AliTMinuitToolkit - Robust M-estimator

https://alice.its.cern.ch/jira/browse/ATO-410

Requirement

AliTMinuitToolkit

- user interface
- comparison of new and “default” fitting algorithm

Status and to do

Requirements

Requirement for “batch” QA/calibration production (not user)

- Robust non linear minimizer
- N dimensional space
- User defined cost function to minimize
  - e.g chi2, log-likelihood + constraints + week constraints
- Robust error estimator

Usage:

- Distortion fits - line charge density - 2D(dr,drϕ) x 3D(r,rϕ,z)
- Analytical Performance fits for QA trending
  - Automatic alarms examples
  - DCA/Angular/Qpt resolution/pulls as function of pt and eta
  - Nsigma band alarm

M-estimator

In statistics, M-estimators are a broad class of estimators, which are obtained as the minima of sums of functions of the data. **Least-squares estimators** are a special case of M-estimators.

More generally, an M-estimator may be defined to be a zero of an estimating function.

For example, a **maximum-likelihood estimate** is often defined to be a zero of the derivative of the likelihood function with respect to the parameter;

https://en.wikipedia.org/wiki/M-estimator

Maximum likelihood estimator:

Another popular M-estimator is maximum-likelihood estimation.

For a family of probability density functions f parameterized by θ, a maximum likelihood estimator of θ, is computed for each set of data by maximizing the likelihood function over the parameter space $\{ \theta \}$

Maximum-likelihood estimators are often inefficient and biased for finite samples (e.g **local minima**)

For many regular problems, maximum-likelihood estimation performs well for "large samples", being an approximation of a posterior mode.

Methods

cross validation

- Cross-validation, sometimes called rotation estimation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. (Wikipedia)
  - https://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics)&oldid=769508660
  - http://www.milanor.net/blog/cross-validation-for-predictive-analytics-using-r/

bootstraping

- bootstrapping is any test or metric that relies on random sampling with replacement.
- Bootstrapping allows assigning measures of accuracy (defined in terms of bias, variance, confidence intervals, prediction error or some other such measure) to sample estimates
  - https://en.wikipedia.org/w/index.php?title=Bootstrapping_(statistics)&oldid=762130479

Random sample consensus (RANSAC)

- Random sample consensus **(RANSAC)** is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an **outlier detection method.**

- A basic assumption is that the data consists of **"inliers",** i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise, and "outliers" which are data that do not fit the model.

https://en.wikipedia.org/wiki/Random_sample_consensus

- RANSAC also assumes that, given a (usually small) set of inliers, there **exists a procedure** which can **estimate the parameters** of a model that optimally explains or fits this data.

https://en.wikipedia.org/wiki/Random_sample_consensus

Minimization benchmark

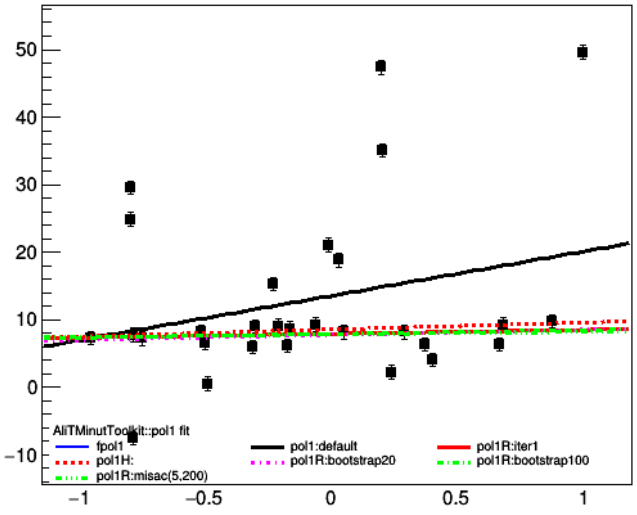
Input - linear model

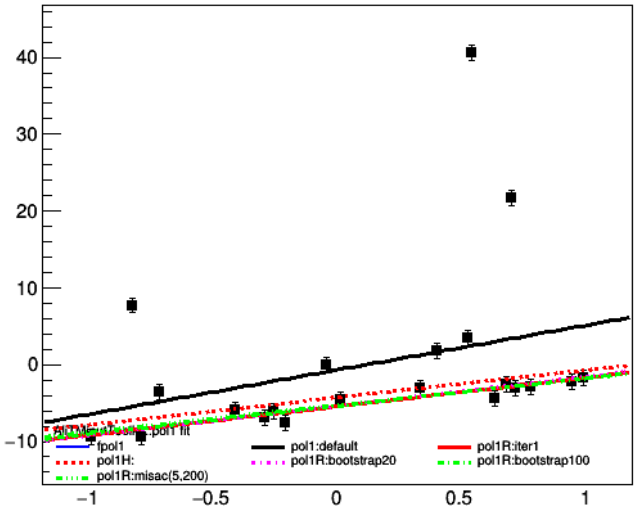
- noise = (60-80%) Gaus(1) + (20%-40%) Gaus(20)

