1DI2235:A - Metody optymalizacji, lab.
Levenberg–Marquardt algorithm

(opracował: M.T., ostatnia modyfikacja: 17 listopada 2023)

1 Nonlinear least squares problem (NLSP)

Nonlinear least squares method is one of the most important tools for estimation of the parameters of nonlinear models. This method is most often implemented using the Levenberg-Marquardt (LM) algorithm. The Levenberg-Marquardt algorithm is used to solve form optimization problems

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \|f(x)\|^2, \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}^N$

$$f(x)=\begin{bmatrix} f_1(x)\\ \vdots \\ f_N(x)\end{bmatrix} \tag{2}$$

is a given (differentiable) function [1]. Let us denote its solution by $x^\star$.
The problem (1) may be casted as

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(x)^2, \tag{3}$$

where $f_i:\mathbb{R}^n\to\mathbb{R}$, $i=1,\ldots,m$ are the components of f. If $f(x)=Ax-y$, where $A\in\mathbb{R}^{N\times n}$, $y\in\mathbb{R}^N$, then we talk about a "linear" least squares problem, the solution of which is given by the formula

$$x^\star = A^\dagger y. \tag{4}$$

If $\operatorname{rank} A=n$, then there exists exactly one miniizer given by (4), in this case, when $\operatorname{rank} A=n$, that is, when the columns of the matrix A are linearly independent, we have

$$A^\dagger=(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}. \tag{5}$$

On the other hand, if $\operatorname{rank} A<n$ then any vector of the form $A^\dagger y+\zeta$, where ζ is such a vector that $A\zeta=0$, is a solution to the optimization problem (1). If f is of other form than $f(x)=Ax-y$, we speak about a nonlinear least square problem, or just about a nonlinear least-squares problem.

2 Gauss-Newton algorithm

Let us note, that in proximity of a given fixed point $x^{(k)}$ we have

$$f(x)\approx f(x^{(k)})+\mathrm{D}f(x^{(k)})(x-x^{(k)}), \tag{6}$$

where

$$\mathrm{D}f(x^{(k)})=\begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x^{(k)}) & \frac{\partial f_1}{\partial x_2}(x^{(k)}) & \ldots & \frac{\partial f_1}{\partial x_n}(x^{(k)})\\ \frac{\partial f_2}{\partial x_1}(x^{(k)}) & \frac{\partial f_2}{\partial x_2}(x^{(k)}) & \ldots & \frac{\partial f_2}{\partial x_n}(x^{(k)})\\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1}(x^{(k)}) & \frac{\partial f_N}{\partial x_2}(x^{(k)}) & \ldots & \frac{\partial f_N}{\partial x_n}(x^{(k)})\end{bmatrix}, \tag{7}$$

thus

$$\|f(x)\|^2\approx\|f(x^{(k)})+\mathrm{D}f(x^{(k)})(x-x^{(k)})\|^2 \tag{8}$$

Let us denote $v=x-x^{(k)}$, we know, that the minimizer of

$$\|f(x^{(k)})+\mathrm{D}f(x^{(k)})v\|^2. \tag{9}$$

with respect to v is given by

$$v^\star_{(k)}=-[\mathrm{D}f(x^{(k)})]^\dagger f(x^{(k)}), \tag{10}$$

thus the minimizer of

$$\|f(x^{(k)})+\mathrm{D}f(x^{(k)})(x-x^{(k)})\|^2 \tag{11}$$

with respect to x is given by

$$x^\star_{(k)}=x^{(k)}-[\mathrm{D}f(x^{(k)})]^\dagger f(x^{(k)}). \tag{12}$$

If $\operatorname{rank}[\mathrm{D}f(x^{(k)})]=n$, then

$$[\mathrm{D}f(x^{(k)})]^\dagger=\left[[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)})\right]^{-1}[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}, \tag{13}$$

and taking $x^{(k+1)}=x^\star_{(k)}$ we obtain the so called Gauss-Netwon recursive formula

$$x^{(k+1)}=x^{(k)}-\left[[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)})\right]^{-1}[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}f(x^{(k)}). \tag{14}$$

Summarizing, in Gaussa-Netwon method, a point $x^{(k+1)}$ is obtained as a solution of

