Modification of the subroutine Fumili (in memory of Prof. I.N.Silin)

I.M. Sitnik

Joint Institute for Nuclear Research

Abstract

More convenient and friendly user interface. No restriction for numbers of parameters and experimental points. Speed advantage when number of parameters is high enough. Additional packet of subroutines, including fit tracks via straws. Programms are written by using FORTRAN-90.

1 Introduction

This work is done in memory of outstanding mathematician Prof. I.N.Silin, who went out on 21 Nov 2006. His famous program FUMILI[1-4] was the first one The algorithm of transformation of he error matrix was permanently improved by him. The last changes were done in 2002. On the other hand, the user interface, which was fixed in sixties as beta-version, was kept unchangeable. New user interface, suggested in this paper, was discussed between me and I.N.Silin in part and was supported by him, but at this time he was fascinated by (he was devoted completely to) the another task (constrained data fitting)[4].

Another problem is the broad class of tasks arose when the number of parameters much more than 100, which is declared as the maximum number of parameters for FUMILI. Let us consider the simplest example of a source of a huge number of parameters. Let we have

$$N_t = N_0 + N_1 N_s, \tag{1}$$

where N_t is total number of the parameters, N_0 is the number of parameters, common for all events, N_1 is the number of parameters, common for some subgroup, N_s is the number of subgroups. When adjusting big setup offsets, such (or more complicated) combination arises frequently with large enough N_s. In author's practice the total number of parameters and events achieved several hundreds and about of 2 millions, respectively. The last library version of the source program allows one to have only 70 free parameters, and only an advanced user is capable to improve the situation, if he have the FORTRAN source. The matter is in this case it is necessary to redefine not only user COMMON-blocks, but also several COMMON-blocks for only inner use. On the other hand, the huge length of COMMON-blocks (especially for error matrixes, about of N_t^2) can be inconvenient for tasks with the small numbers of parameters. In new program all arrays, declared earlier in COMMON-blocks were replaced by adjustable ones. Unavoidable slowing down of the program in this case was compensated by changing of programming style proceeding from no limitation for the task-memory. Recall, that memory of the USSR leader of sixties computer, BESM-6, had about 32k words for user tasks, and the original was written with care of computer memory, often to the detriment of speed. The basic cod of FUMILI (the error matrix transform) during this optimization was kept unchangeable.

To reduce number of "a, the..." let us assume below that User is the name.

2 User guide

2.1 FUMILI

For better understanding of the new program, let us recall user guide for old one. The CALL has the form

CALL FUMILI(S,NPAR,N1,N2,NIT,EPS,AKAP,ALAM,IPR,IEND).

Here **input parameters:**
NPAR – number of parameters,
N1=2, N2=1 (recommended values),
NIT – restriction for number of iterations,
EPS – desirable accuracy (recommended values are 0.01-0.1),
IPR – to print each ith iteration (IPR < 0 – no print),
IEND – flag of conditions of the exit from the program.
output parameters:
S=$\chi^2/2$,
AKAP, ALAM, NC – parameters, characterized convergence.
An array of experimental points should be placed at
COMMON/EXDA/EXDA(LEXD,NEXD),
where LEXD and NEXD are number of words in each experimental point and number of experimental points, respectively. These values should be defined in
COMMON/NED/NED(2)
as NED(1)=NEXD and NED(2)=LEXD. The content of one experimental point is the following:
EXDA(1,I) – y_i-value of an experimental point;
EXDA(2,I) – σ_i, its error;
EXDA(3,I) – $x_i(1)$ (the first argument) of an experimental point:
EXDA(4,I) – $x_i(2)$ (the second argument) of an experimental point:
EXDA(5...N,I) – x_i(3...(N-2)).
There should be LEXD$\geq$3.

Initial values of parameters (which will be replaced after the fit by final ones), and possible restrictions for them should be defined in

COMMON/A/A(NEXD)/AL/AMIN(NEXD)/AU/AMAX(NEXD),
respectively. User should also define restrictions for parameter steps in
COMMON/PL/PL(NEXD).
If they are not defined or negative, the corresponding parameters will be treated as fixed. After fit the parameter errors are placed in
COMMON/SIGMA/SIGMA(NEXD).

