Theoretical Physics

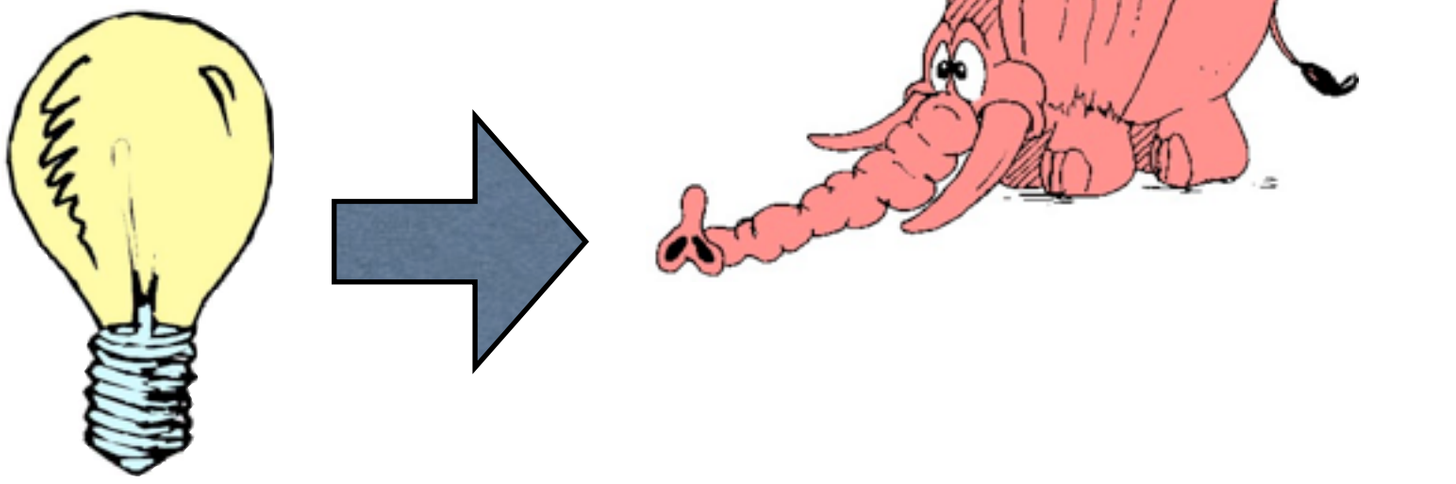
Monte Carlo Development and Tuning

P. Skands (PPD/TH, now at CERN)

Event Generators

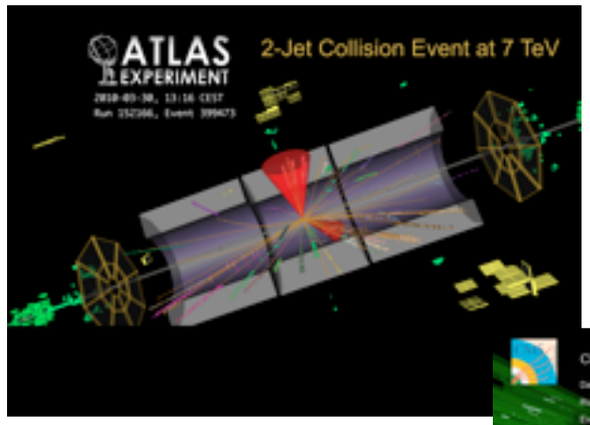
Comparisons to Collider observables

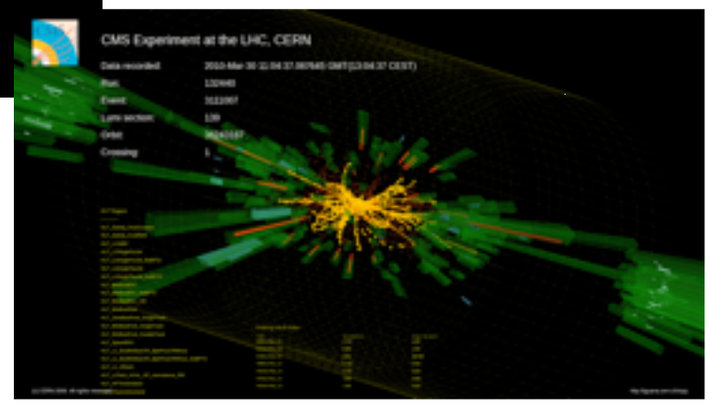
