ЧИСЛЕННЫЕ МЕТОДЫ ОПТИМИЗАЦИИ

Оптимизация. Алгоритм Левенберга—Марквардта

Константин Ловецкий

Ноябрь 2012

Кафедра систем телекоммуникаций

Метод Левенберга—Марквардта

- **Алгоритм Левенберга — Марквардта** — метод оптимизации, направленный на решение задач о наименьших квадратах. Является альтернативой методу Гаусса — Ньютона. Может рассматриваться как комбинация последнего с методом градиентного спуска или как метод доверительных интервалов. Алгоритм был сформулирован независимо Левенбергом (1944) и Марквардтом (1963).

- Kenneth Levenberg (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares". *The Quarterly of Applied Mathematics* **2**: 164–168.
- Donald Marquardt (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". *SIAM Journal on Applied Mathematics* **11**(2): 431–441. doi:10.1137/0111030.
- Philip E. Gill and Walter Murray (1978). "Algorithms for the solution of the nonlinear least-squares problem". *SIAM Journal on Numerical Analysis* **15** (5): 977–992. doi:10.1137/0715063.

The algorithm was first published by Kenneth Levenberg, while working at the Frankford Army Arsenal. It was rediscovered by Donald Marquardt who worked as a statistician at DuPont and independently by Girard, Wynn and Morrison.

Метод Левенберга—Марквардта

- **Алгоритм Левенберга—Марквардта** (Levenberg-Marquardt Algorithm, LMA) является наиболее распространенным алгоритмом оптимизации. Он превосходит по производительности метод наискорейшего спуска и другие методы сопряженных градиентов в различных задачах. Изначально считалось, что LMA – это ***комбинация*** простейшего **градиентного метода** и **метода Гаусса-Ньютона**, однако впоследствии выяснилось, что данный алгоритм можно также рассматривать как **метод доверительных интервалов**.

Метод Левенберга—Марквардта

LMA решает задачу нелинейной минимизации методом наименьших квадратов. Это означает, что функция, которую необходимо минимизировать, выглядит следующим образом:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)$$

где $x = (x_1, x_2, ..., x_n)$ - вектор, а r_j - функция отображения из R^n в R .

Функцию r_j называют невязкой в предположении, что $m \geq n$.

Для простоты функция f представляется вектором невязки вида:

$$r(x) = (r_1(x), r_2(x), ..., r_m(x))$$

Метод Левенберга—Марквардта

Теперь f можно переписать как

$$f(x) = \frac{1}{2}\|r(x)\|^2,$$

а ее производные представить с помощью матрицы Якоби

$$J(x) = \frac{\partial r_j}{\partial x_i}, \quad 1 \le j \le m, \quad 1 \le i \le n$$

Рассмотрим линейный случай, когда каждая функция r_j линейна. Здесь якобиан равен константе, r можно представить как гиперплоскость в пространстве, а

$$f(x) = \frac{1}{2}\|Jx + r(0)\|^2$$

Метод Левенберга—Марквардта

Тогда градиент функции $\nabla f(x) = J^T(Jx + r)$ и $\nabla^2 f(x) = J^T J$. Решая задачу минимума $\nabla f(x) = 0$, получим

$$x_{\min} = -(J^T J)^{-1} J^T r$$

то есть $x_{\min}$ - решение системы нормальных уравнений

$$(J^T J)x = -J^T r.$$

Возвращаясь к общему (нелинейному) случаю, получим:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^T r(x)$$

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x)$$

Метод Левенберга—Марквардта

Отличительной особенностью метода наименьших квадратов является то, что, имея матрицу Якоби $J(x)$, легко получить гессиан $\nabla^2 f(x)$, если функции r_j можно аппроксимировать линейными приближениями (т.е. $\nabla^2 r_j(x)$ малы) или если $r_j(x)$ малы сами по себе.

