LVMini - a fitting tool alternative to Minuit
Experiences using it for calorimeter calibration

Christian Autermann

Universität Hamburg

- L-BFGS Algorithm
- LVMini Examples
- Global Fit Calorimeter Calibration

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

LVMini

Volker Blobel's LVMini

- Detailed manual,
- Fortran program source code,
- Examples,
- available on http://www.desy.de/~blobel/largesc.html

Large Scale Optimization with limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)

- BFGS derived from Newton-step optimization methods (hill climbing)
- Seeks the stationary-point of a function f, where $\nabla f=0$ using first and second derivatives.
- For large scale problems (or limited memory) the Hessian matrix containing the second derivatives cannot be stored.
- Instead (Quasi Newton method) the Hessian is updated by analyzing successive gradient vectors
- Good convergence properties of Newton-step methods are maintained, while the memory consumption is modest:

$O(10^5)$ fit parameters on a standard PC via L-BFGS

BFGS algorithm

In quadratic approximation of a given function F(k) depending on a parameter vector **k**, an iteration with a step *d* can be expressed as:

$$F(\mathbf{k}_i + \mathbf{d}) = F_k + \nabla F_k^T \mathbf{d} + \tfrac{1}{2}\, \mathbf{d}^T C_k\, \mathbf{d}$$ with C_k being the Hessian

A quasi-Newton update of C_{k+1} is given by

$$C_{k+1} = \nabla F_{k+1} - \nabla F_k$$

- The Hessian matrix C can be initialized with the unity matrix, so that the first step is in gradient direction.
- Convergence can be checked using the norm of the gradient $|\nabla F_k|$

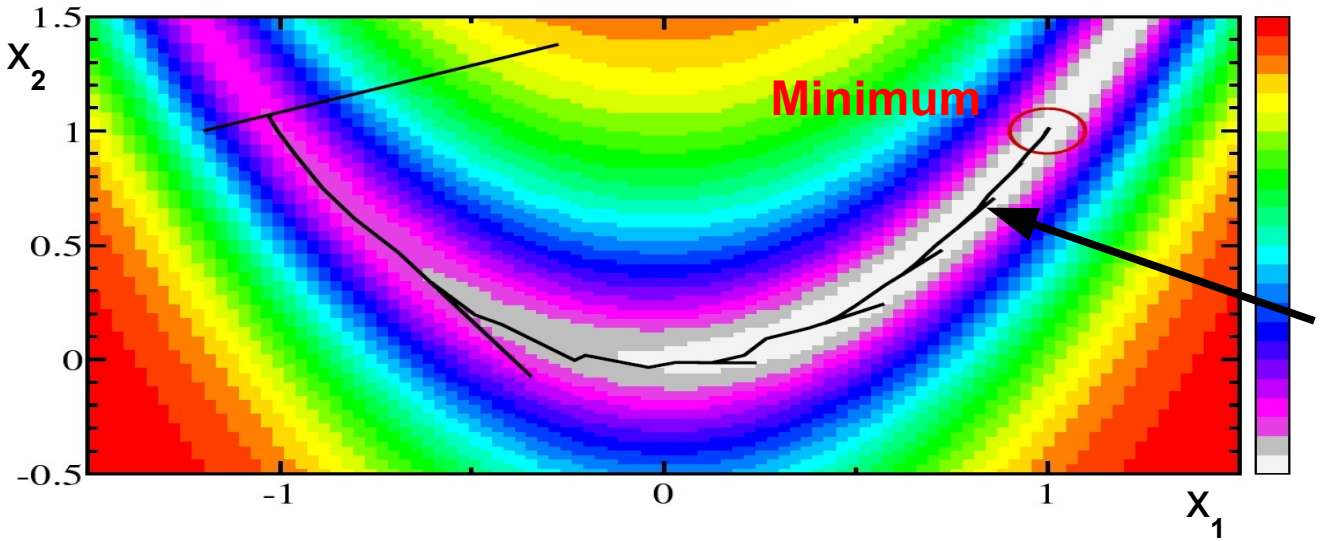
References:

- V. Blobel, Large-scale optimization with the limited-memory BFGS program LVMINI, (2007).
- J. Nocedal and S.J.Wright, Numerical Optimization, Springer Series in Operations Research, (1999).
- http://en.wikipedia.org/wiki/BFGS_method

Example from LVMini manual

Rosenbrock function: minimum (1,1)

$$F(\vec{x}) = 100 \cdot (x_2 - x_1^2) + (1 - x_1)^2$$



LVMini convergence path

Number of function evaluations

Method	$\boldsymbol{x}_0 = (-1, 1.2)$	$\boldsymbol{x}_0 \in \{-10 \ldots +10\}$
Steepest descent	> 5000	
LVMINI	61	$25 \ldots 129$ 71 ± 22
MINUIT, gradient	115	$40 \ldots 252$ 144 ± 53
MINUIT, no gradient	211	$68 \ldots 504$ 286 ± 96

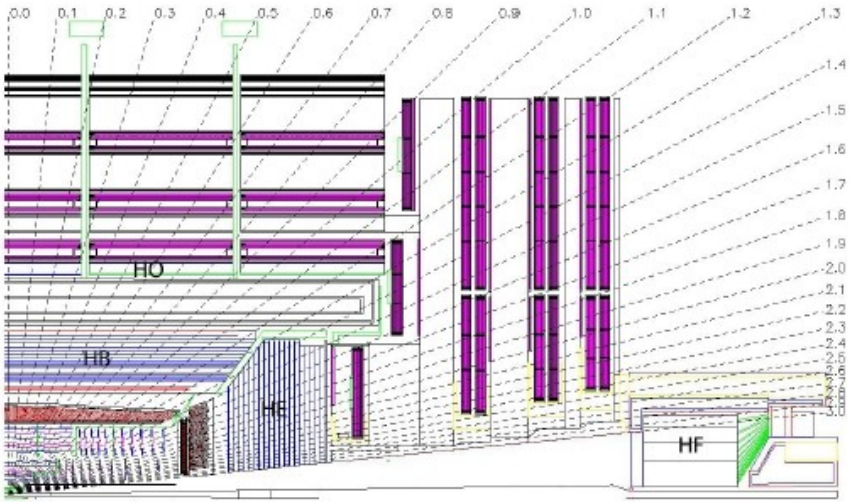
Example

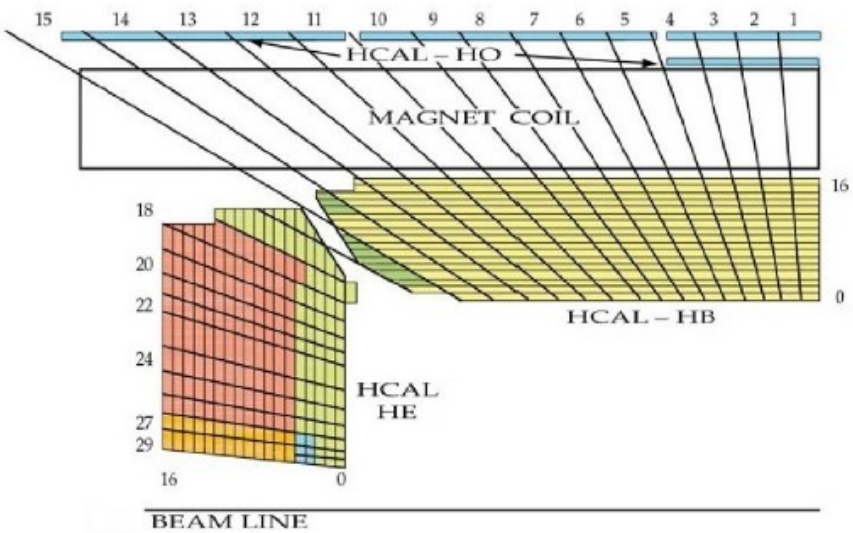

```
void rosenbrock(int &NPAR, double *GRAD, double &FSUM, double *PAR, int &IFLAG){
  FSUM = pow((1.0-PAR[0]),2)+100.0*pow(PAR[1]-pow(PAR[0],2),2);
  // gradient
  GRAD[0] = -2.0*(1.0-PAR[0])-400.0*(PAR[1]-pow(PAR[0],2))*PAR[0];
  GRAD[1] = 200.0*(PAR[1]-pow(PAR[0],2));
}

//Rosenbrock minimization
int main(){
  int    NAUX=1000;
  double AUX[NAUX];
  int    NPAR=2;
  double PAR[NPAR];          //parameters to mi
  double FSUM;               //actual function
  double FOPT,FEDM,DUMMY;

  PAR[0]=-1.2; // minimize by LVMINI
  PAR[1]=+1.0;

  double eps=1.E-9;
  double wlf1=-1.;
  double wlf2=-1.;
  lvmeps_(eps,wlf1,wlf2);
  int NPARneg=-NPAR;
  lvmini_(NPARneg,NPARneg,NAUX,AUX); //initia
  int IFLAG, IRET;
  do {
    rosenbrock(NPAR,AUX,FSUM,PAR,IFLAG);  //e
    lvmfun_(PAR,FSUM,IRET,AUX);
  } while (IRET<0);
}
```