User should write the function FUNCT(X), where X (one- or multidimensional) is arguments of experimental points, depending on parameters A (the corresponding common-block should declared in this function). The numerical computation of derivatives is used in this case. If User prefers to define also first derivatives, he should replace existing in the packet subroutine ARITHM(Y), in which
COMMON/DF/DF(NEXD) (derivative array)
should be also declared. In some linkers the conflict between User's and library subroutines is possible. **Here we have the following alternative: either all or no one derivatives should be defined.**

2.2 FUMILIM

The CALL has the form

H=FUMILIM(A,NAM,NPAR,EXDA,LEXD,NEXD,USFUN,NIT,LUN)

Here (input parameters only) NPAR and NIT are defined above,
A(NPAR,5) - array of initial (and final) values for parameters. The equivalence between old and new arrays is the following:
A(I,1) – A(I);
A(I,2) – SIGMA(I);
A(I,3) – PL(I);
A(I,4) – AMIN(I);
A(I,5) – AMAX(I);
NAM(NPAR)- character array of the names of parameters (up to 15 symbols);
EXDA(LEXD,NEXD) – array of experimental points, having the structure described above;
USFUN(X,A,DF) – User's function for data fit (should be declared via EXTERNAL in the calling program),
X,A,DF are argument, parameters and parameter derivatives, respectively; LUN – number of the output unit (lun=0,6 mean the screen output,

other numbers should be associated with file names. Output is switched off when LUN¡0).
The value of the function is χ^2/NDF (number of degrees of freedom).

If User does not want to deal with the parameter names, he can write:

H=FUMILIM(A,'*',NPAR,EXDA,LEXD,NEXD,USFUN,NIT,LUN)

In the new program restrictions for parameters are ±huge numbers, exactly as in the old one. The difference is: in old version these values are defined via DATA-operator, but in new one the value of 0.d0 is considered as an undefined one, and redefinition (huge value) follows in this case.

The treatment of PL(I) is slightly changed. A(I,3)==0.d0 now is considered as undefined value, and the redefinition follows (rather large value). The advantage of such approach is the following. **If, for example, USFUN is linear respectively i-th parameters, the definition of A(3,I) is rather harmful, then senseless (the same result with, may be, larger number of iterations). In particular, when USFUN is linear respectively all parameters, User is free to define nothing in the array A. And a task will be solved during one iteration.** The definition of A(3,I) is hardly useful, when the corresponding first derivative are defined analytically. There are many other cases, when User is free from definition of A(I,3). So, it is recommended not to define A(I,3) always when starting the fit. And only if the result is suspicious, to start to experimentalize with values of A(3,I).

One can see, that the input parameters N1,N2,EPS from FUMILI now are not in the list of parameters. They are defined inside the program via DATA-operator by recommended values, but advanced User can redefine them in

$$\mathrm{COMMON/FUMINPUT/EPS, N1, N2, NPAROPT.} \tag{2}$$

The disappeared output parameters AKAP,ALAM,IEND can be found in
COMMON/FUMOUTPUT/HISQ,IEND,NITER,AKAPPA,ALAMBD.
Here HISQ=χ^2, NITER is the number of real iterations, values of IEND are the following:

1 –MINIMIZATION IS FINISHED SUCCESSFULLY
2 –MINIMIZATION IS FINISHED SUCCESSFULLY IN PART WHILE

SOME PARAMETERS ARE FIXED ON BOUND

3 –MINIMIZATION TERMINATED AS ALL PARAMETERS ARE FIXED ON BOUND

4 –ITERATION LIMIT IS REACHED

5 –MINIMIZATION IS FINISHED SUCCESSFULLY IN PART WHILE SOME PARAMETERS ARE UNDEFINED

6 –MINIMIZATION TERMINATED AS NO FURTHER DECREASE IN CHISQ IS OBTAINABLE

7 –ALL PARAMETERS ARE FIXED

8 –MINIMIZATION TERMINATED AS ALL PARAMETERS EITHER FIXED ON BOUND OR HAVE INFINITE ERROR ESTIMATION.

User should declare A,DF (and, may be, X) in USFUN(X,A,DF) as arrays. **In this function User is free to define or not each DF(i), and that is the serious advantage in comparison with the old version.**

It is used 2 level of the speed optimization. First of them is applied, when parameter boundaries are completely absent, and number of free parameters is greater than 15. This value can be redefined in NPAROPT (Eq.2). This optimization has a sense, when N_s (Eq.1)is large enough and $N1 > 0$. The error-matrix has many empty elements in this case. Special kind of optimization is applied when NPAR=2 and LUN¡0 (typical for the track reconstruction). In such a case the handle is delivered to FUMI2PAR. The direct call

H=FUMI2PAR(A,EXDA,LEXD,NEXD,USFUN,NIT) is also possible.

FUMILIM starts to work from User's error check.

