JSS30, Summer School, COM5: Machine learning in inverse and ill-posed problems

Larisa Beilina*

Department of Mathematical Sciences, Chalmers University of Technology and Gothenburg University, SE-42196 Gothenburg, Sweden

https://www.jyu.fi/en/research/

Linear and nonlinear Least Squares Problems
Lecture 3

Linear Least Squares Problems

- Suppose that we have a matrix A of the size $m \times n$ and the vector b of the size $m \times 1$. The linear least square problem is to find a vector x of the size $n \times 1$ which will minimize $\|Ax - b\|_2$.
- In the case when $m = n$ and the matrix A is nonsingular we can get solution to this problem as $x = A^{-1}b$.
- When $m > n$ (more equations than unknows) the problem is overdetermined
- When $m < n$ (more unknows than equations) the problem is underdetermined
- Applications: curve fitting, statistical modelling.

Matrix Factorizations that Solve the Linear Least Squares Problem

The linear least squares problem has several explicit solutions that we will discuss:

1. normal equations: the fastest but least accurate; it is adequate when the condition number is small.
2. QR decomposition,

   is the standard one and costs up to twice as much as the first method.
3. SVD, is of most use on an ill-conditioned problem, i.e., when A is not of full rank; it is several times more expensive again.
4. Iterative refinement to improve the solution when the problem is ill-conditioned. Can be adapted to deal efficiently with sparse matrices [Å. Björck. Numerical Methods for Least Squares Problems].

We assume initially for methods 1 and 2 that A has full column rank n.

Linear Least Squares Problems

Further we assume that we will deal with overdetermined problems when we have more equations than unknowns. This means that we will be interested in the solution of linear system of equations

$$Ax = b, \tag{1}$$

where A is of the size $m \times n$ with $m > n$, b is vector of the size m, and x is vector of the size n.
In a general case we are not able to get vector b of the size m as a linear combination of the n columns of the matrix A and n components of the vector x, or there is no solution to (1) in the usual case. We will consider methods which can minimize the residual $r = b - Ax$ as a function on x in principle in any norm, but we will use 2-norm because of the convenience from theoretical (relationships of 2-norm with the inner product and orthogonality, smoothness and strict convexity properties) and computational points of view. Also, because of using 2-norm method is called least squares.

We can write the least squares problem as problem of the minimizing of the squared residuals

$$\|r\|_2^2 = \sum_{i=1}^{m} r_i^2 = \sum_{i=1}^{m} (Ax_i - b)^2. \tag{2}$$

In other words, our goal is to find minimum of this residual using least squares:

$$\min_x \|r\|_2^2 = \min_x \sum_{i=1}^{m} r_i^2 = \min_x \sum_{i=1}^{m} (Ax_i - b)^2. \tag{3}$$

Normal Equations

Our goal is to minimize the residual $\|r(x)\|_2^2 = \|Ax - b\|_2^2$. To find minimum of this functional and derive the *normal equations*, we look for the x where the gradient of $\|Ax - b\|_2^2 = (Ax - b)^T(Ax - b)$ vanishes, or where $\|r'(x)\|_2^2 = 0$. So we want

$$\begin{aligned} 0 &= \lim_{\|e\|\to 0} \frac{(A(x+e)-b)^T(A(x+e)-b) - (Ax-b)^T(Ax-b)}{\|e\|_2} \\ &= \lim_{\|e\|\to 0} \frac{2e^T(A^TAx - A^Tb) + e^TA^TAe}{\|e\|_2} \end{aligned}$$

The second term $\frac{|e^TA^TAe|}{\|e\|_2} \le \frac{\|A\|_2^2\|e\|_2^2}{\|e\|_2} = \|A\|_2^2\|e\|_2$ approaches 0 as e goes to 0, so the factor $A^TAx - A^Tb$ in the first term must also be zero, or $A^TAx = A^Tb$. This is a system of n linear equations in n unknowns, the normal equations.

Normal Equations

Thus, normal equations are

$$A^T Ax = A^T b, \tag{4}$$

which is a symmetric linear system of the $n \times n$ equations.
Using $\|r(x)\|_2^2 = \|Ax - b\|_2^2$ we can compute the Hessian matrix $H = 2A^T A$. If the Hessian matrix $H = 2A^T A$ is positive definite, then x is indeed a minimum. We can show that the matrix $A^T A$ is positive definite if, and only if, the columns of A are linearly independent, or when $r(A) = n$.
If the matrix A has a full rank ($r(A) = n$) then the system (4) is of the size n-by-n and is symmetric positive definite system of normal equations. It has the same solution x as the least squares problem $\min_x \|Ax - b\|_2^2$ of the size m-by-n.