Тогда гессиан, как и в линейном случае, будет равен:

$$\nabla^2 f(x) = J(x)^T J(x)$$

Важно отметить, что это уравнение верно только для малых невязок. Проблемы больших невязок не могут быть решены с помощью квадратичной аппроксимации, и, следовательно, производительность алгоритма, представленного выше, в таких случаях невелика.

Метод Левенберга—Марквардта

LMA как комбинация простейшего градиентного метода и метода Ньютона— Гаусса

Простейший градиентный метод – это наиболее интуитивно понятный способ нахождения минимума функции. Вычисление параметра на очередном шаге выполняется путем вычитания градиента функции, умноженного на заданный положительный коэффициент:

$$x_{i+1} = x_i - \lambda \nabla f$$

Однако при таком подходе имеют место различные проблемы сходимости. Логично предположить, что желательно было бы осуществлять большие шаги по направлению градиента там, где градиент мал (т.е. наклон пологий), и, наоборот, маленькие шаги там, где градиент большой, чтобы не пропустить минимум.

Однако в формуле выполняются прямо противоположные действия.

Метод Левенберга—Марквардта

Другая проблема заключается в том, что кривизна поверхности невязки может быть не одинаковой по всем направлениям. К примеру, если есть длинная и узкая впадина на поверхности невязки, компонент градиента в направлении, указывающем вдоль основания впадины, очень мал, а компонент градиента вдоль стенок впадины, наоборот, велик. Это приводит к движению по направлению к стенкам впадины, тогда как необходимо перемещаться на большие расстояния вдоль основания впадины и на малые – вдоль ее стенок.

Ситуацию можно улучшить, если учитывать информацию о кривизне и градиенте, т. е. вторые производные. Один из способов сделать это – использовать метод Ньютона для решения уравнения $\nabla f(x) = 0$.

Раскладывая градиент f в ряд Тейлора вокруг текущего состояния x_0, получим

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0)$$

Метод Левенберга—Марквардта

Пренебрегая членами более высокого порядка (считая f квадратичной вблизи x_0) и решая задачу минимума, приравняв левую часть уравнения

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0)$$

к нулю, получим правило вычисления параметра на очередном шаге по методу Ньютона:

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i)$$

Поскольку метод Ньютона напрямую использует предположение о квадратичности (пренебрегая членами более высоких порядков при разложении в ряд Тейлора), нет необходимости точно вычислять гессиан, а достаточно использовать его аппроксимацию.
Главное достоинство такого подхода – быстрая сходимость. Однако скорость сходимости зависит от начального положения (если быть более точным - от линейности вокруг начального положения).

Метод Левенберга—Марквардта

Легко заметить, что простейший градиентный метод и метод Ньютона— Гаусса дополняют друг друга с точки зрения предоставляемых преимуществ. Основываясь на этом наблюдении, Левенберг предложил алгоритм, в котором правило вычисления параметра

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i)$$

есть комбинация правил: $x_{i+1} = x_i - \lambda \nabla f$

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i)$$

где H – матрица Гессе, вычисленная в точке x_i.

Метод Левенберга—Марквардта

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i)$$

Данное правило используется следующим образом: если на очередной итерации невязка сокращается, это значит, что предположение о квадратичности $f(x)$ работает, и мы уменьшаем λ (обычно в 10 раз), чтобы понизить влияние градиентного спуска. С другой стороны, если невязка увеличивается, необходимо следовать направлению градиента, и мы увеличиваем λ (во столько же раз).

Метод Левенберга—Марквардта

Таким образом, алгоритм Левенберга представляется в виде последовательности действий:

- 1. Вычислить параметр на очередной итерации по правилу
$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i)$$
- 2. Оценить невязку в новом векторе параметров.
- 3. Если в результате вычисления параметра невязка увеличилась, вернуться на шаг назад (т.е. восстановить прежние значения весов) и увеличить λ в 10 раз. Затем повторить выполнение, начиная с шага 1.
- 4. Если в результате вычисления параметра невязка уменьшилась, принять текущий шаг (т.е. оставить новые значения весов) и уменьшить λ в 10 раз.

