Event generator tuning and validation tools

Rivet, AGILe and Professor

Andy Buckley

Institute for Particle Physics Phenomenology, Durham University

HERA-LHC meeting, 2008-05-29

Validation and tuning

- Event generators have plenty of input parameters, particularly in hadronisation
- Apart from hand-waving arguments, we can't calculate "correct" param values
- Have to judge tunings on how well they describe the data
- Two classes of generator optimisation:
  - Validation: testing performance of a given param set
  - Tuning: choosing a particular optimal param set

  We'll discuss validation first...

Generator validation with Rivet

Global comparisons

- Selective tunings are insufficient: they may
  1. have no sensitivity to certain params
  2. allow unchecked distributions to become poor fits to data
- Systematic global validation is essential when developing general purpose tunings
- Since event records are largely generator independent (as long as you don't ask silly unphysical questions!) it doesn't make sense to write separate sets of validation analyses for every generator…

Rivet

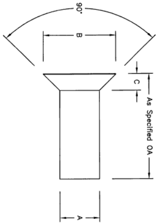
Rivet is a C++ replacement for FORTRAN HZTool

- **A validation framework for generators:**
  - generator steering (via AGILe interfaces)
  - tools for data analysis (e.g. event shape and jet calculators)
  - a library of experimental analyses
- Separation of steering from data analysis:
  - HepMC event could have been just generated or read from file
- Rivet 1.0 released in Jan 2008,
  Rivet 1.1.0 this week

Durham University

Rivet design

- Observables calculated via "projections" — auto-cached, nested and shared
- Analyses use projections to produce histograms comparable to reference data
- Rivet tries to avoid hard-coding any reference data
- AIDA XML files exported from HepData are bundled with the Rivet release
- Binnings are automatically taken from HepData records, ensuring that reference and MC histos are always in sync
- User analyses can be loaded at runtime as "plugins"

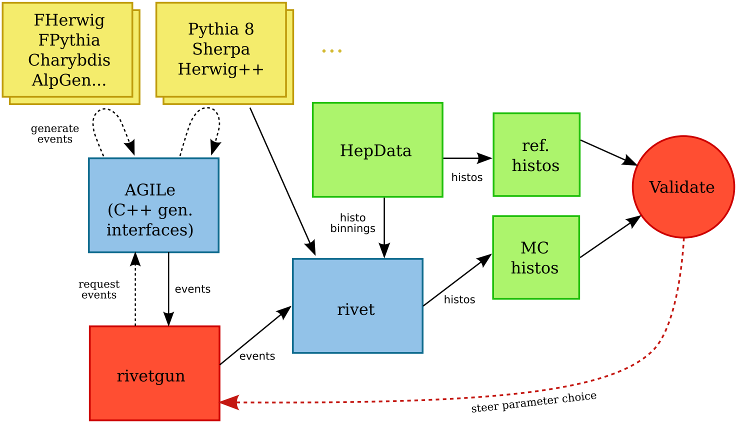


Durham University

Rivetgun and AGILe

- Rivet's primary interface is the **`rivetgun`** command line tool
- Allows reading from HepMC IO_GenEvent file or on the fly via AGILe
- AGILe is **A** (C++) **G**enerator **I**nterface **L**ibrary, **e**rm...
- Interfaces for FHerwig, FPythia with AlpGen, Charybdis, Jimmy (many versions), Pythia 8, Sherpa, Herwig++
- Ensuring that AGILe works with LCG gen packages → improvements in LCG GENSER

How it all fits together



Rivet history

Good things come to those who wait

We started work on Rivet almost 3 years ago!

- **June 2007: 0.9 release**
  - Basic structure, hard-coded analyses, few examples
- **February 2008: 1.0 release (at last!)**
  - Dynamic runtime loading of analyses
  - More analyses
- **May 2008: 1.1 release**
  - Central projection repository: memory issues simplified/eliminated
  - Atlas interface (by James Monk)

Durham University

- Projections no longer stored as member pointers
- Instead register projections with a name in the constructor:

```
class MyAnalysis : public Analysis {
  MyAnalysis() {
    setBeams(PROTON, PROTON);
    ChargedFinalState cfs;
    addProjection(cfs, "CFS");
    ...
  }
```

Rivet 1.1 (cont.)

New and shiny features…

- Apply them in the analyze method via that name:

```
void MyAnalysis::analyze(const Event& evt) {
  ...
  const FinalState& fs =
    applyProjection<FinalState>(evt, "CFS");
  ...
}
```

- Memory management is handled automatically, so no nasty hidden problems with class slicing or premature destruction!
- You wouldn't believe how long it took to make this work…

Rivet projections

A quick selection:

- **Final states:** normal, DIS, "vetoed", charged, hadronic, unstable (for flavour studies)…
- **Event shapes:** thrust, sphericity (regularisable), Parisi C & D params, hemispheres…
- **Jets:** k_T, CDF "track jet", DØ ILC, SISCone, CDF RunII Midpoint (Durham, JADE via FastJet patch)
- **Misc:** jet shapes, primary vertex position, secondary vertices…

+ more. Collection will grow.