Normal Equations

To solve system (4) one can use Cholesky decomposition

$$A^T A = LL^T \tag{5}$$

with L lower triangular matrix. Then the solution of (4) will be given by the solution of triangular system

$$\begin{aligned} Ly &= A^T b, \\ L^T x &= y. \end{aligned} \tag{6}$$

Normal Equations

However, in practice the method of normal equations can be inaccurate by two reasons.

- The condition number of A^TA is twice more than twice more than the condition number of the original matrix A:

$$cond(A^TA) = cond(A)^2. \tag{7}$$

  Thus, the method of normal equations can give a squared condition number even when the fit to data is good and the residual is small. This makes the computed solution more sensitive. In this sense the method of normal equations is not stable.

- Information can be lost during computation of the product of A^TA.

Normal Equations: loss of information in a given floating-point system

Example

$$A = \begin{pmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{pmatrix} \tag{8}$$

with $0 < \delta < \sqrt{\varepsilon}$ in a given floating-point system. In floating-point arithmetics we can compute $A^T A$:

$$A^T A = \begin{pmatrix} 1 & \delta & 0 \\ 1 & 0 & \delta \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{pmatrix} = \begin{pmatrix} 1+\delta^2 & 1 \\ 1 & 1+\delta^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \tag{9}$$

which is singular matrix in the working precision.

Data fitting

In this example we present the typical application of least squares called data or curve fitting problem. This problem appears in statistical modelling and experimental engineering when data are generated by laboratory or other measurements.
Suppose that we have data points (x_i, y_i), $i = 1, ..., m$, and our goal is to find the vector of parameters c of the size n which will fit best to the data y_i of the model function $f(x_i, c)$, where $f : R^{n+1} \to R$, in the least squares sense:

$$\min_c \sum_{i=1}^{m} (y_i - f(x_i, c))^2. \tag{10}$$

If the function $f(x, c)$ is linear then we can solve the problem (10) using least squares method.

The function $f(x, c)$ is linear if we can write it as a linear combination of the functions $\phi_j(x), j = 1, ..., n$ as:

$$f(x, c) = c_1\phi_1(x) + c_2\phi_2(x) + ... + c_n\phi_n(x). \tag{11}$$

Functions $\phi_j(x), j = 1, ..., n$ are called basis functions.
Let now the matrix A will have entries $a_{ij} = \phi_j(x_i), i = 1, ..., m; j = 1, ..., n$, and vector b will be such that $b_i = y_i, i = 1, ..., m$. Then a linear data fitting problem takes the form of (1) with $x = c$:

$$Ac \approx b \tag{12}$$

Elements of the matrix A are created by basis functions $\phi_j(x), j = 1, ..., n$.
We will consider now different examples of choosing basis functions $\phi_j(x), j = 1, ..., n$.

Problem of the fitting to a polynomial

In the problem of the fitting to a polynomial

$$f(x, c) = \sum_{i=1}^{d} c_i x^{i-1} \tag{13}$$

of degree $d-1$ to data points $(x_i, y_i), i = 1, ..., m$, basis functions $\phi_j(x), j = 1, ..., n$ can be chosen as $\phi_j(x) = x^{j-1}, j = 1, ..., n$. The matrix A constructed by these basis functions in a polynomial fitting problem is a Vandermonde matrix:

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{d-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^{d-1} \end{bmatrix}. \tag{14}$$

Here, $x_i, i = 1, ..., m$ are discrete points on the interval for $x = [x_{left}, x_{right}]$.

