



WIKIPEDIA
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

[Current events](#)

[Random article](#)

[Donate to Wikipedia](#)

[Wikipedia store](#)

Interaction

[Help](#)

[About Wikipedia](#)

[Community portal](#)

[Recent changes](#)

[Contact page](#)

Tools

[What links here](#)

[Related changes](#)

[Upload file](#)

[Special pages](#)

[Permanent link](#)

[Page information](#)

[Wikidata item](#)

[Cite this page](#)

Print/export

[Create a book](#)

[Download as PDF](#)

[Printable version](#)

Languages

[Български](#)

[Català](#)

[Deutsch](#)

[Español](#)

[Français](#)

[Bahasa Indonesia](#)

[עברית](#)

[日本語](#)

[Português](#)

[Edit links](#)

[Create account](#) [Log in](#)

Article

[Talk](#)

[Read](#)

[Edit](#)

[View history](#)

Gauss–Newton algorithm

From Wikipedia, the free encyclopedia

The **Gauss–Newton algorithm** is a method used to solve [non-linear least squares](#) problems. It is a modification of [Newton's method](#) for finding a [minimum](#) of a [function](#). Unlike Newton's method, the Gauss–Newton algorithm can only be used to minimize a sum of squared function values, but it has the advantage that second derivatives, which can be challenging to compute, are not required.

Non-linear least squares problems arise for instance in [non-linear regression](#), where parameters in a model are sought such that the model is in good agreement with available observations.

The method is named after the mathematicians [Carl Friedrich Gauss](#) and [Isaac Newton](#).

Contents [\[hide\]](#)

- [1 Description](#)
- [2 Notes](#)
- [3 Example](#)
- [4 Convergence properties](#)
- [5 Derivation from Newton's method](#)
- [6 Improved versions](#)
- [7 Other applications](#)
- [8 Related algorithms](#)
- [9 Notes](#)
- [10 References](#)

Description [\[edit\]](#)

Given m functions $\mathbf{r} = (r_1, \dots, r_m)$ of n variables $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$, with $m \geq n$, the Gauss–Newton algorithm [iteratively](#) finds the minimum of the sum of squares^[1]

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

Starting with an initial guess $\boldsymbol{\beta}^{(0)}$ for the minimum, the method proceeds by the iterations

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} - (\mathbf{J}_r^T \mathbf{J}_r)^{-1} \mathbf{J}_r^T \mathbf{r}(\boldsymbol{\beta}^{(s)})$$

where, if \mathbf{r} and $\boldsymbol{\beta}$ are [column vectors](#), the entries of the [Jacobian matrix](#) are

$$(\mathbf{J}_r)_{ij} = \frac{\partial r_i(\boldsymbol{\beta}^{(s)})}{\partial \beta_j},$$

and the symbol \mathbf{T} denotes the [matrix transpose](#).

If $m = n$, the iteration simplifies to

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} - (\mathbf{J}_r)^{-1} \mathbf{r}(\boldsymbol{\beta}^{(s)})$$

which is a direct generalization of [Newton's method](#) in one dimension.

In data fitting, where the goal is to find the parameters $\boldsymbol{\beta}$ such that a given model function $y = f(x, \boldsymbol{\beta})$ best fits some data points (x_i, y_i) , the functions r_i are the [residuals](#)

$$r_i(\boldsymbol{\beta}) = y_i - f(x_i, \boldsymbol{\beta}).$$

Then, the Gauss–Newton method can be expressed in terms of the Jacobian \mathbf{J}_f of the function f as

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} + (\mathbf{J}_f^T \mathbf{J}_f)^{-1} \mathbf{J}_f^T \mathbf{r}(\boldsymbol{\beta}^{(s)}).$$

Notes [\[edit\]](#)

The assumption $m \geq n$ in the algorithm statement is necessary, as otherwise the matrix $\mathbf{J}_r^T \mathbf{J}_r$ is not invertible and the normal equations cannot be solved (at least uniquely).