Theoretical Idea



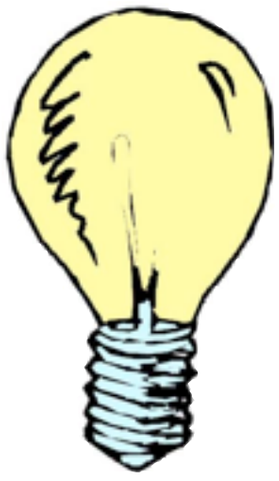
Event Generators

Comparisons to Collider observables





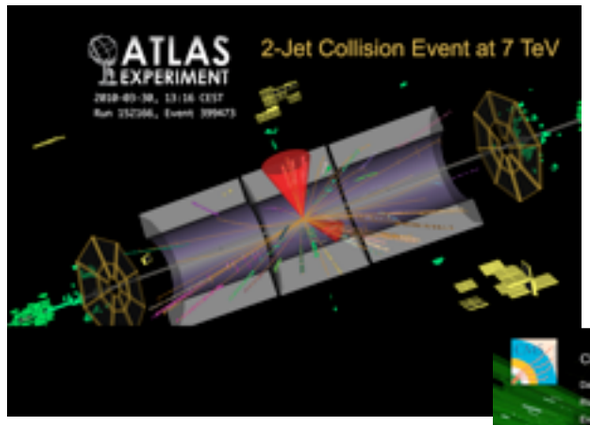
Theoretical Idea

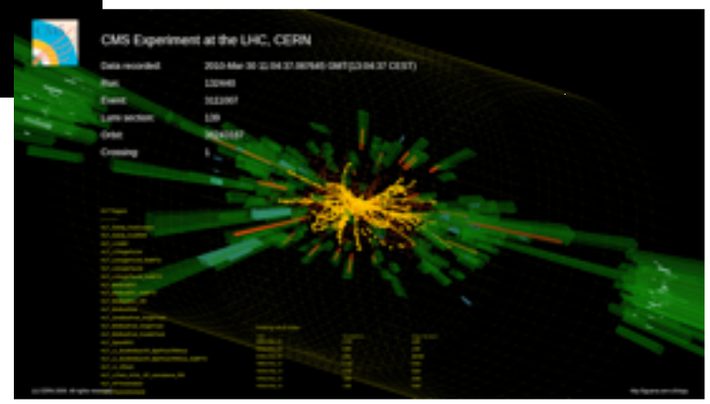


Event Generator (+ detector sim)

