Nonlinear least squares problems

This lecture is based on the book

> P. C. Hansen, V. Pereyra and G. Scherer,
> *Least Squares Data Fitting with Applications*,
> Johns Hopkins University Press, to appear
> (the necessary chapters are available on CampusNet)

and we cover this material:

- Section 8.1: Intro to nonlinear data fitting.
- Section 8.2: Unconstrained nonlinear least squares problems.
- Section 9.1: Newton's method.
- Section 9.2: The Gauss-Newton method.
- Section 9.3: The Levenberg-Marquardt method.

Non-linearity

A parameter α of the function f appears nonlinearly if the derivative $\partial f/\partial\alpha$ is a function of α.

The model $M(\boldsymbol{x}, t)$ is *nonlinear* if at least one of the parameters in $\boldsymbol{x}$ appear nonlinearly.

For example, in the exponential decay model

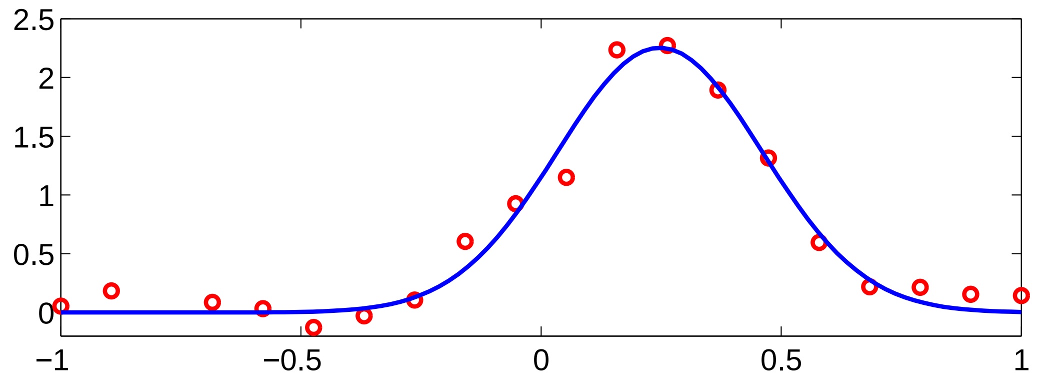
$$M(x_1, x_2, t) = x_1 e^{-x_2 t}$$

we have:

- $\partial M/\partial x_1 = e^{-x_2 t}$ which is independent of x_1,
- $\partial M/\partial x_2 = -t\, x_1 e^{-x_2 t}$ which depends on x_2.

Thus M is a nonlinear model with the parameter x_2 appearing nonlinearly.

Fitting with a Gaussian model



The non-normalized Gaussian function:

$$M(\boldsymbol{x},t) = x_1 e^{-(t-x_2)^2/(2x_3^2)}, \qquad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

where x_1 is the amplitude, x_2 is the time shift, and x_3 determines the width of the Gaussian function.

The parameters x_2 and x_3 appear nonlinearly in this model.

Gaussian models also arise in many other data fitting problems.

The nonlinear least squares problem

Find a minimizer $\boldsymbol{x}^*$ of the nonlinear objective function f:

$$\boxed{\min_{\boldsymbol{x}} f(\boldsymbol{x}) \equiv \min_{\boldsymbol{x}} \tfrac{1}{2} \|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \min_{\boldsymbol{x}} \tfrac{1}{2} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2,}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and, as usual,

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{pmatrix} r_1(\boldsymbol{x}) \\ \vdots \\ r_m(\boldsymbol{x}) \end{pmatrix} \in \mathbb{R}^m,$$

$$r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x}, t_i), \qquad i = 1, \dots, m \; .$$

Here y_i are the measured data corresponding to t_i.

The nonlinearity arises **only** from $M(\boldsymbol{x}, t)$.

