Simulation of Nanosystems for Energy Conversion
Department of Electrical and Computer Engineering
Technical University of Munich

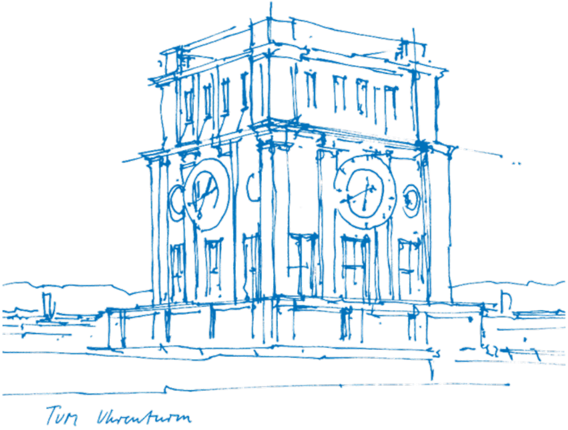
Python for Engineering Data Analysis

Least squares fitting

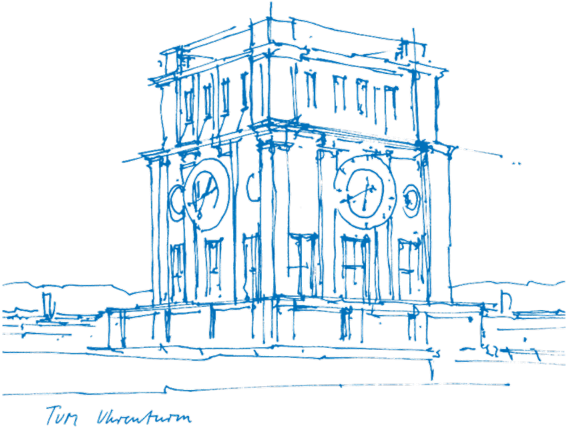
Michael Rinderle and Felix Mayr

Simulation of Nanosystems for Energy Conversion
Department of Electrical and Computer Engineering
Technical University of Munich

June 6, 2020



Linear least squares

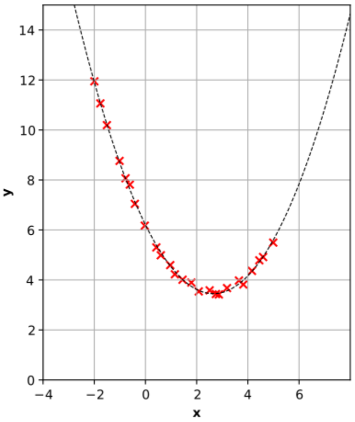


Linear least squares

- given a set of N measurements $(\vec{x}_i, y_i)$ with $i = 1, 2, \ldots, N$.
- find the best fitting curve $f(\vec{x}, \vec{\beta})$ with a set of parameters $\vec{\beta}$.
- in other words: find the parameters $\vec{\beta}$ that minimize the squared error ϵ^2.

$$\epsilon_i = y_i - f(\vec{x}_i, \vec{\beta}) \tag{1}$$

$$\epsilon^2 = \sum_{i=1}^{N} \epsilon_i^2 = \vec{\epsilon}^{\mathrm{T}} \cdot \vec{\epsilon} \tag{2}$$



Linear least squares

- define a target function $f(\vec{x}, \vec{\beta})$.
- if the target function is linear in the parameters $\vec{\beta}$ one can decompose f into

$$f(\vec{x}, \vec{\beta}) = \beta_0 + \beta_1 f_1(\vec{x}) + \beta_2 f_2(\vec{x}) + \ldots \tag{3}$$

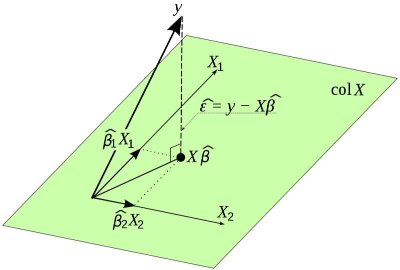
$$= (1, f_1, f_2, \ldots) \cdot \vec{\beta} = \boldsymbol{X} \cdot \vec{\beta} \tag{4}$$

- $\Longrightarrow$ solve (approximate) the linear system

$$\boldsymbol{X} \cdot \vec{\beta} \approx \vec{y} = \vec{y}_{\parallel} + \vec{y}_{\perp} \tag{5}$$

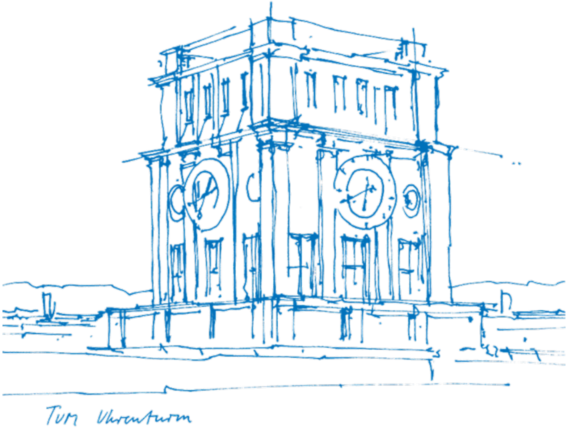
$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \cdot \vec{\beta} = \boldsymbol{X}^{\mathrm{T}} \cdot \vec{y}_{\parallel} + \cancel{\boldsymbol{X}^{\mathrm{T}} \cdot \vec{y}_{\perp}} \tag{6}$$

$$\vec{\beta} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1} \cdot \boldsymbol{X}^{\mathrm{T}}\vec{y} \tag{7}$$



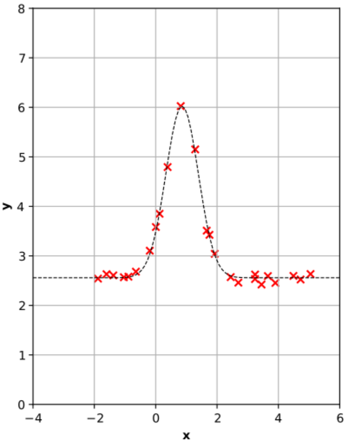
Geometric interpretation:
Find the closest point in the column space of $\boldsymbol{X}$ to the point $\vec{y}$.

