Optics Minimization in LISE^{++}

Purpose:
Minimization of optics of existed separators
First step: quad fields

v.9.10.100
from 05/22/15

1. Introduction
2. New Block "Fitting constraints"
3. Selecting a block to minimize
4. Run minimization
5. Examples
6. levmar example

Recently the first stage of optics minimization procedure was introduced, based on the "levmar" package by M.I.A. Lourakis using the Levenberg-Marquardt nonlinear least square algorithm. At this stage only the quadrupole fields can be varied to minimize user constraints for matrix and beam ellipse elements. In the future this minimization procedure will be used to define curved profile shape, fragment spatial distributions in Monte Carlo mode, and optimize intensity/purity combination.

Based on

levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. M.I.A. Lourakis July 2004. http://users.ics.forth.gr/~lourakis/levmar

Minimization for

- *E-blocks (extended configurations)*
- *with non-linked matrices*
- *set the option "Allow remote matrices recalculation"*

Introduction: "levmar" package

users.ics.forth.gr/~lourakis/levmar/

levmar : Levenberg-Marquardt nonlinear least squares algorithms in C/C++

Home About FAQ Changelog Download Contact

If you are looking for a general-purpose sparse Levenberg-Marquardt C/C++ implementation, please have a look at sparseLM.

Introduction

This site provides GPL native ANSI C implementations of the Levenberg-Marquardt optimization algorithm, usable also from C++, Matlab, Perl, Python, Haskell and Tcl and explains their use. Both unconstrained and constrained (under linear equations, inequality and box constraints) Levenberg-Marquardt variants are included. The Levenberg-Marquardt (LM) algorithm is an iterative technique that finds a local minimum of a function that is expressed as the sum of squares of nonlinear functions. It has become a standard technique for nonlinear least-squares problems and can be thought of as a combination of steepest descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method.

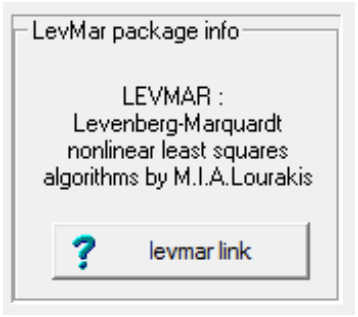
Technical Overview

`levmar` includes double and single precision LM C/C++ implementations, both with analytic and finite difference approximated Jacobians. It is provided free of charge, under the terms of the GNU General Public License. The mathematical theory behind unconstrained `levmar` is described in detail in the lecture notes entitled *Methods for Non-Linear Least Squares Problems*, by K. Madsen, H.B. Nielsen and O. Tingleff, Technical University of Denmark; Matlab implementations of the algorithms presented in the lecture notes are also available. Note however that the formulation of the minimization problem adopted here is slightly different from that described in the lecture notes. There is also a short note, providing a quick overview of the material in the lecture notes.

To deal with linear equation constraints, `levmar` employs variable elimination based on QR factorization, as described in ch. 15 of the book Numerical Optimization by Nocedal and Wright. For the box-constrained case, `levmar` implements the algorithm proposed by C. Kanzow, N. Yamashita and M. Fukushima, Levenberg-Marquardt methods for constrained nonlinear equations with strong local convergence properties, Journal of Computational and Applied Mathematics 172, 2004, pp. 375-397.

`levmar` provides the following two options regarding the solution of the linear systems formed by the augmented normal equations:

1. If you have LAPACK (or an equivalent vendor library such as Intel's MKL, AMD's AMCL, Sun's performance library, IBM's ESSL, SGI's SCSL, NAG, ...), the included LAPACK-based solvers can be used. This is the default option. The employed solver is based on the LU decomposition. Additionally, for experimenting with other approaches, linear solvers based on the Cholesky and QR decompositions have been supplied.
2. If LAPACK is unavailable, a LAPACK-free, LU-based linear systems solver can be used by undefining `HAVE_LAPACK` in `levmar.h`.



- Unconstrained optimization
  - `dlevmar_der()`: double precision, analytic Jacobian
  - `dlevmar_dif()`: double precision, finite difference approximated Jacobian
  - `slevmar_der()`: single precision, analytic Jacobian
  - `slevmar_dif()`: single precision, finite difference approximated Jacobian
- Constrained optimization
  - `dlevmar_lec_der()`: double precision, linear equation constraints, analytic Jacobian
  - `dlevmar_lec_dif()`: double precision, linear equation constraints, finite difference approximated Jacobian
  - `slevmar_lec_der()`: single precision, linear equation constraints, analytic Jacobian
  - `slevmar_lec_dif()`: single precision, linear equation constraints, finite difference approximated Jacobian
  - `dlevmar_bc_der()`: double precision, box constraints, analytic Jacobian
  - `dlevmar_bc_dif()`: double precision, box constraints, finite difference approximated Jacobian
  - `slevmar_bc_der()`: single precision, box constraints, analytic Jacobian
  - `slevmar_bc_dif()`: single precision, box constraints, finite difference approximated Jacobian
  - `dlevmar_blec_der()`: double precision, box & linear equation constraints, analytic Jacobian
  - `dlevmar_blec_dif()`: double precision, box & linear equation constraints, finite difference approximated Jacobian
  - `slevmar_blec_der()`: single precision, box & linear equation constraints, analytic Jacobian
  - `slevmar_blec_dif()`: single precision, box & linear equation constraints, finite difference approximated Jacobian
  - `dlevmar_bleic_der()`: double precision, box, linear equation & inequality constraints, analytic Jacobian
  - `dlevmar_bleic_dif()`: double precision, box, linear equation & inequality constraints, finite difference approximated Jacobian
  - `slevmar_bleic_der()`: single precision, box, linear equation & inequality constraints, analytic Jacobian
  - `slevmar_bleic_dif()`: single precision, box, linear equation & inequality constraints, finite difference approximated Jacobian
  - Convenience wrappers `xlevmar_blic_der()`/`xlevmar_blic_dif()`, `xlevmar_leic_der()`/`xlevmar_leic_dif()` & `xlevmar_lic_der()`/`xlev` are also provided.

Introduction: minimization scheme

LISE++

1. Select a optical block to minimize,
   Check in a parameter to minimize,
   Set bounds constraint

2. Create a block ”Fitting constraints”
   Set constraints

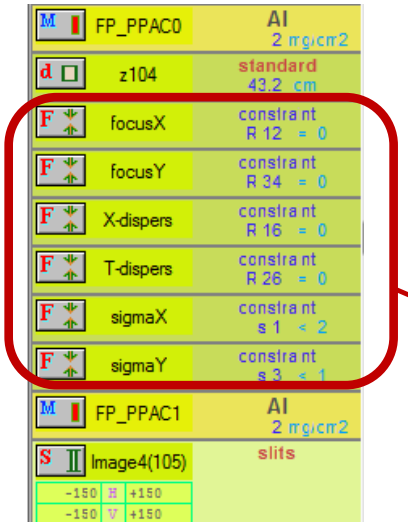
3. Run minimization

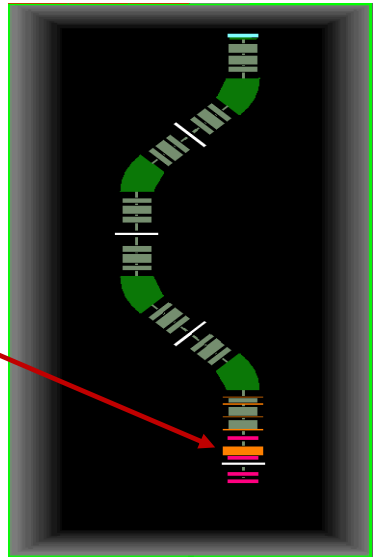
Transport

Command
5.01 "q1B " 0.7 -1.86164 9.75 ;

Command
10.0 "fit1" -2. 6. 0.0 .001 ;

3. Run minimization

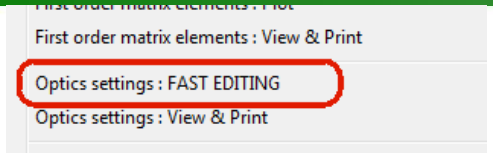




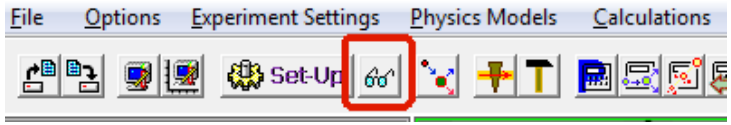
files\examples\NSCL\
A1900_extended_LISE_FIT.lpp

The next file is to append standard constraint blocks
files\examples\
FITconstraints.lpp

Introduction: information about blocks used for minimization



or



Optics settings (fast editing)

Block	Given Name	Start(m)	Length(m)	B0(kG)	Br(Tm)cor/*real	DriftM/*Angle	Rapp(cm)/*R(...	Leff(m)/*Ldip(m)	2 nd order	CalcMatr/*Z-Q	AngAcc,Apps,Slits	COSY | Fit	SE
drift	z089	28.783	0.5640			standard					-- HV --	-	e
= Dipole	D4	29.347	2.4300	+9.6965	* 3.0000	* +45.0	* 3.0939	* 2.4299	yes	* 0	-- HV --	-	S
drift	z097	31.777	0.5260			standard					-- HV --	-	e
Fit	sigY	32.303	0.0000								-- -- --	s3 < 50	e
<Quad>	Q098-8TA	32.303	0.4300	+7.0851	3.0000	QUAD	15.0000	0.4300	yes	1 R	-- HV --	FIT	e
Fit	sigY	32.733	0.0000								-- -- --	s3 < 50	e
drift	z099	32.733	0.1720			standard					-- HV --	-	e
<Quad>	Q100-8TB	32.905	0.7480	-8.1167	3.0000	QUAD	13.3000	0.7480	yes	1 R	-- HV --	FIT	e
Fit	sigY	33.653	0.0000								-- -- --	s3 < 50	e
drift	z101	33.653	0.1756			standard					-- HV --	-	e
<Quad>	Q102-8TC	33.828	0.7480	+4.2117	3.0000	QUAD	13.3000	0.7480	yes	1 R	-- HV --	FIT	e
Fit	sigY	34.576	0.0000								-- -- --	s3 < 50	e
drift	z103	34.576	0.3750			standard					-- HV --	-	e
drift	z104	34.951	0.4320			standard					-- HV --	-	e
Fit	focusX	35.383	0.0000								-- -- --	R12 = 0	e
Fit	focusY	35.383	0.0000								-- -- --	R34 = 0	e
Fit	X-dispers	35.383	0.0000								-- -- --	R16 = 0	e
Fit	T-dispers	35.383	0.0000								-- -- --	R26 = 0	e
Fit	sigmaX	35.383	0.0000								-- -- --	s1 < 2	e
Fit	sigmaY	35.383	0.0000								-- -- --	s3 < 1	e
slits	Image4(105)	35.383	0.0000			SLITS					-- -- HV	-	e
drift	z105	35.383	0.3490			standard					-- -- --	-	e
drift	z106	35.732	0.0890			standard					-- -- --	-	e