The Jacobian and the gradient of $f(\boldsymbol{x})$

The *Jacobian* $J(\boldsymbol{x})$ of the vector function $\boldsymbol{r}(\boldsymbol{x})$ is defined as the matrix with elements

$$[J(\boldsymbol{x})]_{ij} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_j} = -\frac{\partial M(\boldsymbol{x}, t_i)}{\partial x_j}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

The ith row of $J(\boldsymbol{x})$ equals the transpose of the gradient of $r_i(\boldsymbol{x})$:

$$[J(\boldsymbol{x})]_{i,:} = \nabla r_i(\boldsymbol{x})^T = -\nabla M(\boldsymbol{x}, t_i)^T, \qquad i = 1, \ldots, m.$$

Thus the elements of the gradient of $f(\boldsymbol{x})$ are given by

$$[\nabla f(\boldsymbol{x})]_j = \frac{\partial f(\boldsymbol{x})}{\partial x_j} = \sum_{i=1}^{m} r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}$$

and it follows that the gradient is the vector

$$\boxed{\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})\ .}$$

The Hessian matrix of $f(\boldsymbol{x})$

The elements of the *Hessian* of f, denoted $\nabla^2 f(\boldsymbol{x})$, are given by

$$[\nabla^2 f(\boldsymbol{x})]_{k\ell} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_k \partial x_\ell} = \sum_{i=1}^{m} \frac{\partial r_i(\boldsymbol{x})}{\partial x_k} \frac{\partial r_i(\boldsymbol{x})}{\partial x_\ell} + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_k \partial x_\ell},$$

and it follows that the Hessian can be written as

$$\boxed{\nabla^2 f(\boldsymbol{x}) = J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}),}$$

where

$$\begin{aligned} \left[\nabla^2 r_i(\boldsymbol{x})\right]_{k\ell} &= -\left[\nabla^2 M(\boldsymbol{x}, t_i)\right]_{k\ell} \\ &= -\frac{\partial^2 M(\boldsymbol{x}, t_i)}{\partial x_k \partial x_\ell}, \qquad k, \ell = 1, \ldots, m \ . \end{aligned}$$

The optimality conditions

First-order necessary condition:

$$\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0} \ .$$

Second-order sufficient condition:

$$\nabla^2 f(\boldsymbol{x}) = J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}) \quad \text{is positive definite.}$$

The first – and often dominant – term $J(\boldsymbol{x})^T J(\boldsymbol{x})$ of the Hessian contains only the Jacobian matrix $J(\boldsymbol{x})$, i.e., only first derivatives!

In the second term, the second derivatives are multiplied by the residuals. If the model is adequate then the residuals will be small near the solution and this term will be of secondary importance.

In this case one gets an important part of the Hessian "for free" if one has already computed the Jacobian.

Local linear LSQ problem

If we introduce a Taylor expansion around the LSQ solution $\boldsymbol{x}^*$, the local least squares problem for $\boldsymbol{x}$ close to $\boldsymbol{x}^*$ can be written

$$\min_{\boldsymbol{x}} \|J(\boldsymbol{x}^*)\,(\boldsymbol{x}-\boldsymbol{x}^*)+\boldsymbol{r}(\boldsymbol{x}^*)\|_2 =$$

$$\min_{\boldsymbol{x}} \Big\|J(\boldsymbol{x}^*)\,\boldsymbol{x}-\big(J(\boldsymbol{x}^*)\boldsymbol{x}^*+\boldsymbol{r}(\boldsymbol{x}^*)\big)\Big\|_2 \,.$$

It follows from the results in Chapter 1 that:

$$\begin{aligned}
\mathrm{Cov}(\boldsymbol{x}^*) &\simeq J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}\big(J(\boldsymbol{x}^*)\,\boldsymbol{x}^*-\boldsymbol{r}(\boldsymbol{x}^*)\big)(J(\boldsymbol{x}^*)^\dagger)^T \\
&= J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}\big(\boldsymbol{r}(\boldsymbol{x}^*)-J(\boldsymbol{x}^*)\,\boldsymbol{x}^*\big)(J(\boldsymbol{x}^*)^\dagger)^T \\
&= J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}(\boldsymbol{y})(J(\boldsymbol{x}^*)^\dagger)^T.
\end{aligned}$$

This provides a way to approximately assess the uncertainties in the least squares solution $\boldsymbol{x}^*$ for the nonlinear problem.