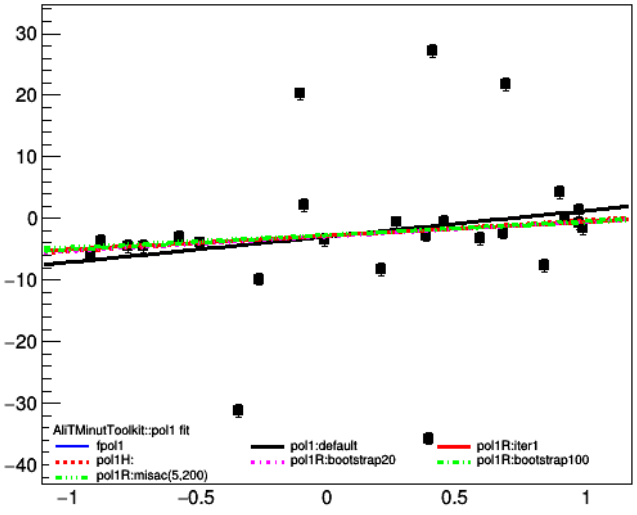
Methods compared:

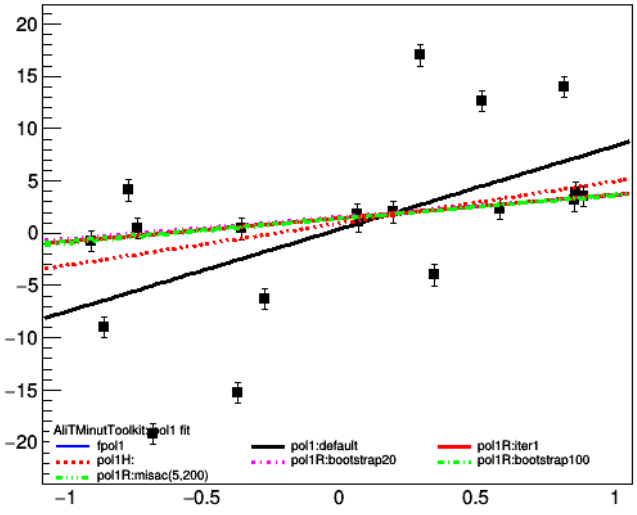
- ROOT:Default
- ROOT:Robust linear fitter (2005, AK,MI)
  - LTS estimator
- AliTMinuitToolkit:Default (chi2)
- LogLike: 80% Gaus+20% Cauchy
  - 1 iter
- LogLike:Huber
- Bootstrap
  - 20,50 iteration
  - LogLike: 80% Gaus+20% Cauchy
- MISAC/RANSAC
  - 20,50 iteration
  - LogLike: 80% Gaus+20% Cauchy