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \|f(x^{(k)})+\mathrm{D}f(x^{(k)})(x-x^{(k)})\|^2, \tag{15}$$

we can hope, that for properly chosen $x^{(0)}$, we get a sequence

$$x^{(0)},x^{(1)},x^{(2)},\ldots \tag{16}$$

that converges to the solution of the original problem (1), i.e.,

$$x^{(0)},x^{(1)},x^{(2)},\ldots\overset{k\to\infty}{\longrightarrow}x^\star. \tag{17}$$

Unfortunately, the Gauss-Newton formula has some drawbacks. The first one is that if the matrix $[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)})$ in (14) is singular, the formula itself is meaningless and cannot be applied. The second issue is the possibility, that relation

$$\|f(x^{(k+1)})\|^2\leqslant\|f(x^{(k)})\|^2. \tag{18}$$

doses not hold. This may happen if in $x^\star_{(k)}$ the approximation (8) does not work, because the distance from $x^\star_{(k)}$ to $x^{(k)}$ is to large.

3 Levenberg-Marquardt algorithm

The Levenberg-Marquardt (LM) method can be treated as a modification of the Gauss-Netwon method. The modification consists in the fact that in the LM method the point $x^{(k+1)}$ is a solution to the optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \|f(x^{(k)})+\mathrm{D}f(x^{(k)})(x-x^{(k)})\|^2+\lambda^{(k)}\|x-x^{(k)}\|^2, \tag{19}$$

for a given fixed $\lambda^{(k)}>0$. In other words, the modification involves adding a regularization term (also called a penalty term)

$$\lambda^{(k)}\|x-x^{(k)}\|^2, \tag{20}$$

to the objective function in (15), which imposes an additional cost (i.e. an increase in the value of the objective function) proportional to the distance of the solution from the point $x^{(k)}$. We can solve (19) using the fact that

$$\lambda^{(k)}\|x - x^{(k)}\|^2 = \|\sqrt{\lambda^{(k)}}(x - x^{(k)})\|^2 \tag{21}$$

and

$$\begin{aligned}&\|f(x^{(k)}) + \mathrm{D}f(x^{(k)})(x - x^{(k)})\|^2 + \|\sqrt{\lambda^{(k)}}(x - x^{(k)})\|^2 = \\ &\left\|\begin{bmatrix} f(x^{(k)}) + \mathrm{D}f(x^{(k)})(x - x^{(k)}) \\ \sqrt{\lambda^{(k)}}(x - x^{(k)}) \end{bmatrix}\right\|^2.\end{aligned} \tag{22}$$

what implies the equivalence of (19) and

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \left\|\begin{bmatrix} f(x^{(k)}) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathrm{D}f(x^{(k)}) \\ \sqrt{\lambda^{(k)}}I \end{bmatrix}(x - x^{(k)})\right\|^2, \tag{23}$$

from where we get

$$x^{\star}_{(k)} = x^{(k)} - \begin{bmatrix} \mathrm{D}f(x^{(k)}) \\ \sqrt{\lambda^{(k)}}I \end{bmatrix}^{\dagger} \begin{bmatrix} f(x^{(k)}) \\ 0 \end{bmatrix}. \tag{24}$$

Setting $x^{(k+1)} = x^{\star}_{(k)}$ and taking into account that

$$\begin{aligned}&\begin{bmatrix} \mathrm{D}f(x^{(k)}) \\ \sqrt{\lambda^{(k)}}I \end{bmatrix}^{\dagger} \begin{bmatrix} f(x^{(k)}) \\ 0 \end{bmatrix} \\ &= \left[\begin{bmatrix} \mathrm{D}f(x^{(k)}) \\ \sqrt{\lambda^{(k)}}I \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathrm{D}f(x^{(k)}) \\ \sqrt{\lambda^{(k)}}I \end{bmatrix}\right]^{-1} \begin{bmatrix} \mathrm{D}f(x^{(k)}) \\ \sqrt{\lambda^{(k)}}I \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} f(x^{(k)}) \\ 0 \end{bmatrix} \\ &= \left[[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)}) + \lambda^{(k)}I\right]^{-1} [\mathrm{D}f(x^{(k)})]^{\mathrm{T}} f(x^{(k)}).\end{aligned} \tag{25}$$

we obtain a formula

$$\begin{aligned}&x^{(k+1)} = \\ &x^{(k)} - \left[[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)}) + \lambda^{(k)}I\right]^{-1} [\mathrm{D}f(x^{(k)})]^{\mathrm{T}} f(x^{(k)}),\end{aligned} \tag{26}$$