Метод Левенберга—Марквардта

Недостатком данного алгоритма является то, что если значение λ велико, вычисленная матрица Гессе никак не используется. Однако можно извлечь некоторую выгоду из второй производной даже в этом случае, масштабируя каждый компонент градиента согласно кривизне. Это должно привести к увеличению шага вдоль направлений, где градиент мал, так что классическая проблема впадины больше не возникнет. Этот ключевой момент был замечен Марквардтом. Он заменил единичную матрицу в формуле на диагональ гессиана, получив таким образом следующее правило:

$$x_{i+1} = x_i - (H + \lambda \ \mathrm{diag}[H])^{-1} \nabla f(x_i)$$

(Тихоновская регуляризация)

Метод Левенберга—Марквардта

- Поскольку гессиан пропорционален кривизне f, правило приведет к большим шагам при малой кривизне (т.е. для почти плоской поверхности) и к малым шагам при большой кривизне (т.е. для крутого наклона).
- Стоит отметить, что хотя LMA является не оптимальным, а лишь эвристическим методом, он очень хорошо работает на практике. Единственный его недостаток заключается в необходимости обращения матрицы на каждом шаге. Даже несмотря на то, что нахождение обратной матрицы обычно выполняется с использованием быстрых методов псевдообращения (таких, как разложение по сингулярным числам матрицы), время одной итерации становится неприемлемым для нескольких тысяч параметров. Для моделей же средних размеров (с несколькими сотнями параметров) LMA работает даже быстрее, чем простейший градиентный метод.

Метод Левенберга—Марквардта

Метод доверительных интервалов

Historically, the LM algorithm was presented by Marquardt as given in the previous section where the parameter, λ, was manipulated directly to find the minimum. Subsequently, a trust-region approach to the algorithm has gained ground.

Trust-region algorithms work in a fundamentally different manner than those presented in the previous section, which are called *line-search methods. In a line search method, we decide on a* direction in which to descend the gradient and are then concerned about the step size, i.e. if $p^{(k)}$ is the direction of descent, and α_k *the stepsize, then our step is given by* $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$ *and* the stepsize is obtained by solving the sub-problem

$$\min_{\forall \alpha_k > 0} f(x^{(k)} + \alpha_k p^{(k)})$$

Метод Левенберга—Марквардта

Метод доверительных интервалов

By contrast, in a trust-region algorithm we build a model ***m(k)*** *that approximates the function* ***f*** in a finite region near ***x(k)****. This region,* ***Δ****, where the model is a good approximation of* ***f*** *, is called* the trust-region. Trust-region algorithms maintain ***Δ*** and update it at each iteration using ***heuristics***.

The model ***m(k)*** *is most often a quadratic obtained by a Taylor series expansion of* ***f*** *around* ***x(k)****,* i.e.

$$m^{(k)} = f(x^{(k)}) + \nabla f(x^{(k)})^{\mathrm{T}} p + \frac{1}{2} p^T H p$$

where ***H*** *is the Hessian (or an approximation of the Hessian) matrix. The sub-problem to be solved* to find the step to take during the iteration is

$$\min_{\|p\| \le \Delta} f(x^{(k)}) + \nabla f(x^{(k)})^{\mathrm{T}} p + \frac{1}{2} p^T H p$$

Метод Левенберга—Марквардта

Метод доверительных интервалов

The iteration step itself is $x^{(k+1)} = x^{(k)} + p$. *A trust-region algorithm can thus be conceived of* as a sequence of iterations, in each of which we model the function ***f*** *by a quadratic and then jump* to the minimum of that quadratic.

The solution of problem is given by a theorem which is as follows

p^* is a global solution of $min_{\|p\|<\Delta} f\left(x^{(k)}\right) + \nabla f\left(x^{(k)}\right) \cdot p + \frac{1}{2} p^T H p$ iff

$\| p^* \| \leq \Delta$ and there is a scalar λ s.t. :

$$\begin{aligned} (H + \lambda I)\, p^* &= -g \\ \lambda\,(\Delta - p^*) &= 0 \end{aligned}$$

and $(H + \lambda I)$ is positive semi-definite

where $g, f \in \Re^n$.