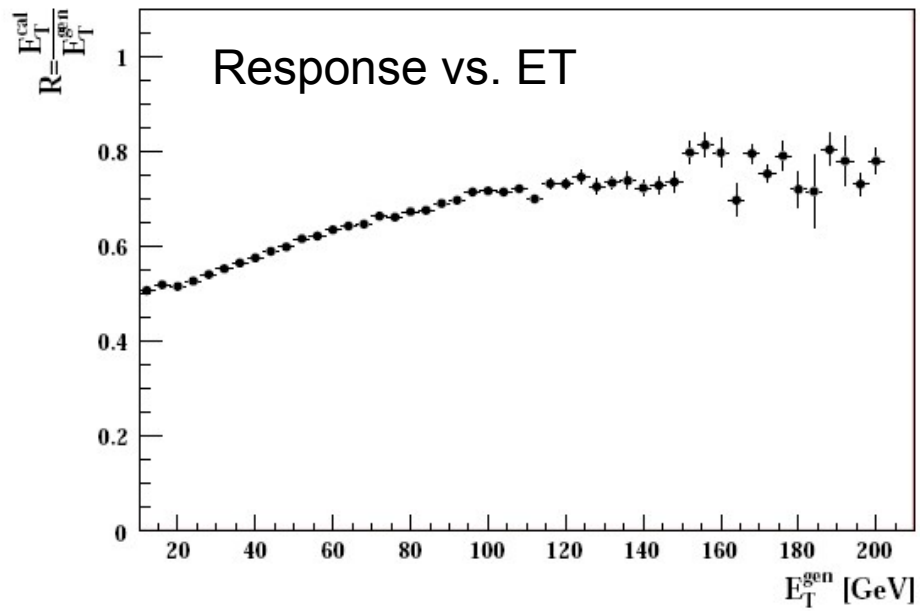
```
Limited-memory BFGS routine LVMINI initialized=======================
   with N = 2  dimension parameter
        M = 2  number of history pairs
       NF =      1000   max number of function calls
      eps = 0.100E-08   convergence parameter
      c_1 = 0.100E-03   first Wolfe parameter
      c_2 = 0.900       second Wolfe parameter
Requires AUX array with at least 34 words

  Initial parameter values:
  par_i       value       par_i       value       par_i       value
     1    -1.20000          2     1.00000


Iter Nfcn Npair      Fcn_value   dFcn__ex   dFcn__ac   fstep    degr
   0    1     0        24.2000   0.00E+00   0.00E+00   1.000    0.00
   1    2     0       171.3360   0.12E+03  -0.15E+03   1.000    0.00
        3     0         4.2252              0.20E+02   0.197
   2    4     1         4.1272   0.82E-01   0.98E-01   1.000    0.29
   3    5     2         4.1203   0.53E-02   0.69E-02   1.000    3.43
   4    6     2         4.1152   0.29E-02   0.51E-02   1.000   25.85
   5    7     2         4.1055   0.18E-01   0.97E-02   0.280   47.48
   6    8     2         4.0867   0.62E-01   0.19E-01   0.159   60.63
        9     2         4.0297              0.76E-01   0.793
   7   10     2         3.9055   0.10E+00   0.12E+00   0.737   81.76
   8   11     2         3.7053   0.25E+00   0.20E+00   0.499   84.32
   9   12     2         3.3621   0.65E+00   0.34E+00   0.312   82.05
  10   13     2         2.8239   0.43E+00   0.54E+00   0.809   78.85
  11   14     2         3.0675   0.36E+00  -0.24E+00   1.000   79.72
```

