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

Minimization for
• E-blocks (extended configurations)
• with non-linked matrices
• set the option “Allow remote matrices recalculation”
levmar : Levenberg-Marquardt nonlinear least squares algorithms in C/C++

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

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, this 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++ 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 ACML, 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 or 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.

levmar package info

LEVMAR : Levenberg-Marquardt nonlinear least squares algorithms by M.I.A.Lourakis

levmar

? levmar link
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 ;
```

The next file is to append standard constraint blocks
files\examples\NSCL\A1900_extended_LISE_FIT.lpp

files\examples\NSCL\FITconstraints.lpp
Introduction: information about blocks used for minimization
New Optic Block “Fitting Constraints”

![Spectrometer design interface](image)

- **Block**
  - ID
  - Given Name
  - D
  - Length
  - Enable
- **Insert Mode**
  - before
  - after
- **Optical**
  - dispersive
  - non-dispersive
- **Selected block**
  - Enable
  - Dispersive (M-dipole)
- **Total**
  - Number of Blocks
  - Length (m)
  - Sequence number
- **Fitting constraints**

OT, 26-May-2015, East Lansing
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).

TRANSPORT notification of selected constraint. Second order constraint input under development
Levmar functions for “equal_to” constraints are used. Important to have limit constraints in LISE++ for apertures. New Functions should be 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!
The “Optics Fit” dialog. The left panel shows optical blocks with varying parameters, whereas blocks with fitting constraints.
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)

```c
/* Osborne's data fitting problem */
{ double x[3][]=
    5.14E-1, 9.08E-1, 9.35E-1, 9.35E-1, 9.08E-1, 5.14E-1,
    9.35E-1, 9.10E-1, 7.35E-1, 7.35E-1, 9.10E-1, 9.35E-1,
    9.35E-1, 5.08E-1, 5.85E-1, 5.85E-1, 5.08E-1, 5.35E-1,
    4.98E-1, 4.79E-1, 4.67E-1, 4.98E-1, 4.49E-1, 4.38E-1,
    4.38E-1, 4.26E-1, 4.14E-1, 4.14E-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)*sizeof(double));
    if (!work)
    { printf(stderr, "memory allocation request failed in main()\n");
      exit(1);
    }

    covar=work+LM_DIF_WORKSZ(m, n);

    if (box)
    {
        if (jacob) ret=dlevmar_bc_der(osborne, jacbosborne, p, x[3], m, n, lb, ub, NULL, info);
        else ret=dlevmar_bc_diff(osborne, p, x[3], m, n, lb, ub, NULL, info);
    }
    else
    {
        if (jacob) ret=dlevmar_der(osborne, jacbosborne, p, x[3], m, n, opt, info);
        else ret=dlevmar_diff(osborne, p, x[3], m, n, opt, info);
    }
    break;

} //////////////////////////////////////////////////////////////////////////////////////////

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*t1;
        x[i]=p[0]+p[1]*exp(-p[3]*t)+p[2]*exp(-p[4]*t);
    }
}

} //////////////////////////////////////////////////////////////////////////////////////////

void jacbosborne(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*t1;
        tmp1=exp(-p[3]*t);
        tmp2=exp(-p[4]*t);
        jac[i+j+1]=1.0;
        jac[i+j]=tmp1;
        jac[j+i]=tmp2;
        jac[i+j+2]=p[1]*t*tmp1;
        jac[i+j+3]=p[2]*t*tmp2;
    }
}

} //////////////////////////////////////////////////////////////////////////////////////////

OT, 26-May-2015, East Lansing
Levmar’s examples (#5) : fitting results

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

With Boxes is slower!!
Levmar’s examples (#4) : Excel vs. Levmar

<table>
<thead>
<tr>
<th>i</th>
<th>ui</th>
<th>f</th>
<th>measur</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.45</td>
<td>34.92</td>
<td>34.78</td>
<td>5.49E-04</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>28.65</td>
<td>28.61</td>
<td>4.47E-05</td>
</tr>
<tr>
<td>2</td>
<td>0.55</td>
<td>23.63</td>
<td>23.65</td>
<td>2.15E-05</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>19.59</td>
<td>19.63</td>
<td>8.29E-05</td>
</tr>
<tr>
<td>4</td>
<td>0.65</td>
<td>16.32</td>
<td>16.37</td>
<td>1.35E-04</td>
</tr>
<tr>
<td>5</td>
<td>0.7</td>
<td>13.67</td>
<td>13.72</td>
<td>2.10E-04</td>
</tr>
<tr>
<td>6</td>
<td>0.75</td>
<td>11.49</td>
<td>11.54</td>
<td>1.77E-04</td>
</tr>
<tr>
<td>7</td>
<td>0.8</td>
<td>9.71</td>
<td>9.744</td>
<td>1.11E-04</td>
</tr>
<tr>
<td>8</td>
<td>0.85</td>
<td>8.24</td>
<td>8.201</td>
<td>5.74E-05</td>
</tr>
<tr>
<td>9</td>
<td>0.9</td>
<td>7.02</td>
<td>7.03</td>
<td>1.69E-05</td>
</tr>
<tr>
<td>10</td>
<td>0.95</td>
<td>6.00</td>
<td>6.005</td>
<td>4.68E-07</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>5.15</td>
<td>5.147</td>
<td>9.77E-06</td>
</tr>
<tr>
<td>12</td>
<td>1.05</td>
<td>4.44</td>
<td>4.427</td>
<td>4.58E-05</td>
</tr>
<tr>
<td>13</td>
<td>1.1</td>
<td>3.84</td>
<td>3.82</td>
<td>1.10E-04</td>
</tr>
<tr>
<td>14</td>
<td>1.15</td>
<td>3.33</td>
<td>3.307</td>
<td>1.95E-04</td>
</tr>
<tr>
<td>15</td>
<td>1.2</td>
<td>2.90</td>
<td>2.672</td>
<td>2.94E-04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>f_{levmar}</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>34.92</td>
<td>0.14</td>
</tr>
<tr>
<td>1</td>
<td>28.65</td>
<td>0.04</td>
</tr>
<tr>
<td>2</td>
<td>23.63</td>
<td>-0.02</td>
</tr>
<tr>
<td>3</td>
<td>19.59</td>
<td>-0.04</td>
</tr>
<tr>
<td>4</td>
<td>16.32</td>
<td>-0.05</td>
</tr>
<tr>
<td>5</td>
<td>13.67</td>
<td>-0.05</td>
</tr>
<tr>
<td>6</td>
<td>11.49</td>
<td>-0.04</td>
</tr>
<tr>
<td>7</td>
<td>9.71</td>
<td>-0.03</td>
</tr>
<tr>
<td>8</td>
<td>8.24</td>
<td>-0.02</td>
</tr>
<tr>
<td>9</td>
<td>7.03</td>
<td>-0.01</td>
</tr>
<tr>
<td>10</td>
<td>6.01</td>
<td>0.00</td>
</tr>
<tr>
<td>11</td>
<td>5.15</td>
<td>0.01</td>
</tr>
<tr>
<td>12</td>
<td>4.43</td>
<td>0.01</td>
</tr>
<tr>
<td>13</td>
<td>3.82</td>
<td>0.02</td>
</tr>
<tr>
<td>14</td>
<td>3.31</td>
<td>0.03</td>
</tr>
<tr>
<td>15</td>
<td>2.87</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Excel results