Selected block: Dispersive (M-dipole); Block Length [m] 0.0001; Let call automatically; Block name = tuning; Length after this block [m] 0.0001; Charge State (Z-Q) = 0; Selected Block Edit; Mutlipole Edit; Cuts (Acceptances); Optical Matrix

Angular acceptance (mrad): Use; Horizontal ±; Vertical ±; Shape: Rectangle / Ellipse

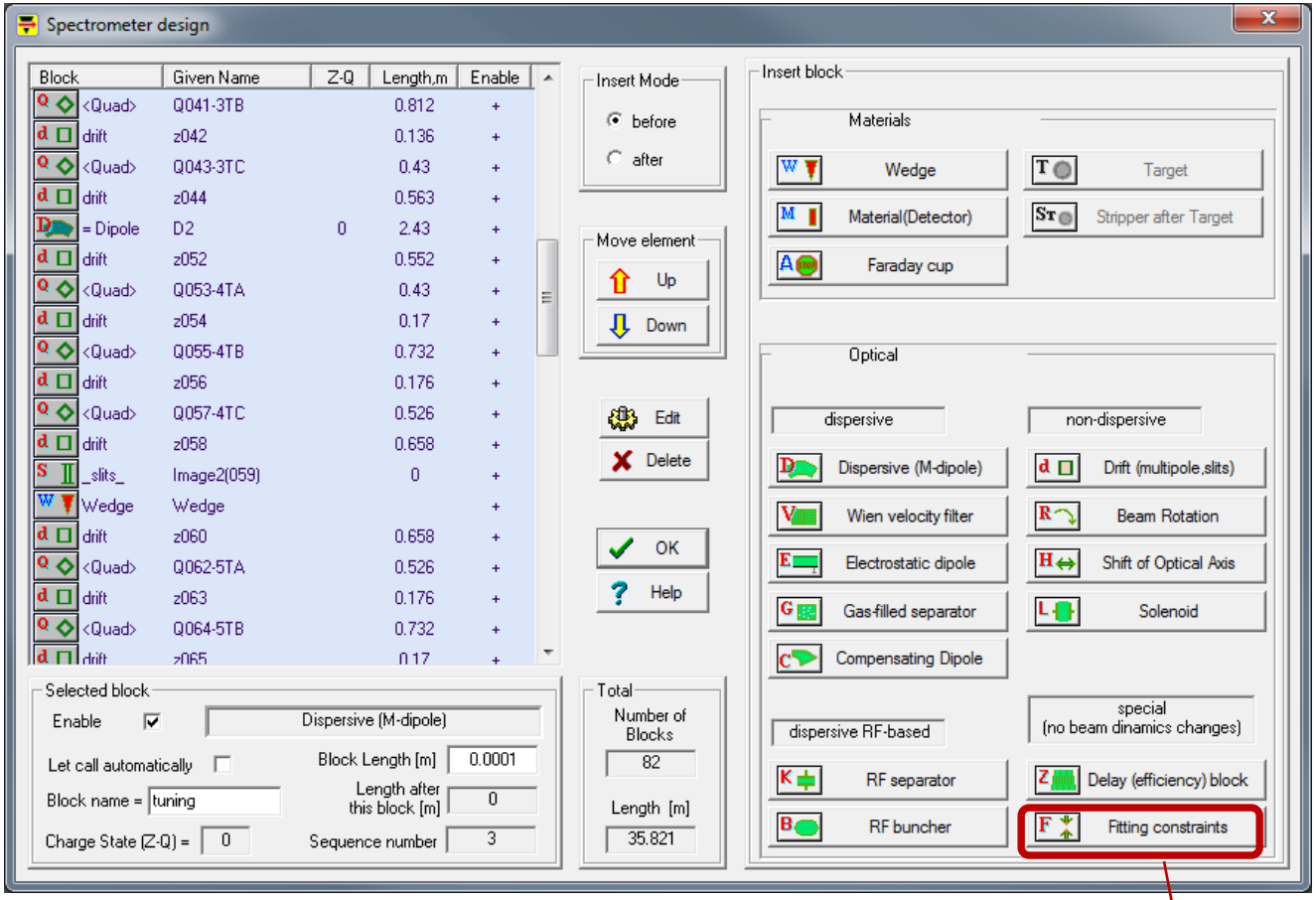
Inside Aperture (mm): min max Use; X = -50 50; Y = -50 50; Shape: Rectangle / Ellipse

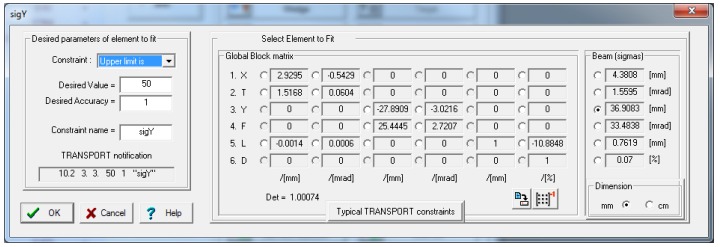
Slits (mm) after this BLOCK: min max Use; X =; Y =; Shape: Rectangle / Ellipse

1-st order Matrix Elements: Matrix Plot; Beam-Sigma Plot; View

Quit; Help

New Optic Block "Fitting Constraints"



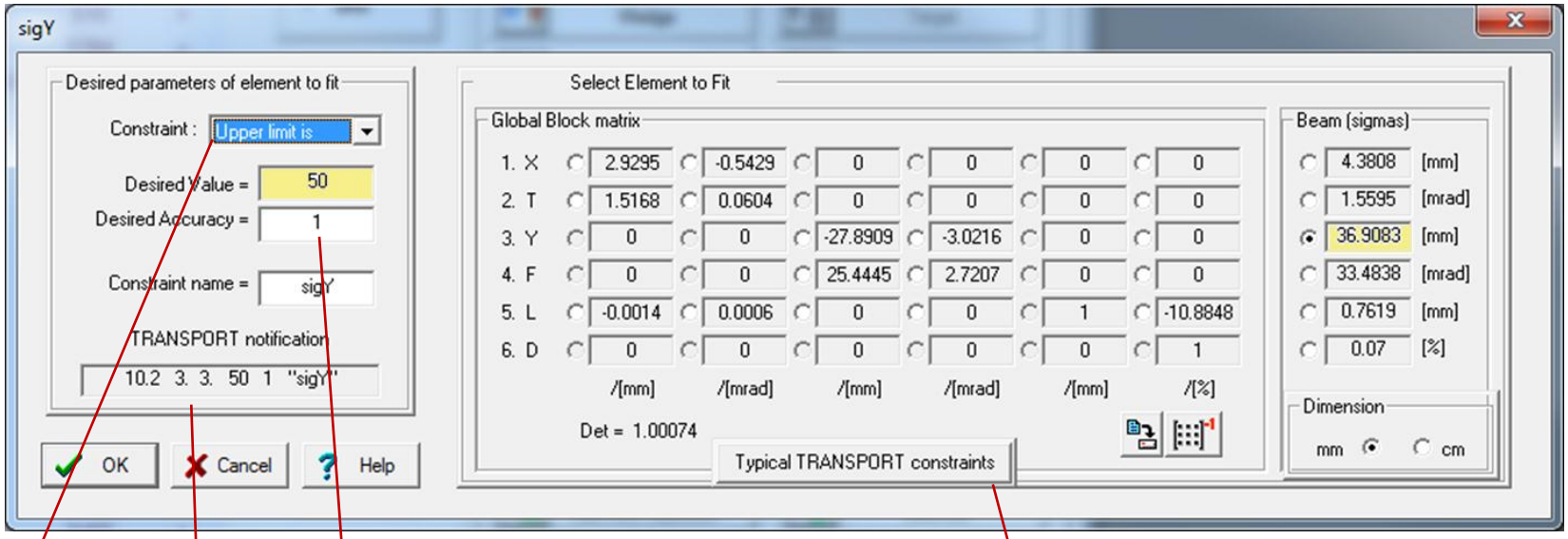


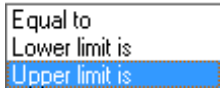
New Optic Block "Fitting Constraints"

42* possible selection for global matrix elements and beam sigma vector

(some matrix elements can be disabled if non rotation or solenoid blocks)

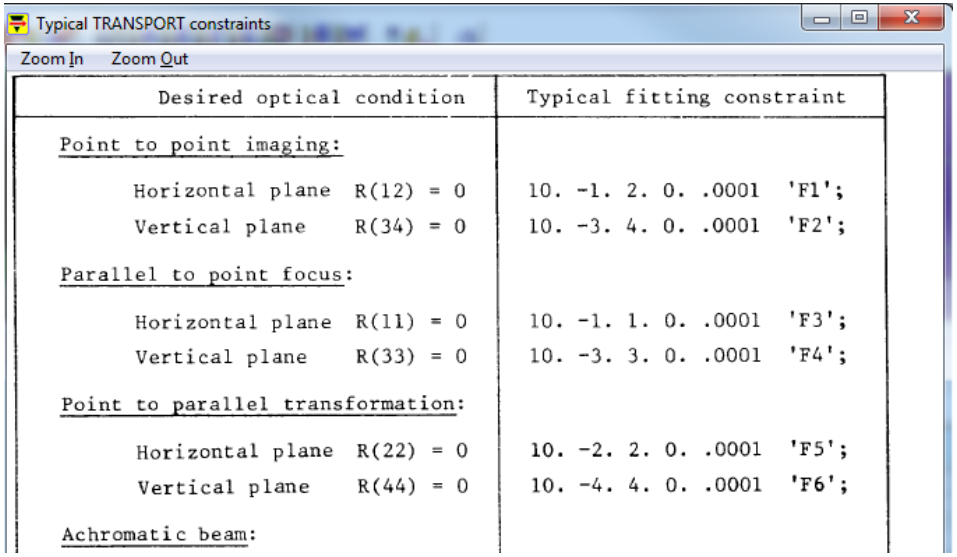
The "Fit constraint" dialog. For a constraint the user selects an element from an optical matrix or beam sigma vector, and set its desired value and precision (weight).