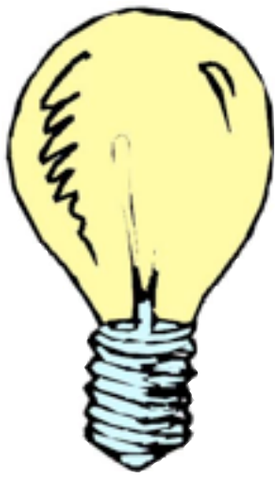
Event Generators

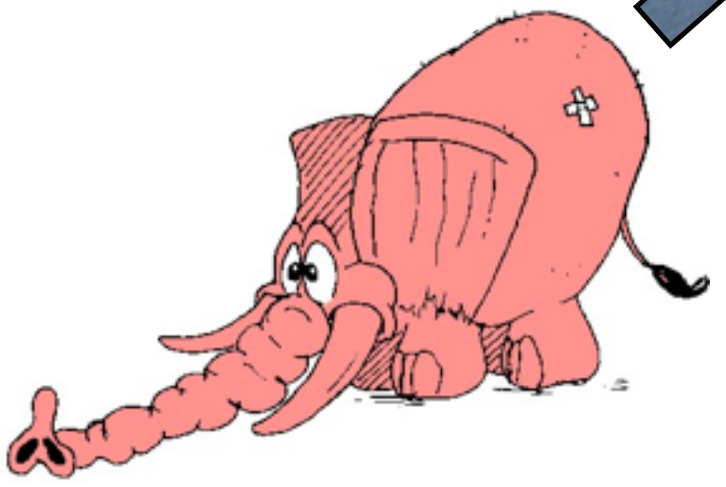
Comparisons to Collider observables





Theoretical Idea



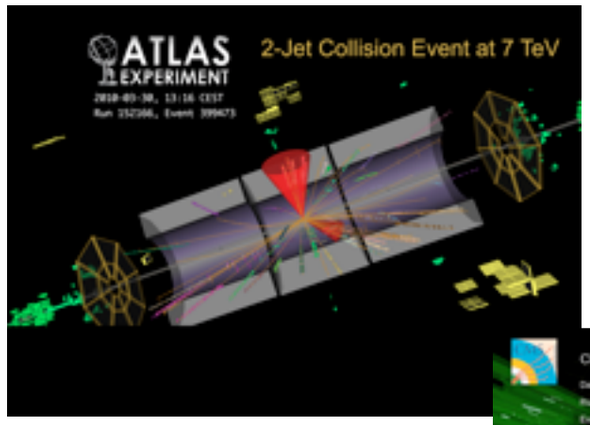


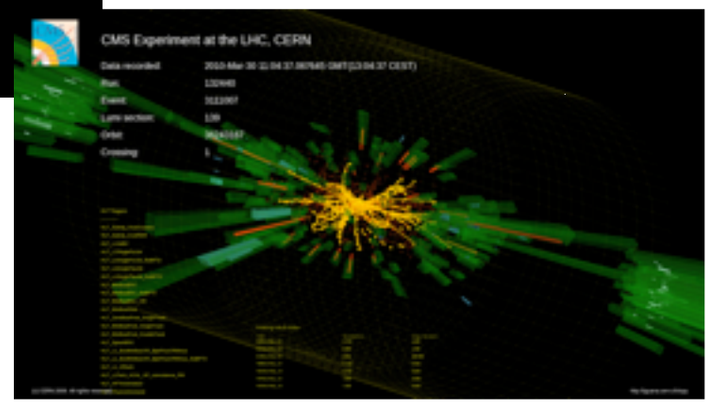
Event Generator (+ detector sim)