which is called **Levenberg-Marquardt recursive formula**. In practice the formula (26) is applied in such a way that, first we solve (with respect to Δ) the equation

$$\left[[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)}) + \lambda^{(k)}I\right]\Delta = -[\mathrm{D}f(x^{(k)})]^{\mathrm{T}} f(x^{(k)}) \tag{27}$$

and then we compute

$$x^{(k+1)} = x^{(k)} + \Delta. \tag{28}$$

This is because the numerical cost of solving a system of linear equations is much lower than the cost of matrix inversion. Let us also note, that for $\lambda^{(k)} > 0$, matrix $\left[[\mathrm{D}f(x^{(k)})]^{\mathrm{T}}\mathrm{D}f(x^{(k)}) + \lambda^{(k)}I\right]$ is always invertible and thus equation (27) always has unique solution.
Let us say a few words about the stop condition. We have

$$\nabla\|f(x)\|_2^2 = 2[\mathrm{D}f(x)]^{\mathrm{T}} f(x). \tag{29}$$

The derivation of the above formula is provided in the Appendix. Hence, and from the necessary condition for a minimum, it follows that the solution $x^{\star}$ of (1) satisfies the following condition

$$2[\mathrm{D}f(x^{\star})]^{\mathrm{T}} f(x^{\star}) = 0. \tag{30}$$

The stopping criterion of the LM method may have various forms. The simplest one is to assume a certain maximum number of iterations. Another solution is to stop the algorithm when the quantity $\|2[\mathrm{D}f(x)]^{\mathrm{T}} f(x)\|^2$ (this is the so called *optimality condition residual*), decreases below certain value, i.e., when

$$\|2[\mathrm{D}f(x)]^{\mathrm{T}} f(x)\|^2 \approx 0. \tag{31}$$

which means that the optimality necessary condition (30) is approximately satisfied.

It should be clearly emphasized that the LM method is only a heuristic and does not always converge. However, it turns out that in practice it works quite well.

The LM algorithm, written in pseudocode, in its basic version, is given in the box below (notation $\leftarrow$ stands for value assignment).

Levenberg-Marquardt Algorithm (A)

given: differentiable function $f : \mathbb{R}^n \to \mathbb{R}^N$, starting point $x^{(0)}$, initial value of trust parameter $\lambda^{(0)} > 0$, maximal number of iterations $k_{\max}$

$k \leftarrow 0$
while $k < k_{\max}$
  $J \leftarrow \mathrm{D}f(x^{(k)})$
  $x^{(k+1)} \leftarrow x^{(k)} - \left[J^{\mathrm{T}}J + \lambda^{(k)}I\right]^{-1} J^{\mathrm{T}} f(x^{(k)})$
  if $\|f(x^{(k+1)})\|^2 < \|f(x^{(k)})\|^2$
    $\lambda^{(k+1)} \leftarrow 0.8\lambda^{(k)}$
  else
    $\lambda^{(k+1)} \leftarrow 2\lambda^{(k)}$
    $x^{(k+1)} \leftarrow x^{(k)}$
  end
  $k \leftarrow k + 1$
end

Written in pseudocode, the LM algorithm, with a more refined stopping criterion, is given in the box below.

Levenberg-Marquardt Algorithm (B)

given: differentiable function $f : \mathbb{R}^n \to \mathbb{R}^N$, starting point $x^{(0)}$, initial value of trust parameter $\lambda^{(0)} > 0$, solution accuracy $\epsilon > 0$

$k \leftarrow 0$
$J \leftarrow \mathrm{D}f(x^{(k)})$
while $\left\| 2J^{\mathrm{T}} f(x^{(k)}) \right\|^2 > \epsilon$
  $J \leftarrow \mathrm{D}f(x^{(k)})$
  $x^{(k+1)} \leftarrow x^{(k)} - \left[J^{\mathrm{T}} J + \lambda^{(k)} I \right]^{-1} J^{\mathrm{T}} f(x^{(k)})$
  if $\|f(x^{(k+1)})\|^2 < \|f(x^{(k)})\|^2$
    $\lambda^{(k+1)} \leftarrow 0.8\lambda^{(k)}$
  else
    $\lambda^{(k+1)} \leftarrow 2\lambda^{(k)}$
    $x^{(k+1)} \leftarrow x^{(k)}$
  end
  $k \leftarrow k + 1$
end

