RFit
Ring Fitter for RICH

A.Soloviev, A.Stadnik and G.Ososkov Version: 2.0 (16/06/2005)

RFit implements an algorithm estimating ring parameters over a set of scattered points (usually given by RICH detectors). It is based on minimization of a deviation between a sought ring and a given data points. RFit uses a MINUIT package to perform the minimization. It supports both standard and robust approaches. It also allows to resolve parameters of two overlapped rings by fitting them at one run.

Contents

1 What is RFit for?

RFit implements an algorithm estimating ring parameters over a set of scattered points (usually given by RICH detectors). The given set of points should contain a ring to be found by RFit. RFit finds this ring among both some parts of nearby rings and noise points that contaminate the given data. If the given data contain more than one ring, RFit may need a ring guidance as a rough approach for a sought ring (initial values of its parameters). RFit also can fit two close overlapping rings at the one time, if user specifies this.

2 What is inside?

RFit algorithm is based on minimization of a least square function (LSF), which is a measure of deviation between a sought ring an a given data points (x_i, y_i):

$$\sigma^2 = \frac{\sum_{i=1}^{n} d_i^2}{n}, \tag{1}$$

where $d_i = \left|\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - r\right|$, point (x_0, y_0) is the ring center, r is its radius.

RFit uses the MINUIT package [1] to perform a minimization. To be more exact, ROOT::TMinuit class is used[1]. This package was originally written in Fortran by Fred James and part of PACKLIB (patch D506) and has been converted to a C++ class by R.Brun. The current implementation in C++ is a straightforward conversion of the original Fortran version.

This approach often fails for highly contaminated data, thus, a robust fit should be used [2, 3, 4]. LSF (1) should be replaced by a weighted LSF:

$$\sigma^2 = \frac{\sum_{i=1}^{n} w_i d_i^2}{\sum_{i=1}^{n} w_i}, \tag{2}$$

where weights w_i are recalculated before a minimization procedure. RFit uses well known Tukey's bi-square weights:

$$w_i = \begin{cases} \left(1 - \frac{d_i^2}{c_T^2 \sigma^2}\right)^2, & d_i < c_T \sigma, \\ 0, & d_i \geq c_T \sigma. \end{cases} \tag{3}$$

In general, the robust approach looks like:

- setting $w_i = 1$, calculating σ^2 by (2), setting $c_T = 5$;
- while $c_T > 1$ doing the loop:
  - calculating w_i by (3) using current values of σ and c_T;
  - minimizing σ^2 given by (2);
  - reducing c_T by $c_T / = 2$.

The robustness slows the algorithm, of course, but it is an inevitable pay for repairing its accuracy and efficiency in highly contaminated cases. RFit supports both standard and robust approaches.

[1] To use RFit both with ROOT and without it, TMinuit class is separated from the ROOT package by slight modifications and is included in the RFit distribution.

The ring guidance when given acts as follows. Weights are used both in standard and in robust approaches (in the first case $w_i = 1$), but they are cut before minimization: $w_i = 0$ if

$$\sqrt{(x_i - x_{guidance})^2 + (y_i - y_{guidance})^2} > 1.5 r_{guidance}.$$

Simultaneous Fit of Two or more Circles (see [6]). For solving the problem of simultaneous fitting of two or more circles it is necessary to create the equation of such a combined curve. It can be done by multiplying the corresponding number of the circle equations. For instance, for the forth order curve joining two circles it gives the six parameter equation

$$F(x_i, y_i; a, b, c, d, R_1, R_2) = \left[\sqrt{(x-a)^2 + (y-b)^2} - R_1\right]\left[\sqrt{(x-c)^2 + (y-d)^2} - R_2\right] = 0. \quad (4)$$

The LSM estimation of all parameters requires the search for the global minimum of the non-linear functional

$$L(a, b, c, d, R_1, R_2) = \sum_i w_i F^2(x_i, y_i; a, b, c, d, R_1, R_2). \quad (5)$$

with the optimal weight function

$$w_{opt}(t) = \frac{1+c}{1 + c\exp(t^2/2)}. \quad (6)$$

The only parameter c is the ratio of the mean number of noise observations within a strip of the width $\sigma\sqrt{2\pi}$ to the mean number of useful observations in the sample. Thus it is determined by the contamination of data not in the whole range of the sample but within its essential part where all useful observations are practically concentrated. The value of c is often roughly known in experimental models. The approximation of $w_{opt}(t)$ by a polynomial of the forth order gives us the famous Tukey weights [7], which are used for calculating of the weight initial values. Coming back to the circle fitting procedure we evaluate (6) using Tukey's weights. Then denoting by k the iteration number one obtains from (6):

$$w_i^{(k+1)} = \frac{1+c}{1 + c\exp(e_i^{(k)}/var^{(k)})^2/2}, \quad (7)$$

where $(var^{(k)})^2 = \sum w_i^{(k)} (e_i^{(k)})^2 / \sum w_i^{(k)}$.

