Toy MC for Calorimeter Calibration

Hartmut Stadie

Calorimeter Calibration Meeting
July, 4th 2008

Motivation

We can use a toy Monte Carlo to

- test the fitter
- study possible biases due to the fitting model
- develop improved fitting procedures to overcome the biases

Generation steps: γ-jet

Steps:

- select photon p_t,η and ϕ from flat spectra
- select corresponding p_t,η and ϕ for the generator jet
- split jet energy in n_c chunks (e.g. $n_c = 200$)
- select η and ϕ for this chunk from a Gaussian distribution around the real jet axis with a width of 0.1
- scale and smear each tower
- sum up the four vectors of the towers to get the jet four momentum

Generation steps: tower treatment

Steps:

- sum up the chunks hitting the same tower
- select em-fration of the tower from a flat distribution between 0.0 and 0.5
- calculate E_{em}
- calculate E_{had}, divide it by the tower constant (c_T) and smear it with a Gaussian with mean $\mu = 1.0$ and width $\sigma = E_{had}\sqrt{c^2_{stochastic}/E_{had} + c^2_{noise}}$
- if($E_{had} < 0$) $E_{had} = 0$
- $E_{outer} = 0$

Response

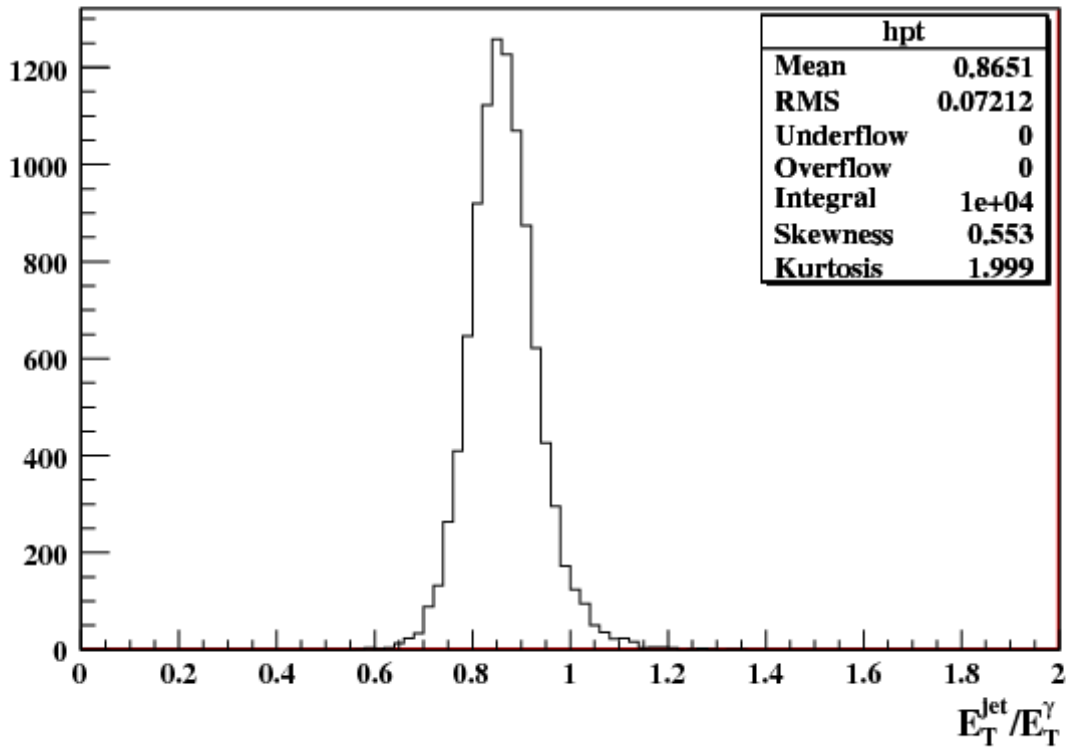
em-fraction

- flat between 0 and 0.5.
- $\bar{f}_{em} = 0.25$,
- $V[f_{em}] = \int_0^{\frac{1}{2}} (f_{em} - \bar{f}_{em})^2 df_{em} = \frac{1}{96}$
- for n_T tower: $\bar{f}_{em} = 0.25 \pm 1/\sqrt{96 n_T}$
  e.g. $n_T = 20; \bar{f}_{em} = 0.25 \pm 0.02$