Problem of the fitting to a polynomial

- On the next slides we will present several examples of polynomial fitting to data generated by the function $b = sin(\pi x/5) + x/5$.
- All examples are supplied by Matlab programs which are available for download from the link of the book [BKK]:

  `https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`
- See description of all available programs at the above link in the file **Programs.pdf**

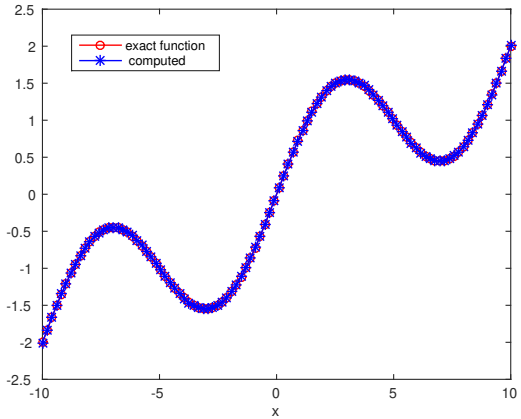
[BKK] L. Beilina, E. Karchevskii, M. Karchevskii, Numerical Linear Algebra: theory and applications, Springer, 2017.

Suppose, that we choose $d = 4$ in (10). Then we can write the polynomial as $f(x, c) = \sum_{i=1}^{4} c_i x^{i-1} = c_1 + c_2 x + c_3 x^2 + c_4 x^3$ and our data fitting problem (12) for this polynomial takes the form

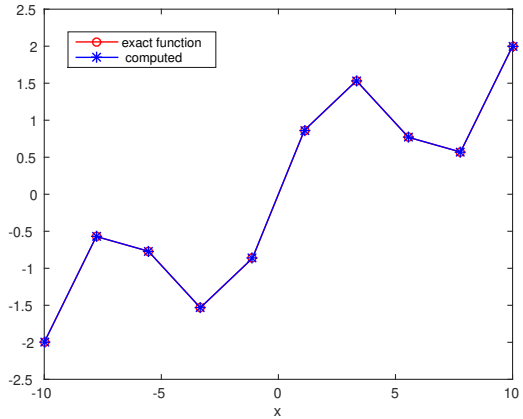
$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & x_m^3 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}. \tag{15}$$

The right hand side of the above system represents measurements or function which we want to fit. Our goal is to find such coefficients $c = \{c_1, c_2, c_3, c_4\}$ which will minimize the residual $r_i = f(x_i, c) - b_i, i = 1..., m$. Since we want minimize squared 2-norm of the residual, or $\|r\|_2^2 = \sum_{i=1}^{m} r_i^2$, then we will solve the linear least squares problem.

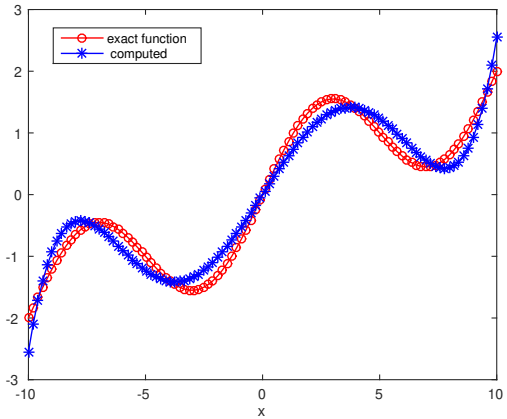
Let us consider an example when the right hand side $b_i, i = 1, ...m$ is taken as a smooth function $b = sin(\pi x/5) + x/5$. Figure on the next slide shows polynomial fitting to the function $b = sin(\pi x/5) + x/5$ for different d in (13) on the interval $x \in [-10, 10]$. Using this figure we observe that with increasing of the degree of the polynomial $d - 1$ we have better fit to the exact function $b = sin(\pi x/5) + x/5$. However, for the degree of the polynomial more than 18 we get erratic fit to the function. This happens because matrix A becomes more and more ill-conditioned with increasing of the degree of the polynomial d. And this is, in turn, because of the linear dependence of the columns in the Vandermonde's matrix A.