4 Parameter estimation using the LM method

Let us assume we have N measurements $y_1, \ldots, y_m$ of a certain process, modelled as $\varphi(x, u)$ for corresponding input signal values $u_1, \ldots, u_m$. We want to determine the vector of parameters x, to minimize the expression

$$\sum_{i=1}^{m} \left[\varphi(x, u_i) - y_i \right]^2. \tag{32}$$

Let us introduce notation

$$f_i(x) = \varphi(x, u_i) - y_i, \tag{33}$$

now, the minimizing of (32) boils down to solving optimization problem (1). We have

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_N(x) \end{bmatrix} = \begin{bmatrix} \varphi(x, u_1) - y_1 \\ \vdots \\ \varphi(x, u_N) - y_N \end{bmatrix} \tag{34}$$

and

$$\mathrm{D}f(x) = \begin{bmatrix} \frac{\partial \varphi(x,u_1)}{\partial x_1} & \frac{\partial \varphi(x,u_1)}{\partial x_2} & \cdots & \frac{\partial \varphi(x,u_1)}{\partial x_n} \\ \frac{\partial \varphi(x,u_2)}{\partial x_1} & \frac{\partial \varphi(x,u_2)}{\partial x_2} & \cdots & \frac{\partial \varphi(x,u_2)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varphi(x,u_N)}{\partial x_1} & \frac{\partial \varphi(x,u_N)}{\partial x_2} & \cdots & \frac{\partial \varphi(x,u_N)}{\partial x_n} \end{bmatrix} \tag{35}$$

Przykład 1. Assume we have N measurements $y_1, \ldots, y_m$ of the process

$$\varphi(x, t) = A \sin(\omega t + \phi), \tag{36}$$

for corresponding time instants $t_1, \ldots, t_m$, where

$$x = \begin{bmatrix} A \\ \omega \\ \phi \end{bmatrix} \tag{37}$$

We want to find the values of the parameters A, ω and ϕ to obtain the best fit of the model $\varphi(x, t)$ to measurement data. In other words we want to minimize

$$\sum_{i=1}^{m} \left[A \sin(\omega t_i + \phi) - y_i \right]^2, \tag{38}$$

with respect to A, ω and ϕ, this is equivalent to minimizing

$$\sum_{i=1}^{m} f_i(x)^2, \tag{39}$$

where

$$f_i(x) = x_1 \sin(x_2 t_i + x_3) - y_i. \tag{40}$$

In other words we want to solve

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|f(x)\|^2, \tag{41}$$

with

$$f(x) = \begin{bmatrix} x_1 \sin(x_2 t_1 + x_3) - y_1 \\ x_1 \sin(x_2 t_2 + x_3) - y_2 \\ \vdots \\ x_1 \sin(x_2 t_N + x_3) - y_N \end{bmatrix}. \tag{42}$$

We have

$$\varphi(x, t_k) = x_1 \sin(x_2 t_k + x_3), \quad k = 1, \ldots, N \tag{43}$$

and therefor

$$\frac{\partial \varphi(x, t_k)}{\partial x_1} = \sin(x_2 t_k + x_3), \tag{44a}$$

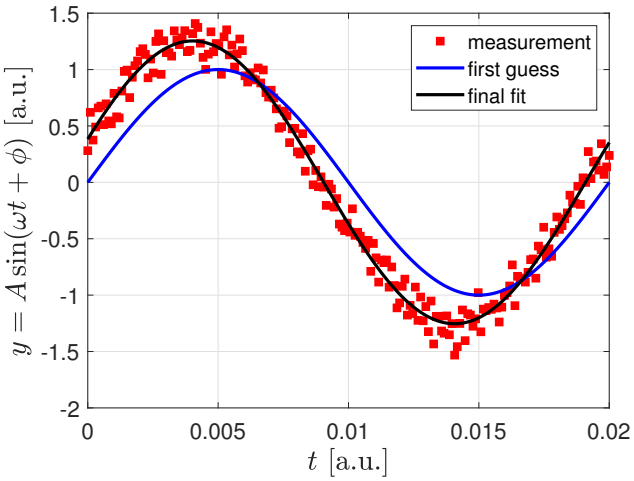
$$\frac{\partial \varphi(x, t_k)}{\partial x_2} = x_1 t_k \cos(x_2 t_k + x_3), \tag{44b}$$