Rivet analyses

Current analyses: code is ⟨*expt*⟩_⟨*year*⟩_S⟨*SpiresID*⟩

- Two illustrative examples
- **LEP:** ALEPH_1991_S2435284, DELPHI_1995_S3137023, DELPHI_1996_S3430090, PDG_HADRON_MULTIPLICITIES
- **Tevatron:** CDF_1994_S2952106, CDF_2001_S4751469, CDF_2004_S5839831, CDF_2005_S6217184, CDF_2006_S6653332, CDF_2007_S7057202, CDF_2008_S7541902, D0_2001_S4674421, D0_2004_S5992206
- **HERA:** H1_1995_S3167097, ZEUS_2001_S4815815
- Want/need more…particularly HERA & B-factories

(DELPHI event shapes and CDF UE analyses highlighted)

Durham University

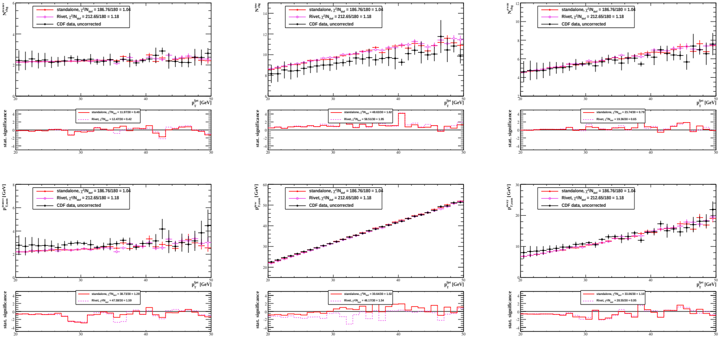
Example: Field-Stuart UE from Rivet

Herwig++ with Rivet & standalone implementations

Transverse | Towards | Away

N_{chg}

$p_{\perp sum}$



Black = CDF data; red = standalone (arXiv:0803.3633); pink = Rivet

Rivet future

The best is yet to come...

- **July/August 2008: 1.2 release**
  - Re-worked histogramming (all pointers eliminated)
  - AIDA histo interfaces replaced with YODA + plain format
  - Mergeable histos, run modes
  - Yet more analyses!
- **Autumn 2008: 1.3 release**
  - Streamed cuts, full use of analysis metadata
  - Even more analyses!

Durham University

Getting started with Rivet

- Easiest way: use bootstrap script at `http://svn.hepforge.org/rivet/bootstrap/rivet-bootstrap`
  - This gets FastJet, AGILe, HepMC and builds/installs them in the right order
  - Alternatively do it by hand with the usual `make && make install`
- AGILe comes with a script to bootstrap a local GENSER repository
- Then run **`rivetgun -h`** — and you're off!

We need your input!

- Since this is a HERA-LHC meeting…
- You might have noticed the HERA section of the analysis list looking a bit sad and empty
- It would be a real shame to not preserve HERA's physics legacy in LHC generator tunings (other than in PDFs). To do that we need people to code up significant HERA analysis papers: diffraction? MPI? Interfacing CASCADE in AGILe?
- Same goes for BaBar — we know there is excellent, unpublished hadron spectum data!
- This would not be wasted time: ATLAS & GENSER already using Rivet
- Audience participation time:
  - Volunteers, suggestions?

Durham University

Generator tuning with Professor

Parameters

We have lots of parameters:

- **PS:** $t_{\min}$, α_s or Λ_{QCD} (really)
- **Hadronisation:** depends strongly on model

  String: string tension σ, Lund symm FF a and b params, baryon suppression, flavour params

  Cluster: constituent masses, flavour params
- **UE:** interaction form factor params (Gaussian width/p(r,h)oton radii), $p_\perp^{\min}$, colour reconnection params
- **CKKW** & friends: ME/PS matching scale

Can sometimes be tuned independently: e.g. kinematics, flavour, UE…depending on analyses

Durham University

Tuning methods

With lots of parameters and a requirement for $\geq$ 100k events per run, tuning is non-trivial. Too slow for serial MCMC sampling approaches to be useful (at present).

Most tunes: by eye / by grad student. Painful, uninspiring and sub-optimal

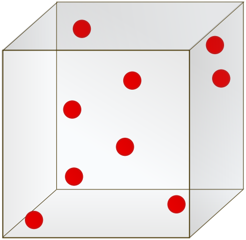
Herwig++ 2.1 default tune: brute-force random on Grid + local param grid scan

DELPHI: following on from linear interpolation tunes, Hamacher et al (1995) did a quadratic interpolation tune. Scalable to many dimensions, interesting and (importantly) it works . . .