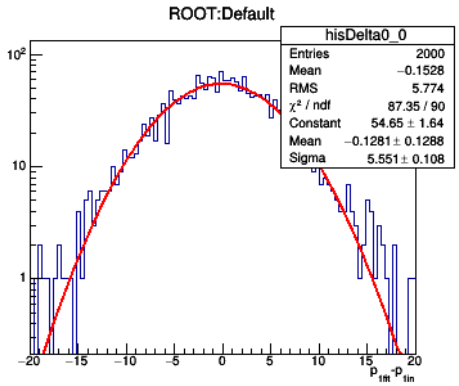
Comparison of M-estimator minimizers

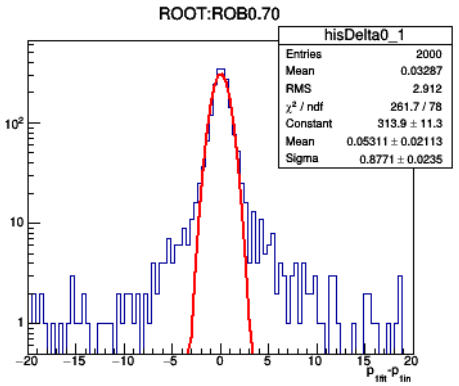


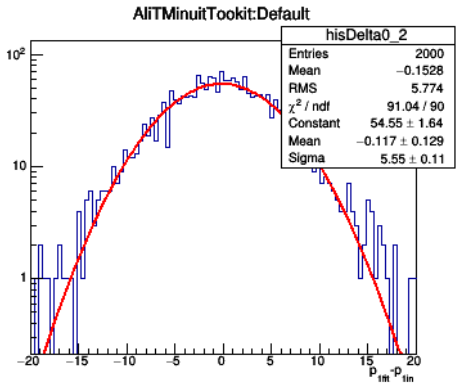


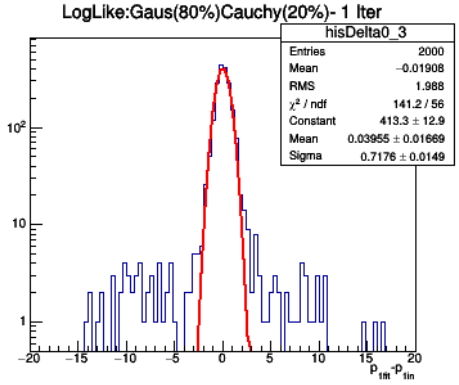


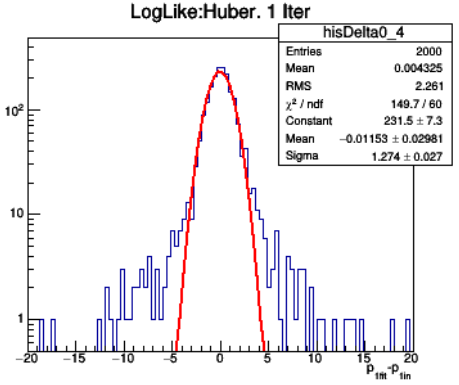


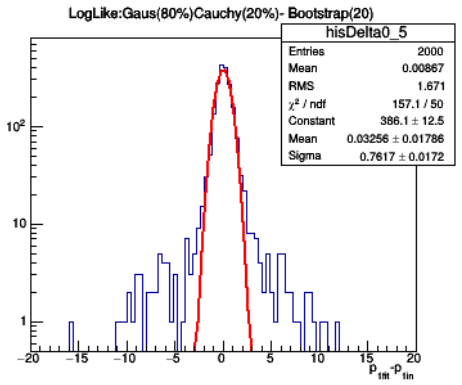

ROOT:Default
hisDelta0_0
Entries 2000
Mean −0.1528
RMS 5.774
χ² / ndf 87.35 / 90
Constant 54.65 ± 1.64
Mean −0.1281 ± 0.1288
Sigma 5.551 ± 0.108
p_tft -p_tin

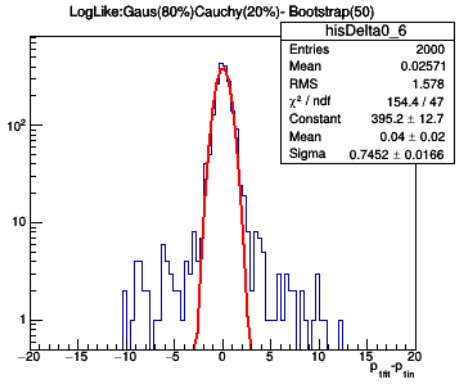

ROOT:ROB0.70
hisDelta0_1
Entries 2000
Mean 0.03287
RMS 2.912
χ² / ndf 261.7 / 78
Constant 313.9 ± 11.3
Mean 0.05311 ± 0.02113
Sigma 0.8771 ± 0.0235
p_tft -p_tin

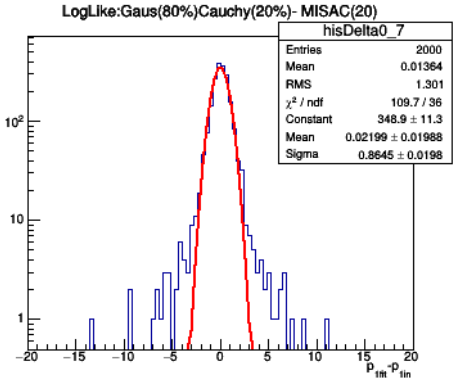

AliTMinuitTookit:Default
hisDelta0_2
Entries 2000
Mean −0.1528
RMS 5.774
χ² / ndf 91.04 / 90
Constant 54.55 ± 1.64
Mean −0.117 ± 0.129
Sigma 5.55 ± 0.11
p_tft -p_tin