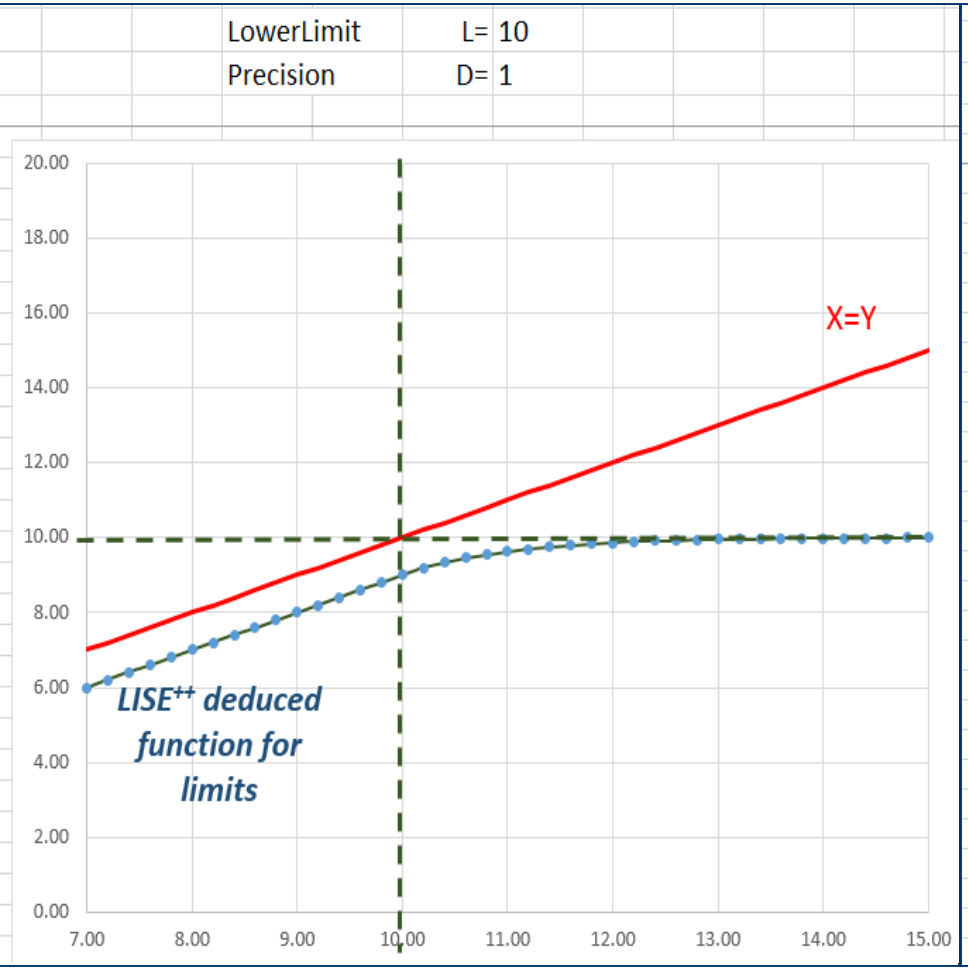


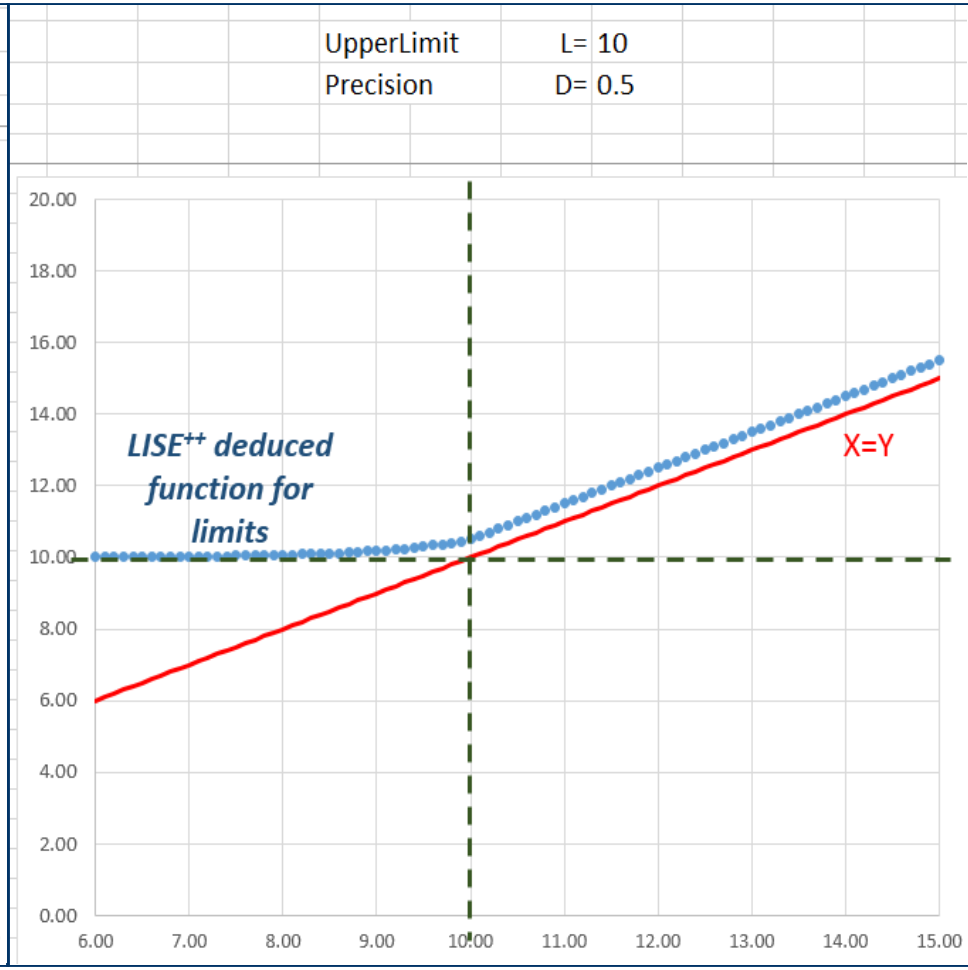
Inverse weight

TRANSPORT notification of selected constraint.
Second order constraint input under development



Special LISE^{++} functions fore limit constraints



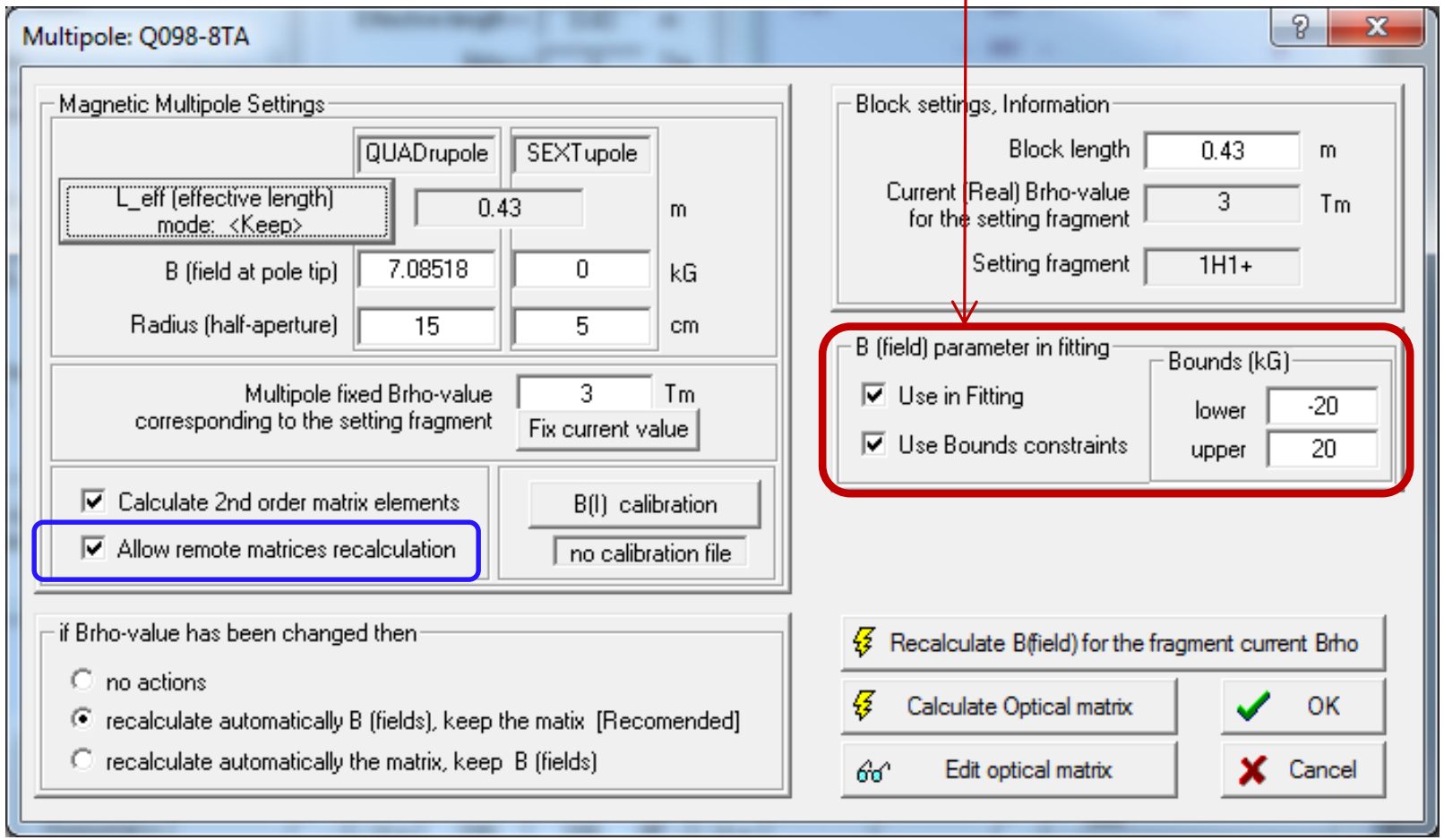


Levmar functions for "equal_to" constraints are used.
Important to have limit constraints in LISE^{++} for apertures
New Functions should continuous!