Durham University

Professor

The Professor tuning project (Durham, Lund, Dresden/Berlin) uses the DELPHI approach in a flexible Python framework:



1. Sample N random MC runs from n-param hypercube
2. For each bin b in each distribution, use the N points to fit an interpolation function using a singular value decomposition.
3. Construct overall χ^2 function and (numerically) minimise
4. Test optimised point by scanning around it in param and lin comb directions

Singular value decomposition (SVD)

- If we wanted to find the params for an exact multi-dimensional problem, we'd use a matrix inverse
- We don't expect the interpolation to be exact $\rightarrow$ Moore-Penrose pseudoinverse
- SVD is a deterministic method to compute the pseudoinverse
- M is an $m \times n$ matrix. $\exists$ the SVD factorization

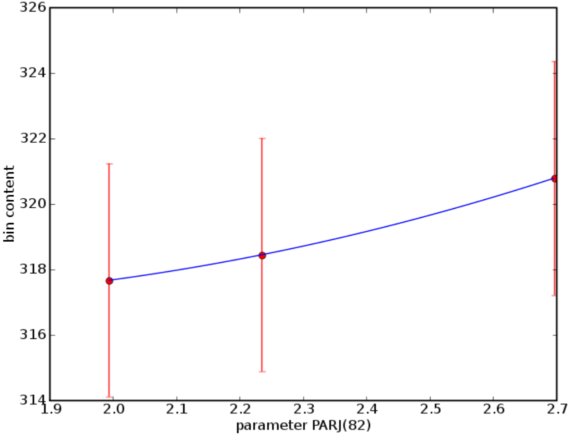
$$M = U \Sigma V^*,$$

  where U is an $m \times m$ and V a $n \times n$ unitary matrix. The Σ matrix is $m \times n$ and diagonal
- Related to eigenvalues — general diagonalisation for all normal matrices
- Equivalent to linear (in coeffs) least squares fit

Durham University

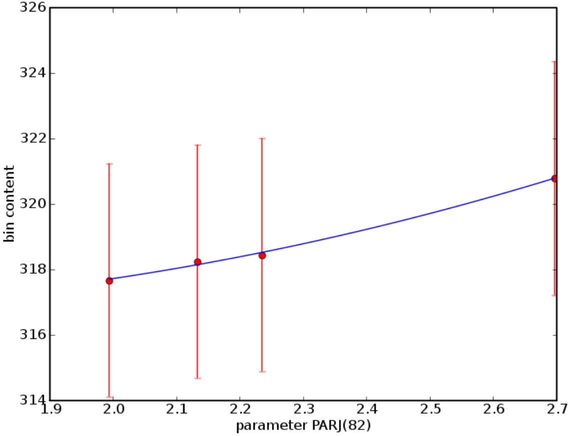
Singular value decomposition demo

Varying number of samples N in a 1D problem:



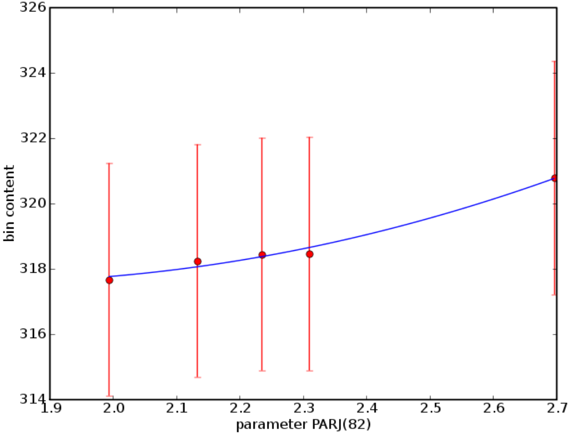
Singular value decomposition demo

Varying number of samples N in a 1D problem:



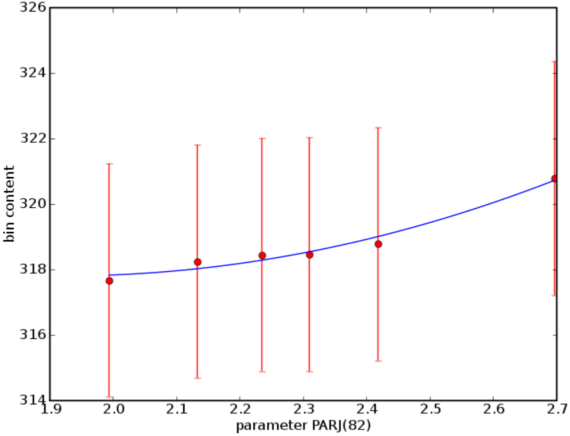
Singular value decomposition demo

Varying number of samples N in a 1D problem:



Singular value decomposition demo

Varying number of samples N in a 1D problem:



The interpolation function

- Obvious interpolation function is the general 2nd order polynomial in n variables:

$$MC_b(\vec{p}) \approx f^{(b)}(\vec{p}) = \alpha_0^{(b)} + \sum_i \beta_i^{(b)} p'_i + \sum_{i \le j} \gamma_{ij}^{(b)} p'_i p'_j$$

  where shifted param vector $\vec{p}\,' \equiv \vec{p} - \vec{p}_0$, with $\vec{p}_0$ chosen as the centre of the param hypercube
- Quadratic f is general-purpose and includes correlations between params
- Remember that this is used to interpolate the actual MC output per bin — not the overall χ^2

Durham University

The interpolation function (cont.)

- Another neat feature: the param centres $\vec{p}_0$ are irrelevant:

$$\begin{aligned} x &\rightarrow x - a \\ &\Downarrow \\ x^2 + bx + c &\rightarrow x^2 + (b - 2a)x + (a^2 - ab + c) \end{aligned}$$

  i.e. still a quadratic in $\vec{p}\,'$ even if we got $\vec{p}_0$ wrong.
- SVD is not a Taylor series!

SVD in context

What matrix are we trying to invert?

- Consider a 2D case in x and y:

$$\underbrace{\begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & \vdots & & \end{pmatrix}}_{M \text{ (sampled param sets)}} \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_x \\ \beta_y \\ \gamma_{xx} \\ \gamma_{xy} \\ \gamma_{yy} \end{pmatrix}}_{C \text{ (coeffs)}} = \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \end{pmatrix}}_{V \text{ (values)}}$$

- Then $C = \tilde{\mathcal{I}}[M]\, V$ ($\tilde{\mathcal{I}}$ = pseudoinverse operator)

Minimum number of samples

- P params require at least $N_{\min}^{(P)}$ samples:

$$N_{\min}^{(P)} = 1 + P + P(P+1)/2$$
$$= (2 + 3P + P^2)/2$$

$$\Rightarrow N_{\min}^{(P+1)} - N_{\min}^{(P)} = P + 2$$

- SVD Σ matrix is square for $N = N_{\min}^{(P)}$
- $N > N_{\min}^{(P)}$ is desirable: overconstraint gives a handle on deviations from exact quadratic behaviour

Minimum number of samples

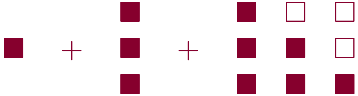
Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3

Durham University

Minimum number of samples

Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3
2	6

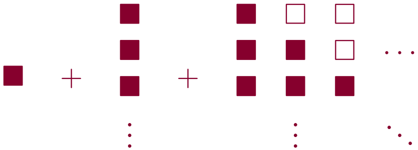
Minimum number of samples



Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3
2	6
3	10

Durham University

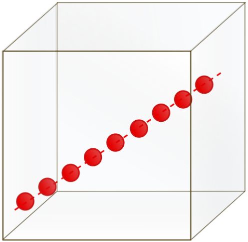
Minimum number of samples

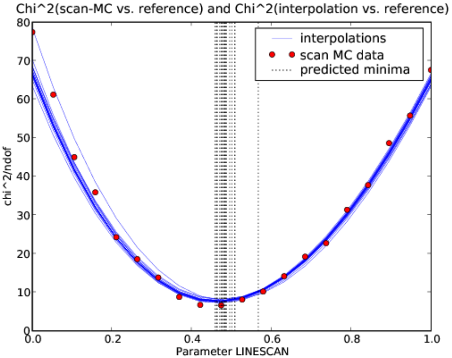


Num params, P	Min num samples, $N_{\text{min}}^{(P)}$
1	3
2	6
3	10
4	15
5	21
$\vdots$	$\vdots$

Verifying the interpolation

A line scan of χ^2/N_{DoF} for FPythia vs DELPHI event shapes through 3D param space (σ_{string}, Λ_{QCD}, Lund a) around a Professor predicted minimum:





Comparable success for 5 params, different observable sets.

Durham University

Professor plans

- Repeat full Delphi tuning: $\sim$24 params
- Tune FPythia 6416 UE and compare to Atlas tune
  - Requires multi-runs, modes in Rivet
- Tune C++ generators

Summary

Summary

- **Rivet** is now in active use by experiments and generator authors: community submission of new analyses à la HZTool is important
- **Professor** is a new interpolation-based generator tuning framework, using output from Rivet: first tunes are being done now
- Lots of room for expansion: weighted tunes, combining different runs/initial states
- **Thankfully, human input will always be needed!**