Gradus Vol 6, No 1 (2019) 1-7
ISSN 2064-8014

RELATIVE EFFECTIVENESS OF THE TRUST-REGION ALGORITHM WITH PRECISE SECOND ORDER DERIVATIVES

*Ambrus Kőházi-Kis**

Department of Natural Sciences and Engineering Basics, GAMF Faculty of Engineering an Computer Science, John von Neumann University, Hungary

Keywords:
Levenberg-Marquardt–method
Trust-region method
Hessian matrix
Eigen-value problem
Numerical stability

Article history:
Received 7 September 2018
Revised 2 April 2019
Accepted 19 March 2019

Abstract
Trust-region methods with precise Hessian matrix have some drawbacks: time consuming calculation of the elements of the second order derivative matrix, and the generally non-definite Hessian matrix causes numerical and methodical troubles. Their applicability depends on how well their substitute, for example the Levenberg-Marquardt–method performs. The Levenberg-Marquardt–method often performs well in least-squares problems. This procedure dynamically mixes the steepest-descent and the Gauss-Newton–methods. Generally one hopes that the more analytical properties of the problem's cost function utilized in an optimization procedure, the faster, the more effective search method can be constructed. It is definitely the case when we use first derivatives together with function values (instead of just function values). In the case of second derivative of the cost function the situation is not so clear. In lot of cases even if second order model is used within the search procedure the Hessian matrix is just approximated, and it is not calculated precisely even if it would be possible to calculate analytically, because of its temporal cost and a big amout of memory needed. In this paper I investigate whether the precise Hessian matrix is worth to be determined, whether one gains more on the increased effectiveness of the search method than looses on the increased temporal costof dealing with the precise Hessian matrix. In this paper it is done by the comparison of the Levenberg-Marquardt–method and a trust-region method using precise Hessian matrix.

1 Introduction

Least-squares problems are quite frequent in numerical data processing [1, 3].

Let us take a function $\boldsymbol{f} : \mathbb{R}^{\mathrm{n}} \mapsto \mathbb{R}^{\mathrm{m}}$ (with $m > n$), called as *residual vector*, and consider the problem of finding the least-squares solution $\boldsymbol{x}^{\dagger}$

$$\boldsymbol{x}^{\dagger} = ArgMin\left\{F\left(\boldsymbol{x}\right)\right\} \ , \tag{1}$$

where function $F : \mathbb{R}^{\mathrm{n}} \mapsto \mathbb{R}$ is called the *cost function* (or *function of merit*) of the problem:

$$F\left(\boldsymbol{x}\right) \equiv \tfrac{1}{2} \sum_{i=0}^{m} \left(f_i\left(\boldsymbol{x}\right)\right)^2 \ . \tag{2}$$

*Corresponding author. Tel.: +3676516436; fax: +3676516299
E-mail address: kohazi-kis.ambrus@uni-neumann.hu

When the components $f_i(\boldsymbol{x})$ of $\boldsymbol{f}(\boldsymbol{x})$ are non-linear functions, we have to use iteration: from a starting point $\boldsymbol{x}_0$ we compute $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, ... and we shall assume that the descending condition

$$F(\boldsymbol{x}_{k+1}) < F(\boldsymbol{x}_k) \tag{3}$$

is satisfied.

The solution gets much faster, if the derivatives of the cost function can be determined. In this paper only this kind of least-squares problems are considered.