A) Theoretical Idea is wrong

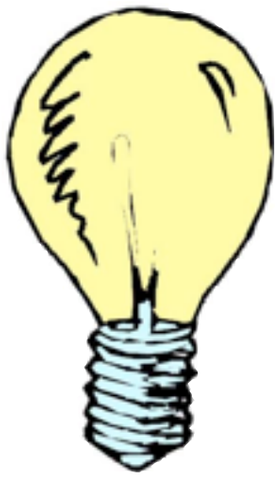
Event Generators

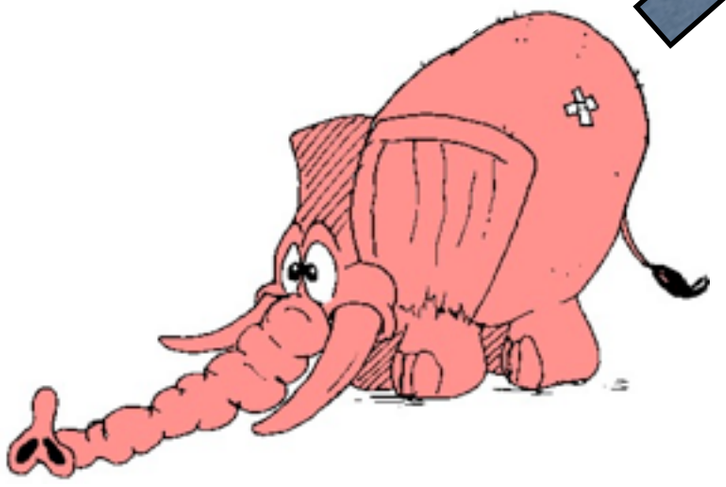
Comparisons to Collider observables





Theoretical Idea





Event Generator (+ detector sim)

A) Theoretical Idea

B) Event Generator Model is wrong

Event Generators

- **Based on Random Number Generator**
  - Use Random Numbers to sample from quantum mechanical probability distributions
  - Factorize complete hadron collision event into sub-steps (MANY substeps) each of which has a fundamental quantum probability to occur
    - Hard Scattering Process ~ a few elementary particles
    - Resonance decays ~ tens of particles
    - Bremsstrahlung (spacelike and timelike) ~ several tens of particles
    - Confinement → (string) hadronization ~ hundreds of hadrons
    - Beam Remnants, Hadron Decays (e.g., $\pi^0 \to \gamma\gamma$) , … + hundreds of photons
  - Put the steps together → Generate completely simulated full-fledged hadron collider physics events

Requirements

Requirements

- **1. Write a good event generator**
  - Development Work: Include all known physics (in as good an approximation as possible)
    - Mostly on paper, then on your desktop, debugging, more desktop, finally on cluster or farm to look at tails (many distributions fall off exponentially)
    - **Short bursts**, then back to blackboard, coding/debugging, another burst

Requirements

- **1. Write a good event generator**
  - Development Work: Include all known physics (in as good an approximation as possible)
    - Mostly on paper, then on your desktop, debugging, more desktop, finally on cluster or farm to look at tails (many distributions fall off exponentially)
      - **Short bursts**, then back to blackboard, coding/debugging, another burst
- **2. Tune the generator**
  - Some (most) free parameters of the generator correspond to things that *cannot* be calculated from first principles theory
    - I.e., finished generator has knobs which are a priori not fixed
    - Must be constrained by comparison to data ⇒ **TUNING**