The Gauss–Newton algorithm can be derived by [linearly approximating](#) the vector of functions r_i . Using [Taylor's theorem](#), we can write at every iteration:

$$\mathbf{r}(\boldsymbol{\beta}) \approx \mathbf{r}(\boldsymbol{\beta}^s) + \mathbf{J}_r(\boldsymbol{\beta}^s) \Delta$$

with $\Delta = \boldsymbol{\beta} - \boldsymbol{\beta}^s$. The task of finding Δ minimizing the sum of squares of the right-hand side, i.e.,

$$\min \|\mathbf{r}(\boldsymbol{\beta}^s) + \mathbf{J}_r(\boldsymbol{\beta}^s) \Delta\|_2^2,$$

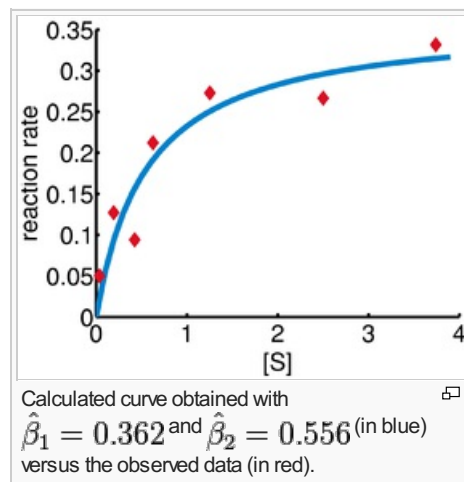
is a [linear least squares](#) problem, which can be solved explicitly, yielding the normal equations in the algorithm.

The normal equations are m linear simultaneous equations in the unknown increments, Δ . They may be solved in one step, using [Cholesky decomposition](#), or, better, the [QR factorization](#) of \mathbf{J}_r . For large systems, an [iterative method](#), such as the [conjugate gradient](#) method, may be more efficient. If there is a linear dependence between columns of \mathbf{J}_r , the iterations will fail as $\mathbf{J}_r^T \mathbf{J}_r$ becomes singular.

Example [\[edit\]](#)

In this example, the Gauss–Newton algorithm will be used to fit a model to some data by minimizing the sum of squares of errors between the data and model's predictions.

In a biology experiment studying the relation between substrate concentration $[S]$ and reaction rate in an enzyme-mediated reaction, the data in the following table were obtained.



| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------|-------|-------|-------|--------|--------|--------|--------|
| $[S]$ | 0.038 | 0.194 | 0.425 | 0.626 | 1.253 | 2.500 | 3.740 |
| rate | 0.050 | 0.127 | 0.094 | 0.2122 | 0.2729 | 0.2665 | 0.3317 |

It is desired to find a curve (model function) of the form

$$\text{rate} = \frac{V_{\max} [S]}{K_M + [S]}$$

that fits best the data in the least squares sense, with the parameters V_{\max} and K_M to be determined.

Denote by x_i and y_i the value of $[S]$ and the rate from the table, $i = 1, \dots, 7$. Let $\beta_1 = V_{\max}$ and $\beta_2 = K_M$. We will find β_1 and β_2 such that the sum of squares of the residuals

$$r_i = y_i - \frac{\beta_1 x_i}{\beta_2 + x_i} \quad (i = 1, \dots, 7)$$

is minimized.

The Jacobian \mathbf{J}_r of the vector of residuals r_i in respect to the unknowns β_j is an 7×2 matrix with the i -th row having the entries

$$\frac{\partial r_i}{\partial \beta_1} = -\frac{x_i}{\beta_2 + x_i}, \quad \frac{\partial r_i}{\partial \beta_2} = \frac{\beta_1 x_i}{(\beta_2 + x_i)^2}.$$

Starting with the initial estimates of $\beta_1=0.9$ and $\beta_2=0.2$, after five iterations of the Gauss–Newton algorithm the optimal values $\hat{\beta}_1 = 0.362$ and $\hat{\beta}_2 = 0.556$ are obtained. The sum of squares of residuals decreased from the initial value of 1.445 to 0.00784 after the fifth iteration. The plot in the figure on the right shows the curve determined by the model for the optimal parameters versus the observed data.

Convergence properties [\[edit\]](#)

It can be shown^[2] that the increment Δ is a [descent direction](#) for S , and, if the algorithm converges, then the limit is a [stationary point](#) of S . However, convergence is not guaranteed, not even [local convergence](#) as in [Newton's method](#).