The derivative function of an m variable, n dimensional vector-function $\boldsymbol{f} : \mathbb{R}^{\mathrm{n}} \mapsto \mathbb{R}^{\mathrm{m}}$ is also called Jacobian-matrix (see Ref. [4]):

$$\boldsymbol{f}'(\boldsymbol{x}_i) \equiv \left.\frac{\mathrm{d}\,\boldsymbol{f}}{d\,\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} = \boldsymbol{J}_f(\boldsymbol{x}_i) \equiv \begin{pmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} & \cdots & \left.\frac{\partial f_1}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} \\ \vdots & . & \vdots \\ \left.\frac{\partial f_m}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} & \cdots & \left.\frac{\partial f_m}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} \end{pmatrix} \quad . \tag{4}$$

The first derivative of the function $F(\boldsymbol{x}) \equiv \frac{1}{2}\,\|\boldsymbol{f}(\boldsymbol{x})\|^2$ can be obtained as (see ref. [1])

$$\boldsymbol{F}'(\boldsymbol{x}_i) = \boldsymbol{J}_f(\boldsymbol{x}_i)^T\ \boldsymbol{f}(\boldsymbol{x}_i) \tag{5}$$

$$\boldsymbol{F}'(\boldsymbol{x}_i) \equiv \begin{pmatrix} \left.\frac{\partial F}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} \\ \vdots \\ \left.\frac{\partial F}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} \end{pmatrix} = \begin{pmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} & \cdots & \left.\frac{\partial f_m}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} \\ \vdots & . & \vdots \\ \left.\frac{\partial f_1}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} & \cdots & \left.\frac{\partial f_m}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} \end{pmatrix} \begin{pmatrix} f_1(\boldsymbol{x}_i) \\ \vdots \\ f_m(\boldsymbol{x}_i) \end{pmatrix} . \tag{6}$$

The second derivative of function F, the Hessian-matrix can be calculated as (see ref. [1])

$$\boldsymbol{F}"(\boldsymbol{x}_i) = \boldsymbol{J}_f(\boldsymbol{x}_i)^T\ \boldsymbol{J}_f(\boldsymbol{x}_i) + \boldsymbol{f}"(\boldsymbol{x}_i)\ \boldsymbol{f}(\boldsymbol{x}_i)\ . \tag{7}$$

There are strategies that use the derivatives of the cost function to solve optimization problems [1]. A short survey is given among the fundamental methods to give some insight to the connection between the Levenberg-Marquardt–algorithm and trust-region methods.

In the following I investigate the possibility to gain effectiveness when we use trust-region algorithm in which the precise Hessian matrix of the cost function is determined instead of the Levenberg-Marquardt–algorithm in which the Hessian matrix is just approximated.

2 Fundamental methods

2.1 Steepest descent direction method

A $\boldsymbol{h}$ descent direction satisfies

$$\boldsymbol{h}^T\,\boldsymbol{F}' < 0\,., \tag{8}$$

In the steepest descent direction method linear search is proceed though a direction

$$\boldsymbol{h}_{sd} = -\boldsymbol{F}' \quad . \tag{9}$$

The method is robust even if $\boldsymbol{x}$ is far from $\boldsymbol{x}^\dagger$, but it has poor convergence. Especially the low speed of final convergence is problematic.

2.2 Newton's method

Second order expansion of the residual vector is used to calculate the second derivative of the cost function

$$F\left(\boldsymbol{x}+\boldsymbol{\xi}\right) \simeq F\left(\boldsymbol{x}\right)+\boldsymbol{\xi}^{T}\,\boldsymbol{F}'\left(\boldsymbol{x}\right)+\tfrac{1}{2}\,\boldsymbol{\xi}^{T}\,\boldsymbol{F}"\left(\boldsymbol{x}\right)\,\boldsymbol{\xi}+\mathcal{O}\left(\left\|\boldsymbol{\xi}\right\|^{3}\right)\,, \tag{10}$$

$$\boldsymbol{F}'\left(\boldsymbol{x}+\boldsymbol{\xi}\right) \simeq \boldsymbol{F}'\left(\boldsymbol{x}\right)+\boldsymbol{F}"\left(\boldsymbol{x}\right)\,\boldsymbol{\xi}\,. \tag{11}$$

