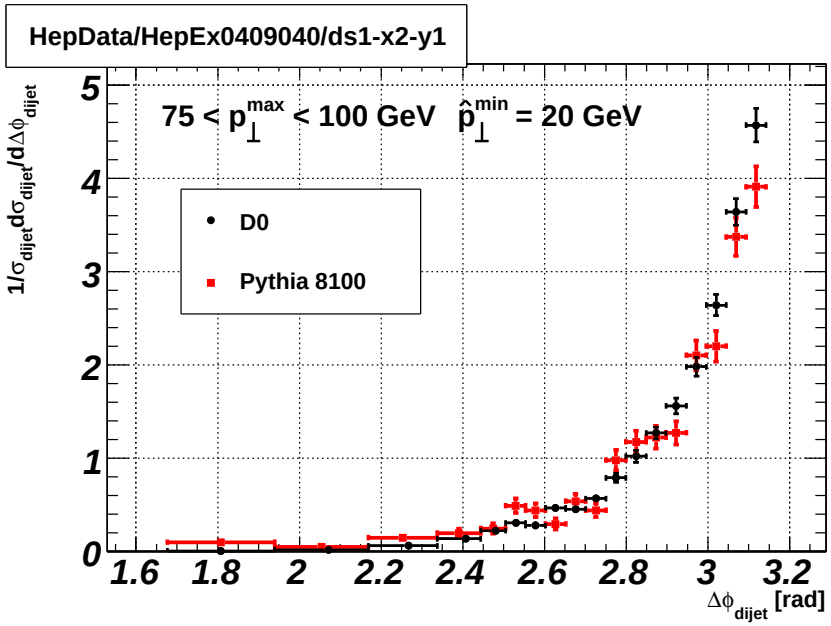
- Use `rivetgun` option `--histotype ROOT`
  and analyze with Root
  `root -l Rivet.root`
  `TGraph's` are stored `(TGraphAsymmErrors)`
  $\Rightarrow$ use Draw option `A` to get axes (e.g. tg$->$Draw("AP"))

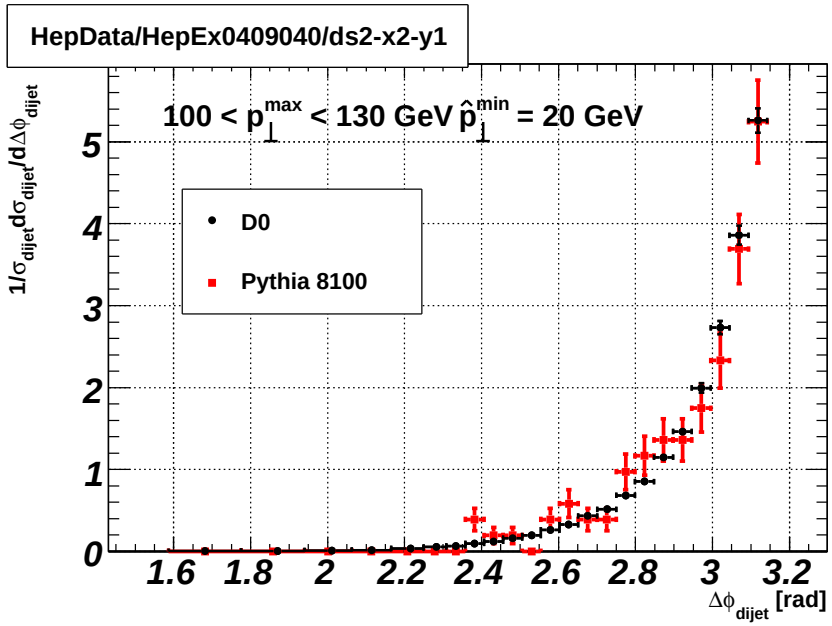
- As default AIDA format files are produced:
  Rivet.aida

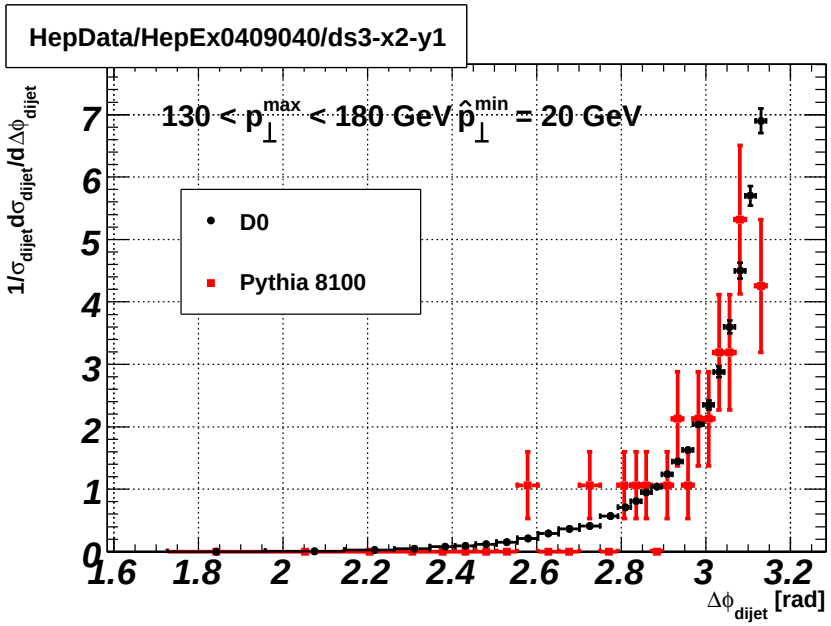
Further RivetGun flags

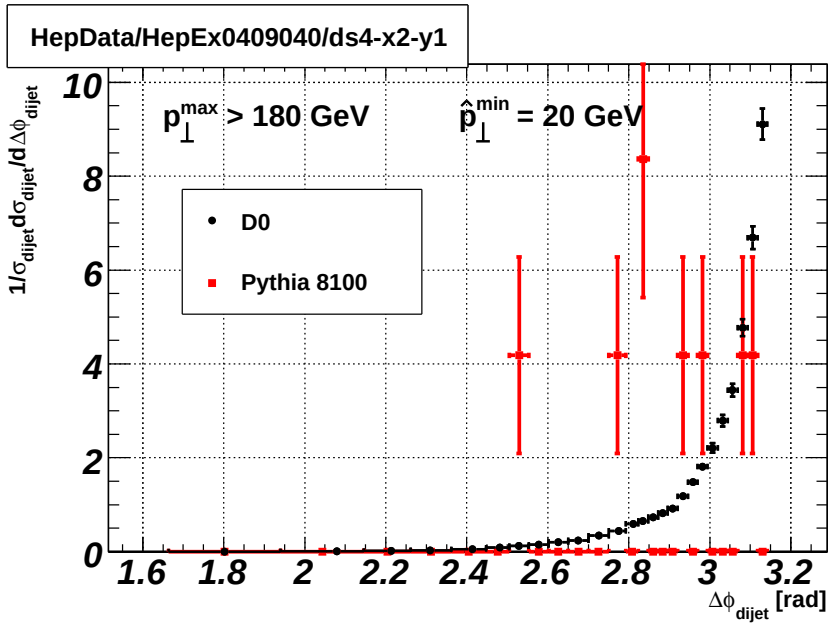
- `-h`: help, including available analyses and generators
- `-P param_file`: Event generator parameters read in from text file
- `-p key=val`: Specifying a parameter on the command line
- `-H name`: choose output histo file base name (without extension)
- `-o evtfile`: write out HepMC events in ASCII format to file (via `IO_GenEvt`)
- `-l namespace:LEVEL`: control logging level