One often uses Tukey's biweights [7] to speed up calculations

$$w^T(x) = \begin{cases} (1 - (x/c\sigma_0)^2)^2, & |x| < c\sigma_0, \\ 0 & \text{otherwise,} \end{cases}$$

with $3 \leq c \leq 6$.

3 Main features

3.1 RFit algorithm features

The features, such as efficiency and accuracy, depend on several parameters. We require

- a mean number of points in a sought ring $\widehat{m} > 10$
- a distance between a sought ring and a neighbor one $\delta > 1.2r$
- a relative number of noise points **0.1**

Satisfying these requirements, RFit provides the following main features:

- efficiency: $\approx$ **80%**
- accuracy for radius: $\approx$ **25%** of the unit used
- accuracy for center coordinates: $\approx$ **30%** of the unit used
- productivity: $\approx$ **100** calculations of σ (FCN calls)

3.2 RFit package features

The features are:

- group: stand-alone console application
- language: C++
- platforms: UNIX/Linux, Windows (without visualization)
- interface: command line
- input: ASCII file
- output: console
- visualization: ROOT-based

4 Testing

To test RFit, the local model of CBM RICH data was used. The program implementing it generates a requested number of pads situated randomly along a circle with the requested parameters and contaminated by some noise pads. Those pads, which imitate a photon signal, form a cluster simulating a discrete gaussian-like energy distribution with a constant semi-width. Then a circle parameters will be estimated by FIT subroutine. The requested (and estimated) parameters are:

- $\boldsymbol{npt}$ — total number of pads of the circle and noise pads ($\boldsymbol{npt \leq 500}$),
- $\boldsymbol{r}$, $\boldsymbol{a}$, $\boldsymbol{b}$ — radius, co-ordinates of the center of the circle,
- $\boldsymbol{sig}$ is a photon uncertainty mean square,
- $\boldsymbol{sigp}$ is a photon semi-width,
- $\boldsymbol{noise}$ — number of noise pads, uniformly distributed in a rectangle surrounding the circle,
- $\boldsymbol{var}$ - mean square deviation of points from a fitted circle.

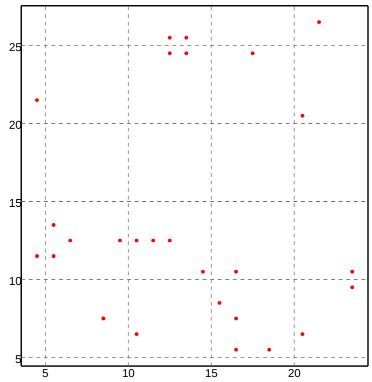

Figure 1: Model data example: one event.

4.1 Efficiency

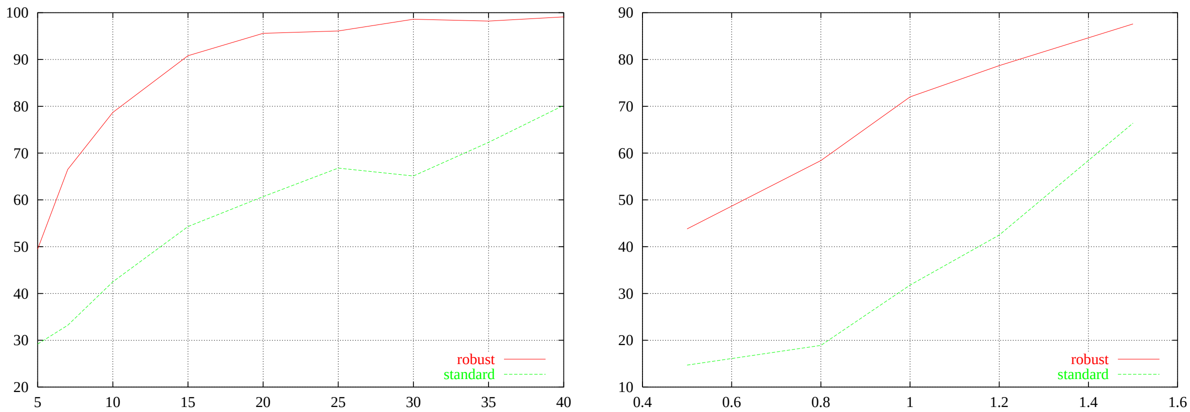

Figure 2: Efficiency vs mean number of photons per ring $\widehat{\boldsymbol{m}}$ for $\boldsymbol{\delta = 1.2r}$ (left) and vs distance to neighbor ring $\boldsymbol{\delta}$ for $\widehat{\boldsymbol{m}} = \mathbf{10}$ (right).