The optimization step $\boldsymbol{h}$ is the solution of the equation,

$$\boldsymbol{F}"\left(\boldsymbol{x}\right)\,\boldsymbol{h}_{N}=-\boldsymbol{F}'\left(\boldsymbol{x}\right)\,. \tag{12}$$

Newton method performs well in the final stage of the iteration, where $\boldsymbol{x} \simeq \boldsymbol{x}^{\star}$ and the Hessian matrix is positive definite, we get quadratic convergence. In the opposite situation, i.e. when the Hessian matrix is negative definite or indefinite, generally one cannot get even a descent step when one uses equation 12. If the eigenvalues and eigenvectors of the Hessian are determined the Newton's-method can be made an effective tool even if the Hessian is not positive definite [2].

2.3 Gauss-Newton method

This method is also called quasi-newton method because the Hessian matrix is approximated, the second term of the Hessian is omitted in (7):

$$\boldsymbol{F}"\left(\boldsymbol{x}_{i}\right) \simeq \boldsymbol{J}_{f}\left(\boldsymbol{x}_{i}\right)^{T}\,\boldsymbol{J}_{f}\left(\boldsymbol{x}_{i}\right)\,. \tag{13}$$

The optimization step is the solution to

$$\boldsymbol{J}_{f}\left(\mathbf{x}_{i}\right)^{T}\,\boldsymbol{J}_{f}\left(\mathbf{x}_{i}\right)\,\boldsymbol{h}_{NG}=-\boldsymbol{F}'\left(\mathbf{x}_{i}\right) \tag{14}$$

The obtained direction $\boldsymbol{h}_{NG}$ is always descent [1] if he Jacobi-matrix $\boldsymbol{J}_{f}\left(\boldsymbol{x}\right)$ has full rank.

The approximation is good, the convergence is quadratic if $\boldsymbol{f}\left(\boldsymbol{x}_{i}\right)$ is small enough, or $\boldsymbol{f}"\left(\boldsymbol{x}_{i}\right)$ is negligible (quasi-linear least square problem).

The value of $F\left(\boldsymbol{x}^{\dagger}\right)$ controls the final convergence speed.

The method with line search can be shown to have guaranteed convergence, provided that $\left\{\boldsymbol{x} \mid F\left(\boldsymbol{x}\right) \leq F\left(\boldsymbol{x}_{0}\right)\right\}$ is bounded, and the Jacobian-matrix $\boldsymbol{J}_{f}\left(\boldsymbol{x}\right)$ has full rank in all steps [1].

If the applied approximation for the second derivative of the cost function is not good, the convergence is generally slow.

3 Levenberg-Marquardt method

Levenberg (1944) and Marquardt (1963) suggested a method where the step $\boldsymbol{h}$ is computed by the following equation:

$$\left[\boldsymbol{J}_{f}\left(\mathbf{x}_{i}\right)^{T}\,\boldsymbol{J}_{f}\left(\mathbf{x}_{i}\right)+\mu\,\boldsymbol{I}\right]\,\boldsymbol{h}=-\boldsymbol{F}'\left(\mathbf{x}_{i}\right)\,,\qquad \boldsymbol{F}'\left(\mathbf{x}_{i}\right)=\boldsymbol{J}_{f}\left(\mathbf{x}_{i}\right)^{T}\,\boldsymbol{f} \tag{15}$$

where $\boldsymbol{I}$ is the unity matrix, and $\mu > 0$ is the so called damping parameter. This parameter controls the size and also the type of the step. For all $\mu > 0$ the coefficient matrix is positive definite, and this ensures that $\boldsymbol{h}$ is a descent direction.

This method is a combination of the Gauss-Newton method and the steepest-decent method [1, 4, 5]. For large values of μ we get

$$\boldsymbol{h} \simeq -\frac{1}{\mu}\,\mathbf{F}'\left(\mathbf{x}_{i}\right)\,, \tag{16}$$

i.e. a short step in the steepest-descent direction.