Rivet Analysis D0_2004_S5992206: Azimuthal dijet decorrelations

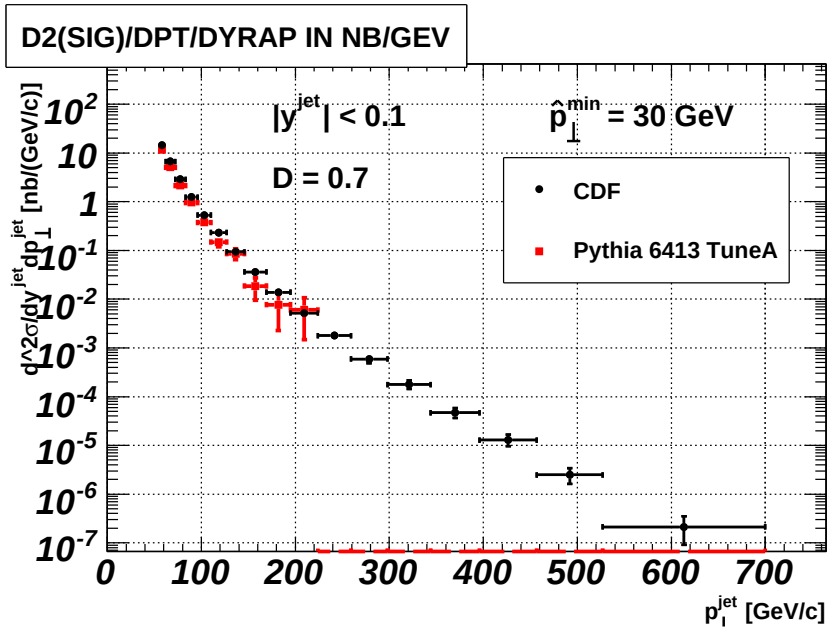


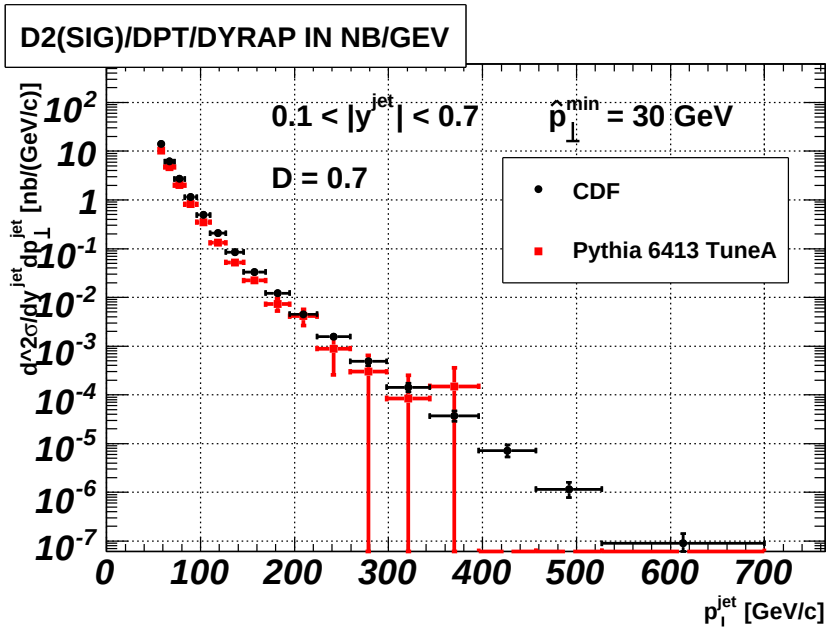


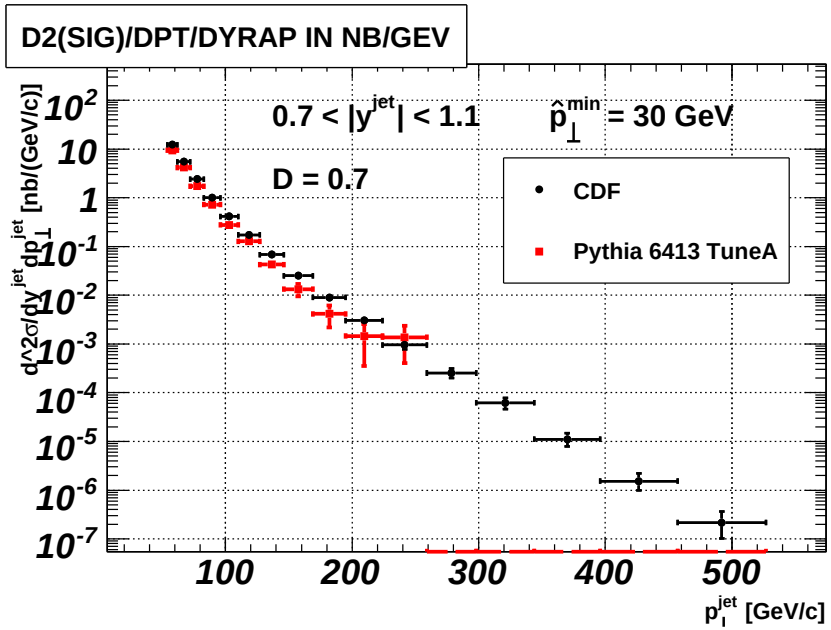


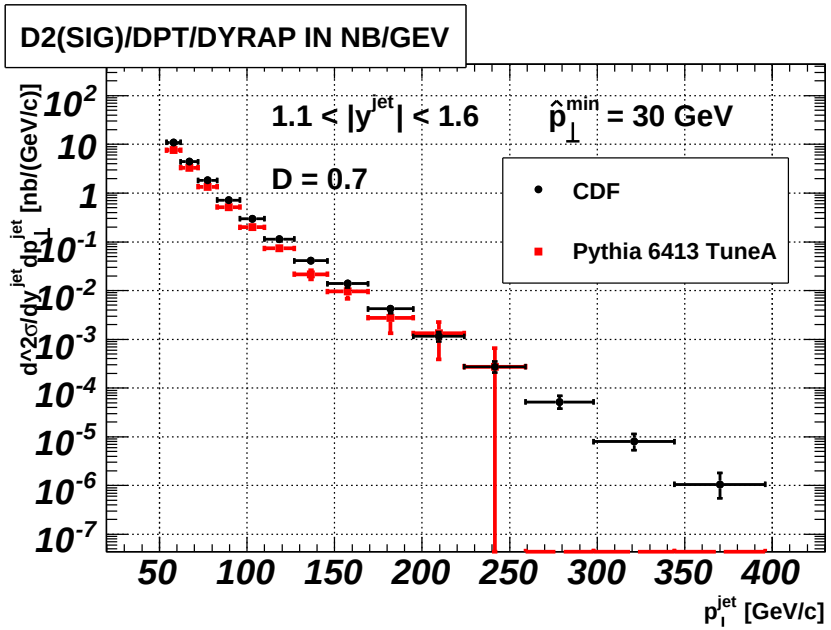


Rivet Analysis: CDF_2007_S7057202: Inclusive $k_\perp$ jet cross section



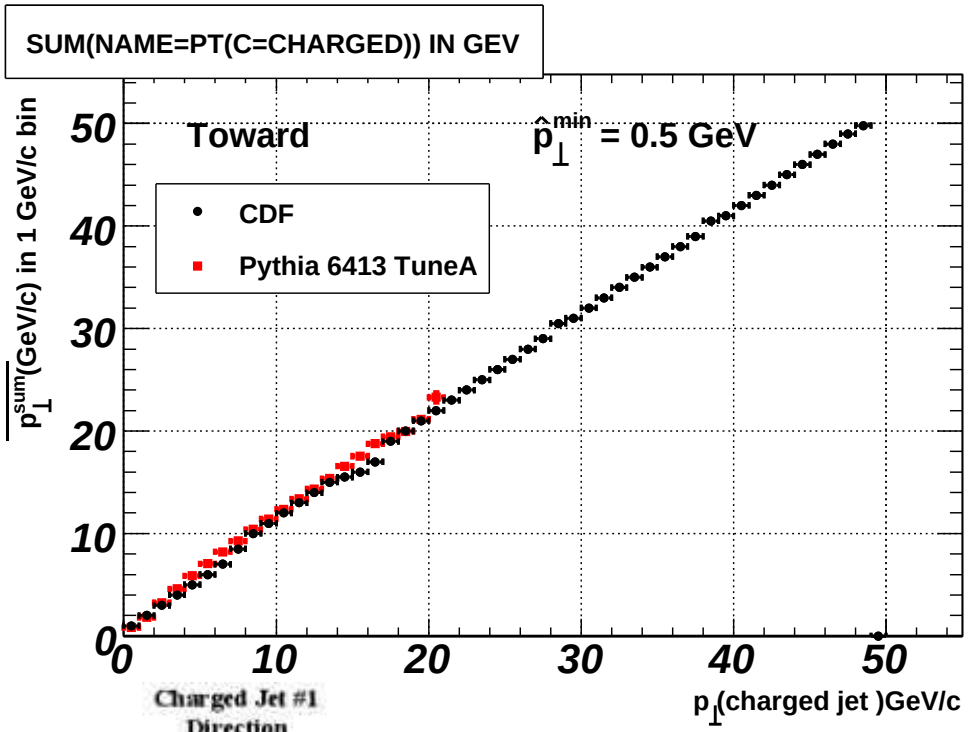


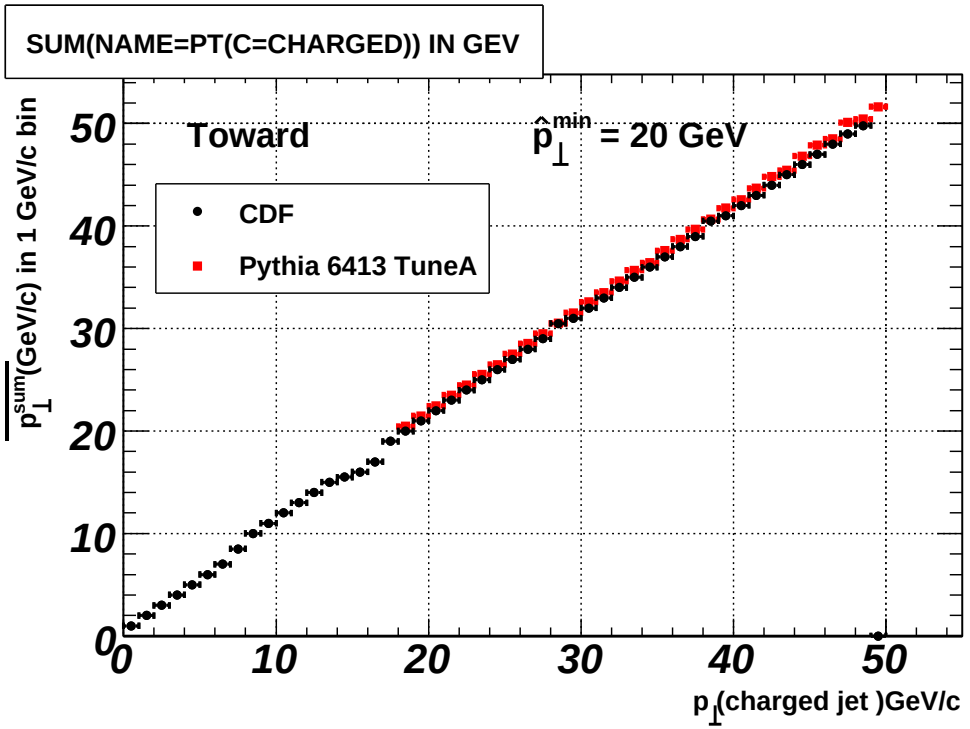