Generator Tuning

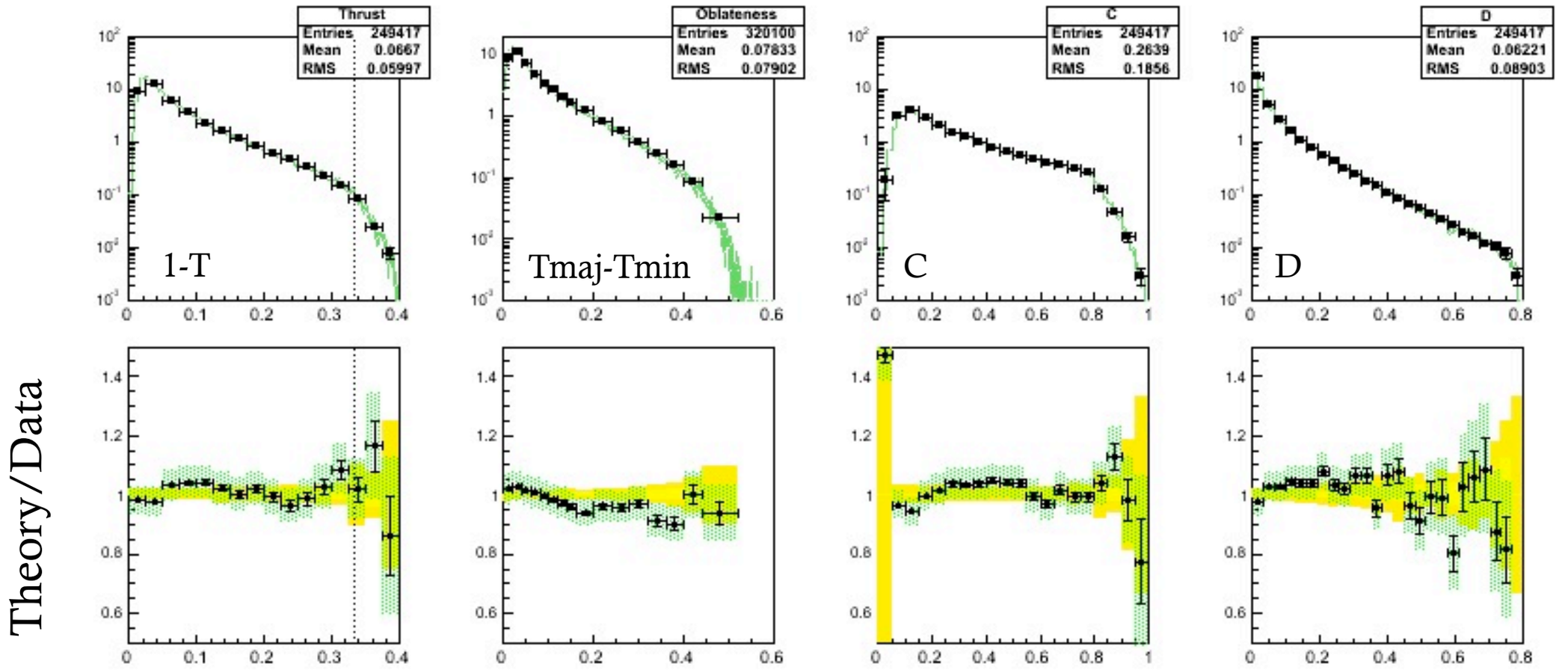
(and validation)

- **After Development** (once physics model itself is ~ finished)
  - Compare to large set of collider measurements
    - Adjust parameters until "best fit" obtained
      - → need to run the generator MANY times, with slightly different parameters, and for different measurements
      - Need GOOD statistics to distinguish fluctuations from physical behavior for rare events
      - **Rare events important!** (most likely to mistake for new physics)
- **Long runs**
  - Repeated long run cycles during tuning
  - Single long run for validating a new version (frozen tuning)

Example: PYTHIA 8

- After run:

In this case looking at:
event shapes at LEP



Tells you how well the generator is reproducing known physics

Tuning & Development

- **Tuning is a developing field**
  - **New distributions** are added → new runs
  - **New lessons** are learned about which distributions are important → new runs
  - **New lessons** are learned about out which *parts* of distributions are important → new runs
  - **Physics Models and other inputs** (e.g., proton structure functions) updated → new runs
  - Efforts *do* exist attempting to optimize/reduce load (e.g., by interpolations in a hypercube of random parameter points), but …

Tuning & Development

- **Tuning is a developing field**
  - **New distributions** are added → new runs
  - **New lessons** are learned about which distributions are important → new runs
  - **New lessons** are learned about out which *parts* of distributions are important → new runs
  - **Physics Models and other inputs** (e.g., proton structure functions) updated → new runs
  - Efforts *do* exist attempting to optimize/reduce load (e.g., by interpolations in a hypercube of random parameter points), but …

Constant demand for up-to-date tunes (and uncertainty evaluations) from the experiments → never 'done'

Size and Scaleability

- **The indivisible quantum is 1 event**
  - Typically ~ 1 - 10 ms on a "standard" CPU
  - Negligible memory requirements (~ 10M)
- **Number of events needed for tuning**
  - Typically from 100k events (low stats)
  - to 100M events (high stats)
  - For each collider (~ 10 from LEP to Tevatron, LHC)
    - For each beam energy (~ 3-4 for each collider)
      - for each process (~ 10 different ones from min-bias, Drell-Yan, dijets, top, …)