Selecting optic block and its parameters to use in the minimization

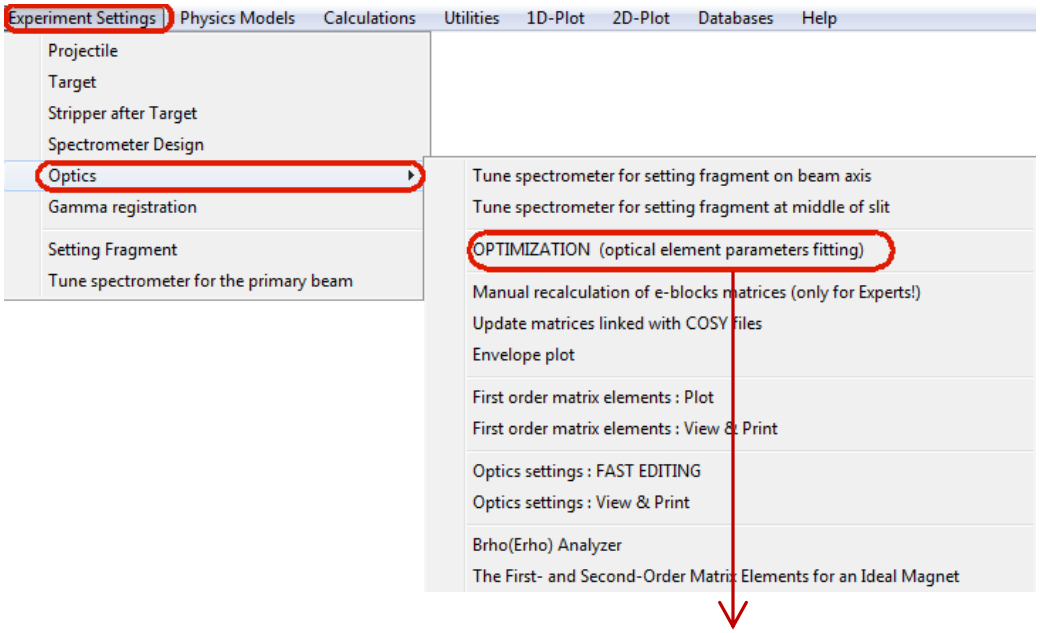
In current version only M-Quad B-fields and E-Quad voltages

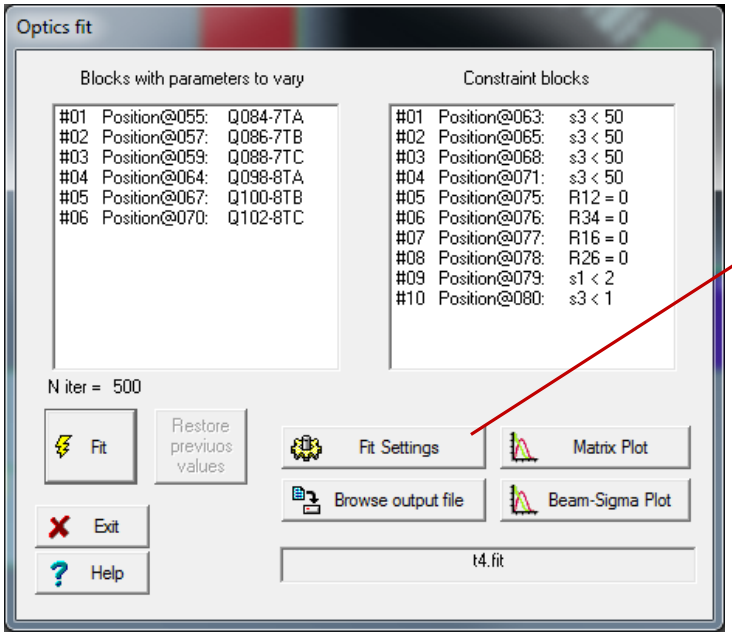
Set in it!

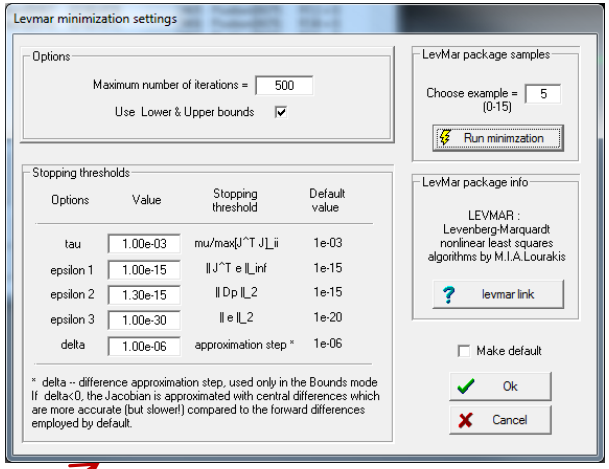


No matrix link to external file!

Optimization dialog





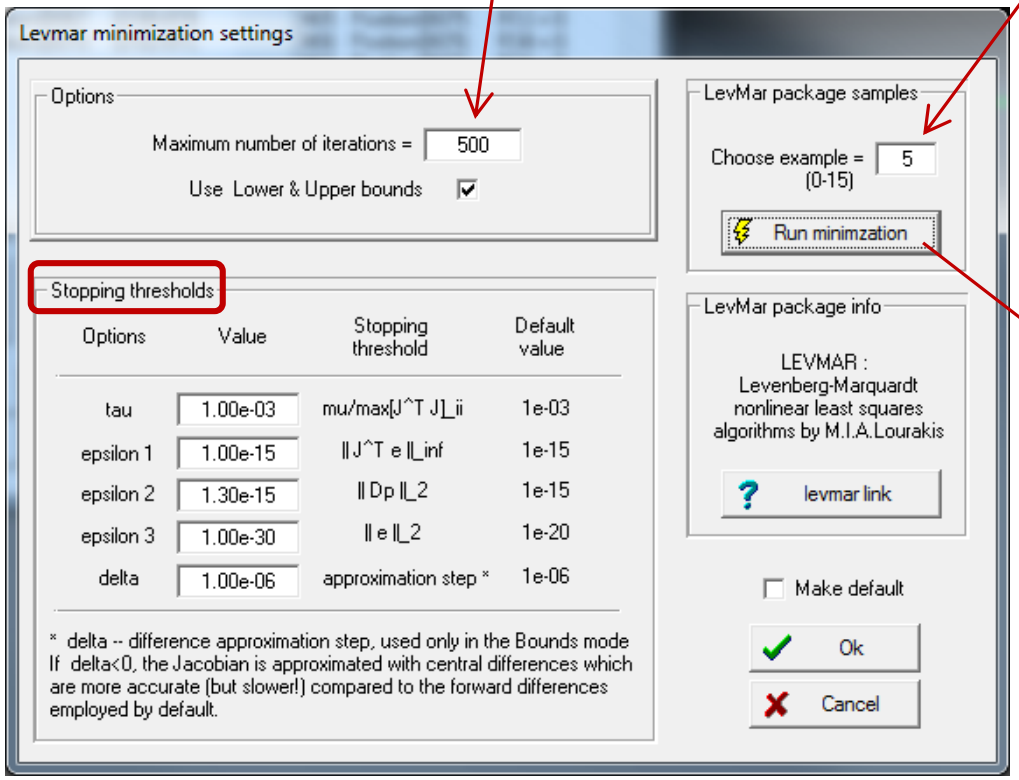


The "Optics Fit" dialog. The left panel shows optical blocks with varying parameters, whereas blocks with fitting constraints.

Fit Settings Dialog

For the first step use “50-100”

“levmar” package examples to play with settings



“see the next page

Levmar's examples (#5 – Osbrone data fitting)

```
case 5:
/* Osborne's data fitting problem */
{
  double x33[]={
    8.44E-1, 9.08E-1, 9.32E-1, 9.36E-1, 9.25E-1, 9.08E-1, 8.81E-1,
    8.5E-1, 8.18E-1, 7.84E-1, 7.51E-1, 7.18E-1, 6.85E-1, 6.58E-1,
    6.28E-1, 6.03E-1, 5.8E-1, 5.58E-1, 5.38E-1, 5.22E-1, 5.06E-1,
    4.9E-1, 4.78E-1, 4.67E-1, 4.57E-1, 4.48E-1, 4.38E-1, 4.31E-1,
    4.24E-1, 4.2E-1, 4.14E-1, 4.11E-1, 4.06E-1};

  m=5; n=33;
  p[0]=0.5; p[1]=1.5; p[2]=-1.0; p[3]=1.0E-2; p[4]=2.0E-2;


        work=malloc((LM_DIF_WORKSZ(m, n)+m*m)*sizeof(double));
        if(!work){
          fprintf(stderr, "memory allocation request failed in main()\n");
          exit(1);
          }
        covar=work+LM_DIF_WORKSZ(m, n);

  if(box)
    {
    lb[0]=lb[1]=lb[2]=lb[3]=lb[4]= -20;
    ub[0]=ub[1]=ub[2]=ub[3]=ub[4]=  20;

        if(jacob) ret=dlevmar_bc_der(osborne, jacosborne, p, x33, m, n, lb, ub, NULL,
        else      ret=dlevmar_bc_dif(osborne,             p, x33, m, n, lb, ub, NULL,
    }
  else
    {
        if(jacob) ret=dlevmar_der(osborne, jacosborne, p, x33, m, n, N, opts, info, w
        else      ret=dlevmar_dif(osborne,             p, x33, m, n, N, opts, info, w
    }
}
break;
```

```
//##############################################################################
//##############################################################################
/* Osborne's problem, minimum at (0.3754, 1.9358, -1.4647, 0.0129, 0.02
void osborne(double *p, double *x, int m, int n, void *data)
{
register int i;
double t;

      for(i=0; i<n; ++i){
    t=10*i;
    x[i]=p[0] + p[1]*exp(-p[3]*t) + p[2]*exp(-p[4]*t);
      }
}
//##############################################################################
void jacosborne(double *p, double *jac, int m, int n, void *data)
{
register int i, j;
double t, tmp1, tmp2;

  for(i=j=0; i<n; ++i){
    t=10*i;
        tmp1=exp(-p[3]*t);
    tmp2=exp(-p[4]*t);

        jac[j++]=1.0;
        jac[j++]=tmp1;
        jac[j++]=tmp2;
    jac[j++]=-p[1]*t*tmp1;
    jac[j++]=-p[2]*t*tmp2;
  }
}
//##############################################################################
```