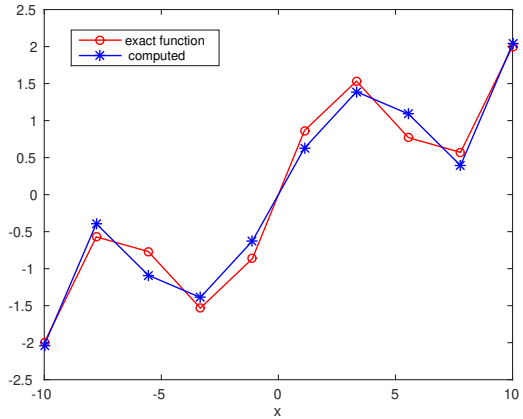
a) d=10



b) d=10



c) d=5



d) d=5

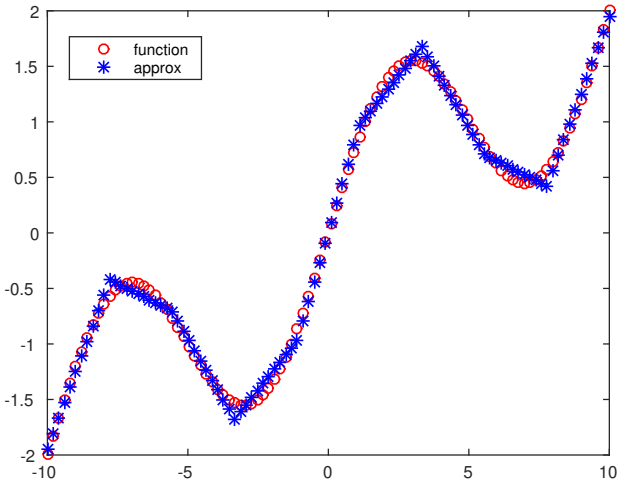
Figure: *Polynomial fitting for different d in (13) to the function* $b = sin(\pi x/5) + x/5$ *on the interval* $x \in [-10, 10]$ *using the method of normal equations. On the left figures: fit to the 100 points* $x_i, i = 1, ..., 100$*; on the right figures: fit to the 10 points* $x_i, i = 1, ..., 10$*. Lines with blue stars represent computed function and with red circles - exact one.*

Approximation using linear splines

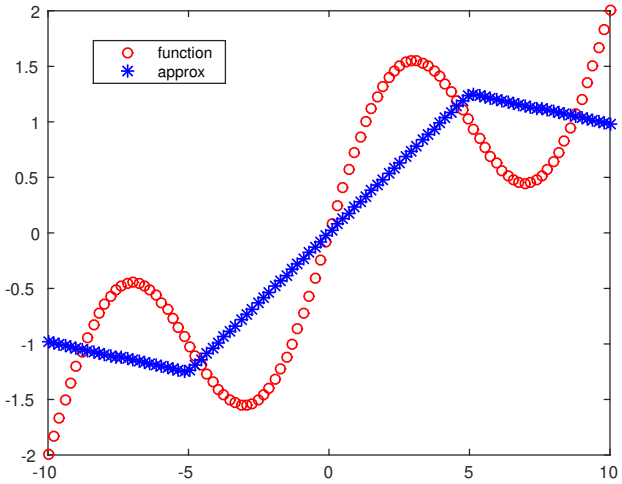
When we want to solve the problem (10) of the approximation to the data vector $y_i, i = 1, ..., m$ with linear splines we use following basis functions $\phi_j(x), j = 1, ..., n$, in (11) which are called also hat functions:

$$\phi_j(x) = \begin{cases} \frac{x - T_{j-1}}{T_j - T_{j-1}}, & T_{j-1} \le x \le T_j, \\ \frac{T_{j+1} - x}{T_{j+1} - T_j}, & T_j \le x \le T_{j+1}. \end{cases} \tag{16}$$

Here, the column j in the matrix A is constructed by the given values of $\phi_j(x)$ at points $T_j, j = 1, .., n$, which are called conjunction points and are chosen by the user. Using (16) we can conclude that the first basis function is $\phi_1(x) = \frac{T_2 - x}{T_2 - T_1}$ and the last one is $\phi_n(x) = \frac{x - T_{n-1}}{T_n - T_{n-1}}$.
Figure on the next slide shows approximation of a function $b = sin(\pi x/5) + x/5$ on the interval $x \in [-10, 10]$ using linear splines with different number n of conjunction points $T_j, j = 1, ..., n$.



a) n=10



b) n=5

Figure: *Polynomial fitting to the function* $b = sin(\pi x/5) + x/5$ *on the interval* $x \in [-10, 10]$ *using linear splines with different number n of conjunction points* $T_j, j = 1, ..., n$ *in (16). Blue stars represent computed function and red circles - exact one.*