RFit efficiency is calculated as

$$\epsilon = \frac{N - N_{failed}}{N},$$

where $N = 1000$ is the total number of events processed by RFit, N_{failed} is the number of unrecognized events: those for that $|x_0 - x_0^{exact}| > 1$ or $|y_0 - y_0^{exact}| > 1$ or $|r - r^{exact}| > 1$.

4.2 Accuracy

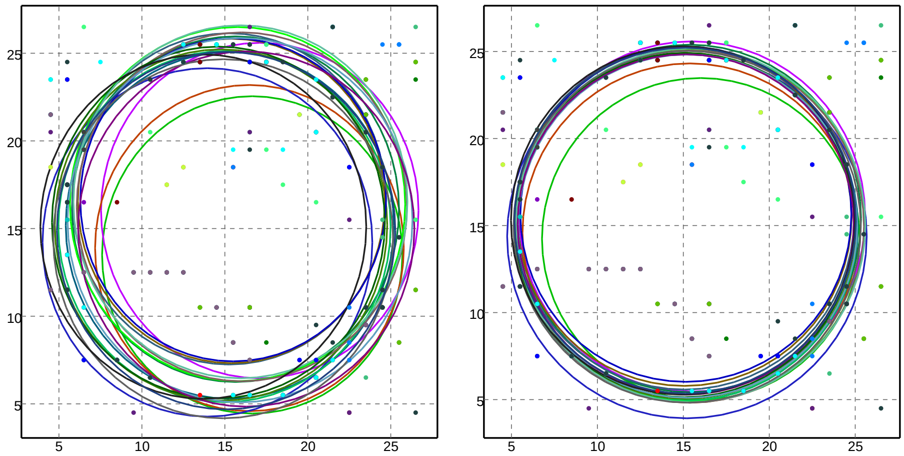

Figure 3: Rings recognition example: in a standard approach (left) and in a robust one (right). Total number of events is 20, $\widehat{m} = 10$ and $\delta = 1.2r$ for each event.

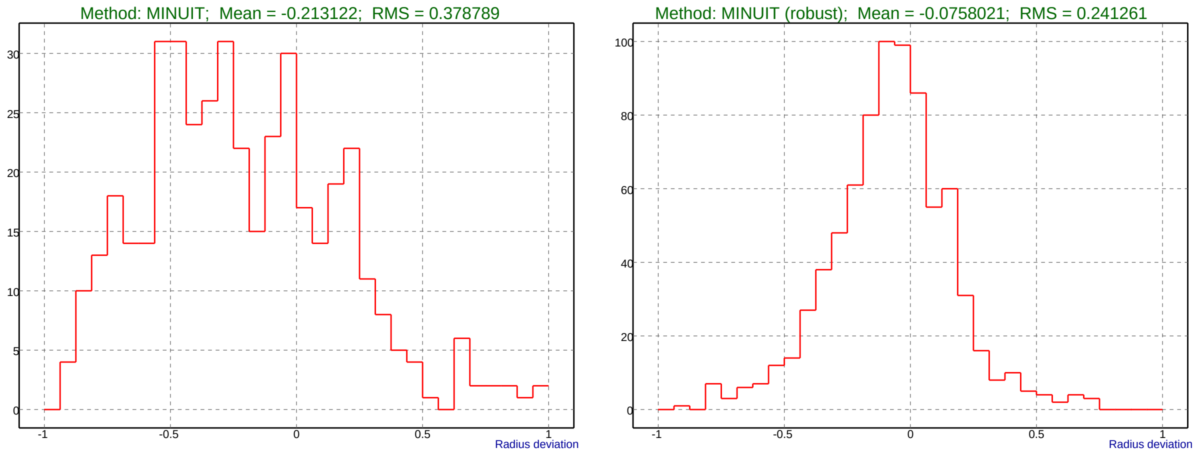

Figure 4: Ring radius deviations: in a standard approach (left) and in a robust one (right). Total number of events is 1000, $\widehat{m} = 10$ and $\delta = 1.2r$ for each event.