Метод Левенберга—Марквардта

Метод доверительных интервалов

It can be seen that $\lambda(\Delta - p^*) = 0$ basically states that if $\|p^*\| < \Delta$ *then* $\lambda = 0$ *but* not otherwise. Hence, we reach the same parameter update equation for the LM algorithm using a trust-region framework as we obtained using the line-search method. The heuristic to update the size of the trust-region usually depends on the ratio of the expected change in ***f*** *to the predicted change, i.e.*

$$\rho_k = \frac{f(w^{(k)}) - f(w^{(k)} + p^*)}{f(w^{(k)}) - m^{(k)}(p^*)}$$

If there is a good agreement between predicted and actual values($\rho_k \approx 1$), *then* ***Δ*** *is increased; if* the agreement is poor (ρ_k *is small), then* ρ_k *is decreased. If* ρ_k *is smaller than a threshold value* (10^{-4}), the step is rejected and the value of $w^{(k)}$ *is retained but* ***Δ*** *is decreased as before. Thus,* the algorithm is similar to the previous one but the value that is changed with each iteration is ***Δ*** and not λ .

Метод Левенберга—Марквардта

Descriptions

1. Detailed description of the algorithm can be found in Numerical Recipes in C, Chapter 15.5: Nonlinear models
2. C. T. Kelley, *Iterative Methods for Optimization*, SIAM Frontiers in Applied Mathematics, no 18, 1999, ISBN 0-89871-433-8. Online copy
3. History of the algorithm in SIAM news
4. A tutorial by Ananth Ranganathan
5. Methods for Non-Linear Least Squares Problems by K. Madsen, H.B. Nielsen, O. Tingleff is a tutorial discussing non-linear least-squares in general and the Levenberg-Marquardt method in particular
6. T. Strutz: *Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond)*. Vieweg+Teubner, ISBN 978-3-8348-1022-9.

Метод Левенберга—Марквардта

Implementations

- ***The oldest implementation still in use is lmdif, from MINPACK, in Fortran, in the public domain.*** See also:
  - lmfit, a translation of lmdif into C/C++ with an easy-to-use wrapper for curve fitting, public domain.
  - The GNU Scientific Library library has a C interface to MINPACK.
  - C/C++ Minpack includes the Levenberg–Marquardt algorithm.
  - Several high-level languages and mathematical packages have wrappers for the MINPACK routines, among them:
    - Python library scipy, module scipy.optimize.leastsq,
    - IDL, add-on MPFIT.
    - R (programming language) has the minpack.lm package.
- levmar is an implementation in C/C++ with support for constraints, distributed under the GNU General Public License.
  - levmar includes a MEX file interface for MATLAB
  - Perl (PDL), python and Haskell interfaces to levmar are available: see PDL::Fit::Levmar, PyLevmar and HackageDB levmar.
- sparseLM is a C implementation aimed at minimizing functions with large, arbitrarily sparse Jacobians. Includes a MATLAB MEX interface.
- InMin library contains a C++ implementation of the algorithm based on the eigen C++ linear algebra library. It has a pure C-language API as well as a Python binding
- ***ALGLIB has implementations of improved LMA in C# / C++ / Delphi / Visual Basic. Improved algorithm takes less time to converge and can use either Jacobian or exact Hessian***.