Approximation using bellsplines

In the case when we want to solve the problem (10) using bellsplines, the number of bellsplines which can be constructed are $n + 2$, and the function $f(x, c)$ in (10) is written as

$$f(x, c) = c_1\phi_1(x) + c_2\phi_2(x) + ... + c_{n+2}\phi_{n+2}(x). \tag{17}$$

We define

$$\phi_j^0(x) = \begin{cases} 1, & T_j \le x \le T_{j+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Then all other basis functions, or bellsplines, $\phi_j^k(x), j = 1, ..., n+2; k = 1, 2, 3$ are defined as follows:

$$\phi_j^k(x) = (x - T_k)\frac{\phi_j^{k-1}(x)}{T_{j+k} - T_j} + (T_{j+k+1} - x)\frac{\phi_{j+1}^{k-1}(x)}{T_{j+k+1} - T_{j+1}}. \tag{19}$$

Here, the column j in the matrix A is constructed by the given values of $\phi_j(x)$ at conjunction points $T_j, j = 1, .., n$ which are chosen by the user. If in (19) we obtain ratio $0/0$, then we assign $\phi_j^k(x) = 0$. We define additional three points T_{-2}, T_{-1}, T_0 at the left side of the input interval as $T_{-2} = T_{-1} = T_0 = T_1$, and correspondingly three points $T_{n+1}, T_{n+2}, T_{n+3}$ on the right side of the interval as $T_n = T_{n+1} = T_{n+2} = T_{n+3}$. All together we have $n + 6$ conjunction points $T_j, j = 1, ..., n + 6$. Number of bellsplines which can be constructed are $n + 2$.

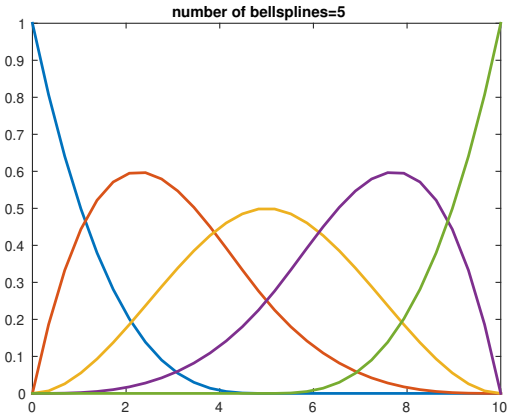
If conjunction points T_j are distributed uniformly, then we can introduce the mesh size $h = T_{k+1} - T_k$ and bellsplines can be written explicitly as

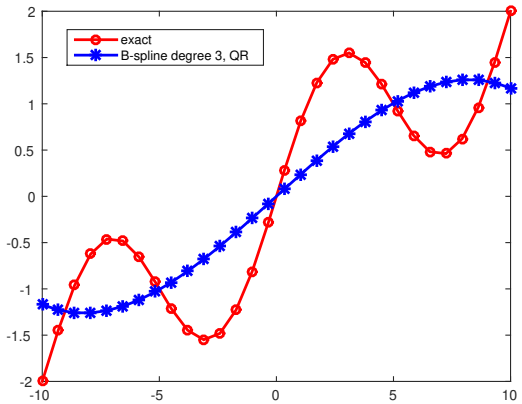
$$\phi_j(x) = \begin{cases} \frac{1}{6}t^3 & \text{if } T_{j-2} \le x \le T_{j-1},\ t = \frac{1}{h}(x - T_{j-2}), \\ \frac{1}{6} + \frac{1}{2}(t + t^2 - t^3) & \text{if } T_{j-1} \le x \le T_j,\ t = \frac{1}{h}(x - T_{j-1}), \\ \frac{1}{6} + \frac{1}{2}(t + t^2 - t^3) & \text{if } T_j \le x \le T_{j+1},\ t = \frac{1}{h}(T_{j+1} - x), \\ \frac{1}{6}t^3 & \text{if } T_{j+1} \le x \le T_{j+2}, t = \frac{1}{h}(T_{j+2} - x). \end{cases} \quad (20)$$

In the case of uniformly distributed bellsplines we place additional points at the left side of the input interval as
$T_0 = T_1 - h, T_{-1} = T_1 - 2h, T_{-2}T_1 - 3h$, and correspondingly on the right side of the interval as $T_{n+1} = T_n + h, T_{n+2} = T_n + 2h, T_{n+3} = T_n + 3h$.
Then the function $f(x, c)$ in (10) will be the following linear combination of $n + 2$ functions $\phi_j(x)$ for indices $j = 0, 1, ..., n + 1$:

$$f(x, c) = c_1\phi_0(x) + c_2\phi_1(x) + ... + c_{n+2}\phi_{n+1}(x). \quad (21)$$

Figure on the next slide shows approximation of a function $b = sin(\pi x/5) + x/5$ on the interval $x \in [-10, 10]$ using bellsplines.