In the distributions for the ring radius deviation only recognized rings are taken into account. Distributions for ring center coordinates look the same.

4.3 Productivity

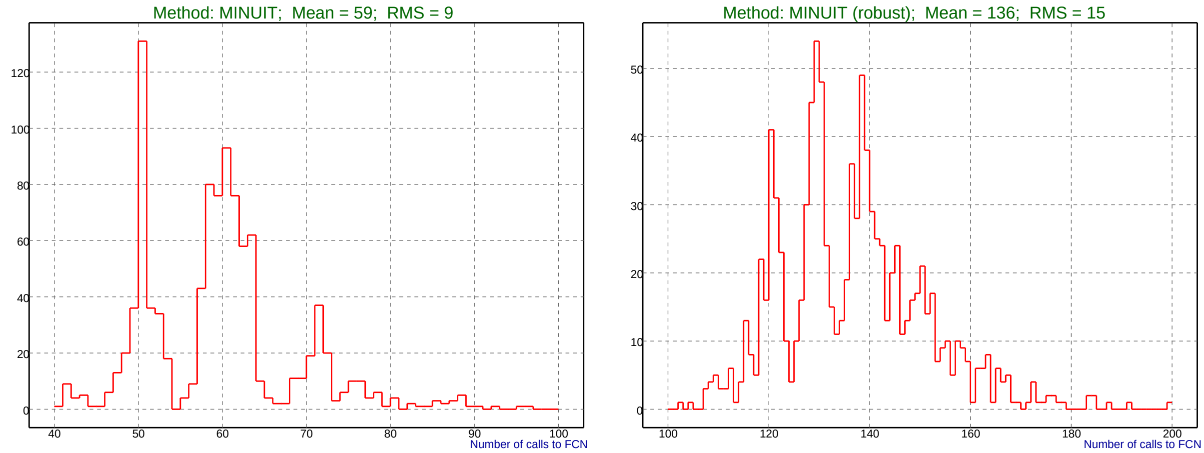

Figure 5: Needed iterations number distribution: in a standard approach (left) and in a robust one (right).

Each event processing takes $\approx \mathbf{3} \times \mathbf{10^{-4}}$ s in a standard approach and $\approx \mathbf{7} \times \mathbf{10^{-4}}$ s in a robust one on the Pentium III, 1000MHz.

4.4 Ring guidance feature

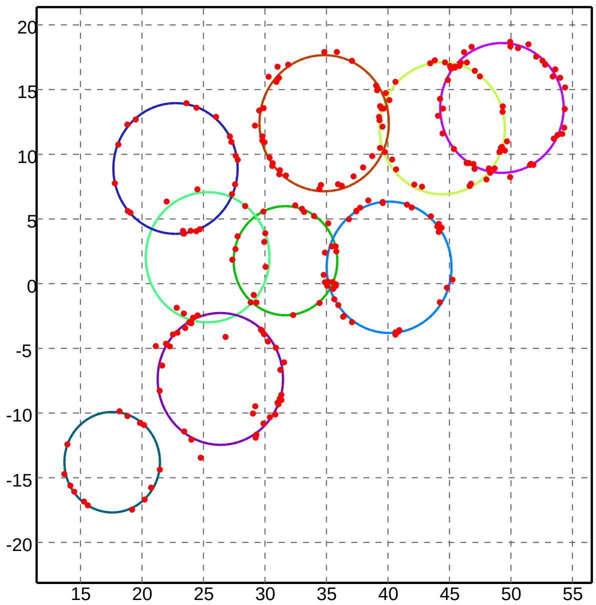

Figure 6: Multiple rings recognition using ring guidances

Due to high multiplicity of CBM events, one has several hundreds of overlapped rings in the each event. In such conditions a ring guidance is really needed. It could be provided by prolongating particle tracks from STS to the focal plane of RICH detector to determine the rough ring center. Besides, the rough estimation of each particle momentum should be also done to provide the ring radius. RFit works well with the data containing more than one ring using the given guidances for each of them (see fig.6).