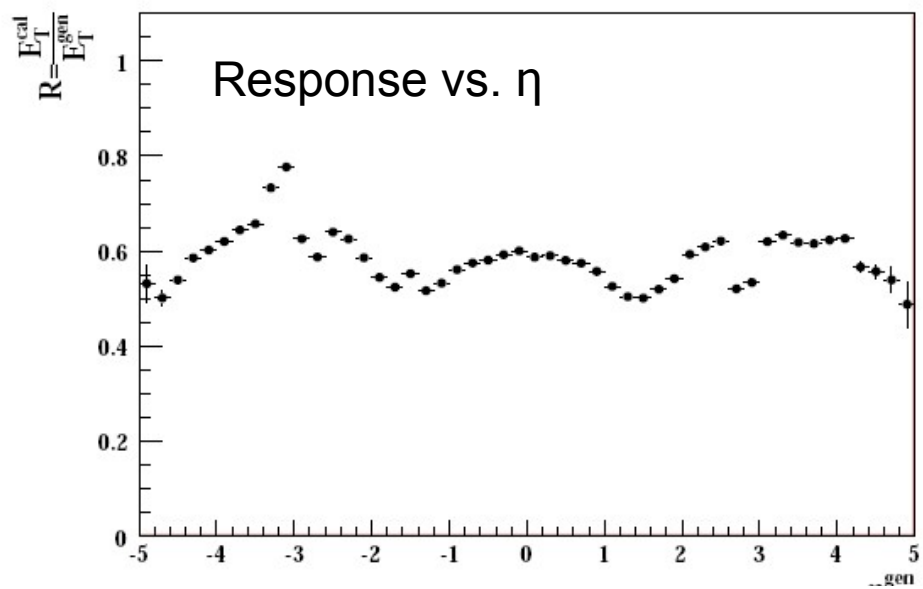
do {
  Evaluate function with PAR[ ] ⇒FSUM
  Fit Parameters with FSUM ⇒ PAR[ ]
} while not converged

CMS Calorimeters









Global Fit Calibration

$$\chi^2_{global} = \chi^2_{\gamma/jet} + \chi^2_{track/tower} + \chi^2 + \ldots$$

$$\chi^2 = \sum_i \frac{(t_i - f(m_i))^2}{[\delta(t_i - f(m_i))]^2}$$

$f(m_i)$: some function(s) parametrizing all the necessary corrections to the measurements m_i

<u>Examples:</u>

Track/tower: t = track momentum
m = tower energy

Gamma/jet: t = gamma energy
$m_i = \sum_i f$ (tower$_i$ energy)