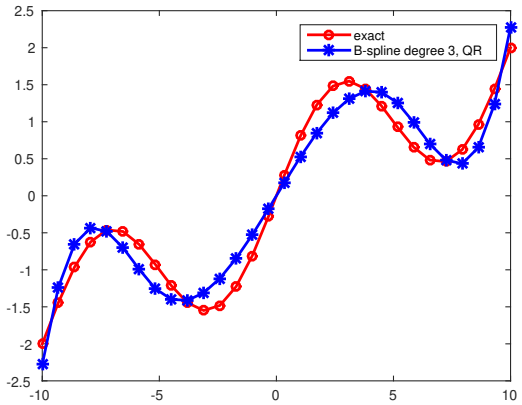


Figure: *Polynomial fitting to the function* $b = sin(\pi x/5) + x/5$ *on the interval* $x \in [-10, 10]$ *with different number of bellsplines. Blue stars represent computed function and red circles - exact one.*

Nonlinear least squares problems

Suppose that for our data points $(x_i, y_i), i = 1, ..., m$ we want to find the vector of parameters $c = (c_1, ..., c_n)$ which will fit best to the data $y_i, i = 1, ..., m$ of the model function $f(x_i, c), i = 1, ..., m$. We consider the case when the model function $f : R^{n+1} \to R$ is nonlinear now. Our goal is to find minimum of the residual $r = y - f(x, c)$ in the least squares sense:

$$\min_c \sum_{i=1}^{m} (y_i - f(x_i, c))^2. \tag{22}$$

To solve problem (22) we can still use the linear least squares method if we can transform the nonlinear function $f(x, c)$ to the linear one. This can be done if the function $f(x, c)$ can be represented in the form $f(x, c) = A \exp^{cx}, A = const$. Then taking logarithm of $f(x, c)$ we get: $\ln f = \ln A + cx$, which is already linear function. Then linear least squares problem after this transformation can be written as

$$\min_c \sum_{i=1}^{m} (\ln y_i - \ln f(x_i, c))^2. \tag{23}$$

Another possibility how to deal with nonlinearity is consider the least squares problem as an optimization problem. Let us define the residual $r : R^n \to R^m$ as

$$r_i(c) = y_i - f(x_i, c), \quad i = 1, ..., m. \tag{24}$$

Our goal is now minimize the function

$$F(c) = \frac{1}{2} r(c)^T r(c) = \frac{1}{2} \|r(c)\|_2^2. \tag{25}$$

To find minimum of (25) we should have

$$\nabla F(c) = \frac{\partial F(c)}{\partial c_i} = 0, \quad i = 1, ..., m. \tag{26}$$

Direct computations show that the gradient vector $\nabla F(c)$ is

$$\nabla F(c) = \frac{dF}{dc} = J^T(c) r(c), \tag{27}$$

where J^T is the transposed Jacobian matrix of the residual $r(c)$.

For a sufficiently smooth function $F(c)$ we can write its Taylor expansion as

$$F(c) = F(c_0) + \nabla F(c_0)(c - c_0) + O(h^2), \tag{28}$$

with $|h| = \|c - c_0\|$. Since our goal is to find minimum of $F(c)$, then at a minimum point c^* we should have $\nabla F(c^*) = 0$. Taking derivative with respect to c from (28) we obtain

$$H(F(c_0))(c - c_0) + \underbrace{\nabla F(c_0)}_{\text{compare with (27)}} = 0, \tag{29}$$

where H denotes the Hessian matrix of the function $F(c_0)$. Using (27) we also can write

$$\nabla F(c_0) = \frac{dF}{dc}(c_0) = J^T(c_0) r(c_0). \tag{30}$$

Using (30) in (29) we obtain

$$H(F(c_0))(c - c_0) + J^T(c_0)r(c_0) = 0, \tag{31}$$

and from this expression we observe that we have obtained a system of linear equations

$$H(F(c_0))(c - c_0) = -J^T(c_0)r(c_0) \tag{32}$$

which can be solved again using linear least squares method. The Hessian matrix $H(F(c_0))$ can be obtained from (30)

$$\nabla F(c_0) = \frac{dF}{dc}(c_0) = J^T(c_0)r(c_0). \tag{33}$$

as

$$H(F(c_0)) = J^T(c_0)J(c_0) + \sum_{i=1}^{m} r_i(c_0)H(r_i), \tag{34}$$

where $H(r_i)$ denotes the Hessian matrix of the residual function $r_i(c)$. These m matrices $H(r_i)$ are inconvenient to compute, but since they are multiplied to the small residuals $r_i(c_0)$, the second term in (34) is often very small at the solution c_0 and this term can be dropped out.