Size and Scaleability

- **The indivisible quantum is 1 event**
  - Typically ~ 1 - 10 ms on a "standard" CPU
  - Negligible memory requirements (~ 10M)
- **Number of events needed for tuning**
  - Typically from 100k events (low stats)
  - to 100M events (high stats)
  - For each collider (~ 10 from LEP to Tevatron, LHC)
    - For each beam energy (~ 3-4 for each collider)
      - for each process (~ 10 different ones from min-bias, Drell-Yan, dijets, top, …)

Very roughly up to ~10^8s per high-stat run

Size and Scaleability

- **The indivisible quantum is 1 event**
  - Typically ~ 1 - 10 ms on a "standard" CPU
  - Negligible memory requirements (~ 10M)
- **Number of events needed for tuning**
  - Typically from 100k events (low stats)
  - to 100M events (high stats)
  - For each collider (~ 10 from LEP to Tevatron, LHC)
    - For each beam energy (~ 3-4 for each collider)
      - for each process (~ 10 different ones from min-bias, Drell-Yan, dijets, top, …)

Very roughly up to ~10^8s per high-stat run

Use of Fermilab GP

- **With 10^7 - 10^8s per high-stat run**
  - With ~ 100 cores → 10^5 - 10^6s per core
  - ~ 1 - 10 days
  - = Length of a typical high-stat tuning/validation run

Use of Fermilab GP

- **With 10^7 - 10^8s per high-stat run**
  - With ~ 100 cores → 10^5 - 10^6s per core
  - ~ 1 - 10 days
  - = Length of a typical high-stat tuning/validation run
- **On my desktop**
  - 2 cores: ~ 1 year **PER ITERATION!**

Results

- **Major Tuning & Physics Reference Web Site** (~ 10000 plots)
  - http://home.fnal.gov/~skands/leshouches-plots/
  - Used by experimental collaborations and for TH/EXP interplay

Results

- **Major Tuning & Physics Reference Web Site** (~ 10000 plots)
  - http://home.fnal.gov/~skands/leshouches-plots/
  - Used by experimental collaborations and for TH/EXP interplay
- **HEP phenomenology paper** (being submitted to a journal)
  - *The "Perugia" Tunes*, arXiv:1005.3457 [hep-ph]

Results

- **Major Tuning & Physics Reference Web Site** (~ 10000 plots)
  - http://home.fnal.gov/~skands/leshouches-plots/
  - Used by experimental collaborations and for TH/EXP interplay
- **HEP phenomenology paper** (being submitted to a journal)
  - *The "Perugia" Tunes*, arXiv:1005.3457 [hep-ph]
- **PYTHIA itself**
  - Tunes distributed together with PYTHIA
  - Used by Tevatron, LHC collaborations, + others
    - E.g., *Pythia 6.4 Physics and Manual* (FERMILAB-PUB-06-052-CD-T)
    - 924 citations in arXiv (18th most cited paper on arXiv in 2009)
    - (published in JHEP)
  - Data comparisons and extensive tuning highly used and valuable

Results

- **Major Tuning & Physics Reference Web Site** (~ 10000 plots)
  - http://home.fnal.gov/~skands/leshouches-plots/
  - Used by experimental collaborations and for TH/EXP interplay
- **HEP phenomenology paper** (being submitted to a journal)
  - *The "Perugia" Tunes*, arXiv:1005.3457 [hep-ph]
- **PYTHIA itself**
  - Tunes distributed together with PYTHIA
  - Used by Tevatron, LHC collaborations, + others
    - E.g., *Pythia 6.4 Physics and Manual* (FERMILAB-PUB-06-052-CD-T)
    - 924 citations in arXiv (18th most cited paper on arXiv in 2009)
    - (published in JHEP)
  - Data comparisons and extensive tuning highly used and valuable

THANK YOU!