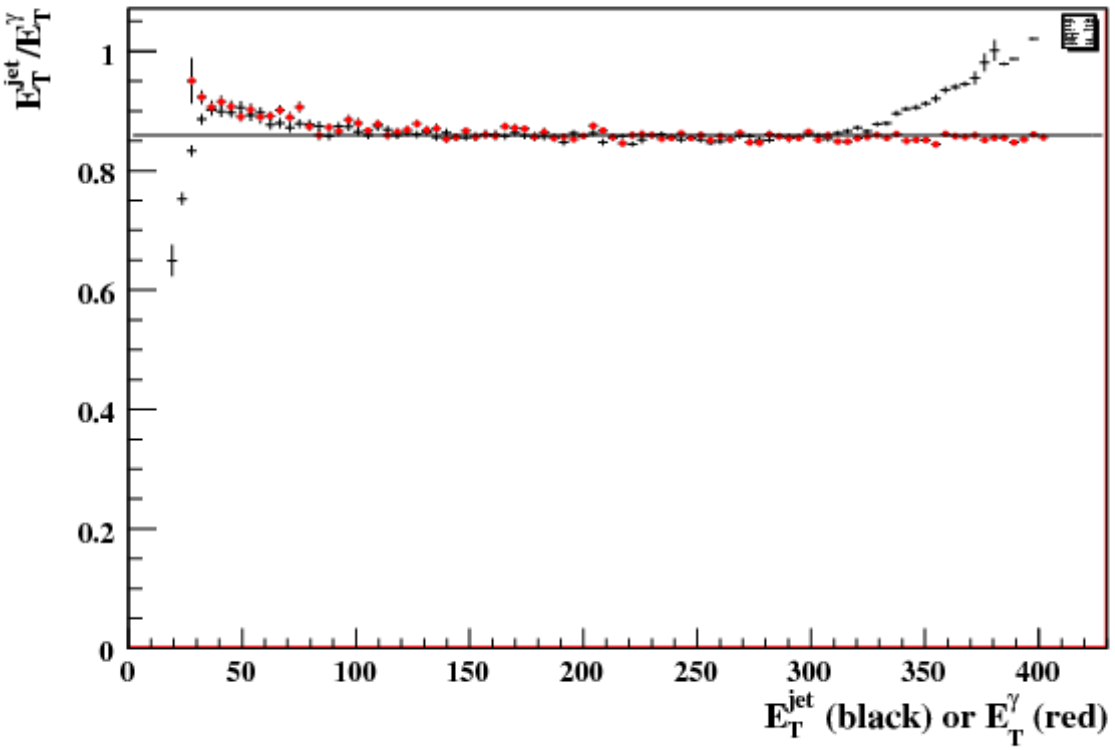
response

- $\bar{R} = \bar{f}_{em} \cdot 1.0 + (1 - \bar{f}_{em}) \cdot \frac{1}{c_T}$
- for $c_T = 1.25$:
  $\bar{R} = 0.25 + 0.75/1.25 = 0.85$

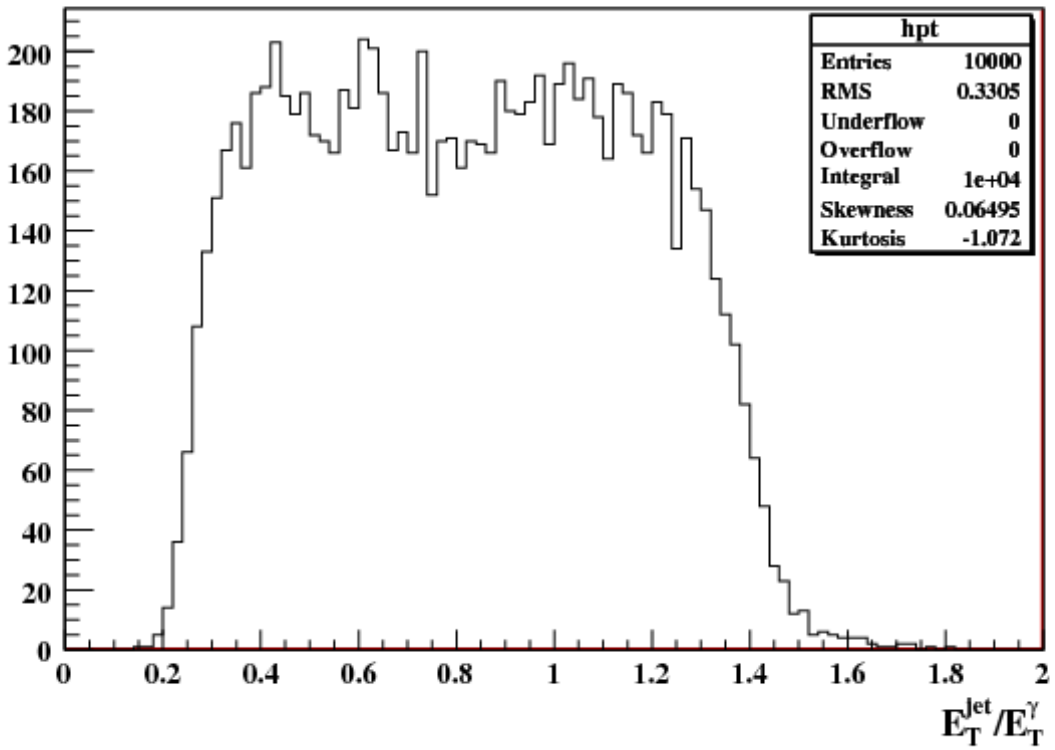
Tower constant c_T = 1.25



Tower constant c_T = 1.25



Tower constant random



Usage

Configuration in toy.cc

```
int main() {
  ToyMC* mc = new ToyMC();
  mc->mMinEta         = -2.5;
  mc->mMaxEta         =  2.5;
  mc->mMinPt          = 30;
  mc->mMaxPt          = 400;
  mc->mTowConst       =  1.25;
  mc->mResoStochastic =  1.25;
  mc->mResoNoise      =  0.056;
  mc->mJetSpread      =  0.10;
  mc->mNoOutOfCone    = true;
  //settings for symmetric distributions
  //mc->mModel          = ToyMC::Gauss;
  //setting for flat distribution (noise)
  mc->mModel          = ToyMC::Flat;
  mc->makePhotonJet("input/toy_photonjet.root",10000);
}
```

Building and running

```
make toy
./toy
```