Example

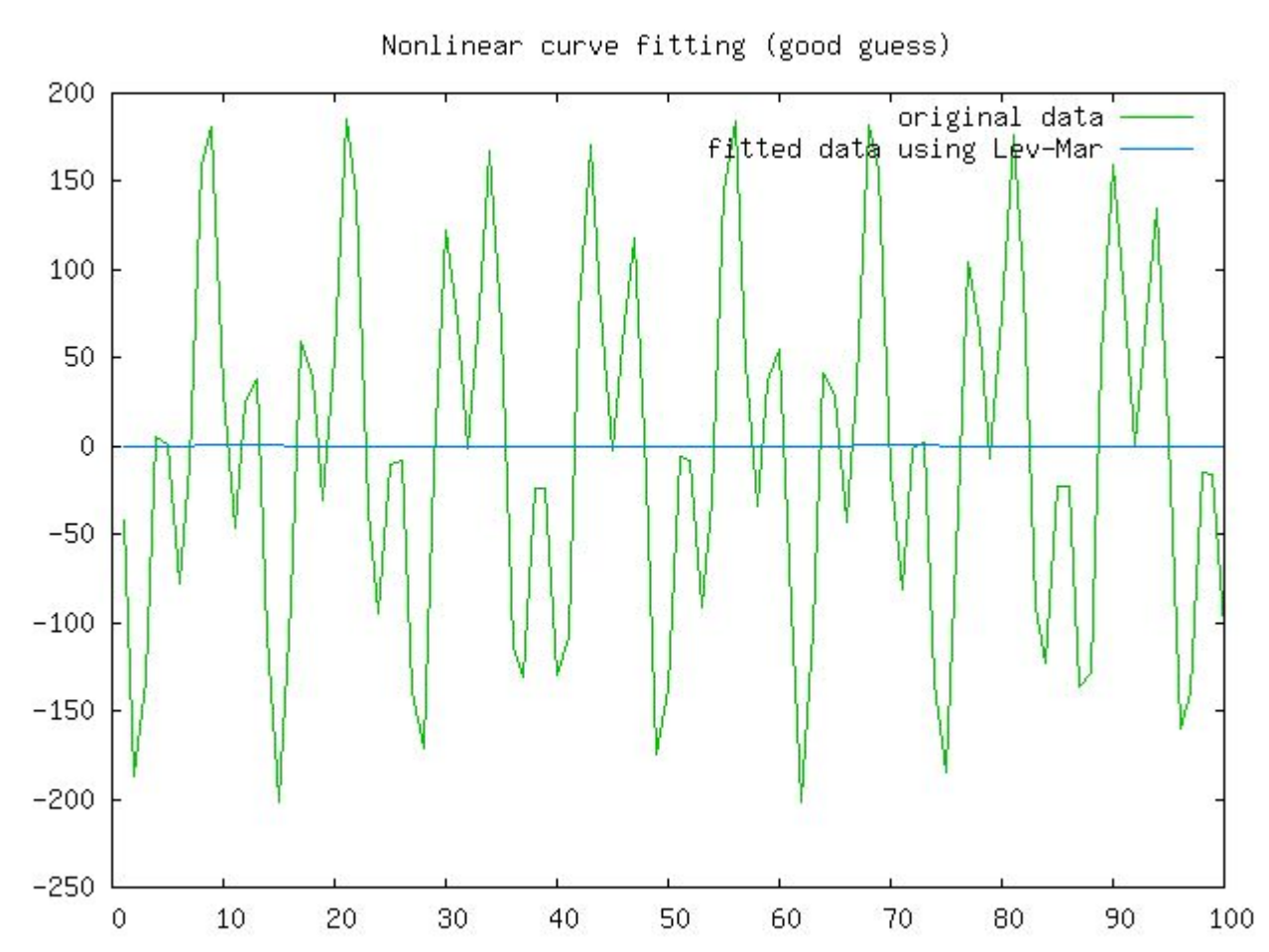
In this example we try to fit the function

$$y = a\cos(bX) + b\sin(aX)$$

using the Levenberg–Marquardt algorithm implemented in GNU Octave as the ***leasqr*** function.