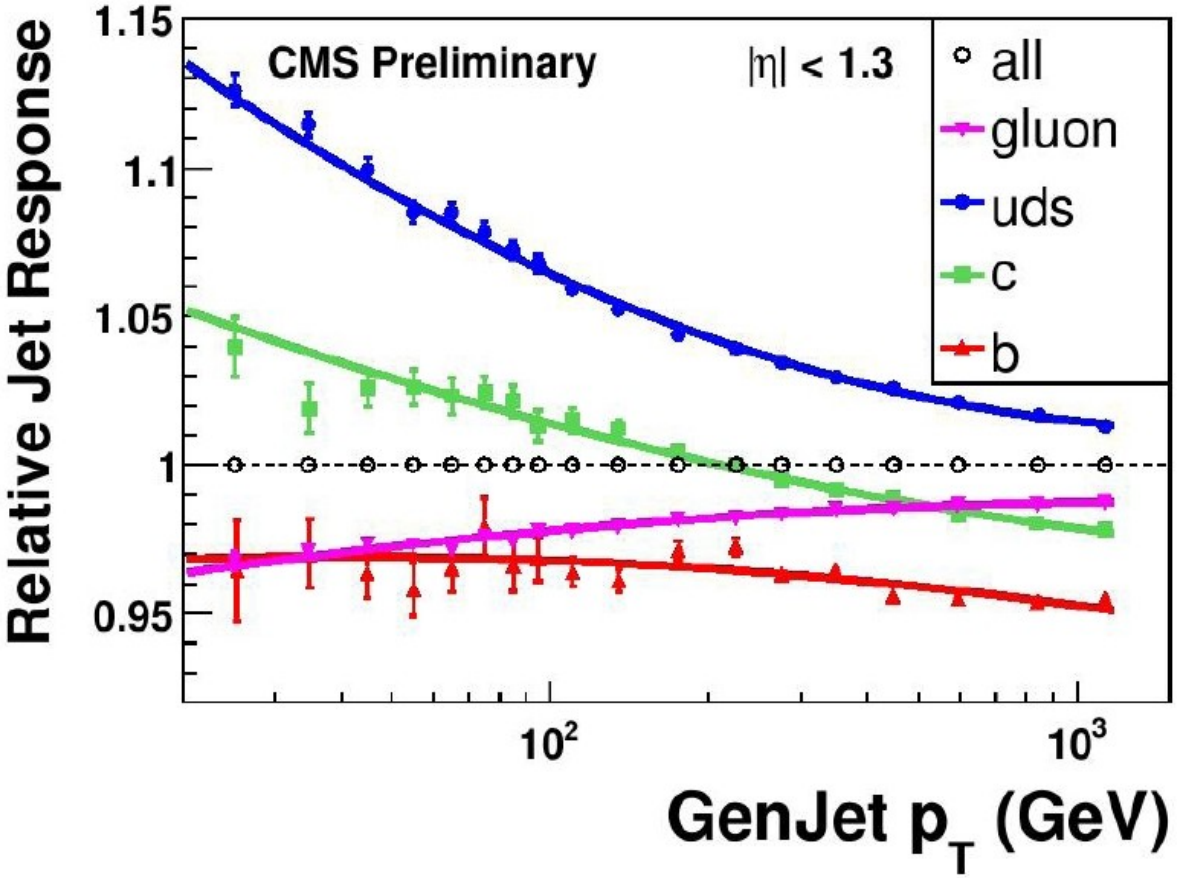
...

Jet-Energy calibration is incorporated in the parametrization function f, too

Classical Calorimeter Calibration

- factorized approach: Use *global fit* to include correlations!

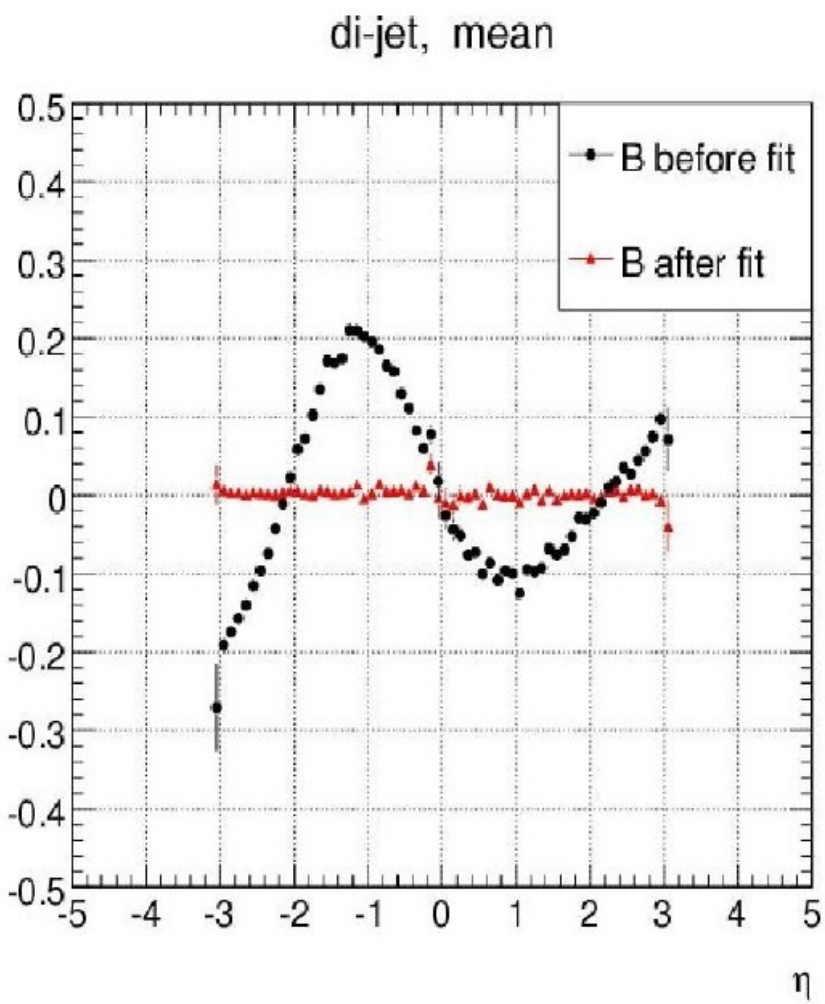
- different response for different jet flavors: try tower level