Newton's method

If $f(\boldsymbol{x})$ is twice continuously differentiable then we can use Newton's method to solve the nonlinear equation

$$\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}$$

which provides local stationary points for $f(\boldsymbol{x})$. This version of the Newton iteration takes the form, for $k = 0, 1, 2, \ldots$

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \left(\nabla^2 f(\boldsymbol{x}_k)\right)^{-1} \nabla f(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - \left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)\right)^{-1} J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k), \end{aligned}$$

where $S(\boldsymbol{x}_k)$ denotes the matrix

$$S(\boldsymbol{x}_k) = \sum_{i=1}^{m} r_i(\boldsymbol{x}_k) \nabla^2 r_i(\boldsymbol{x}_k).$$

Convergence. Quadratic convergence, but expensive – requires mn^2 derivatives to evaluate $S(\boldsymbol{x}_k)$.

The Gauss-Newton method

If the problem is only mildly nonlinear or if the residual at the solution is small, a good alternative is to neglect the second term $S(\boldsymbol{x}_k)$ of the Hessian altogether.

The resulting method is referred to as the Gauss-Newton method, where the computation of the step $\boldsymbol{\Delta x}_k^{\mathrm{GN}}$ involves the solution of the linear system

$$\big(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\big)\, \boldsymbol{\Delta x}_k^{\mathrm{GN}} = -J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k).$$

Note that in the full-rank case this is actually the normal equations for the linear least squares problem

$$\min_{\boldsymbol{\Delta x}_k^{\mathrm{GN}}} \left\| J(\boldsymbol{x}_k)\, \boldsymbol{\Delta x}_k^{\mathrm{GN}} - (-\boldsymbol{r}(\boldsymbol{x}_k)) \right\|_2^2 .$$

This is a descent step if $J(\boldsymbol{x}_k)$ has full rank.

Damped Gauss-Newton = G-N with line search

Implementations of the G-N method usually perform a line search in the direction $\Delta\boldsymbol{x}_k^{\mathrm{GN}}$, e.g., requiring the step length α_k to satisfy the Armijo condition:

$$\begin{aligned} f(\boldsymbol{x}_k + \alpha_k \Delta\boldsymbol{x}_k^{\mathrm{GN}}) &< f(\boldsymbol{x}_k) + c_1\, \alpha_k \nabla f(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k^{\mathrm{GN}} \\ &= f(\boldsymbol{x}_k) + c_1\, \alpha_k \boldsymbol{r}(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k^{\mathrm{GN}}, \end{aligned}$$

with a constant $c_1 \in (0,1)$.

This ensures that the reduction is (at least) proportional to both the parameter α_k and the directional derivative $\nabla f(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k^{\mathrm{GN}}$.

Line search make the algorithm (often) globally convergent.

Convergence. Can be quadratic if the neglected term in the Hessian is small. Otherwise it is linear.

Algorithm: Damped Gauss-Newton

- Start with the initial point $\boldsymbol{x}_0$, and iterate for $k = 0, 1, 2, \ldots$
- Solve $\min_{\Delta \boldsymbol{x}} \| J(\boldsymbol{x}_k)\, \Delta \boldsymbol{x} + \boldsymbol{r}(\boldsymbol{x}_k) \|_2$ to compute the step direction $\Delta \boldsymbol{x}_k^{\mathrm{GN}}$.
- Choose a step length α_k so that there is enough descent.
- Calculate the new iterate: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \Delta \boldsymbol{x}_k^{\mathrm{GN}}$.
- Check for convergence.

The Levenberg-Marquardt method

Very similar to G-N, except that we replace the line search with a trust-region strategy where the norm of the step is limited.

$$\min \| J(\boldsymbol{x}_k)\, \Delta\boldsymbol{x} + r(\boldsymbol{x}_k) \|_2^2 \qquad \text{subject to} \qquad \|\Delta\boldsymbol{x}\|_2 \leq \text{bound.}$$

Constrained optimization is outside the scope of this course (it is covered in 02612).

Computation of the L-M Step

The computation of the step in Levenberg-Marquardt's method is implemented as:

$$\Delta \boldsymbol{x}_k^{\mathrm{LM}} = \operatorname{argmin}_{\Delta \boldsymbol{x}} \left\{ \|J(\boldsymbol{x}_k)\,\Delta \boldsymbol{x} + r(\boldsymbol{x}_k)\|_2^2 + \lambda_k \, \|\Delta \boldsymbol{x}\|_2^2 \right\}$$

where $\lambda_k > 0$ is a so-called *Lagrange parameter* for the constraint at the kth iteration.