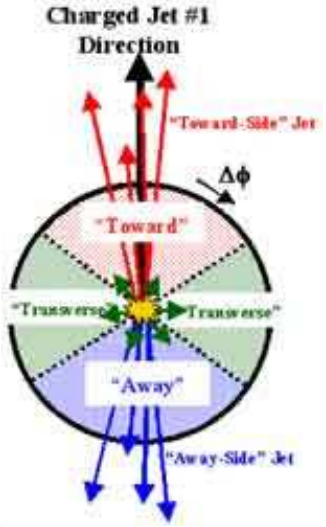


Rivet Analysis: CDF_2001_S4751469: Field & Stuart, Underlying event

Toward region



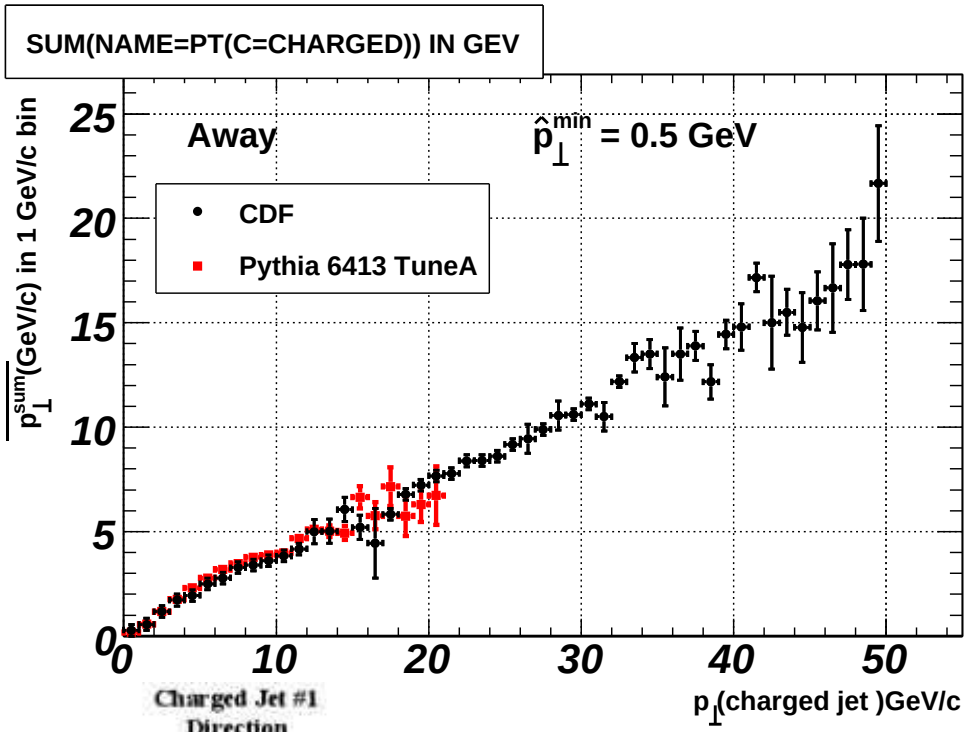


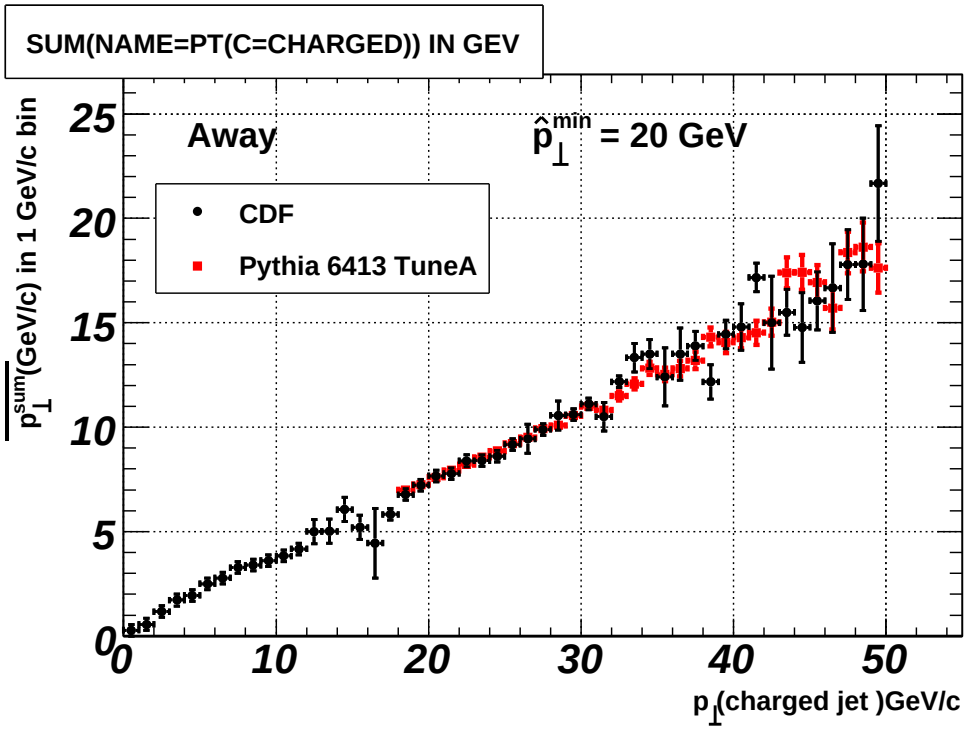


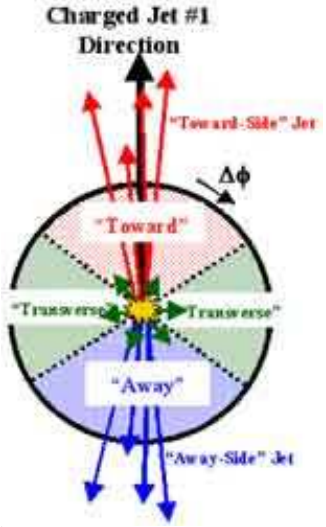
Need to launch different $\hat{p}_\perp^{\min}$ runs to get left and right side of histogram reasonably filled