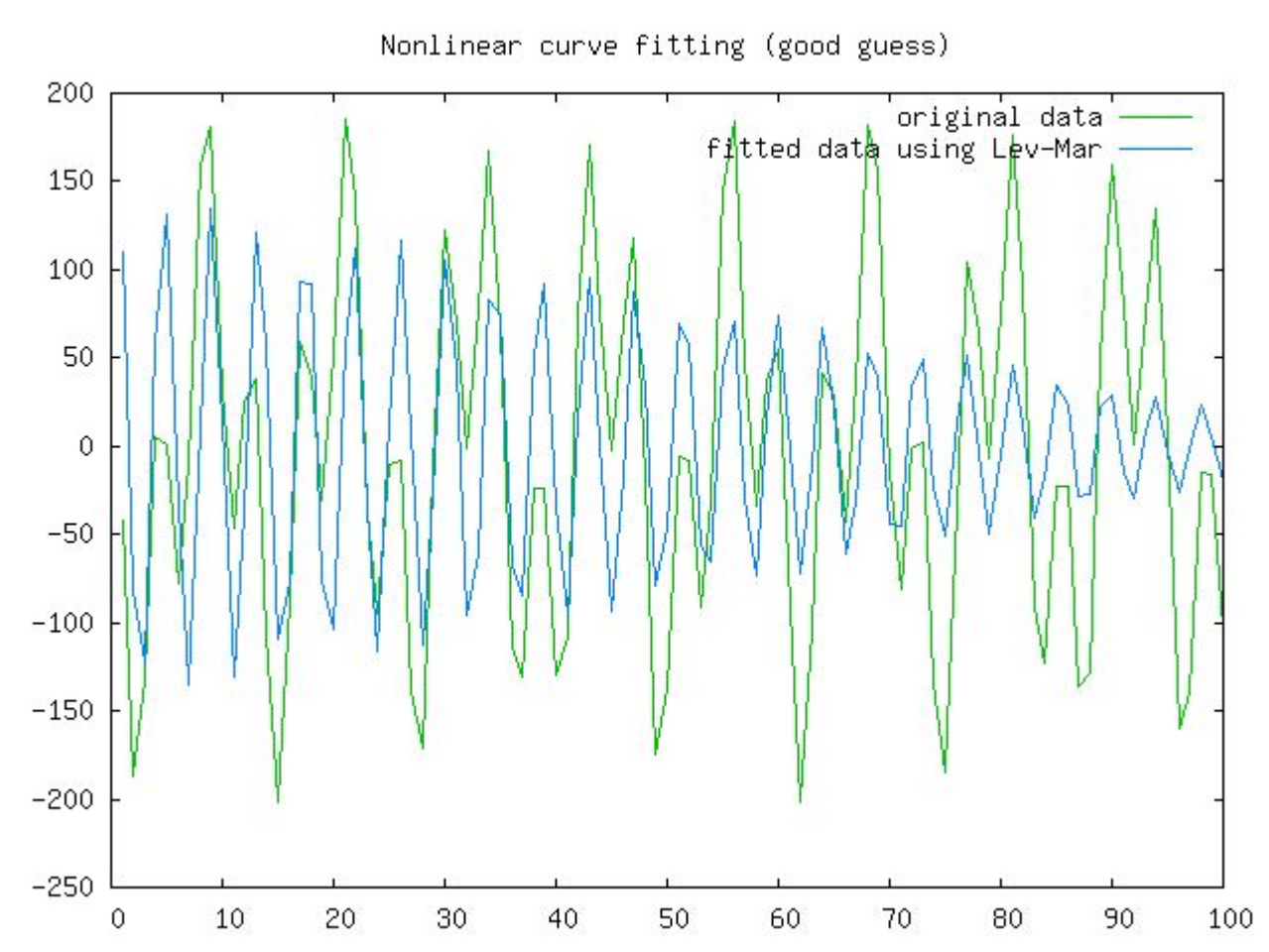
The 3 graphs Fig 1,2,3 show progressively better fitting for the parameters a=100, b=102 used in the initial curve. Only when the parameters in Fig 3 are chosen closest to the original, are the curves fitting exactly. This equation is an example of very sensitive initial conditions for the Levenberg–Marquardt algorithm. One reason for this sensitivity is the existence of multiple minima — the function $\cos(\beta x)$ has minima at parameter value b* and b*+2pi.