If μ, however, is very small, then $\boldsymbol{h}=\boldsymbol{h}_{NG}$, which is a very good step in the final stages of the iteration (when $\boldsymbol{x}_{i}$ is close to $\boldsymbol{x}^{\dagger}$, the solution) if the residual value $\boldsymbol{f}\left(\mathbf{x}^{\dagger}\right)$ is small enough.

The method is self-controlled by the gain factor

$$\rho = \frac{F(\boldsymbol{x}) - F(\boldsymbol{x}+\boldsymbol{h})}{L(\boldsymbol{0}) - L(\boldsymbol{h})}\,, \tag{17}$$

where the denominator is the gain, predicted by the linear model:

$$L(\boldsymbol{h}) = F(\mathbf{x}_i) + \boldsymbol{h}^T\,\boldsymbol{F}'(\mathbf{x}_i) + \tfrac{1}{2}\,\boldsymbol{h}^T\,\boldsymbol{J}_f(\mathbf{x}_i)^T\;\boldsymbol{J}_f(\mathbf{x}_i)\;\boldsymbol{h} \tag{18}$$

$$L(\boldsymbol{0}) - L(\boldsymbol{h}) = \tfrac{1}{2}\,\boldsymbol{h}^T\;\left[\mu\,\boldsymbol{h} - \boldsymbol{F}'(\mathbf{x}_i)\right] > 0 \tag{19}$$

If ρ is large then $L(\boldsymbol{h})$ is a good approximation to $F(\boldsymbol{x}+\boldsymbol{h})$, we can decrease μ that the next step should be closer to the Gauss-Newton step. If ρ is small but positive then $\boldsymbol{h}$ is a descent step but μ must be increased, that the next step should be closer to the steepest-direction step because the second order approximation of $F(\boldsymbol{x}+\boldsymbol{h})$ is not good enough. If ρ is negative then $\boldsymbol{h}$ is not even a descent step, the step must be canceled, μ must be increased, that the next step should be closer to the steepest-direction step because the second order approximation of $F(\boldsymbol{x}+\boldsymbol{h})$ is not good enough.

4 Trust-region methods

A trust-region optimization method defines a region (with its radius) around a test point. Within this region a simplified (usually not more than second order) model is built. The optimization step is determined by the optimal point of the model function within this region where the model function is trusted to be a good model of the cost function. [1]. If the improvement is well modelled by the model function than the step is executed and the radius of the trust region may be increased, otherwise the step is cancelled and the radius is decreased.

The earliest use of the term seems to be by Sorensen in 1982 [6].

There exist also trust-region methods without derivatives [7] that form a linear or quadratic models by interpolation to values of the cost function.

There are trust-region methods that uses only the first derivative of the residual vector $\boldsymbol{f}$. A classic method of this kind is the Powell's dog-leg method [10]: it combines the steepest-descent ($\boldsymbol{h}_{sd}$) and the Gauss-Newton ($\boldsymbol{h}_{GN}$) steps to get a step ($\boldsymbol{h}_{dl}$) not longer than the sugar (R) of the trust-region.

In some sense Levenberg-Marqhardt method is also a trust region method: by controlling the damping parameter μ the step length is controlled.

In the case of the true second order derivative Hessian matrix, the trust-region step can be calculated only if the eigenvalues and eigenvectors of the Hessian is determined [2]. Without solving the eigen-problem of the Hessian the trust region step can be calculated only iteratively [8, 9].

In the following I will concentrate on second order trust-region methods that build a true second order model of the cost function.