Then the system (31) is transformed to the following linear system

$$J^T(c_0)J(c_0)(c - c_0) = -J^T(c_0)r(c_0), \tag{35}$$

which actually is a system of normal equations for the $m \times n$ linear least squares problem

$$J(c_0)(c - c_0) = -r(c_0). \tag{36}$$

The system (35) determines the Gauss-Newton method for the solution of the least squares problem as an iterative process

$$c^{k+1} = c^k - [J^T(c_k)J(c_k)]^{-1}J^T(c_k)r(c_k), \tag{37}$$

where k is the number of iteration.

An alternative to the Gauss-Newton method is Levenberg-Marquardt method. This method is based on the finding of minimum of the regularized function

$$F(c) = \frac{1}{2} r(c)^T r(c) + \frac{1}{2}\gamma (c - c_0)^T (c - c_0) = \frac{1}{2}\|r(c)\|_2^2 + \frac{1}{2}\gamma \|c - c_0\|_2^2, \quad (38)$$

where c_0 is a good initial guess for c and γ is a small regularization parameter. Then we repeat all steps which we have performed for the obtaining the Gauss-Newton method, see (27)-(34).

Finally, In the Levenberg-Marquardt method the linear system which should be solved at every iteration k is

$$(J^T(c^k)J(c^k) + \gamma_k I)(c^{k+1} - c^k) = -J^T(c^k)r(c^k), \tag{39}$$

and the corresponding linear least squares problem is

$$\begin{bmatrix} J(c^k) \\ \sqrt{\gamma_k} I \end{bmatrix} \cdot (c^{k+1} - c^k) \approx \begin{bmatrix} -r(c^k) \\ 0 \end{bmatrix}. \tag{40}$$

Example 1

Let us consider the nonlinear model equation

$$A\mathrm{e}^{\mathrm{E/T-T_0}} = \mathrm{y}. \tag{41}$$

Our goal is to determine parameters A, E and T_0 in this equation by knowing y and T. We rewrite (41) as a nonlinear least squares problem in the form

$$\min_{A,E,T_0} \sum_{i=1}^{m} (y_i - A\mathrm{e}^{\mathrm{E/T_i-T_0}})^2. \tag{42}$$

We will show how to obtain from the nonlinear problem (42) the linear one. We take logarithm of (41) to get

$$\ln A + \frac{E}{T - T_0} = \ln y. \tag{43}$$

Now multiply both sides of (43) by $T - T_0$ to obtain:

$$\ln A(T - T_0) + E = \ln y(T - T_0). \tag{44}$$

and rewrite the above equation as

$$T \underbrace{\ln A}_{c_2} \underbrace{-T_0 \ln A + E}_{c_3} + \underbrace{T_0}_{c_1} \ln y = T \ln y. \tag{45}$$

Let now define the vector of parameters $c = (c_1, c_2, c_3)$ with $c_1 = T_0, c_2 = \ln A, c_3 = E - T_0 \ln A$. Now the problem (45) can be written as

$$c_1 \ln y + c_2 T + c_3 = T \ln y, \tag{46}$$

which is already a linear problem. Now we can rewrite (46) denoting by $f(c, y, T) = c_1 \ln y + c_2 T + c_3$ as a linear least squares problem in the form

$$\min_c \sum_{i=1}^{m} (T_i \ln y_i - f(c, y_i, T_i))^2. \tag{47}$$

The system of linear equations which is needed to be solved is

$$\begin{bmatrix} \ln y_1 & T_1 & 1 \\ \ln y_2 & T_2 & 1 \\ \vdots & \vdots & \vdots \\ \ln y_m & T_m & 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} T_1 \ln y_1 \\ T_2 \ln y_2 \\ \vdots \\ T_m \ln y_m \end{bmatrix} \tag{48}$$

Example 2

Suppose that the nonlinear model function is given as

$$f(x, c) = A\mathrm{e}^{c_1 x} + B\mathrm{e}^{c_2 x}, \quad A, B = \text{const.} > 0, \tag{49}$$

and our goal is to fit this function using Gauss-Newton method. In other words, we will use iterative formula (36) for iterative update of $c = (c_1, c_2)$. The residual function will be

$$r(c) = y - f(x, c) = y - A\mathrm{e}^{c_1 x} - B\mathrm{e}^{c_2 x}, \tag{50}$$

where $y = y_i, i = 1, ..., m$ are data points.