Example (1-st of 3)



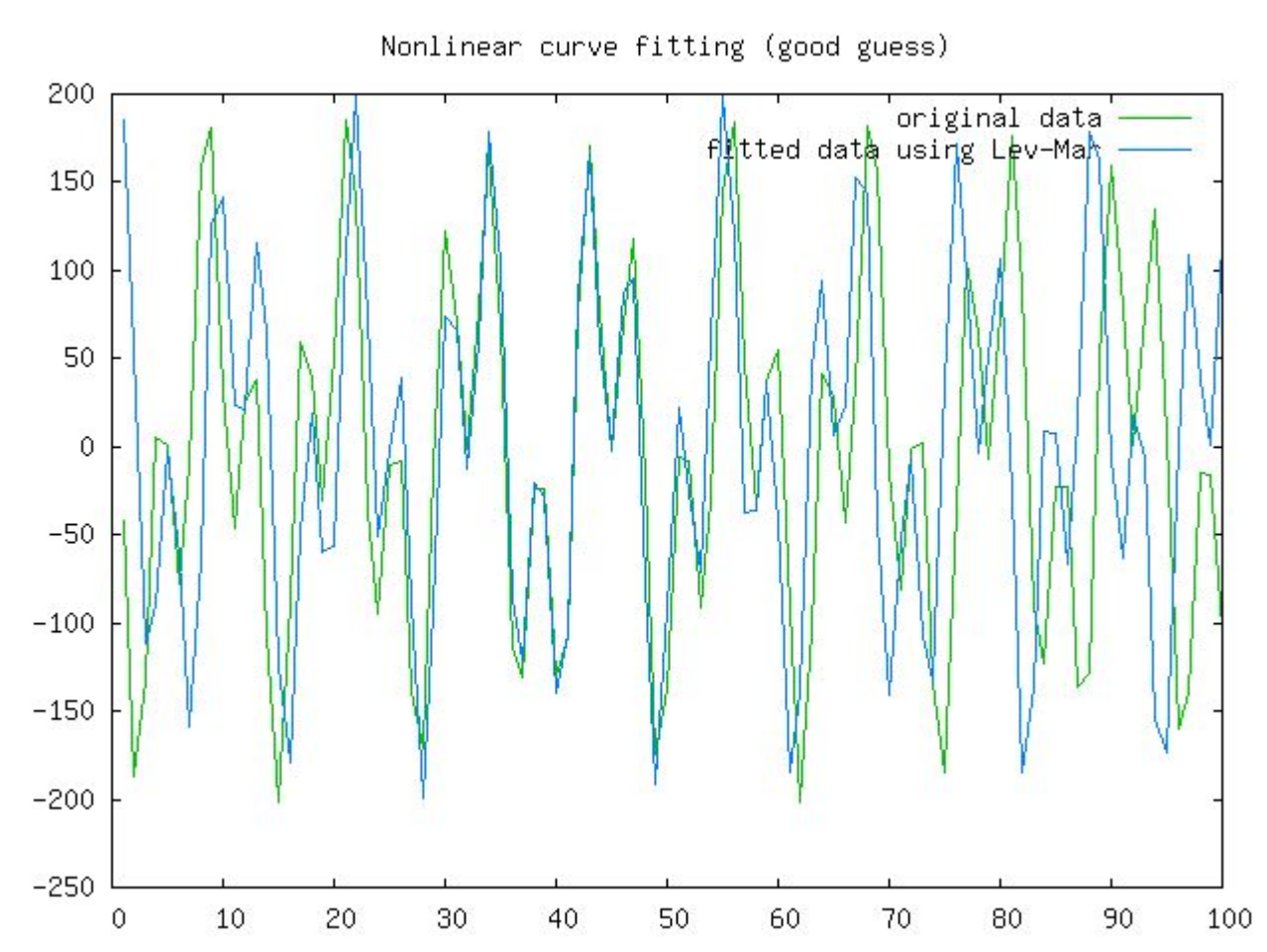
- Fig 1. Poor Fit

Example (2-nd of 3)



- Fig 2. Better Fit

Example (the last picture)



- Fig 3. Best Fit