Levmar results

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

It is planning to use different colors and fonts to underline, to select key moments
Example for A1900 (1)

Let's destroy it manually

Multipole: Q100-8T8

Magnetic Multipoles Settings

- Quadrupole
- Sextupole

Block settings, Information

- Block length: 0.748 m
- Current (Real) B-field value for the setting fragment: 3 Tm
- Setting fragment: 1H1+

B (field) parameter in fitting

- Use in Fitting
- Use Bounds constraints

If B-field value has been changed then:

- Recalculate B-field for the fragment current B-field
- Recalculate automatically B-fields, keep the matrix [Recommended]
- Recalculate automatically the matrix, keep B-fields

Optics fit

Blocks with parameters to vary

<table>
<thead>
<tr>
<th>Block</th>
<th>Constraint blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>#01</td>
<td>Position@054: C004-71A</td>
</tr>
<tr>
<td>#02</td>
<td>Position@055: C008-71B</td>
</tr>
<tr>
<td>#03</td>
<td>Position@057: C008-71C</td>
</tr>
<tr>
<td>#04</td>
<td>Position@050: C009-71A</td>
</tr>
<tr>
<td>#05</td>
<td>Position@066: C10051B</td>
</tr>
<tr>
<td>#06</td>
<td>Position@067: C10051C</td>
</tr>
</tbody>
</table>

N iter = 500

Fit Settings

- Fit
- Restore previous values

Matrix Plot

- Browse output file
- Beam Sigma Plot

OT, 26-May-2015, East Lansing
The Quad field value was not restored exactly.
Example for A1900 (2) -- only last triplet to use in fit

The Quad field value was restored

Initial +870.782 and Final +870.376 LISE fit reduced values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial</th>
<th>Final</th>
<th>Precision (Fit-Des)/P</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>#01: Q100-8TB</td>
<td>-4.000e+00 &lt; -4.000e+00 &lt; +2.000e+01</td>
<td>-8.046e+00</td>
<td>1.000e+00 1.014e+11</td>
<td>&lt;+5 000e+01</td>
</tr>
<tr>
<td>#02: Q102-81C</td>
<td>-1.000e+01 &lt; +4.212e+00 &lt; +4.000e+01</td>
<td>+4.040e+00</td>
<td>1.000e+00 2.536e+06</td>
<td>&lt;+5 000e+01</td>
</tr>
<tr>
<td>#11: sigY</td>
<td>+4.846e+01</td>
<td>+3.322e+01</td>
<td>1.000e+00 5.224e+08</td>
<td>&lt;+5 000e+01</td>
</tr>
<tr>
<td>#12: sigY</td>
<td>+5.902e+01</td>
<td>+1.355e+01</td>
<td>1.000e+00 2.178e+03</td>
<td>&lt;+5 000e+01</td>
</tr>
<tr>
<td>#13: focusX</td>
<td>-7.758e-01</td>
<td>-1.789e-06</td>
<td>1.000e+03 1.789e-03</td>
<td>&lt;+5 000e+01</td>
</tr>
<tr>
<td>#14: focusX</td>
<td>-7.758e-01</td>
<td>-1.789e-06</td>
<td>1.000e+03 1.789e-03</td>
<td>&lt;+5 000e+01</td>
</tr>
<tr>
<td>#15: sigY</td>
<td>+7.190e-05</td>
<td>+3.580e-05</td>
<td>1.000e-03 3.580e-03</td>
<td>&lt;+0 000e+00</td>
</tr>
<tr>
<td>#16: sigY</td>
<td>+7.190e-05</td>
<td>+3.580e-05</td>
<td>1.000e-03 3.580e-03</td>
<td>&lt;+0 000e+00</td>
</tr>
</tbody>
</table>

All constraints are good!
Acknowledgements

to

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

for fruitful discussions.