1. Number of experimental points should be not less than number of parameters: NEXD $\geq$ NPAR
2. Length of Block of one experimental point should be LEXP ≥ 3
3. $\mathrm{EXDA}(2, \mathrm{I})$ (experimental error) should be positive
4. initial value of parameter should be within up- and down-limits for it: $\mathrm{A}(4, \mathrm{I}) \leq \mathrm{A}(1, \mathrm{I}) \leq \mathrm{A}(5, \mathrm{I})$

Error number 4 arises frequently when the real structure of EXDA-array does not match with its definition via LEXD,NEXD. Each negative check is **FATAL** for User's program. Error check is switched off, when LUN < 0.

2.3 FUMIDOUB

It is widely spread such procedure as the double fit, when the second step is fulfilled after removing of so called "bad points". When adjusting physical detectors offsets, one deal with huge number of events, and "bad points" form statistically meaningful wings of the $(y_i - Y(x_i))/\sigma_i$-distribution. Here $Y(x_i)$ is the value of fitted function for x_i arguments of the i-th experimental point. If such a distribution is far from even(symmetrical respectively zero), that is guarantee of systematic shifts in fitted parameters respectively right values. The next steps becomes necessary in this case. If to use the simplest way, based on comparison $\chi_i^2 < C$, the correct solution demands several iterations, shrinking step by step the corridor for "good points". Here the two-step procedure is suggested. The algorithm is described in section 3. The call is looking as

H=FUMIDOUB(A,NAM,NPAR,EXDA,LEXD,NEXD,USFUN,NIT,LUN)

with the same list of parameters as in FUMILIM. The work of FUMIDOUB is organized by the following way. Firstly the ordinary call to FUMILIM is fulfilled. Then, the parameters y_0, σ of the $(y_i - Y(x_i))/\sigma_i$-distribution is calculated. (For symmetric distributions $y_0 \simeq= 0$). Then the internal EXDA-array is formed with the conditions $(y_i - y_0 < k\sigma$, where k=2.5 is defined in COMMON/FDOUB/k via DATA-operator and can be redefined by User. Then the second call to FUMILIM is fulfilled with reference to new EXDA-array and recalculated value of NEXD.

2.4 FUMSTRAW

The track reconstruction via straws $(x = A * z + B)$is not linear respectively A and B parameters, and there is large probability to find the local minimum. To find the global minimum, four sets of initial parameters values are analyzed firstly in FUMSTRAW (1-iteration call to FUMILIM for each set), then for the set, provided the least χ^2 the call to FUMILIM with rather large iteration limit is being done. But, as a rule (99%), the convergence is achieved during 0–1 iterations. The initial conditions are four possible sets of parameters for two tubes with smallest and largest z-coordinates.

The call have the form:

H=FUMSTRAW(A,B,EXDA,NEX),

where
EXDA : input array of experimental points;
EXDA(1,i): space radius (not time);
EXDA(2,i): radius error;
EXDA(3,i): z-coordinate of wire (along the beam);
EXDA(4,i): x-coordinate of wire (transversal to z).
All values in EXDA should be in the same units. The ascending (descending) order of z-coordinates is assumed. At least, the tubes with largest and smallest z-coordinates should be the first and the last.

2.5 ERREXDA

This program is the analog of the original program ERRORF. It calculates theoretical confidence band for each experimental point. The results, together with the experimental point data will be written to the file, associated with LUN. Here the Silin's algorithm is used. The CALL has the form

CALL ERREXDA(A,EXDA,FUNCT,LUN),

where all parameters are described above.

2.6 ERRFUNCT

This program calculates the confidence band as set of points, which allows one to present data as a graph. The same, as in the previous program, algorithm is used. The CALL has the form

CALL ERRFUNCT(ALIM,BLIM,A,EXDA,FUNCT,LUN).

Here ALIM and BLIM are lowest and highest values of the argument, respectively. The other parameters are described above. The number of points is of 100. It is defined in COMMON/ERRNUMB/N via the DATA-operator and can be redefined by User. It is assumed that experimental points have only one argument. The structure of he output array is the following:
$x(1), y(1), y_{min}(1), y_{max}(1)$
...
$x(N), y(N), y_{min}(N), y_{max}(N)$

For another case the following program is envisaged.