4.1 Second-order trust-region methods

These kind of trust-region methods are based on the second order model of the merit function obtained from the Taylor-expansion of $\boldsymbol{f}$,

$$F(\mathbf{x}_i + \boldsymbol{\xi}) \simeq L(\boldsymbol{\xi}) = F(\mathbf{x}_i) - \boldsymbol{\xi}^T\,\boldsymbol{b} + \tfrac{1}{2}\,\boldsymbol{\xi}^T\,\boldsymbol{H}\,\boldsymbol{\xi}\,, \tag{20}$$

where the derivatives of the cost function $F(\mathbf{x}_i)$ are exactly calculated

$$\boldsymbol{b} = -\boldsymbol{J}_f(\mathbf{x}_i)^T\;\boldsymbol{f} \quad , \qquad \boldsymbol{H} = \boldsymbol{J}_f(\boldsymbol{x}_i)^T\;\boldsymbol{J}_f(\boldsymbol{x}_i) + \boldsymbol{f}"(\boldsymbol{x}_i)\;\boldsymbol{f}(\boldsymbol{x}_i)\;. \tag{21}$$

A constrained optimization is made in the trust-region to find the minimizer of the model function.

The subspace optimization can be done inexactly but in this way the benefit of using the exact Hessian matrix is (partly) wasted.

Exact subspace optimization even of the model function can be very time-consuming because of the need for solving of the eigen-problem of the Hessian-matrix. However, when the calculation of the cost function is quite time consuming then the extra time of the evaluation of the eigen-problem of the Hessian can be negligible.

4.2 Path-finder trust-region algorithm

I proposed a trust-region method that is easy to implement: instead of searching for an optimal value, the optimal way to the minimizer of the original problem is followed within the trust region [11].

It can be supposed that the way along the local gradient to the local minimizer can be parametrized with some scalar (time-like) parameter t. The path towards the minimizer of the model function can be determined, as long as the eigen-system of the symmetric Hessian matrix $\boldsymbol{H}$ is found. The eigen-vectors and eigen-values can be effectively determined with the *tred2* and *tqli* functions of the Numerical Recipes [12]. The *tred2* and *tqli* functions work well up to 100 dimensions when one uses *long double* real variable in C.

The way toward the minimizer leads anti-parallel to the local gradient:

$$\frac{\mathrm{d}\,\boldsymbol{x}}{\mathrm{d}\,t} = \boldsymbol{b} - \boldsymbol{H}\,\boldsymbol{x}\,. \tag{22}$$

This equation is separated in the eigen-system of $\boldsymbol{H}$

$$\frac{\mathrm{d}\,x_j}{\mathrm{d}\,t} = b_j - \lambda_j\,x_j\,, \tag{23}$$

where x_j and g_j are the j-th component of the vectors $\boldsymbol{x}$ and $\boldsymbol{g}$, respectively; and λ_j is the j-th diagonal element of $\boldsymbol{H}$ in its eigen-system of coordinate. This equation can be integrated. If $\lambda_j \neq 0$ (or more precisely $|\lambda_j| > \varepsilon$, where ε is a small number) then

$$x_j\,(t) - x_{i\,j} = \frac{b_j}{\lambda_j}\,\left[1 - \exp\left(-\lambda_j\,t\right)\right]\,, \tag{24}$$

where $\boldsymbol{x}_i$ is the starting point. Otherwise (if $\lambda_j \approx 0$) (or more precisely $|\lambda_j| \leq \varepsilon$)

$$x_j\,(t) - x_{i\,j} = b_j\,t\,. \tag{25}$$

The region can be trusted only within an environment of radius R, that is the length of our step

$$r\,(t) = \sum_{i=1}^{m} \left\|\mathbf{x}\,(t) - \boldsymbol{x}_i\right\|^2 \tag{26}$$

could not be longer than R. Even if the Hessian matrix $\boldsymbol{H}$ is positive definite (all of its eigenvalues are positive) it cannot be longer than R. The restriction of $r\,(t)$ can be done by a one-dimensional successive approximation in t.

Finally one has to check that how good the second order approximation was. Depending on the value of the gain factor ρ in Eq. (17) one has to increase ($\rho > \rho_0 > 0$) or decrease ($\rho \leq \rho_0$) the radius of our trust region, and one even has to cancel the step if $\rho \leq 0$.