Rivet Analysis: CDF_2001_S4751469: Field & Stuart, Underlying event

Away region



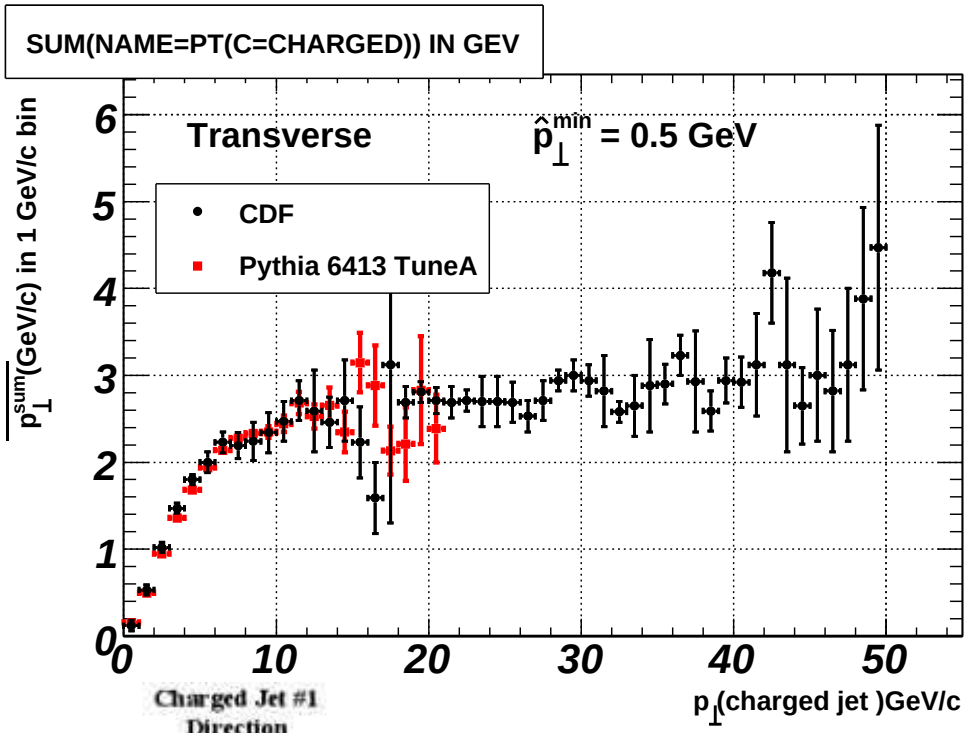


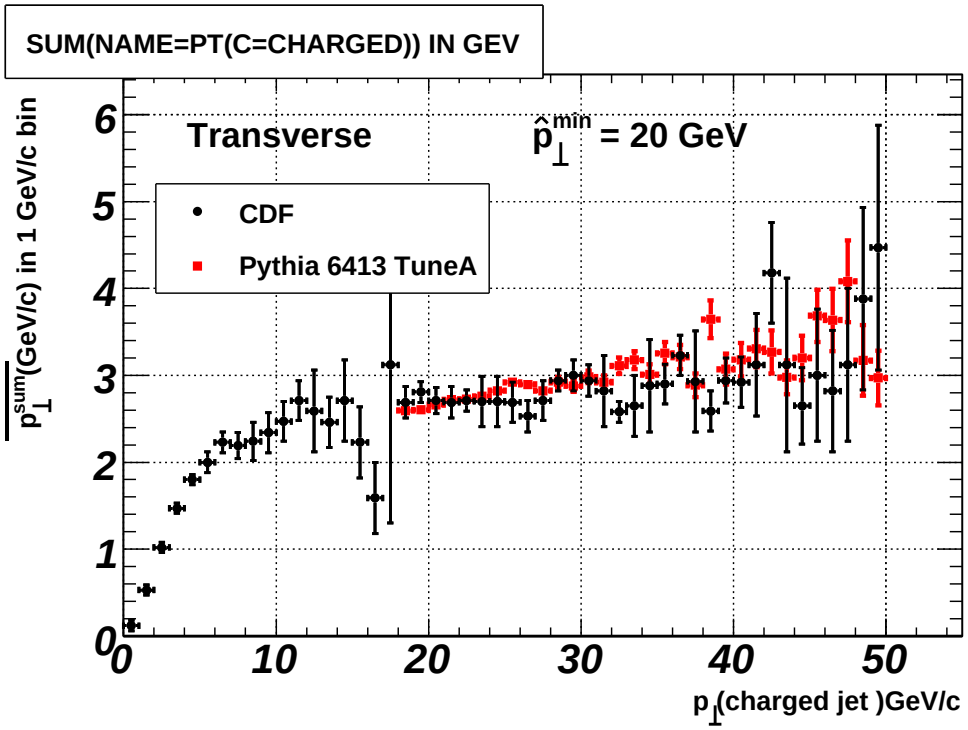


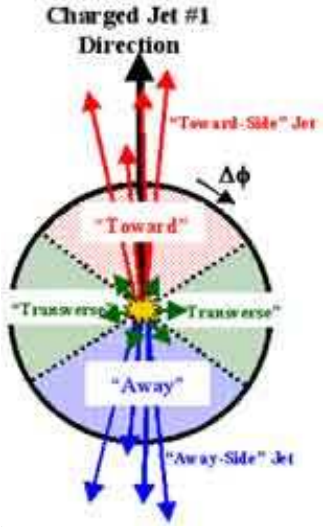
Need to launch different $\hat{p}_\perp^{\min}$ runs to get left and right side of histogram reasonably filled

Rivet Analysis: CDF_2001_S4751469: Field & Stuart, Underlying event

Transverse region







Need to launch different $\hat{p}_\perp^{\min}$ runs to get left and right side of histogram reasonably filled

Important for experimental measurements

- Authors of analyses: Correct your data for detector effects (acceptance /efficiency/background) particle level/hadronic final state not further!
- If corrected it can be always compared to event generators
- Else your analysis will be obsolete sooner or later (typically rather soon)
- Present and past collider Centre of Mass Energies provide unique points of operation
- Matrix of correlated errors are provided by QCD group analyses. This information can not be recovered from published plots and is therefore extremely important to be documented, too!
- Event generator authors (Herwig++, Pythia8, Sherpa, ...) appreciate very much corrected LEP analyses. Hadronisation corrections are larger than detector corrections $\Rightarrow$ Constraining fragmentation models. The most important ones are already in Rivet.
- Constraints are needed for further generator development. The more the better. You will benefit from it in the next iteration!