Levmar's examples (#5) : fitting results

```
/* Osborne's problem, minimum at (0.3754, 1.9358, -1.4647, 0.0129, 0.0221) */
```

c:\program files (x86)\lise\results\LevMar 5 00.log

Print

```
Jacob: No,  Box: No

------------------------------------
==>  Results for Osborne's problem(5):
Levenberg-Marquardt returned 90 in 90 iter, reason 2
Solution: 0.3754101  1.935847  -1.464687  0.01286753  0.0221227

Options info:
0: 1.000e-03        mu
1: 1.000e-15        epsilon1  ||J^T e||_inf
2: 1.000e-15        epsilon2  ||Dp||_2
3: 1.000e-20        epsilon3  ||e||_2
4: 1.000e-06        delta       aprrox.step

Minimization info:
0: 8.790e-01        ||e||_2 at initial p
1: 5.465e-05        ||e||_2
2: 4.899e-10        ||J^T e||_inf
3: 5.068e-31        ||Dp||_2
4: 2.233e+01        mu/max[J^T J]_ii
5: 90               # iterations
6: 2                reason for terminating
7: 141              # function evaluations
8: 10               # Jacobian evaluations
9: 91               # linear systems solved, i.e. # attempts for reducing error

Termination reason: 2 - stopped by small Dp

Covariance of the fit :
+4.293184e-06 +4.168233e-04 -4.204934e-04 +8.751658e-07 -1.635539e-06
+4.168233e-04 +4.853246e-02 -4.884843e-02 +9.847883e-05 -1.962537e-04
```

c:\program files (x86)\lise\results\LevMar 5 01.log

Print

```
Jacob: No,  Box: Yes
Low Bounds:-20  -20  -20  -20  -20
Upp Bounds:+20  +20  +20  +20  +20

------------------------------------
==>  Results for Osborne's problem(5):
Levenberg-Marquardt returned 24 in 24 iter, reason 2
Solution: 0.3754098  1.935822  -1.464662  0.01286748  0.0221228

Options info:
0: 1.000e-03        mu
1: 1.000e-15        epsilon1  ||J^T e||_inf
2: 1.000e-15        epsilon2  ||Dp||_2
3: 1.000e-20        epsilon3  ||e||_2
4: 1.000e-06        delta       aprrox.step

Minimization info:
0: 8.790e-01        ||e||_2 at initial p
1: 5.465e-05        ||e||_2
2: 1.902e-07        ||J^T e||_inf
3: 3.009e-33        ||Dp||_2
4: 7.378e-11        mu/max[J^T J]_ii
5: 24               # iterations
6: 2                reason for terminating
7: 983              # function evaluations
8: 24               # Jacobian evaluations
9: 24               # linear systems solved, i.e. # attempts for reducing error

Termination reason: 2 - stopped by small Dp

Covariance of the fit :
```

Jacobian: No

Box: No

Box: Yes

c:\program files (x86)\lise\results\LevMar 5 10.log

Print

```
Jacob: Yes,  Box: No

------------------------------------
==>  Results for Osborne's problem(5):
Levenberg-Marquardt returned 31 in 31 iter, reason 2
Solution: 0.3754101  1.935847  -1.464687  0.01286753  0.0221227

Options info:
0: 1.000e-03        mu
1: 1.000e-15        epsilon1  ||J^T e||_inf
2: 1.000e-15        epsilon2  ||Dp||_2
3: 1.000e-20        epsilon3  ||e||_2
4: 1.000e-06        delta       aprrox.step

Minimization info:
0: 8.790e-01        ||e||_2 at initial p
1: 5.465e-05        ||e||_2
2: 4.522e-09        ||J^T e||_inf
3: 5.227e-30        ||Dp||_2
4: 4.923e+01        mu/max[J^T J]_ii
5: 31               # iterations
6: 2                reason for terminating
7: 40               # function evaluations
8: 31               # Jacobian evaluations
9: 40               # linear systems solved, i.e. # attempts for reducing error

Termination reason: 2 - stopped by small Dp

Covariance of the fit :
+4.294491e-06 +4.169215e-04 -4.205927e-04 +8.751991e-07 -1.636147e-06
+4.169215e-04 +4.853941e-02 -4.885547e-02 +9.847325e-05 -1.963069e-04
```

c:\program files (x86)\lise\results\LevMar 5 11.log

Print

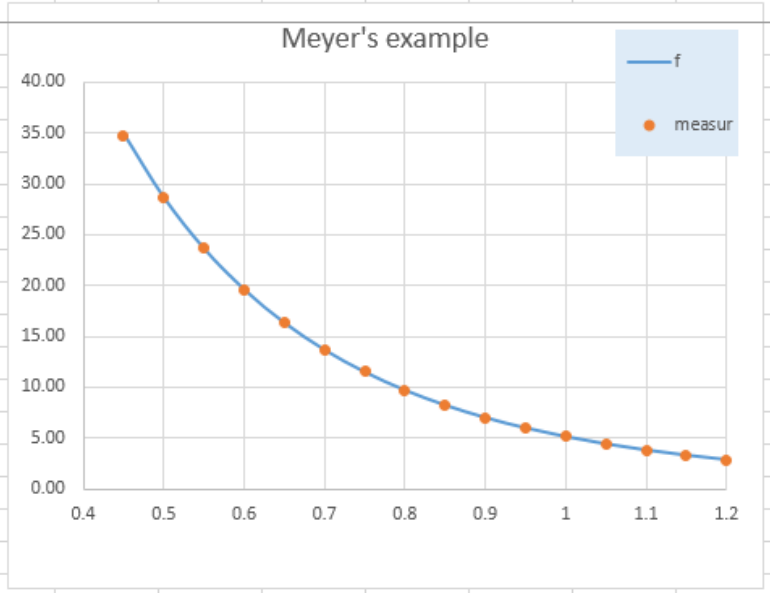
```
Jacob: Yes,  Box: Yes
Low Bounds:-20  -20  -20  -20  -20
Upp Bounds:+20  +20  +20  +20  +20

------------------------------------
==>  Results for Osborne's problem(5):
Levenberg-Marquardt returned 28 in 28 iter, reason 2
Solution: 0.3754101  1.935847  -1.464687  0.01286753  0.0221227

Options info:
0: 1.000e-03        mu
1: 1.000e-15        epsilon1  ||J^T e||_inf
2: 1.000e-15        epsilon2  ||Dp||_2
3: 1.000e-20        epsilon3  ||e||_2
4: 1.000e-06        delta       aprrox.step

Minimization info:
0: 8.790e-01        ||e||_2 at initial p
1: 5.465e-05        ||e||_2
2: 1.435e-09        ||J^T e||_inf
3: 0.000e+00        ||Dp||_2
4: 7.488e-11        mu/max[J^T J]_ii
5: 28               # iterations
6: 2                reason for terminating
7: 1359             # function evaluations
8: 28               # Jacobian evaluations
9: 28               # linear systems solved, i.e. # attempts for reducing error

Termination reason: 2 - stopped by small Dp

Covariance of the fit :
```

Jacobian: Yes

With Boxes is slower!!

Levmar's examples (#4) : Excel vs. Levmar

i	ui	f	measur	delta
0	0.45	34.92	34.78	5.49E-04
1	0.5	28.65	28.61	4.47E-05
2	0.55	23.63	23.65	2.15E-05
3	0.6	19.59	19.63	8.29E-05
4	0.65	16.32	16.37	1.35E-04
5	0.7	13.67	13.72	2.10E-04
6	0.75	11.49	11.54	1.77E-04
7	0.8	9.71	9.744	1.11E-04
8	0.85	8.24	8.261	5.74E-05
9	0.9	7.02	7.03	1.69E-05
10	0.95	6.00	6.005	4.68E-07
11	1	5.15	5.147	9.77E-06
12	1.05	4.44	4.427	4.58E-05
13	1.1	3.84	3.82	1.10E-04
14	1.15	3.33	3.307	1.95E-04
15	1.2	2.90	2.872	2.94E-04
				2.06E-03



f excel	f_levmar	delta	delta/f
34.92	34.78	0.14	0.39%
28.65	28.61	0.04	0.13%
23.63	23.64	-0.02	-0.07%
19.59	19.63	-0.04	-0.22%
16.32	16.38	-0.05	-0.32%
13.67	13.72	-0.05	-0.37%
11.49	11.54	-0.04	-0.38%
9.71	9.74	-0.03	-0.35%
8.24	8.26	-0.02	-0.28%
7.02	7.03	-0.01	-0.17%
6.00	6.01	0.00	-0.04%
5.15	5.15	0.01	0.13%
4.44	4.43	0.01	0.33%
3.84	3.82	0.02	0.55%
3.33	3.31	0.03	0.79%
2.90	2.87	0.03	1.06%

p0	9.460866	2.48	8.85
p1	5.096541	6.15	4
p2	3.112559	3.45	2.5

LevMar	5.72E-06	
		init
p0	2.48	8.85
p1	6.18	4
p2	3.50	2.5

Excel	2.06E-03	
		init
p0	9.46	8.85
p1	5.10	4
p2	3.11	2.5

x[i]=p[0]*exp(10.0*p[1]/(ui+p[2]) - 13.0)