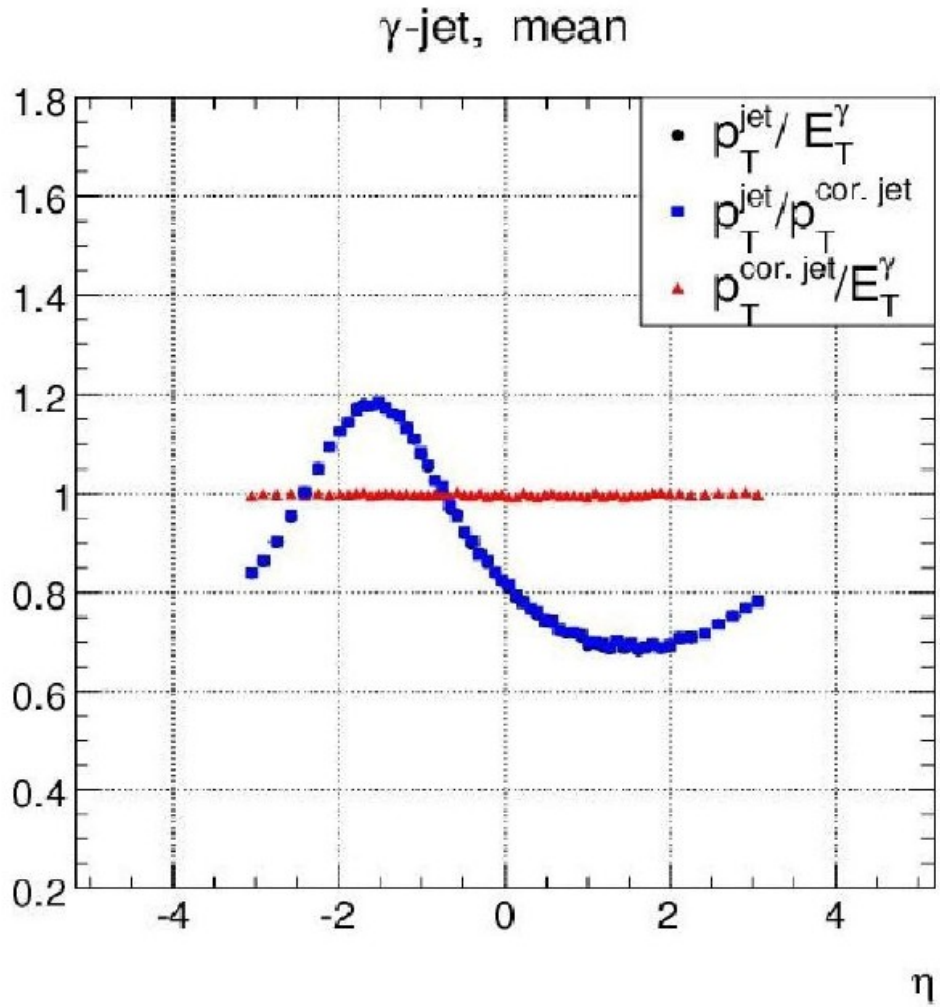


Estimators: Avoid Biases

Relative calibration in η should not affect energy scale!