Conclusions

- Rivet provides a framework for validation
  of experiment and theory
  - Authors should implement their publication results ASAP
    (corrected for detector effects)

- Rivet provides also
  - a sophisticated analysis framework
  - with a rich repository of projections/predefined observables
  - uniform interface to various MC event generators
  - uniform interface to different jet algorithms through FastJet

- Practical Exercises

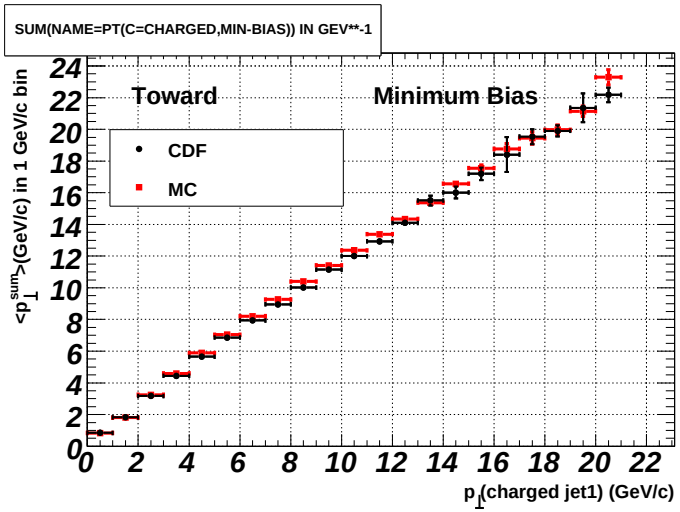
CDF Underlying event analysis

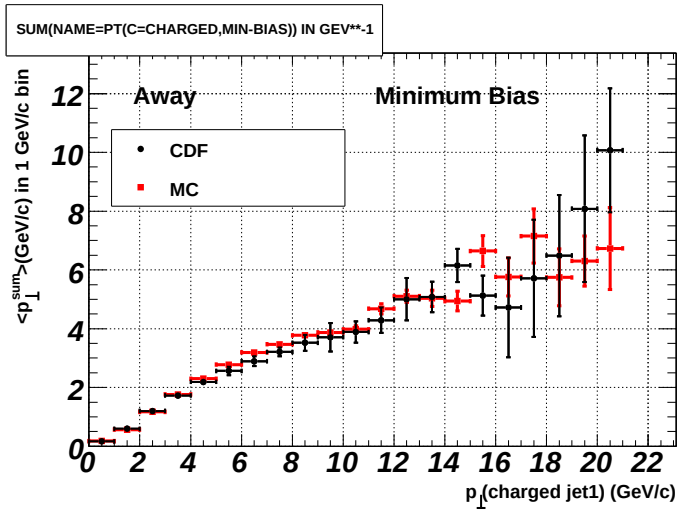
- Run the analysis `CDF_2001_S4751469` (Field & Stuart UE)
- Verify the histograms (are they (reasonably) filled?)
- Run Pythia6413 (`FPythia:6413`) with TuneA: MSTP(5)=100
- Compare to data (located in .../local/share/Rivet/)
- Vary the TuneA parameters and check for changes in the distributions
- Run analysis with Pythia8 (`CCPythia`) and compare
- Run analysis with Herwig6510+Jimmy (`FHerwigJimmy:6510`) and compare
- Run Pythia6413 with new shower S0: MSTP(5)=300
  and compare to data and TuneA

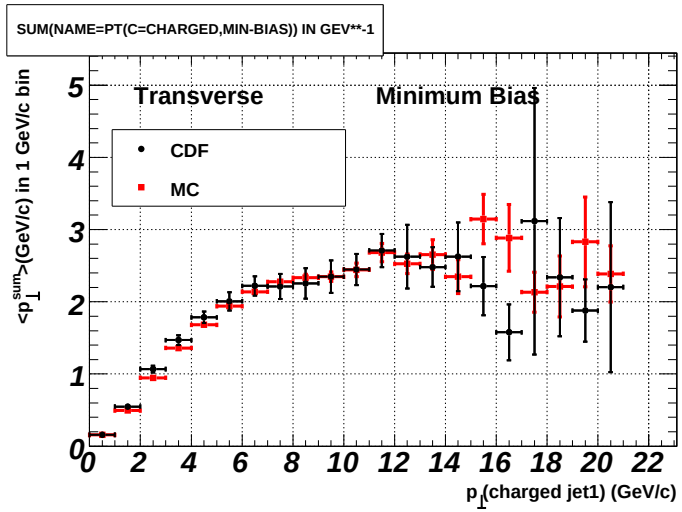
Figures of merit produced with Root script CDF2001UE_DESY.C

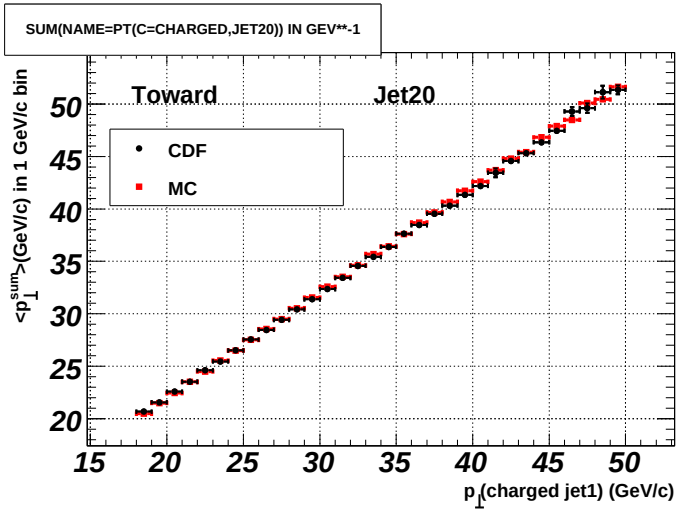
- MC = Pythia 6413, TuneA, 1M events

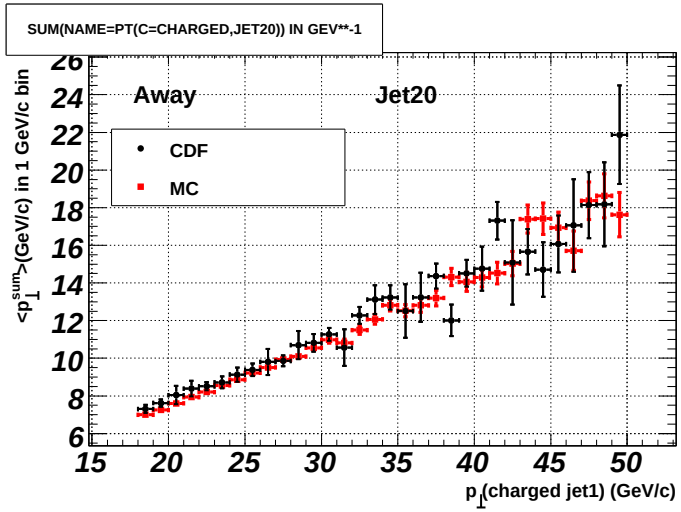
Two separated runs to populate JET20 and Minimum Bias charged jet $p_\perp$ ranges

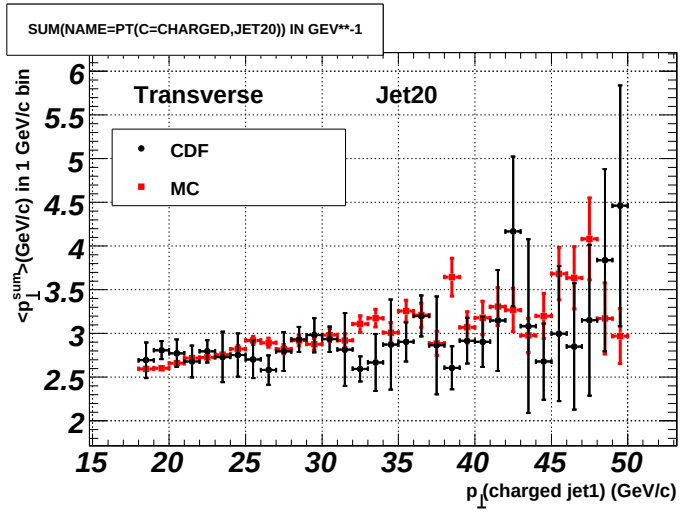












- Details

How Rivet works

- Event reference passed to `AnalysisHandler`
- First event analysed ⇒ Projections get applied
- Each applied projection is attached to event, after running `analyze`
- Subsequent projections will be compared against attached projections of same type to prevent redundant re-evaluation
- `MyProj::compare(MyProj)` defines comparison behaviour
- Filling histograms
- Processing next event
- `finalize` normalises histograms if applicable,
  data point collections are written out in chosen file format