The L-M step is computed as the solution to the linear LSQ problem

$$\min_{\Delta \boldsymbol{x}} \left\| \begin{pmatrix} J(\boldsymbol{x}_k) \\ \lambda_k^{1/2} I \end{pmatrix} \Delta \boldsymbol{x} - \begin{pmatrix} -\boldsymbol{r}(\boldsymbol{x}_k) \\ \mathbf{0} \end{pmatrix} \right\|_2^2 .$$

This method is more robust, in case of an ill conditioned Jacobian.

Algorithm: Levenberg-Marquardt

- Start with the initial point $\boldsymbol{x}_0$ and iterate for $k = 0, 1, 2, \ldots$
- At each step k choose the Lagrange parameter λ_k.
- Solve the linear LSQ problem

$$\min_{\Delta \boldsymbol{x}} \left\| \begin{pmatrix} J(\boldsymbol{x}_k) \\ \lambda_k^{1/2} I \end{pmatrix} \Delta \boldsymbol{x} - \begin{pmatrix} -\boldsymbol{r}(\boldsymbol{x}_k) \\ \boldsymbol{0} \end{pmatrix} \right\|_2^2$$

  to compute the step $\Delta \boldsymbol{x}_k^{\mathrm{LM}}$.
- Calculate the next iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta \boldsymbol{x}_k^{\mathrm{LM}}$.
- Check for convergence.

Note: there is no line search (i.e., no α_k-parameter), its role is taken over by the Lagrange parameter λ_k.

The role of the Lagrange parameter

Consider the L-M step, which we formally write as:

$$\Delta \boldsymbol{x}_k^{\mathrm{LM}} = \left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + \lambda_k I \right)^{-1} J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k).$$

The parameter λ_k influences both the direction and the length of the step.

Depending on the size of λ_k, the step $\Delta \boldsymbol{x}_k^{\mathrm{LM}}$ can vary from a Gauss-Newton step for $\lambda_k = 0$, to a short step approximately in the steepest descent direction for large values of λ_k.

How to choose the Lagrange parameter

A strategy developed by Marquardt. The underlying principles are:

1. The initial value $\lambda_0 \approx \|J(\boldsymbol{x}_0)^T J(\boldsymbol{x}_0)\|_2$.
2. For subsequent steps, an improvement ratio is defined as:

$$\rho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{\frac{1}{2}(\Delta\boldsymbol{x}_k^{\mathrm{LM}})^T (J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k) - \lambda_k \Delta\boldsymbol{x}_k^{\mathrm{LM}})}.$$

Here, the denominator is the reduction in f predicted by the local linear model.

If ρ_k is large then the pure Gauss-Newton model is good enough, so λ_{k+1} can be made smaller than at the previous step. If ρ_k is small (or even negative) then a short steepest descent step should be used, i.e., λ_{k+1} should to be increased.