The rate of convergence of the Gauss–Newton algorithm can approach [quadratic](#).^[3] The algorithm may converge slowly or not at all if the initial guess is far from the minimum or the matrix $\mathbf{J}_r^T \mathbf{J}_r$ is [ill-conditioned](#). For example, consider the problem with $m = 2$ equations and $n = 1$ variable, given by

$$\begin{aligned} r_1(\beta) &= \beta + 1 \\ r_2(\beta) &= \lambda\beta^2 + \beta - 1. \end{aligned}$$

The optimum is at $\beta = 0$. (Actually the optimum is at $\beta = -1$ for $\lambda = 2$, because $S(0) = 1^2 + (-1)^2 = 2$, but $S(-1) = 0$.) If $\lambda = 0$ then the problem is in fact linear and the method finds the optimum in one iteration. If $|\lambda| < 1$ then the method converges linearly and the error decreases asymptotically with a factor $|\lambda|$ at every iteration. However, if $|\lambda| > 1$, then the method does not even converge locally.^[4]

Derivation from Newton's method [\[edit\]](#)

In what follows, the Gauss–Newton algorithm will be derived from [Newton's method](#) for function optimization via an approximation. As a consequence, the rate of convergence of the Gauss–Newton algorithm can be quadratic under certain regularity conditions. In general (under weaker conditions), the convergence rate is linear.^[5]

The recurrence relation for Newton's method for minimizing a function S of parameters, β , is

$$\beta^{(s+1)} = \beta^{(s)} - \mathbf{H}^{-1} \mathbf{g}$$

where \mathbf{g} denotes the [gradient vector](#) of S and \mathbf{H} denotes the [Hessian matrix](#) of S . Since $S = \sum_{i=1}^m r_i^2$, the gradient is given by

$$g_j = 2 \sum_{i=1}^m r_i \frac{\partial r_i}{\partial \beta_j}.$$

Elements of the Hessian are calculated by differentiating the gradient elements, g_j , with respect to β_k

$$H_{jk} = 2 \sum_{i=1}^m \left(\frac{\partial r_i}{\partial \beta_j} \frac{\partial r_i}{\partial \beta_k} + r_i \frac{\partial^2 r_i}{\partial \beta_j \partial \beta_k} \right).$$

The Gauss–Newton method is obtained by ignoring the second-order derivative terms (the second term in this expression). That is, the Hessian is approximated by

$$H_{jk} \approx 2 \sum_{i=1}^m J_{ij} J_{ik}$$

where $J_{ij} = \frac{\partial r_i}{\partial \beta_j}$ are entries of the Jacobian \mathbf{J}_r . The gradient and the approximate Hessian can be written in matrix notation as

$$\mathbf{g} = 2\mathbf{J}_r^T \mathbf{r}, \quad \mathbf{H} \approx 2\mathbf{J}_r^T \mathbf{J}_r.$$

These expressions are substituted into the recurrence relation above to obtain the operational equations

$$\beta^{(s+1)} = \beta^{(s)} + \Delta; \quad \Delta = -(\mathbf{J}_r^T \mathbf{J}_r)^{-1} \mathbf{J}_r^T \mathbf{r}.$$

Convergence of the Gauss–Newton method is not guaranteed in all instances. The approximation

$$\left| r_i \frac{\partial^2 r_i}{\partial \beta_j \partial \beta_k} \right| \ll \left| \frac{\partial r_i}{\partial \beta_j} \frac{\partial r_i}{\partial \beta_k} \right|$$

that needs to hold to be able to ignore the second-order derivative terms may be valid in two cases, for which convergence is to be expected.^[6]

1. The function values r_i are small in magnitude, at least around the minimum.
2. The functions are only "mildly" non linear, so that $\frac{\partial^2 r_i}{\partial \beta_j \partial \beta_k}$ is relatively small in magnitude.

Improved versions [\[edit\]](#)

With the Gauss–Newton method the sum of squares S may not decrease at every iteration. However, since Δ is a descent direction, unless $S(\beta^s)$ is a stationary point, it holds that $S(\beta^s + \alpha\Delta) < S(\beta^s)$ for all sufficiently small $\alpha > 0$. Thus, if divergence occurs, one solution is to employ a fraction, α , of the increment

vector, Δ in the updating formula

$$\beta^{s+1} = \beta^s + \alpha \Delta.$$

In other words, the increment vector is too long, but it points in "downhill", so going just a part of the way will decrease the objective function S . An optimal value for α can be found by using a [line search](#) algorithm, that is, the magnitude of α is determined by finding the value that minimizes S , usually using a [direct search method](#) in the interval $0 < \alpha < 1$.