A Rivet Projection

include/Rivet/Projections/Multiplicity.hh

```
#include "Rivet/Projection.hh"
#include "Rivet/Projections/FinalState.hh"
#include "Rivet/Particle.hh"
#include "Rivet/Event.hh"

namespace Rivet {

  /// Count the final-state particles in an event.
  class Multiplicity : public Projection {

  public:
    /// Constructor. Provided FinalState projection must live throughout run.
    Multiplicity(FinalState& fsp)
      : _fsproj(fsp), _totalMult(0), _hadMult(0)
    {
      addProjection(fsp);
    }

    ~Multiplicity() {
      getLog() << Log::TRACE << "Destroying " << getName() << " at " << this
      << endl;
    }

    /// Return the name of the projection
    string getName() const {
      return "Multiplicity";
    }
```

include/Rivet/Projections/Multiplicity.hh

```
  protected:
    /// Perform the projection on the Event.
    void project(const Event& e);

    /// Compare projections.
    int compare(const Projection& p) const;

  public:
    /// @name Access the projected multiplicities.
    //@ {
    /// Total multiplicity
    const unsigned int totalMultiplicity() const { return _totalMult; }

    /// Hadron multiplicity
    const unsigned int hadronMultiplicity() const { return _hadMult; }
    //@ }


  private:
    /// The FinalState projection used by this projection
    FinalState& _fsproj;

    /// Total multiplicity.
    unsigned int _totalMult;

    /// Hadronic multiplicity.
    unsigned int _hadMult;
  };

}
```

A Rivet Analysis

include/Rivet/Analyses/ExampleAnalysis.hh

```
#include "Rivet/Analysis.hh"
#include "Rivet/Projections/FinalState.hh"
#include "Rivet/Projections/ChargedFinalState.hh"
#include "Rivet/Projections/Multiplicity.hh"
#include "Rivet/Projections/Thrust.hh"
#include "Rivet/Projections/Sphericity.hh"
#include "Rivet/RivetAIDA.fhh"


namespace Rivet {
  /// This class just measures a few random things as an example.
  class ExampleAnalysis : public Analysis {

  public:
    /// Default constructor
    ExampleAnalysis()
      : _cmultproj(_cfsproj), _cnmultproj(_fsproj),
        _thrustproj(_cfsproj), _sphericityproj(_cfsproj)
    {
      addProjection(_fsproj);
      addProjection(_cfsproj);
      addProjection(_cmultproj);
      addProjection(_cnmultproj);
      addProjection(_thrustproj);
      addProjection(_thrustproj);
    }
```

include/Rivet/Analyses/ExampleAnalysis.hh

A Rivet Analysis (continued)

```
    /// Factory method
    static Analysis* create() {
      return new ExampleAnalysis();
    }

  public:
    /// @name Publication metadata
    //@{
    /// Return the name of this analysis
    string getName() const {
      return "Example";
    }
    /// Get a description of the analysis.
    string getSpiresId() const {
      return "NONE";
    }
    /// Get a description of the analysis.
    // string getDescription() const {
    //   return "";
    // }
    /// Experiment which performed and published this analysis.
    string getExpt() const {
      return "NONE";
    }
    /// When published (preprint year according to SPIRES).
    string getYear() const {
      return "NONE";
    }
    //@}
```

e.g. for CDF 2001 UE analysis:

`getReferences()`
Phys.Rev.D65:092002,2002
FNAL-PUB 01/211-E

4751469

Field & Stuart underlying event analysis at CDF.

CDF

2001

include/Rivet/Analyses/ExampleAnalysis.hh

```
  private:
    /// The final state projectors.
    FinalState _fsproj;
    ChargedFinalState _cfsproj;

    /// The multiplicity projectors (charged and all).
    Multiplicity _cmultproj;
    Multiplicity _cnmultproj;

    /// The thrust projector.
    Thrust _thrustproj;

    /// The sphericity projector.
    Sphericity _sphericityproj;

    /// Hide the assignment operator
    ExampleAnalysis& operator=(const ExampleAnalysis&);

    //@{
    /// Histograms
    AIDA::IHistogram1D* _histTot;
    AIDA::IHistogram1D* _histChTot;
    AIDA::IHistogram1D* _histHadrTot;
    AIDA::IHistogram1D* _histHadrChTot;
    AIDA::IHistogram1D* _histThrust;
    AIDA::IHistogram1D* _histMajor;
    AIDA::IHistogram1D* _histSphericity;
    AIDA::IHistogram1D* _histAplanarity;
    //@}
  };
}
```

A Rivet Analysis

src/Analyses/ExampleAnalysis.cc

```
#include "Rivet/Tools/Logging.hh"
#include "Rivet/Analyses/ExampleAnalysis.hh"
#include "Rivet/RivetAIDA.hh"

namespace Rivet {

  // Book histograms
  void ExampleAnalysis::init() {
    // Since this is just a demo analysis, there is no associated paper!
    _histTot   = bookHistogram1D("TotalMult",
                 "Total multiplicity", 100, -0.5, 99.5);
    _histChTot = bookHistogram1D("TotalChMult",
                 "Total charged multiplicity", 50, -1.0, 99.0);
    ...
  }

  // Do the analysis
  void ExampleAnalysis::analyze(const Event& event) {
    Log log = getLog();

    // Analyse and print some info
    const Multiplicity& cm = event.applyProjection(_cmultproj);
    const Multiplicity& cnm = event.applyProjection(_cnmultproj);

    const Thrust& t = event.applyProjection(_thrustproj);
    log << Log::DEBUG << "Thrust = " << t.thrust() << endl;
```

src/Analyses/ExampleAnalysis.cc

```
    const Sphericity& s = event.applyProjection(_sphericityproj);
    log << Log::DEBUG << "Sphericity = " << s.sphericity() << endl;
    log << Log::DEBUG << "Aplanarity = " << s.aplanarity() << endl;


    // Fill histograms
    const double weight = event.weight();
    _histTot->fill(cnm.totalMultiplicity(), weight);
    _histChTot->fill(cm.totalMultiplicity(), weight);
    ...
  }


  // Finalize
  void ExampleAnalysis::finalize() {
    normalize(_histTot);
    normalize(_histChTot);
    ...
  }
}
```

Cuts and Beam Constraints

- Projections and analyses specify cuts and beam constraints
- Cuts: binary cut ranges on named variables
- BeamConstraint: order-independent pair of allowed beams, inclusive `All` wildcard
- Each projection and analysis contains an acyclic graph of other projections (Optimised to obtain maximal efficiency/minimal redundancy)
- Analyses can set constraints on minimal cuts/cut ranges, allowed beam types, etc.
- Usable by e.g. RivetGun

Histogramming

- Histogramming via AIDA
- 1D histos, 1D profile, 2D `DataPointSet` supported
- Output in different file formats:
  - AIDA XML (default)
  - flat text (easily readable).
    Requires rivetgun option `--histotype FLAT`
  - Root TGraphAsymmErrors (for histograms),
    TTree (for observables on event by event basis).
    Requires rivetgun option `--histotype ROOT`

External (plugin) analyses

- No need to modify `libRivet` to write and link an analysis
- At runtime, ., `LD_LIBRARY_PATH` and `RIVET_ANALYSIS_PATH` are scanned for `libRivet*.so`
- If such a file contains `dlopen`able analyses, they are added to the list available in `rivetgun -h`
- Projections are not pluggable
- Look at Rivet wiki for more developer info
  `http://projects.hepforge.org/rivet/trac/wiki`
  `http://projects.hepforge.org/rivet/trac/wiki/WritingAnAnalysis`

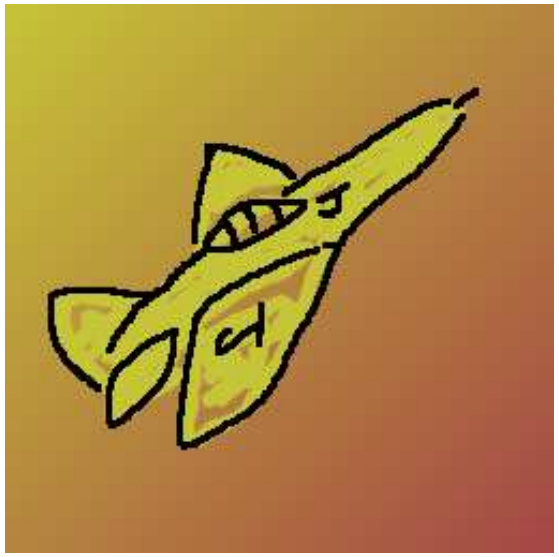
AGILe

- `AGILe` **Generator** interface defines basic messages to which generators respond
- `setInitialState(...)`,
  `setSeed(...)`,
  `setParam(...)`,
  `makeEvent(...)`,
  `getCrossSection()`,
  ...
- It is doing such a good job "behind the scenes" that the user does not realise all the work that went in
- Interface now stabilising: generator state I/O remains to be done