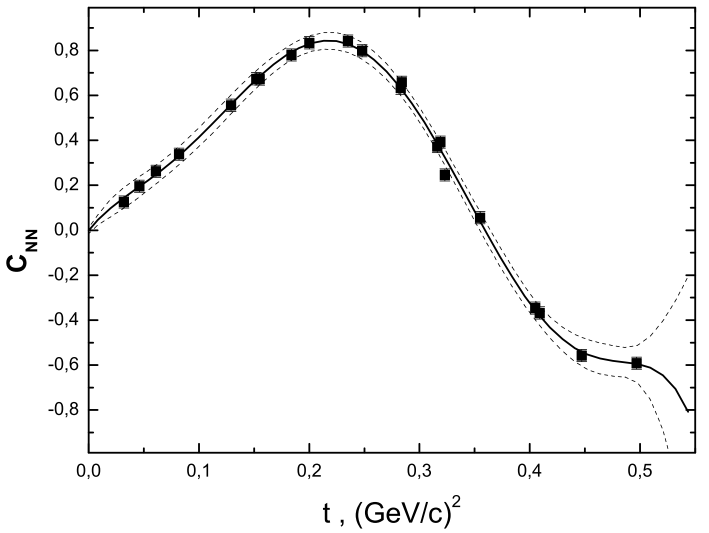

Figure 1: Polinomial Fit of data with calculation of the confidence band.

CALL ERRFMULT(ALIM,BLIM,A,EXDA,FUNCT,IX,X,LUN).

Here IX is the argument number, which will be scanned. The values of other arguments should be defined by User in the X-array, having dimension (LEXD-2).

3 Calculation of distribution parameters for FUMIDOUB

As the first approximation, y_0 and σ are calculated using the following formulas:

$$y_0 = \frac{1}{\mathrm{NEXD}} \sum \frac{y_i - Y(i)}{\sigma_i} \tag{3}$$

$$\sigma^2 = \frac{1}{\mathrm{NEXD}^2} \sum \left(\frac{y_i - Y(i)}{\sigma_i} \right)^2 - y_0^2, \tag{4}$$

where $y_i = (\mathrm{EXDA}(1, \mathrm{I}), \sigma_i = \mathrm{EXDA}(2, \mathrm{I})$, $Y(x_i)$ is the fitted function value for EXDA(3,I)... EXDA(LEXD,I) arguments. ($y_0 \simeq 0$, when the distribution is symmetric.)

Then the 100-bin histogram is filled only for events from the interval

$$|y_i - y_0| < 5\sigma.$$

It is easy to see that the bin width in this case is 0.1σ. Only bins, nearest to the bin with the maximum volume is collected in the second approximation with the condition $w_j < w_{max} \exp(-2)$ (equivalent to condition $|y_j - y_0| < 2\sigma$ for the Gaussian distribution). The following formulas are used:

$$y_0 = \frac{\sum y_j w_j}{\sum w_j} \tag{5}$$

$$\sigma^2 = \frac{\sum y_j^2 w_j}{\sum w_j} - y_0^2, \tag{6}$$

where y_j is the bin argument, w_j is the bin volume. The value of σ, calculated in this approach, is underestimated. For the ideal Gaussian we have is $\sigma = 0.879\sigma_0$, where σ_0 is the right value. This factor is taken into account for the final value of σ. This approach helps to suppress the contribution of the distribution wings and of their asymmetry. It was tested endlessly in author's own histogram packet, close to famous HBOOK. It gives results, close to ones, fitted by Gaussian+background function. For histograms with many peaks, this method gives right values for the highest of them. It would be useful to include these values as default ones into HBOOK and ROOT.

4 Testing

A number of testing program and about decade of real ones was used for testing in pair of the old and new programs. No one case was with nonidentical results was found. When all parameters are common for all events and NPAR > 2, the new program works 5-15% faster. The solution time grows as $t \simeq a + c * NPAR^2$. At NPAR $= 2$ (and LUN < 0) speed advantage of the new program is about of 2. When $N_0 + N_1 \ll N_t$ (Eq.1), the solution time grows as $t \simeq a + c *$ NPAR. These two cases (for FUMLIM) are illustrated in Fig.2. The speed of FUMILI is the same for both cases. Thus, speed advantage of the new program in comparison with old one can achieve one order at 100 parameters and two orders at 1000 parameters.

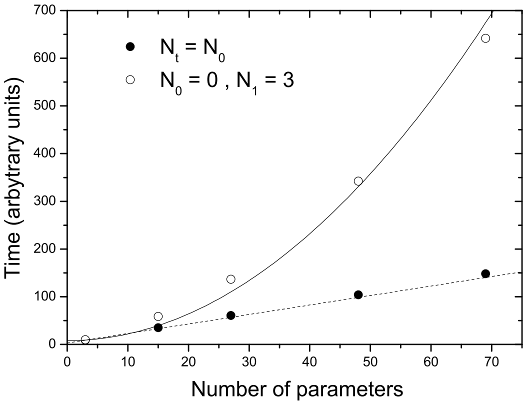

Figure 2: Dependence of the solution time on the number of parameters.