4.5 Overlapping rings fitting feature

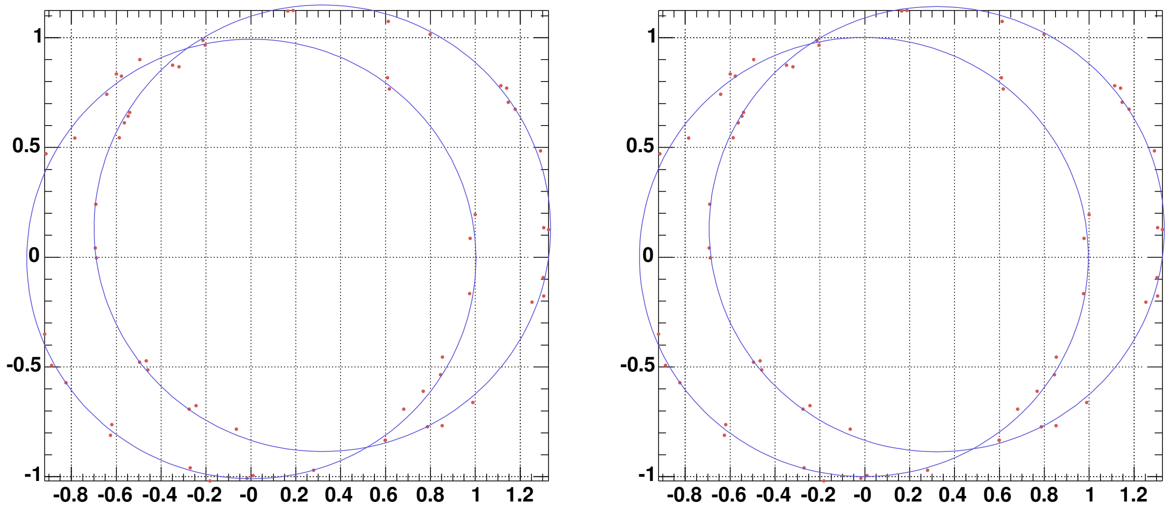

Figure 7: Overlapped rings recognition: in a standard approach (left) and in a robust one (right).

An example of resolving parameters for two close overlapping rings is presented on fig.7. The parameters used for rings simulation were the following: center = (0,0) and radius = 1 for the first ring, center = (0.33,0.12) radius = 1.01 for another.

As one can see from the results of fitting the robust approach gives higher accuracy and requires less calls of the minimization procedure.

```
FCN=2.0908e-05 FROM MIGRAD    STATUS=CONVERGED      92 CALLS          93 TOTAL
center = (0.00170912,-0.007945) radius = 1.00081
center = (0.318086,0.131953) radius = 1.01703

FCN=1.46533e-06 FROM MIGRAD    STATUS=CONVERGED      38 CALLS          170 TOTAL
center = (-0.00415244,0.0011299) radius = 1.00004
center = (0.320362,0.127882) radius = 1.01405
```

5 Installation

In order to compile and install RFit on your system as a stand-alone application, you should type the following in the base directory of the RFit distribution:

```
% ./configure
% make
% make install
```

`configure` script attempts to find ROOT Framework [5]. Thus, you should install it first. Otherwise, RFit will be build with its own MINUIT and with no visualization.

Since RFit uses `autoconf`, no trouble arises when compiling it. If you face any problems, please report about them to the authors.

6 Usage

Type `rfit --help` in your command line to get information on RFit options:

```
Usage: rfit [options] file(s)
Options: [defaults in brackets after descriptions]
Help options:
  -h, --help                display this help and exit
  -V, --version             display version information and exit
Output options:
  -s, --show [file]         display data and resulting ring [no]
  -t, --timer               display elapsed time [no]
Processing options:
  -g, --guidance <value>    sought ring guidance [data driven]
  -r, --robust              robust fitting [no]
  -d, --double              fit two rings in one run [no]
  -v, --verbose             output processing information [no]
  -F, --fast                using experimental fast fit [MINUIT]
```

References

[1] http://wwwinfo.cern.ch/asdoc/minuit/minmain.html

[2] *P.Huber*, Robust statistics. Wiley, New-York (1981)

[3] *G.Ososkov*, Elastic arm methods of data analysis as a robust approach. Tatra Mt. Math. Publ. **26** (2003), 291-306

[4] *N.Chernov, E.Kolganova, G.Ososkov*, Robust Methods for the RICH Ring Recognition and Particle Identification. Nuclear Inst. Meth. A433 (1999) 247-278.
http://thsun1.jinr.ru/~kea/RICH98/r98.html

[5] http://root.cern.ch

[6] *G.Ososkov, G.A.Agakishiev, Yu.Panebratzev*, Robust methods for fast fitting of one or a few crossing circle simultaneously. Proceed. of the 6-th Intern. Conference on Phys.Comp., PC'94, European Phys. Society, Geneva, 1994, 361-364.

[7] *F.Mosteller, W.Tukey*, Data analysis and regression: a second course in statistics. Addison - Wesley (1977).