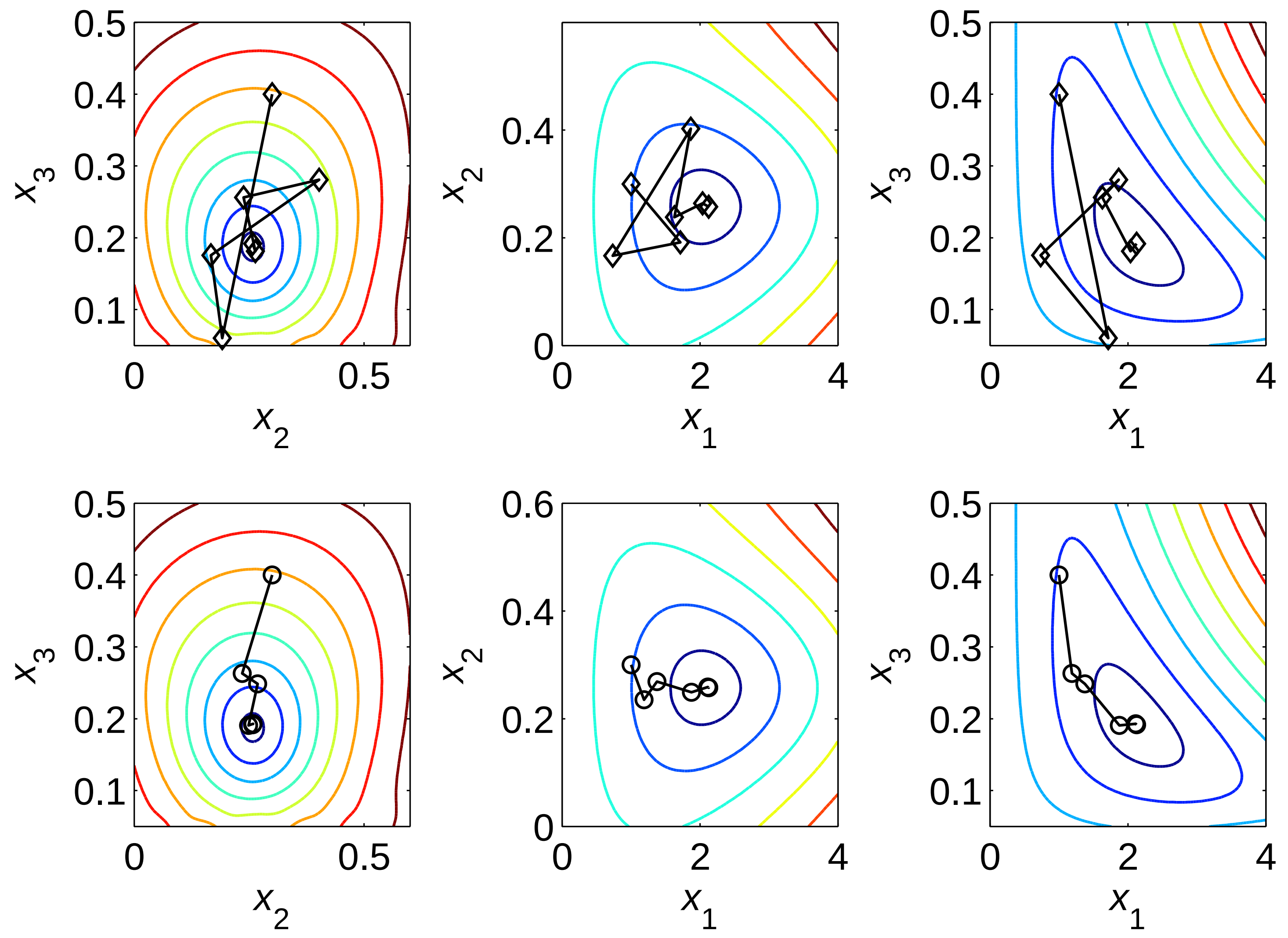
Algorithm: Marquardt's Parameter Updating

- If $\rho_k > 0.75$ then $\lambda_{k+1} = \lambda_k/3$.
- If $\rho_k < 0.25$ then $\lambda_{k+1} = 2\,\lambda_k$.
- Otherwise use $\lambda_{k+1} = \lambda_k$.
- If $\rho_k > 0$ then perform the update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\Delta x}_k^{\mathrm{LM}}$.

As G-N, the L-M algorithm is (often) globally convergent.

Convergence. Can be quadratic of the neglected term in the Hessian is small. Otherwise it is linear.

G-N without damping (top) vs. L-M (bottom)



MATLAB Optimization Toolbox: `lsqnonlin`

`[x,resnorm] = lsqnonlin(fun,x0)` requires an initial point `x0` and a function `fun` that computes the *vector-valued* function

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{pmatrix}$$

and solves the problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{f}(\boldsymbol{x})\|_2^2 = \min_{\boldsymbol{x}} \left(f_1(\boldsymbol{x})^2 + \cdots + f_m(\boldsymbol{x})^2) \right) .$$

Use `optimset` to choose between different optimization methods.

E.g., `'LargeScale'='off'` and `'LevenbergMarquardt'='off'` give the standard G-N method, while `'Jacobian'='on'` and `'Algorithm'='levenberg-marquardt'` give the L-M algorithm.