Initial parameters for both cases

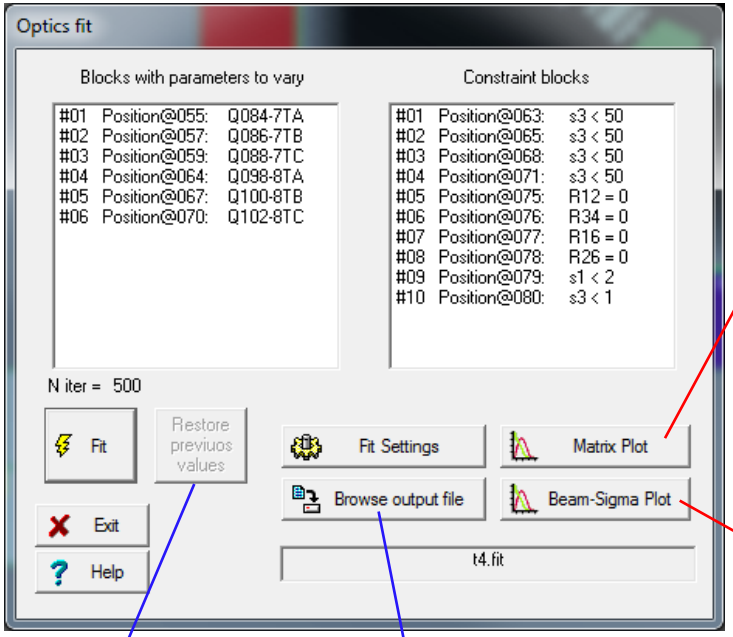
Levmar results

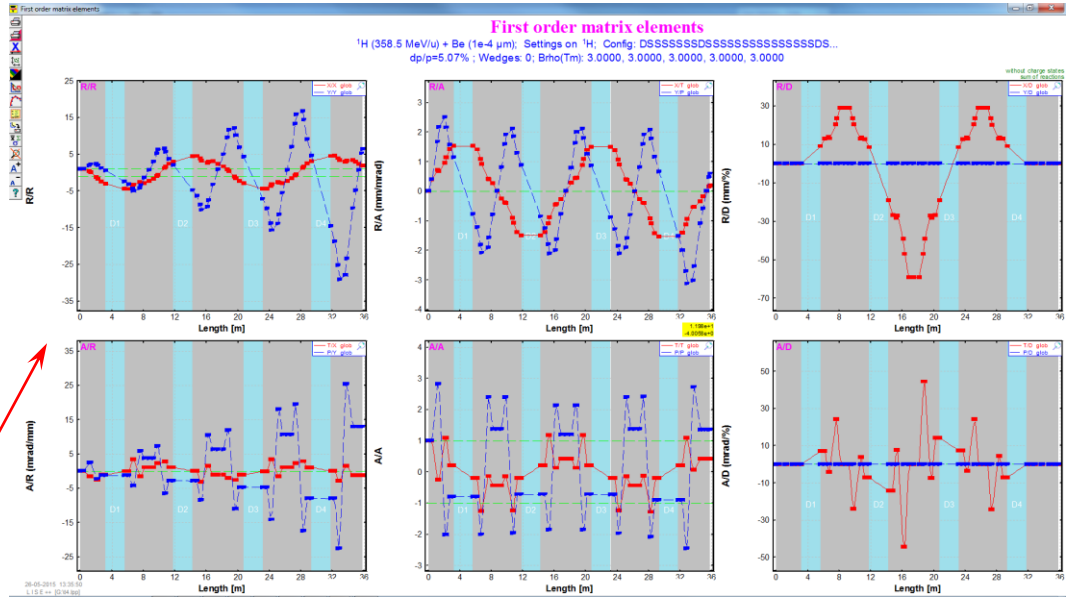
Excel results

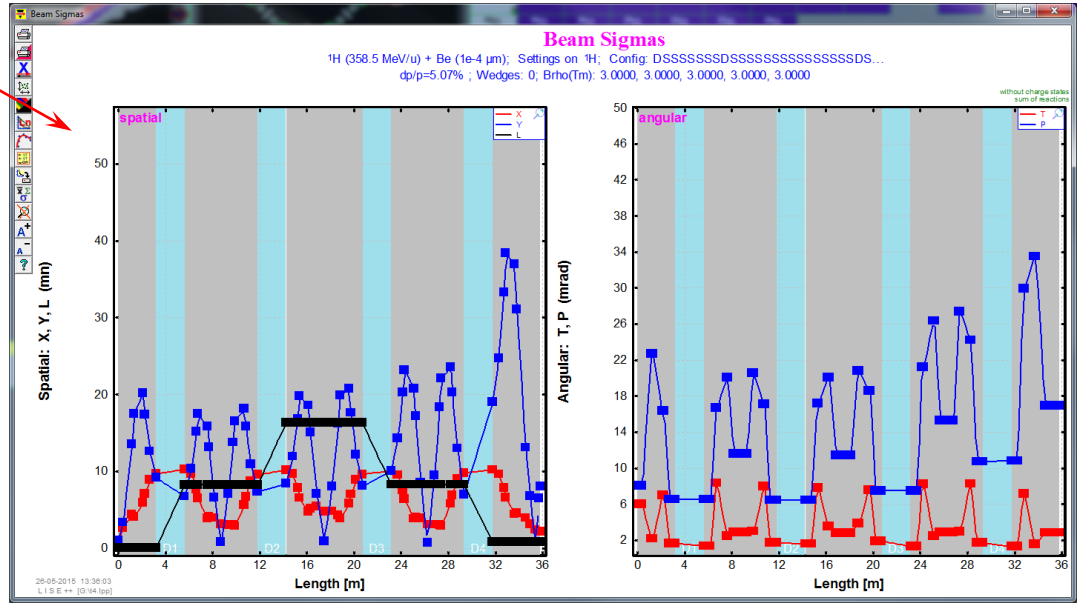
Levmar chi-square result by 3 orders of magnitude is lower, than Excel's result!!!

Matrix elements and Beam sigmas plot

You can get plots before fitting process and after to compare values







After fitting process it is possible to restore initial settings

Initial log-file name is LISE++ filename with the "fit" extension. Located by default in the directory "LISE\result"

Fitting information window (coming from log-file)

t4.fit

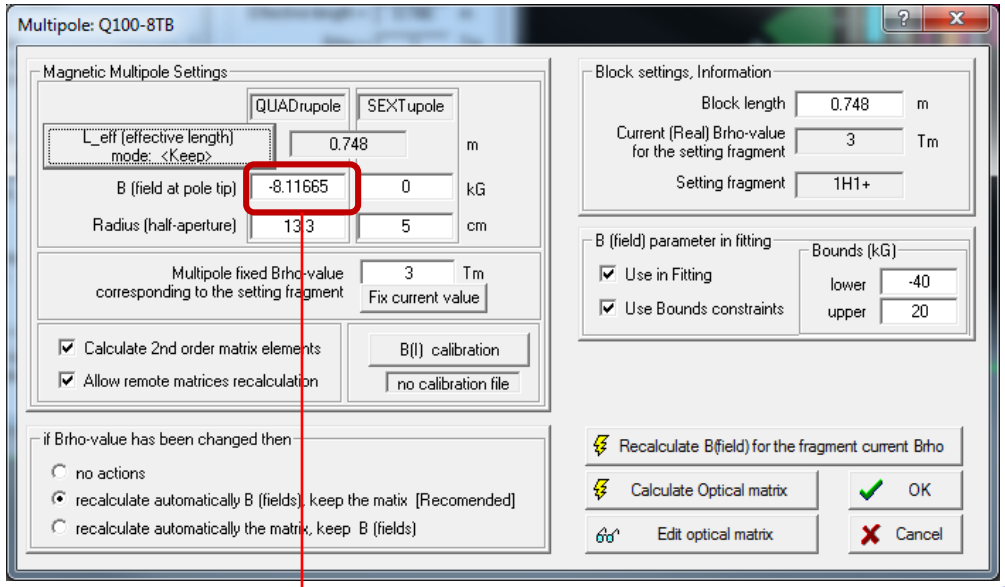
```
Initial  +641.499  and Final +641.173  LISE fit reduced values

Parameters:          LeftBound      Initial     RightBound   |   Final
#01: Q084-7TA      -1.000e+99 < +9.403e+00 < +1.000e+99   | +8.916e+00
#02: Q086-7TB      -1.000e+99 < -1.083e+01 < +1.000e+99   | -9.624e+00
#03: Q088-7TC      -1.000e+99 < +8.752e+00 < +1.000e+99   | +7.892e+00
#04: Q098-8TA      -2.000e+01 < +7.085e+00 < +2.000e+01   | +6.310e+00
#05: Q100-8TB      -4.000e+01 < -5.000e+00 < +2.000e+01   | -6.152e+00
#06: Q102-8TC      -1.000e+01 < +4.212e+00 < +4.000e+01   | +3.994e+00

-------------------------------------------------------------------------

Fitting values:       Initial       Final       Precision  (Fin-Des)/P |    Desired
#01: sigY           +2.469e+01   +1.648e+00    1.000e+00   0.000e+00   | < +5.00e+01
#02: sigY           +3.323e+01   +3.167e+00    1.000e+00   0.000e+00   | < +5.00e+01
#03: sigY           +4.555e+01   +6.177e+00    1.000e+00   0.000e+00   | < +5.00e+01
#04: sigY           +4.716e+01   +7.588e+00    1.000e+00   0.000e+00   | < +5.00e+01
#05: focusX         +6.022e-01   -1.913e-06    1.000e-03   1.913e-03   | = +0.00e+00
#06: focusY         -5.364e+00   -8.493e-01    5.000e-01   1.699e+00   | = +0.00e+00
#07: X-dispers      -1.287e-13   -1.767e-06    1.000e-03   1.767e-03   | = +0.00e+00
#08: T-dispers      +2.814e-13   -2.038e-05    1.000e-02   2.038e-03   | = +0.00e+00
#09: sigmaX         +3.618e+00   +1.991e+00    1.000e-01   9.911e-01   | < +2.00e+00
#10: sigmaY         +6.510e+01   +1.134e+01    1.000e-02   1.035e+03   | < +1.00e+00

-------------------------------------------------------------------------

==>  Results for t4.fit:
Levenberg-Marquardt returned 500 in 500 iter, reason 3
Solution: 8.916349  -9.623625  7.891567  6.309539  -6.152035  3.993751

Minimization info:
0: 4.147e+07        ||e||_2 at initial p
1: 4.111e+07        ||e||_2
2: 2.840e+00        ||J^T e||_inf
3: 2.275e-06        ||Dp||_2
4: 2.378e-04        mu/max[J^T J]_ii
5: 500              # iterations
6: 3                reason for terminating
7: 4499             # function evaluations
8: 500              # Jacobian evaluations
9: 500              # linear systems solved, i.e. # attempts for reducing error

Termination reason: 3 - stopped by itmax

-------------------------------------------------------------------------
```

Appears automatically after fitting process completed

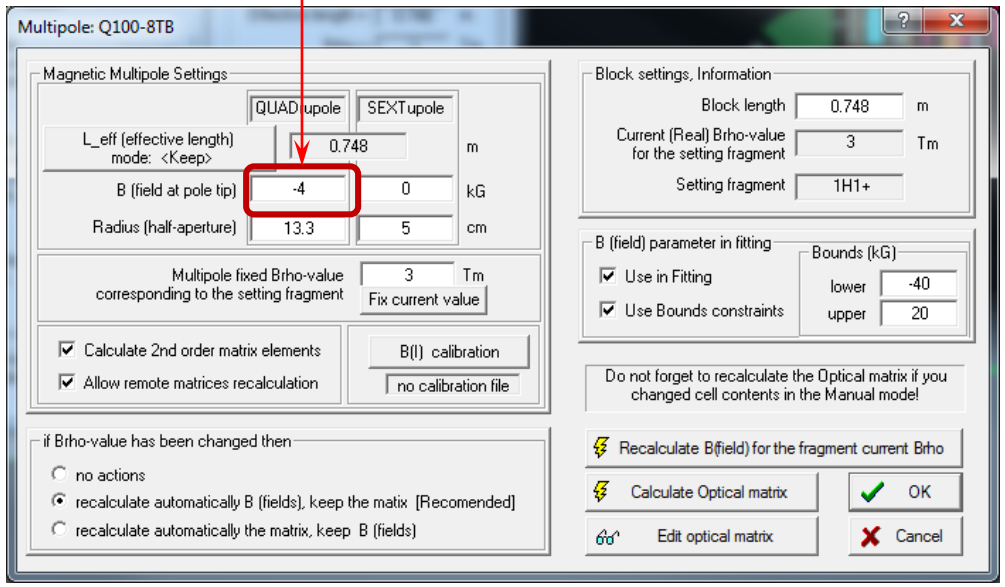
It is planning to use different colors and fonts to underline, to select key moments