First, we compute Jacobian matrix $J(c)$, where two columns in this matrix will be given by

$$J(c)_{i,1} = \frac{\partial r_i}{\partial c_1} = -x_i A \mathrm{e}^{\mathrm{c_1 x_i}}, \quad \mathrm{i} = 1, ..., \mathrm{m},$$
$$J(c)_{i,2} = \frac{\partial r_i}{\partial c_2} = -x_i B \mathrm{e}^{\mathrm{c_2 x_i}}, \quad \mathrm{i} = 1, ..., \mathrm{m}. \tag{51}$$

If we will take initial guess for the parameters $c^0 = (c_1^0, c_2^0) = (1, 0)$, then we have to solve the following problem at iteration $k = 1$:

$$J(c^0)(c^1 - c^0) = -r(c^0), \tag{52}$$

and the next update for parameters $c^1 = (c_1^1, c_2^1)$ in the Gauss-Newton method can be computed as

$$c^1 = c^0 - [J^T(c_0)J(c_0)]^{-1} J^T(c_0) r(c_0). \tag{53}$$

Here, $r(c^0)$ and $J(c_0)$ can be computed explicitly as follows:

$$r(c^0) = y_i - f(x_i, c^0) = y_i - (Ae^{1\cdot x_i} + Be^{0\cdot x_i}) = y_i - Ae^{x_i} - B, i = 1, ..., m, \quad (54)$$

and noting that $c^0 = (c_1^0, c_2^0) = (1, 0)$ two columns in the Jacobian matrix $J(c_0)$ will be

$$\begin{aligned} J(c^0)_{i,1} &= -x_i Ae^{1\cdot x_i} = -x_i Ae^{x_i}, \quad i = 1, ..., m, \\ J(c^0)_{i,2} &= -x_i Be^{0\cdot x_i} = -x_i B, \quad i = 1, ..., m. \end{aligned} \quad (55)$$

Substituting (54), (55) into (52) yields following linear system of equations

$$\begin{bmatrix} -x_1 A\mathrm{e}^{x_1} & -x_1 B \\ -x_2 A\mathrm{e}^{x_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A\mathrm{e}^{x_m} & -x_m B \end{bmatrix} \cdot \begin{bmatrix} c_1^1 - c_1^0 \\ c_2^1 - c_2^0 \end{bmatrix} = - \begin{bmatrix} y_1 - A\mathrm{e}^{x_1} - \mathrm{B} \\ y_2 - A\mathrm{e}^{x_2} - \mathrm{B} \\ \vdots \\ y_m - A\mathrm{e}^{x_m} - \mathrm{B} \end{bmatrix} \tag{56}$$

which is solved for $c^1 - c^0$ using method of normal equations as

$$\begin{bmatrix} -x_1 A e^{x_1} & -x_1 B \\ -x_2 A e^{x_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A e^{x_m} & -x_m B \end{bmatrix}^T \cdot \begin{bmatrix} -x_1 A e^{x_1} & -x_1 B \\ -x_2 A e^{x_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A e^{x_m} & -x_m B \end{bmatrix} \cdot \begin{bmatrix} c_1^1 - c_1^0 \\ c_2^1 - c_2^0 \end{bmatrix}$$
$$= - \begin{bmatrix} -x_1 A e^{x_1} & -x_1 B \\ -x_2 A e^{x_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A e^{x_m} & -x_m B \end{bmatrix}^T \cdot \begin{bmatrix} y_1 - A e^{x_1} - B \\ y_2 - A e^{x_2} - B \\ \vdots \\ y_m - A e^{x_m} - B \end{bmatrix} \tag{57}$$

This system can be solved for $c^1 - c^0$, and next values c^1 are obtained by using (53).