- The bigger picture

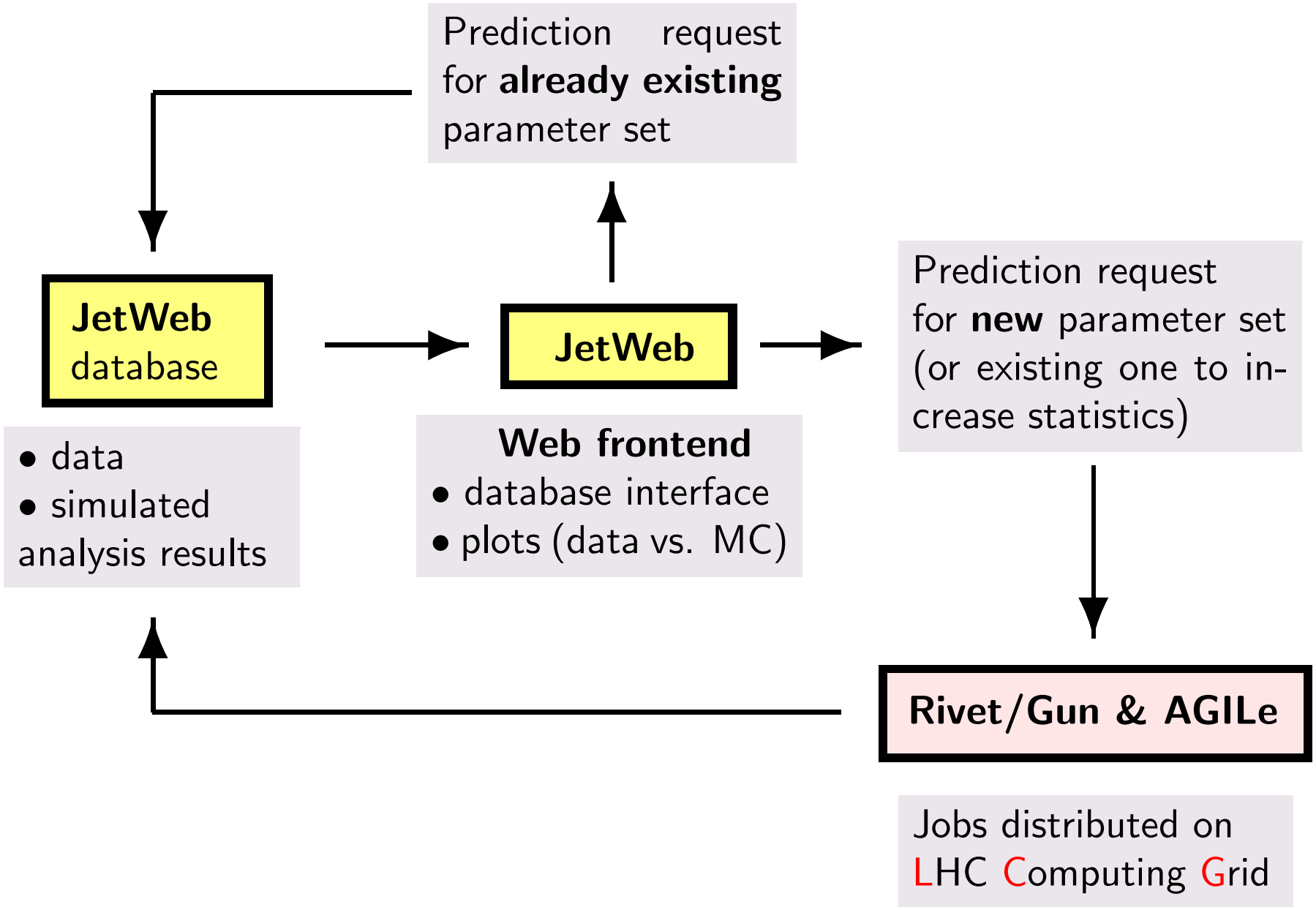
JetWeb

A web based comparison, validation and tuning tool

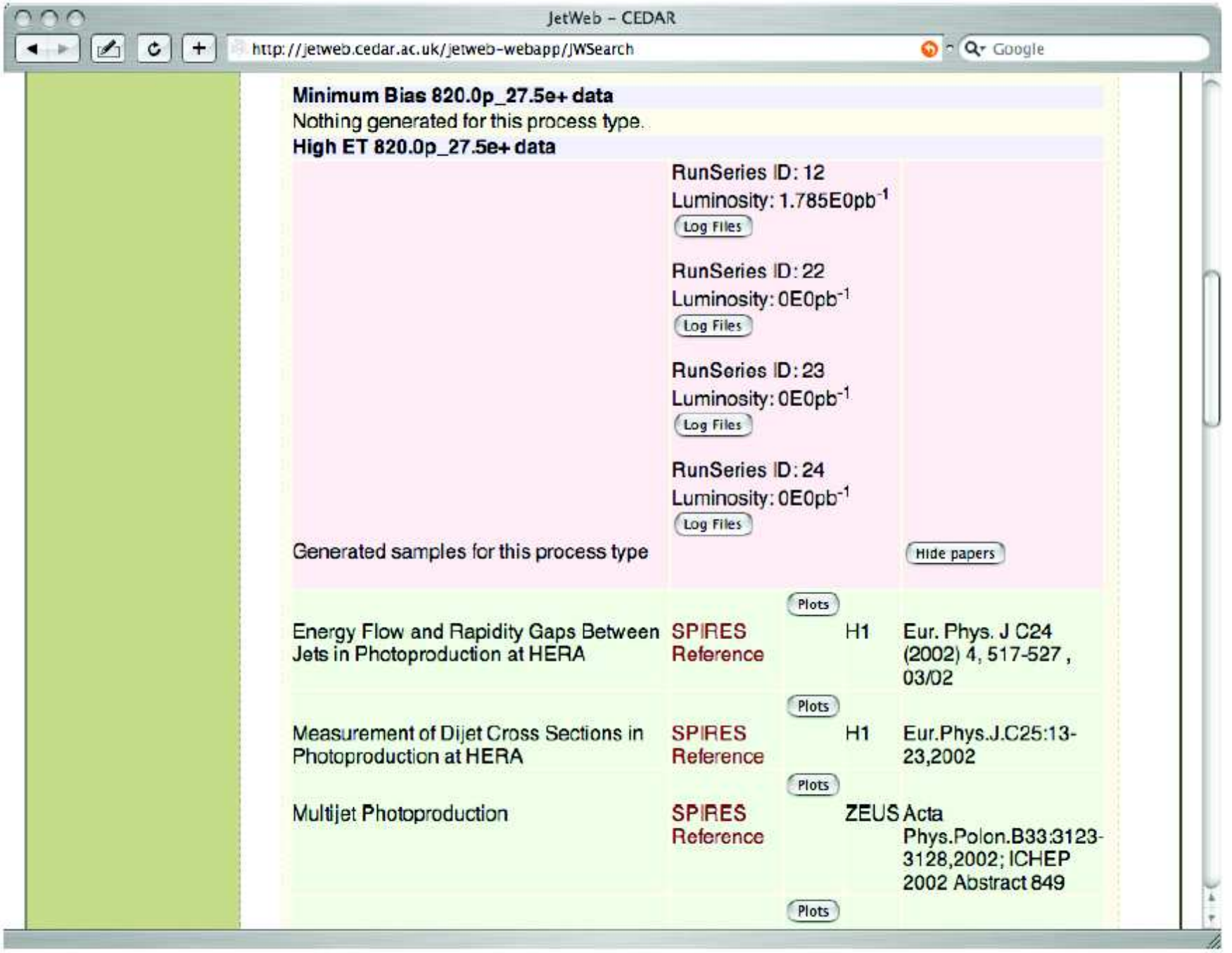


- Archive of simulated analysis results, indexed by model parameters
- Using MySQL to access database (for reference data and simulated analyses)
- Visual inspection of distribution agreement
- Generation and storage of simulated data
- Grid based distribution of simulation jobs
- JetWeb project page at CEDAR: `http://jetweb.cedar.ac.uk`