Non-linear least squares



Non-linear least squares

- for non-linear problems the parameters $\vec{\beta}$ are not independent and a decomposition of f is not possible.
- an iterative algorithm is necessary to find the solution.
  - Gradient descent method
  - Gauss-Newton method
  - Levenberg-Marquardt method
- start with an initial set of parameters β_0 and update them by a step $\Delta\beta$ minimizing the error function ϵ^2 step by step.



Gradient descent method

- Compute the gradient of the error function ϵ^2 with respect to each parameter β_i

$$\frac{\partial \epsilon^2}{\partial \beta_i} = -2 \sum_{j=1}^{N} \epsilon_j \cdot \frac{\partial f(\vec{x}_j, \vec{\beta})}{\partial \beta_i} \tag{8}$$

$$\nabla_\beta \epsilon^2 = -2 \vec{\epsilon}^{\mathrm{T}} \cdot \boldsymbol{J} \tag{9}$$

- with the jacobian matrix $\boldsymbol{J}$ collecting all the derivatives with respect to β_i (columns) evaluated at all measurement points x_j (rows).
- advance a (small) step α along the negative gradient to update the parameters.

$$\Delta \vec{\beta} = \alpha \boldsymbol{J}^{\mathrm{T}} \vec{\epsilon} \tag{10}$$

Gauss-Newton method

- linearize the problem at the current parameter set β^k and find the best fitting parameters for the linearized problem. (k is the iteration counter)
- update the parameters and repeat the linearization at the new position β^{k+1}.

$$f(x, \vec{\beta}^{k+1}) = f(x, \vec{\beta}^k) + \frac{\partial f}{\partial \beta_1} \Delta\beta_1 + \frac{\partial f}{\partial \beta_2} \Delta\beta_2 + \cdots = y \tag{11}$$

$$\boldsymbol{J} \Delta\vec{\beta} = \vec{y} - f(x, \vec{\beta}^k) = \vec{\epsilon} \tag{12}$$

$$\Delta\vec{\beta} = (\boldsymbol{J}^{\mathrm{T}} \boldsymbol{J})^{-1} \cdot \boldsymbol{J}^{\mathrm{T}} \vec{\epsilon} \tag{13}$$

- the Gauss-Newton method converges fast, but only for "well behaving" functions $f(x, \vec{\beta})$.
- the start position β_0 has to be fairly close to the minimum already.

Levenberg-Marquardt method

- combination of the Gauss-Newton method and the gradient descent method.
- A prose description would be: "use small gradient descent steps towards the minimum when necesary and use larger Gauss-Newton steps when possible".

$$(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda \boldsymbol{I}) \cdot \Delta\vec{\beta} = \boldsymbol{J}^{\mathrm{T}}\vec{\epsilon} \tag{14}$$

- for $\lambda = 0$ the method is similar to the Gauss-Newton method.
- for large λ the method is similar to the gradient descent method, because $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$ will become negligible.
- the choice of λ can be optimized according to the particular problem.
- literature suggests a starting value according to the 2-norm of the matrix $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$.

$$\lambda^0 = ||\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}||_2 \tag{15}$$

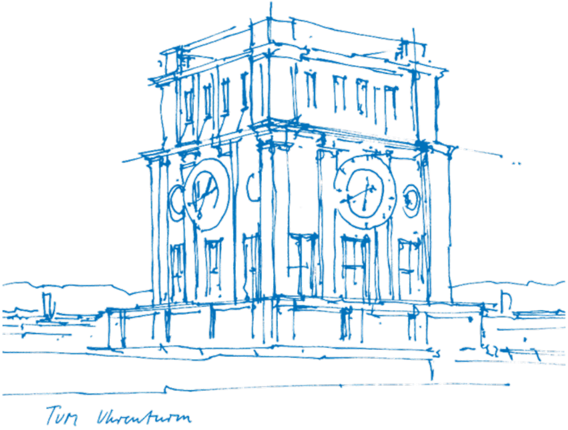
Levenberg-Marquardt method

- Marquardt developed a strategy to update λ every step.
- he introduced a measure for the improvement of an iteration step $\Delta\beta$

$$\rho^k = \frac{\epsilon^2(x, \vec{\beta}^k) - \epsilon^2(x, \vec{\beta}^{k+1})}{\Delta\vec{\beta}^{\mathrm{T}} \cdot (\lambda^k \Delta\vec{\beta} + \boldsymbol{J}^{\mathrm{T}} \vec{\epsilon}(x, \vec{\beta}^k))}, \tag{16}$$

  where the numerator represents the error reduction by the iteration step, and the denominator represents the predicted error reduction by the local linear model.
- 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 $\lambda^{k+1} = \lambda^k$.
- only perform update step $\beta^{k+1} = \beta^k + \Delta\beta$ if $\rho^k > 0$. (if there is improvement at all)

Links



Some links with further information

- http://people.duke.edu/~hpgavin/ce281/lm.pdf
- http://people.compute.dtu.dk/pcha/LSDF/NonlinDataFit.pdf
- https://www.youtube.com/watch?v=lsKIhNkzpbw
- https://www.youtube.com/watch?v=8evmj2L-iCY
- https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.070/ws11_12/Numerik3/Skript/Kapitel1.pdf (german)