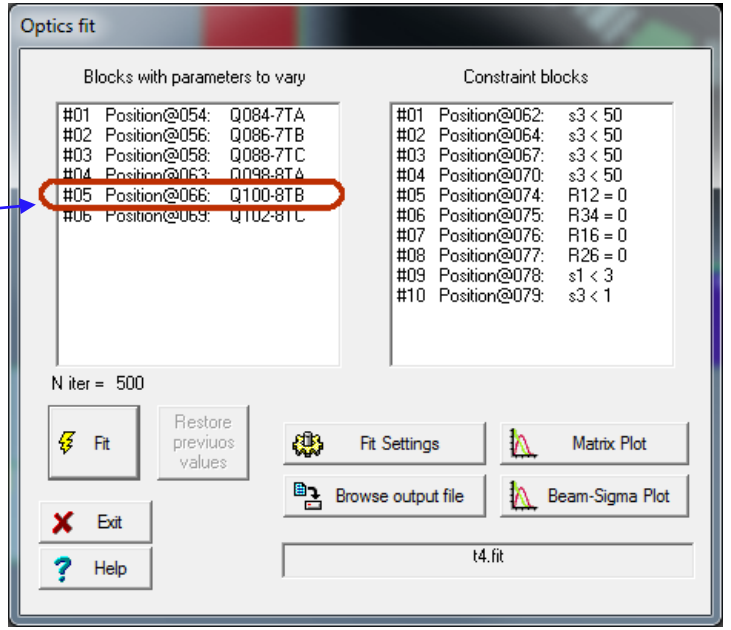
```
==> "sigmaY" : last fitting block global optical matrix and sigma vector

                                                    Format [mm-mrad]
=================== G L O B A L ====  matrix ==================   | Beam(sigma) |
   +1.991e+00  -1.913e-06    0           0          0    -1.767e-06   |    1.99e+00 |
   -1.482e+00  +5.025e-01    0           0          0    -2.038e-05   |    3.36e+00 |
    0           0          -9.083e+00  -8.493e-01   0     0           |    1.13e+01 |
    0           0          -3.325e+00  -4.210e-01   0     0           |    4.73e+00 |
   -1.428e-03  +5.792e-04    0           0          1.0  -1.088e+01   |    7.62e-01 |
    0           0           0           0          0    +1.000e+00   |    7.00e-02 |

-------------------------------------------------------------------------

Covariance of the fit :
+3.071318e+15 -7.383656e+15 +5.433857e+15 +6.802513e+16 -1.067550e+17 +2.799623e+15
-7.383595e+15 +1.775066e+16 -1.306325e+16 -1.635389e+17 +2.566550e+17 -6.742622e+15
+5.433800e+15 -1.306322e+16 +9.613624e+15 +1.203536e+17 -1.888818e+17 +4.964434e+15
+7.835533e+16 -1.883748e+17 +1.386313e+17 +1.182920e+18 -9.006090e+17 -1.998218e+18
-1.408364e+17 +3.385910e+17 -2.491815e+17 -1.296397e+18 -1.118888e+18 +6.699783e+18
+4.149371e+16 -9.976589e+16 +7.342309e+16 -1.150617e+18 +5.385893e+18 -7.714485e+18
```

Example for A1900 (1)

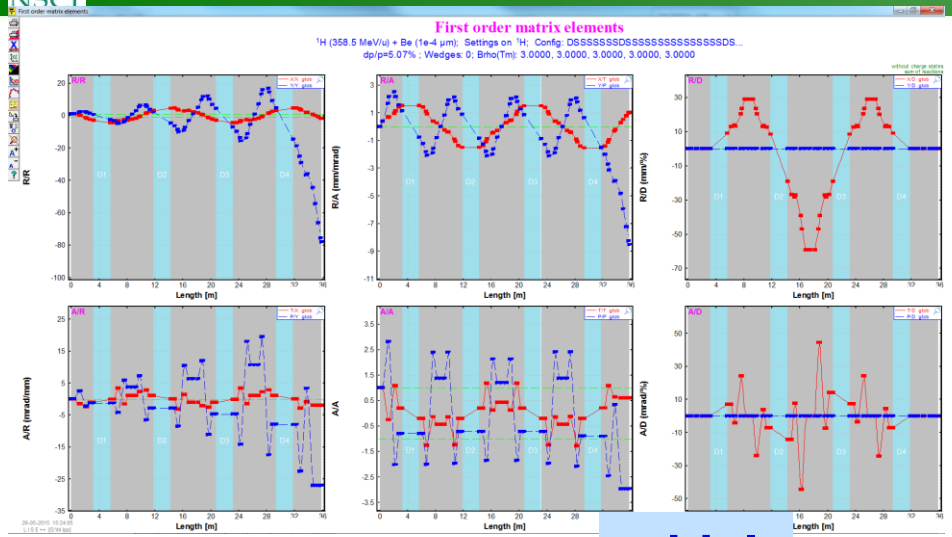


Let's destroy it manually

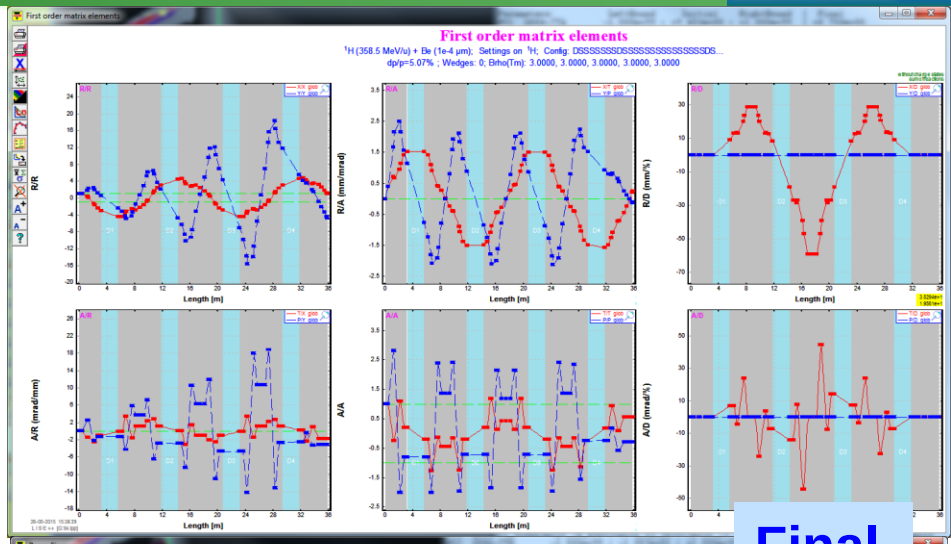




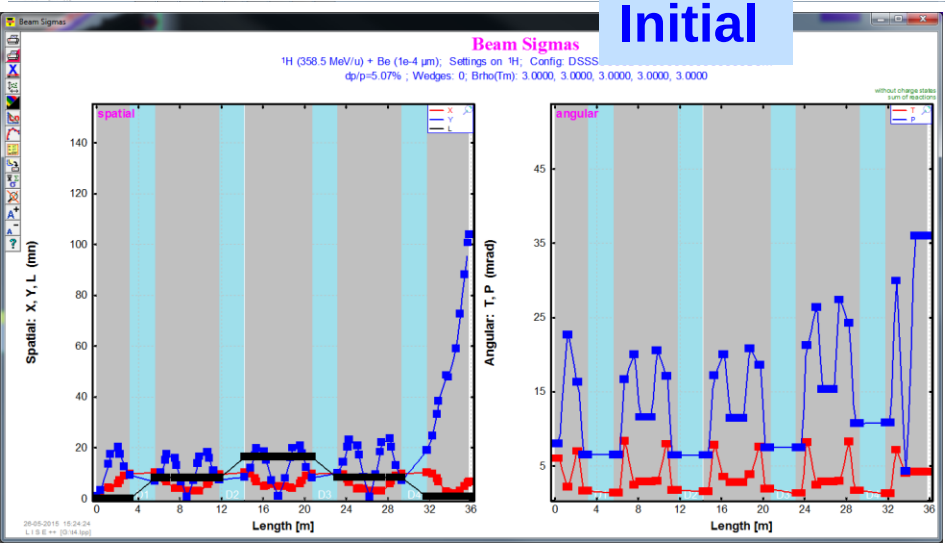
Example for A1900 (1)