For the linear task respectively all parameters and $N_t = N_0$ the solution is 1.5 faster when all derivatives are defined in comparison with the case when no one derivative is defined. When $N_t \gg N_0 + N_1$, the situation changes to the contrary one. This strange on the first glance result is, nevertheless, natural. The matter is, when, after the first call to USEFUN, some DF(i)=0.d0, the second call with numerical computation of DF(i) follows. In case of $N_t \gg N_0 + N_1$ such a situation take place in the majority of cases. And each time all derivatives are calculated again what is the kill-time.

The subroutine FUMIDOUB was tested in the real exp.305 data processing(Saturne-2, Saclay). The apparatus offsets were firstly adjusted with help of FUMILI. The preliminary results of this experiment are presented in [6]. Later the same job was fulfilled with the help of FUMIDOUB. In particular, when adjusting straws offsets (time shifts, wire eccentricity and so on), about 250 parameters and two millions events were involved into the fit. As the result, the number of events, passed through the set of criteria, increased appreciably. The elastic peak resolution also became better. The comparison of old results (2005) and present ones are demonstrated in Fig.3.

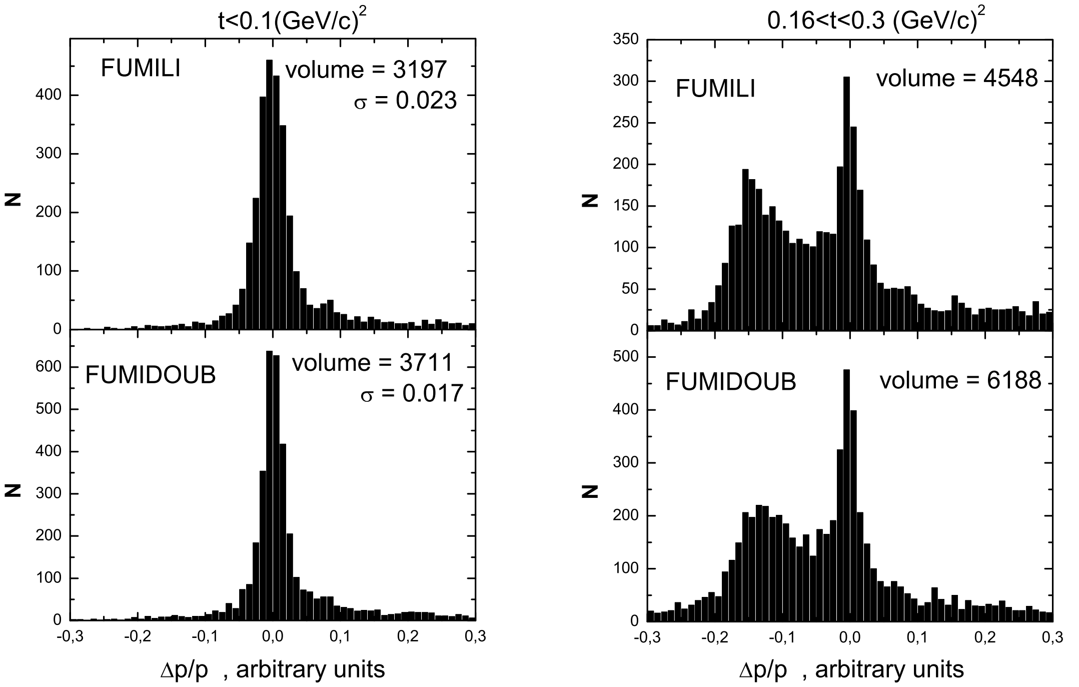

Figure 3: Comparison of the elastic peak separation for two 4-momentum transfer intervals.

References

[1] S.N. Sokolov and I.N. Silin, Preprint JINR D-810, Dubna (1961).

[2] I.N. Silin, CERN Program Library, D510, FUMILI, 1983.

[3] A.J. Ketikian, E.V. Komissarov, V.S. Kurbatov, I.N. Silin, NIM A 314 (1992) 578.

[4] V.S. Kurbatov, I.N.Silin, NIM A 345 (1994) 346.

[5] S.N. Dymov, V.S. Kurbatov, I.N. Silin, S.V. Yashenko, NIM A 440 (2000) 431.

[6] I.M.Sitnik et al.,