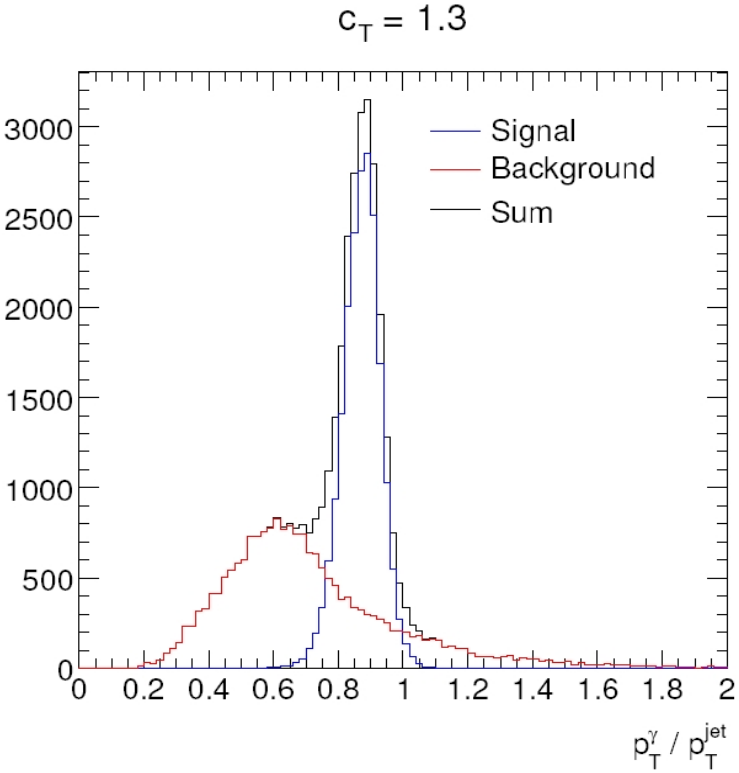
Check on toy MC with $sin(\eta)$ response in η with $B = \frac{2|\vec{p}_{T,1} - \vec{p}_{T,2}|}{p_{T,1} + p_{T,2}}$:





Robustness: Treatment of Outliers

Toy MC example



Simplistic parametrization:

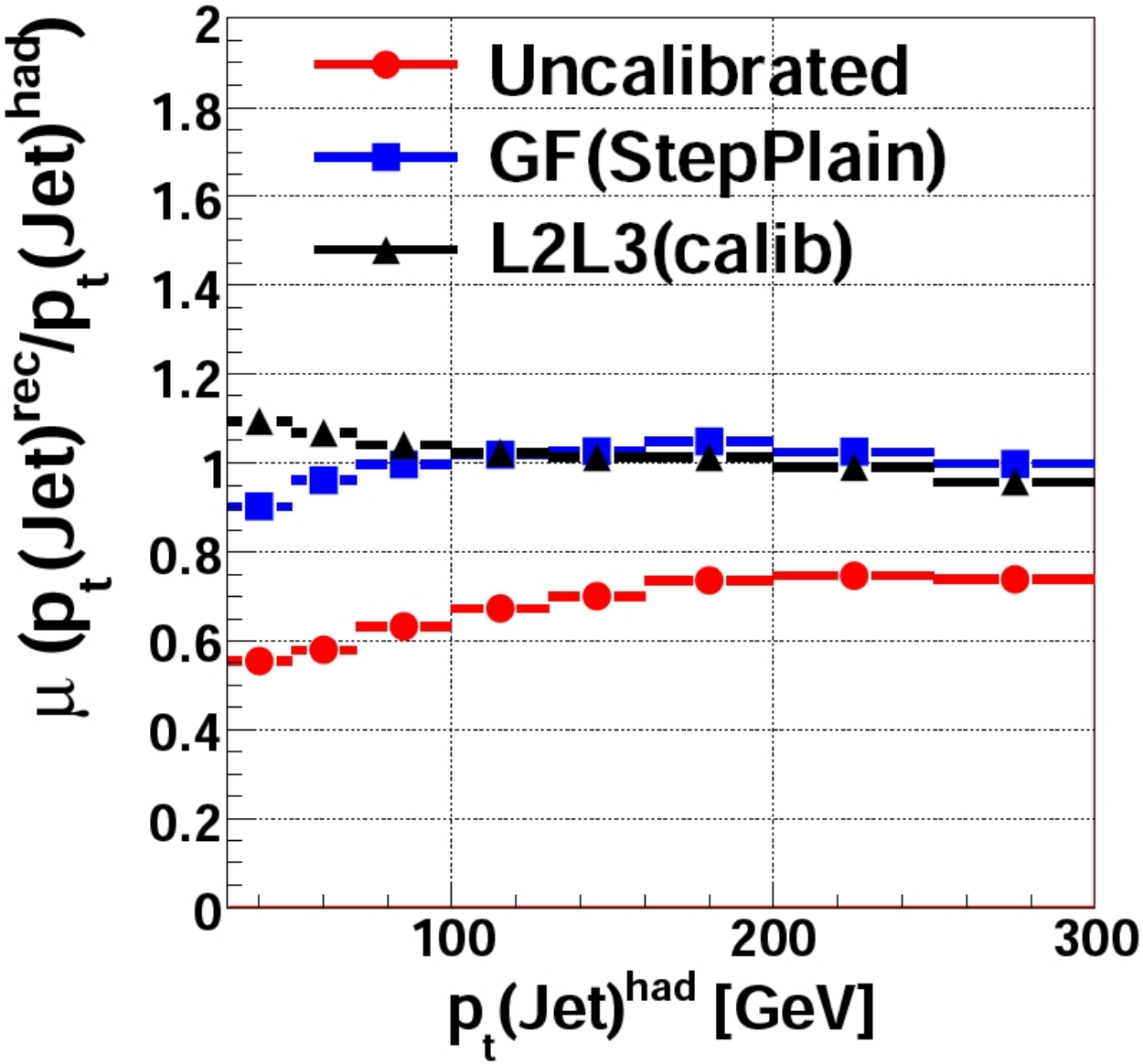
$$p_T^{jet} = \sum_{tow} (p_{T,em}^{tow} + c_{had} \cdot p_{T,had}^{tow})$$

$$\sigma^{jet} = \sqrt{\sum_{tow} \sigma_{tow}^2}, \qquad \sigma_{tow} = 1.25\sqrt{p^{tow}}$$

- $p_{T,meas}^{tow} = \left(\pi^0 + (1-\pi^0)(f_{em} + \frac{1-f_{em}}{c_T})\right) p_{T,true}^{tow}$
- $\overline{\pi}^0 = 0.25$, $\overline{f}_{em} = 0.25$:
  $< \frac{p_{T,meas}^{jet}}{p_T^\gamma} >= 0.4375 + \frac{0.5625}{c_T}$

$N(\text{bgr})/N(\text{sig})$	none	fitted c_{had} CAUCHY	HUBER
0.25	1.32 ± 0.01	1.30 ± 0.01	1.30 ± 0.01
1	1.36 ± 0.01	1.32 ± 0.01	1.33 ± 0.01

Integration and Validation



Our jet corrections integrated in CMS offline framework. Comparisons with factorized calibrations.

Experience and Observations

Started chi2-minimization using TMinuit and LVMini in parallel

- Comparable results and convergence, if using few parameters
- But TMinuit is practically limited to ~20 parameters (no theoretical limit, though)

Successfully used LVMini for fits with $O(10^4)$ parameters

- Fast and reliable convergence
- Runtime is determined by chi2-function evaluation and therefore number of events
- Developed efficient multi-threading code to evaluate f(k) and $\nabla f(k)$.
- Linear time dependence of number of events, parameters and available CPU-cores.

Conclusion and Acknowledgments

The LVMini program is easy to use

- Well written and detailed documentation available
  http://www.desy.de/~blobel/largesc.html
- Performance for a small number of parameters (<20) is comparable to Minuit
- Unique and reliable performance for many parameters up to ~100.000
- C++ examples are available from us

This report is based on the work of many people I wish to acknowledge:

Volker Blobel;

Ulla Gebbert, Sebastian Naumann-Emme, Christian Sander, Matthias Schroeder, Hartmut Stadie, Jan Thomsen, Roger Wolf.