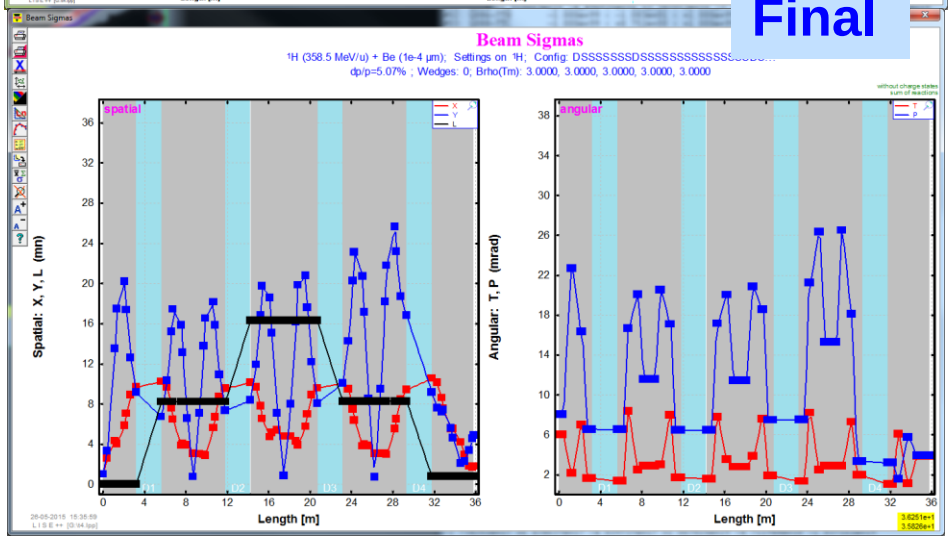


Initial



Final





Initial +870.782 and Final +870.318 LISE fit reduced values

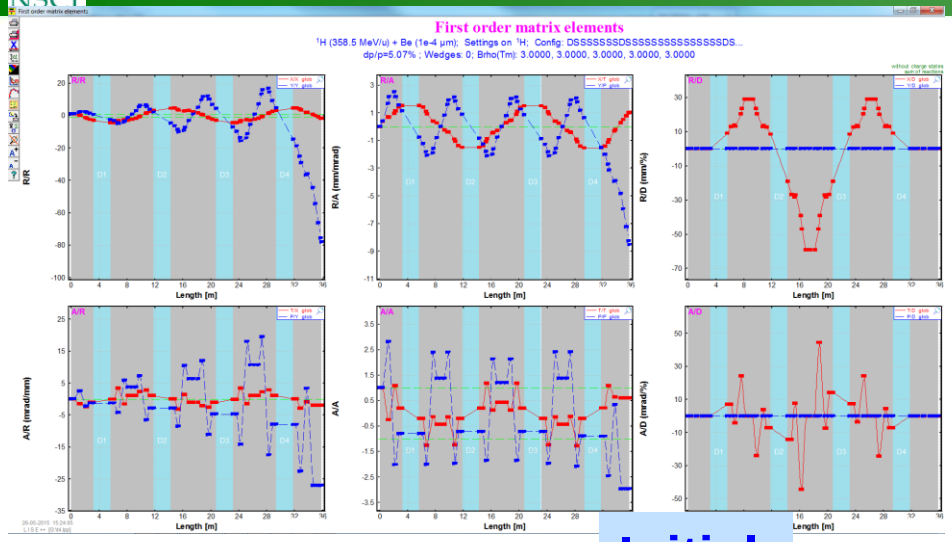
Parameters:	LeftBound	Initial	RightBound	Final
#01: Q084-7TA	-1.000e+99 <	+9.403e+00 <	+1.000e+99	+8.750e+00
#02: Q086-7TB	-1.000e+99 <	-1.083e+01 <	+1.000e+99	-9.228e+00
#03: Q088-7TC	-1.000e+99 <	+8.752e+00 <	+1.000e+99	+7.597e+00
#04: Q098-8TA	-2.000e+01 <	+7.085e+00 <	+2.000e+01	+5.738e+00
#05: Q100-8TB	-4.000e+01 <	-4.000e+00 <	+2.000e+01	-5.247e+00
#06: Q102-8TC	-1.000e+01 <	+4.212e+00 <	+4.000e+01	+4.103e+00

The Quad field value was not restored exactly

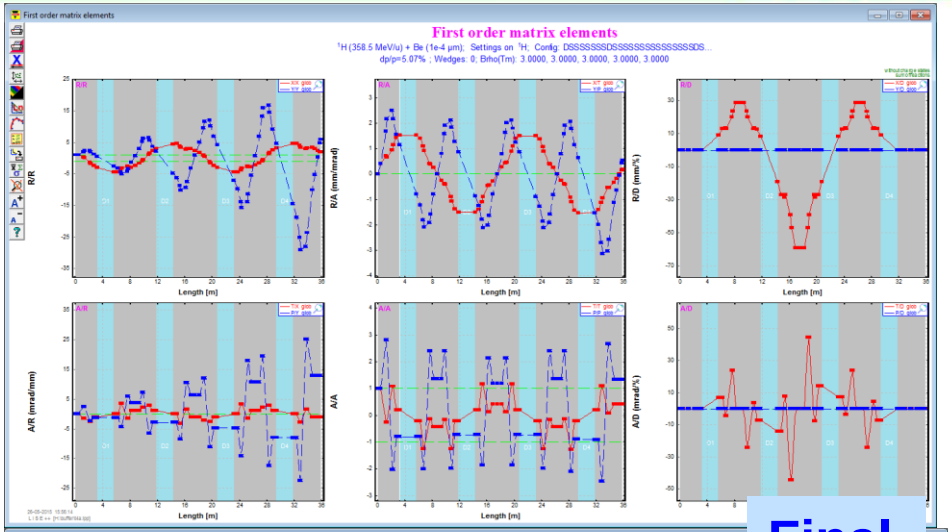
Fitting values:	Initial	Final	Precision	(Fin-Des)/P	Desired
#01: sigY	+2.469e+01	+7.589e+00	1.000e+00	0.000e+00	< +5.00e+01
#02: sigY	+3.323e+01	+7.188e+00	1.000e+00	0.000e+00	< +5.00e+01
#03: sigY	+4.846e+01	+5.595e+00	1.000e+00	0.000e+00	< +5.00e+01
#04: sigY	+5.902e+01	+2.098e+00	1.000e+00	0.000e+00	< +5.00e+01
#05: focusX	+7.758e-01	+5.198e-05	1.000e-03	5.198e-02	= +0.00e+00
#06: focusY	-7.236e+00	+3.918e-05	5.000e-01	7.836e-05	= +0.00e+00
#07: X-dispers	+7.190e-05	+3.171e-15	1.000e-03	3.171e-12	= +0.00e+00
#08: T-dispers	-1.354e-04	-3.960e-15	1.000e-02	3.960e-13	= +0.00e+00
#09: sigmaX	+4.744e+00	+1.780e+00	1.000e-01	2.953e-01	< +3.00e+00
#10: sigmaY	+8.802e+01	+3.395e+00	1.000e-02	2.405e+02	< +1.00e+00

The last constaint was not succesfull

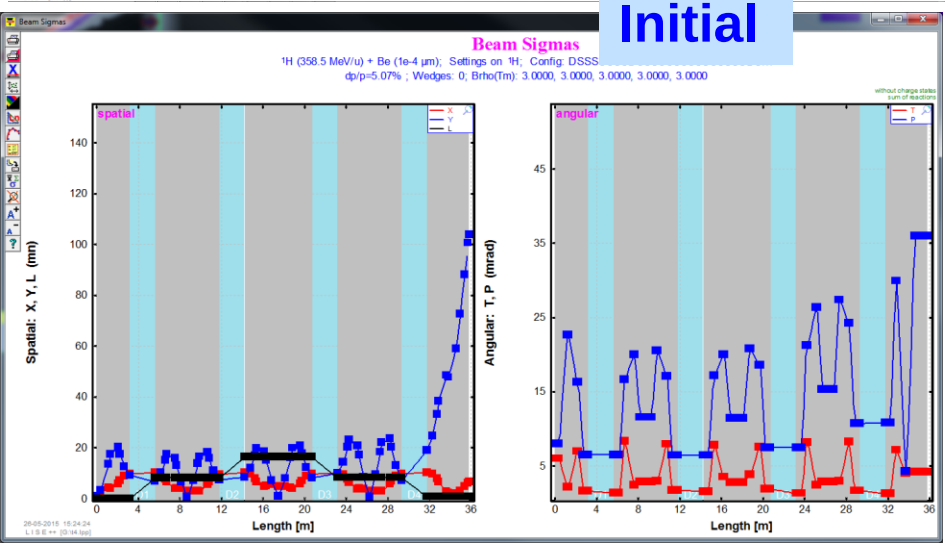
Example for A1900 (2) -- only last triplet to use in fit

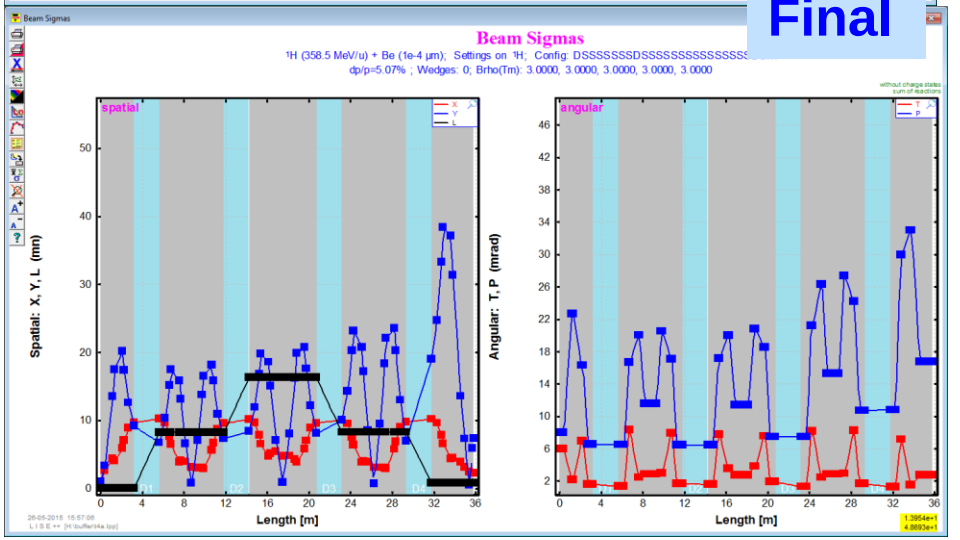


Initial



Final





Initial +870.782 and Final +870.376 LISE fit reduced values

Parameters:	LeftBound	Initial	RightBound	Final
#01: Q100-8TB	−4.000e+01 <	−4.000e+00 <	+2.000e+01	−8.040e+00
#02: Q102-8TC	−1.000e+01 <	+4.212e+00 <	+4.000e+01	+4.040e+00

The Quad field value was restored

Fitting values:	Initial	Final	Precision	(Fin−Des)/P	Desired
#01: sigY	+2.469e+01	+2.469e+01	1.000e+00	1.014e−11	< +5.00e+01
#02: sigY	+3.323e+01	+3.323e+01	1.000e+00	5.224e−08	< +5.00e+01
#03: sigY	+4.846e+01	+3.711e+01	1.000e+00	2.536e−06	< +5.00e+01
14: sigY	+5.902e+01	+1.355e+01	1.000e+00	0.000e+00	< +5.00e+01
15: focusX	+7.758e−01	−1.178e−06	1.000e−03	1.178e−03	= +0.00e+00
16: focusY	−7.236e+00	−4.717e−02	5.000e−01	9.434e−02	= +0.00e+00
17: X-dispers	+7.190e−05	+3.588e−04	1.000e−03	3.588e−01	= +0.00e+00
18: T-dispers	−1.354e−04	−5.358e−05	1.000e−02	5.358e−03	= +0.00e+00
19: sigmaX	+4.744e+00	+2.414e+00	1.000e−01	5.564e−01	< +3.00e+00
#10: sigmaY	+8.802e+01	+4.777e−01	1.000e−02	5.931e−01	< +1.00e+00

All constraints are good!

Acknowledgements

to

Drs. M.Hausmann, M.Portilio, and D.Weisshaar (NSCL/MSU),

for fruitful discussions.