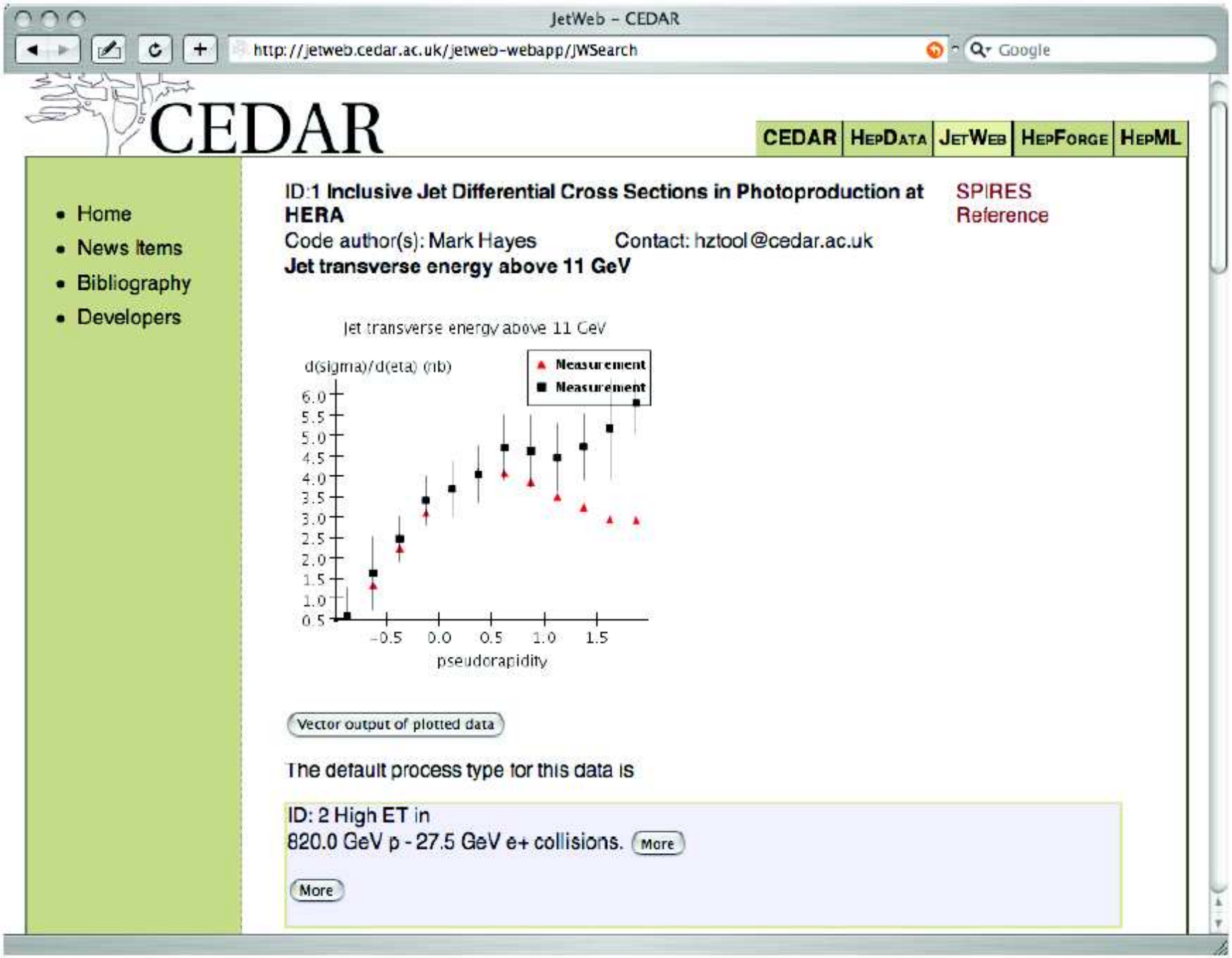
Validation of Experiment and Theory II



JetWeb: Physics analysis publications



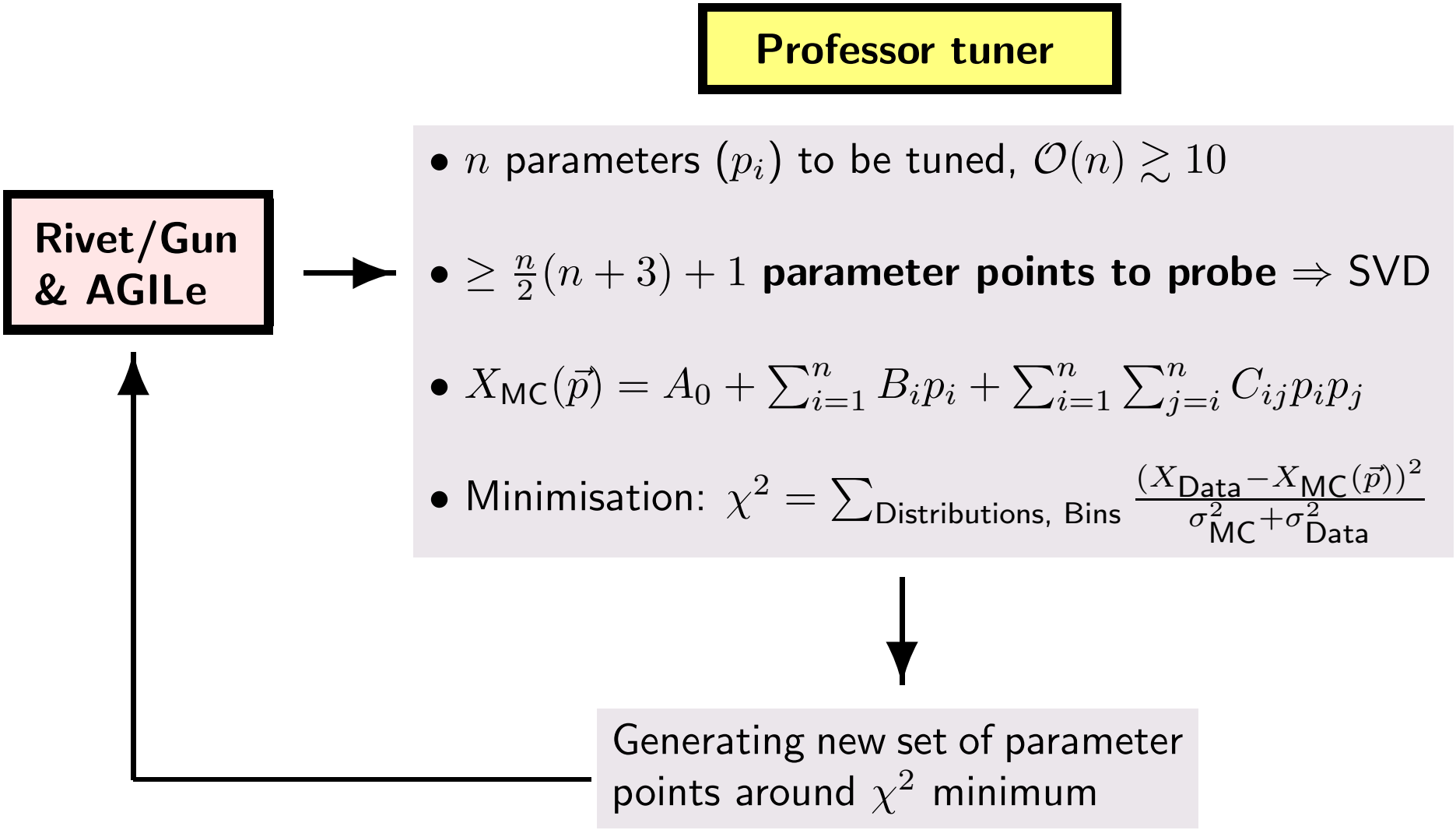
JetWeb: Visual plot verification



Validation of Experiment and Theory III

Procedure for estimating Systematic errors

MCnet project at: `http://projects.hepforge.org/professor`



Conclusions (II)

- Rivet details elaborated
- There are already two embedded Rivet applications
- JetWeb (web based comparison, validation and tuning tool)
- Professor tuner (optimisation of MC event generators)