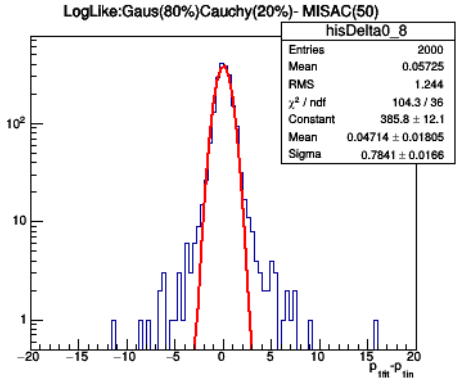

LogLike:Gaus(80%)Cauchy(20%)- 1 Iter
hisDelta0_3
Entries 2000
Mean −0.01908
RMS 1.988
χ² / ndf 141.2 / 56
Constant 413.3 ± 12.9
Mean 0.03955 ± 0.01669
Sigma 0.7176 ± 0.0149
p_tft -p_tin


LogLike:Huber. 1 Iter
hisDelta0_4
Entries 2000
Mean 0.004325
RMS 2.261
χ² / ndf 149.7 / 60
Constant 231.5 ± 7.3
Mean −0.01153 ± 0.02981
Sigma 1.274 ± 0.027
p_tft -p_tin


LogLike:Gaus(80%)Cauchy(20%)- Bootstrap(20)
hisDelta0_5
Entries 2000
Mean 0.00867
RMS 1.671
χ² / ndf 157.1 / 50
Constant 386.1 ± 12.5
Mean 0.03256 ± 0.01786
Sigma 0.7617 ± 0.0172
p_tft -p_tin


LogLike:Gaus(80%)Cauchy(20%)- Bootstrap(50)
hisDelta0_6
Entries 2000
Mean 0.02571
RMS 1.578
χ² / ndf 154.4 / 47
Constant 395.2 ± 12.7
Mean 0.04 ± 0.02
Sigma 0.7452 ± 0.0166
p_tft -p_tin


LogLike:Gaus(80%)Cauchy(20%)- MISAC(20)
hisDelta0_7
Entries 2000
Mean 0.01364
RMS 1.301
χ² / ndf 109.7 / 36
Constant 348.9 ± 11.3
Mean 0.02199 ± 0.01988
Sigma 0.8645 ± 0.0198
p_tft -p_tin


LogLike:Gaus(80%)Cauchy(20%)- MISAC(50)
hisDelta0_8
Entries 2000
Mean 0.05725
RMS 1.244
χ² / ndf 104.3 / 36
Constant 385.8 ± 12.1
Mean 0.04714 ± 0.01805
Sigma 0.7841 ± 0.0166
p_tft -p_tin

User interface

Interface similar in standard fits:

- `AliTMinuitToolkit * AliTMinuitToolkit::Fit(TGraph *graph, const char *fitterName, Option_t* option, Option_t* goption, Option_t* foption, Double_t xmin, Double_t xmax);`
- `AliTMinuitToolkit * AliTMinuitToolkit::Fit(TH1* his, const char *fitterName, Option_t* option, Option_t* goption,Option_t* foption, Double_t xmin, Double_t xmax)'`

Parameters:

- fitterName instead of the fit function (fitters to be registered before)
- foption to specify drawig option for function itself
- other parameters as in standard TH1::Fit and TGraph::Fit
- fitoption can be used to parse fitter parameters - e.g number of fitting iteration for bootstrap and MISAC (see example below)

User interface. Example

Example usage: code to compare different minimizer setting

- `AliTMinuitToolkit*fitter2 = AliTMinuitToolkit::Fit(gr,"pol1","default", "", "funOption(1,3,1)");` **`// standard minuit`**
- `AliTMinuitToolkit*fitter3 = AliTMinuitToolkit::Fit(gr,"pol1R","iter1", "", "funOption(2,3,1)");` **`// logLike:gaus+cauchy`**
- `AliTMinuitToolkit*fitter4 = AliTMinuitToolkit::Fit(gr,"pol1H","", "", "funOption(2,3,2)");` **`// logLike: huber`**
- `AliTMinuitToolkit*fitter5 = AliTMinuitToolkit::Fit(gr,"pol1R","bootstrap20", "","funOption(6,3,4)");` **`// bootstrap20`**
- `AliTMinuitToolkit*fitter6 = AliTMinuitToolkit::Fit(gr,"pol1R","bootstrap100", "", "funOption(3,3,5)" );` **`// bootstrap100`**
- `fit6=*(fitter6->GetParameters());  rms6=*(fitter6->GetRMSEstimator());`
- `AliTMinuitToolkit*fitter7 = AliTMinuitToolkit::Fit(gr,"pol1R","misac(5,200)","","funOption(3,3,6)");` **`// misac 20`**

To do

Test fits in N-dimensions for fit of the distortion models

- fit parameters - space charge density in hotspots

Improve MISAC/RANSAC method and use correlation

Implementation of other robust methods