5 Comparison the efficiencies of LM and my trust-region method

The Path-finder trust-region algorithm described above is just a true second order method. In the following the Levenberg-Marquardt–method is tested: when it performs worse than a true second-order method.

I compared the performances of the Levenberg-Marquardt (LM) and Trust-Region (TR) methods on problems from the Ref. [4].

Test problem	LM	TR
Rosenbrock function (4)	23	22
Freunstein-Roth function (7)	36	7
Bard function (8)	5	11
Kowalik and Osborn function (9)	15	11
Meyer function (10)	176	207
Brown and Dennis function (14)	42	9
Exponential fit, 4 parameters (18)	65	44
Modified Meyer function (20)	83	415

Table 1. Levenberg-Marquardt (LM) and Trust-Region (TR) methods step numbers to achieve the minimizers of different optimization problems to the same accuracy

It can be seen in table 1. that in some problems the true second order trust-region method performs better. It seems that in these problems the approximations applied in the Levenberg-Marquardt–method are not good enough. In other problems the Levenberg-Marquardt–method performs even better than my trust-region method. The most significant case is the Modified Meyer function's problem where I beleive that the numerical instability of my method emerged in this case is the origin its partial failure. In my trust region much more problematic eigen-problems must be solved than in the case of Levenberg-Marquardt– method where only matrix inversion is made.

It needs additional investigations on different problems why the one or the other method performs better on.

Acknowledgement

This research is supported by EFOP-3.6.1-16-2016-00006 "The development and enhancement of the research potential at John von Neumann University" project. The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

References

[1] J. Nocedal, S.J. Wright, "Numerical Optimization", Springer-Verlag (1999).

[2] W. Shiquan, W. Fang, "Computation of a trust region step", Acta Math. Appl. Sinica, 7, 354-362 (1991).

[3] K. Madsen, H.B. Nielsen, O. Tingleff, "Methods for non-linear least square problems", IMM Department of Mathematical Modeling, Technical Paper, booklet 1999. (Available: http://www.imm.dtu.dk/pubdb/views/edoc_download.php/ 3215/pdf/imm3215.pdf Accessed: 2018.11.05.)

[4] H.B. Nielsen, "Damping parameter in Marquardt's method", IMM Department of Mathematical Modeling, Technical Paper, booklet 1999. (Available: http://www2.imm.dtu.dk/documents/ftp/tr99/tr05_99.pdf Accessed: 2018.11.05.)

[5] D. Ramadasan, M. Chevaldonné, T. Chateau, "LMA: A generic and efficient implementation of the Levenberg-Marquardt Algorithm", Sotfware: Praxtise and Experience, doi: 10.1002/spe.2497 (2017).

[6] D.C. Sorensen: Newton's Method with a Model Trust Region Modification, SIAM J. Numer. Anal., 19(2), 409426 (1982).

[7] B. Karasözen, "Survey of trust-region derivative free optimization methods", J. of Industr. and Managm. Optim., 3, 321-334 (2007).

[8] J.B Erway, P.E. Gill, J.D. Griffin, “Iterative methods for finding a trust region step”, SIAM J. Optim., 20, 1110-1131 (2009).

[9] J.B Erway, P.E. Gill, “A subspace minimization method for the trust-region step”, SIAM J. Optim., 20, 1439-1461 (2009).

[10] M.J.D. Powell, “A hybrid method for non-linear Equations”, In P. Rabinowitz (ed): Numerical methods for non-linear algebraic equations, Gordon&Breach, pp. 87-114 (1970).

[11] A. Kőházi-Kis, “Dielektrikumtükrök automatizált tervezése”, GAMF Közleményei, XXIII. évfolyam, 87-95 (2009).

[12] W.H. Press, S.A. Teukolsky, A.W.T. Vetterling, A.B. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, New York, 1992.