In cases where the direction of the shift vector is such that the optimal fraction, α , is close to zero, an alternative method for handling divergence is the use of the [Levenberg–Marquardt algorithm](#), also known as the "trust region method".^[1] The normal equations are modified in such a way that the increment vector is rotated towards the direction of [steepest descent](#),

$$(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{D}) \Delta = -\mathbf{J}^T \mathbf{r},$$

where \mathbf{D} is a positive diagonal matrix. Note that when D is the identity matrix and $\lambda \rightarrow +\infty$, then $\Delta/\lambda \rightarrow -\mathbf{J}^T \mathbf{r}$, therefore the [direction](#) of Δ approaches the direction of the negative gradient $-\mathbf{J}^T \mathbf{r}$.

The so-called Marquardt parameter, λ , may also be optimized by a line search, but this is inefficient as the shift vector must be re-calculated every time λ is changed. A more efficient strategy is this. When divergence occurs increase the Marquardt parameter until there is a decrease in S . Then, retain the value from one iteration to the next, but decrease it if possible until a cut-off value is reached when the Marquardt parameter can be set to zero; the minimization of S then becomes a standard Gauss–Newton minimization.

Other applications [\[edit\]](#)

The Gauss–Newton algorithm is a popular method for solving nonlinear inverse problems. A particular application is generating computational models of oil and gas reservoirs for consistency with observed production data.^[7]

Related algorithms [\[edit\]](#)




In a [quasi-Newton method](#), such as that due to [Davidon](#), [Fletcher and Powell](#) or Broyden–Fletcher–Goldfarb–Shanno ([BFGS method](#)) an estimate of the full Hessian, $\frac{\partial^2 S}{\partial \beta_j \partial \beta_k}$, is built up numerically using first derivatives

$\frac{\partial r_i}{\partial \beta_j}$ only so that after n refinement cycles the method closely approximates to Newton's method in

performance. Note that quasi-Newton methods can minimize general real-valued functions, whereas Gauss–Newton, Levenberg–Marquardt, etc. fits only to nonlinear least-squares problems.

Another method for solving minimization problems using only first derivatives is [gradient descent](#). However, this method does not take into account the second derivatives even approximately. Consequently, it is highly inefficient for many functions, especially if the parameters have strong interactions.

Notes [\[edit\]](#)

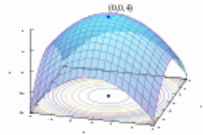
- ^a ^b Björck (1996)
- [^] Björck (1996) p260
- [^] Björck (1996) p341, 342
- [^] Fletcher (1987) p.113
- [^] <http://www.henley.ac.uk/web/FILES/maths/09-04.pdf> 
- [^] Nocedal (1999) p259
- [^] "History matching production data and uncertainty assessment with an efficient TSVD parameterization algorithm" . *Journal of Petroleum Science and Engineering* **113**: 54–71. doi:10.1016/j.petrol.2013.11.025 .

References [\[edit\]](#)

- Björck, A. (1996). *Numerical methods for least squares problems*. SIAM, Philadelphia. ISBN 0-89871-360-9.
- Fletcher, Roger (1987). *Practical methods of optimization* (2nd ed.). New York: John Wiley & Sons. ISBN 978-0-471-91547-8..
- Nocedal, Jorge; Wright, Stephen (1999). *Numerical optimization*. New York: Springer. ISBN 0-387-98793-2.

| | | |
|--|--|------------------------|
| v t e | Least squares and regression analysis | [show] |
| v t e | Optimization: Algorithms, methods, and heuristics | [hide] |

| | |
|---|------------------------|
| Unconstrained nonlinear: Methods calling ... | [show] |
| Constrained nonlinear | [show] |
| Convex optimization | [show] |
| Combinatorial | [show] |
| Metaheuristics | [show] |
| Categories (Algorithms and methods · Heuristics) · Software | |



Categories: [Optimization algorithms and methods](#) | [Least squares](#) | [Statistical algorithms](#)

This page was last modified on 14 August 2015, at 18:19.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