$$\frac{\partial \varphi(x, t_k)}{\partial x_3} = x_1 \cos(x_2 t_k + x_3), \tag{44c}$$

for $k = 1, \ldots, N$, thus according to (35) we get

$$\mathrm{D}f(x) = \begin{bmatrix} \sin(x_2 t_1 + x_3) & x_1 t_1 \cos(x_2 t_1 + x_3) & x_1 \cos(x_2 t_1 + x_3) \\ \sin(x_2 t_2 + x_3) & x_1 t_2 \cos(x_2 t_2 + x_3) & x_1 \cos(x_2 t_2 + x_3) \\ \vdots & \vdots & \vdots \\ \sin(x_2 t_N + x_3) & x_1 t_N \cos(x_2 t_N + x_3) & x_1 \cos(x_2 t_N + x_3) \end{bmatrix}. \tag{45}$$

Let us consider exemplary measurements, depicted in Fig. 1 (red squares). By inspection, we can estimate the crude values of parameters, approximately we have $A = 1$, $\phi = 0$ and $\omega = 100\pi$, where in the latter case we used $\omega = T/2\pi$, where, based on the Fig. 1 we cane estimate $T \approx 0.02$. The exemplary code is given in Listing 1.

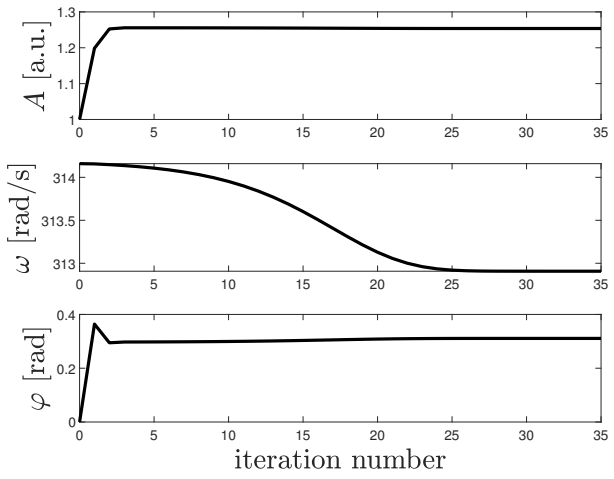


Rysunek 1: Sinusoid fitting ($y = A\sin(\omega t + \phi)$) [Exercise 1].

Listing 1. close all

```
clear all
clc
nfontslatex = 18;
nfonts = 14;

load("LM01Data")
k_max = 35;
h = @(x,t) x(1)*sin(x(2)*t+x(3));
f = @(x) x(1)*sin(x(2)*t+x(3))-y;

x0 = [1.0, 100*pi, 0.0]';  % initial values of model parameters
n = length(x0);

figure
plot01 = plot(t,y,"s","MarkerEdgeColor","r","MarkerFaceColor","r");
hold on
tPlot = linspace(t(1),t(end),1e+3)';
plot02 = plot(tPlot,h(x0,tPlot),"b","LineWidth",2);
grid on

J = @(x) [sin(x(2)*t+x(3)) x(1)*t.*cos(x(2)*t+x(3)) x(1)*cos(x(2)*t+x(3))];

X = zeros(n,k_max+1);
X(:,1) = x0;
L = 1.0;
for k = 1:k_max
    x = X(:,k);
    xNew = x - ((J(x)'*J(x)) + L*eye(n))\J(x)'*f(x);
    if ( norm(f(xNew)) < norm(f(x0)) )
        X(:,k+1) = xNew;
        L = 0.8*L;
    else
        X(:,k+1) = x;
        L = 2*L;
    end
end
xOptimal = X(:,end);

plot03 = plot(tPlot,h(xOptimal,tPlot),"k","LineWidth",2);
legend([plot01,plot02,plot03],"measurement", "first guess", "final fit")
set(gca,"FontSize",nfonts);
ylabel("$y = A\sin(\omega t + \phi)$ [a.u.]","Interpreter","Latex","FontSize",nfontslatex)
xlabel("$t$ [s]","Interpreter","Latex","FontSize",nfontslatex)

figure
set(gca,"FontSize",nfonts);
subplot(3,1,1)
plot((0:k_max),X(1,:),"k","LineWidth",2)
ylabel("$A$ [a.u.]","Interpreter","Latex","FontSize",nfontslatex)

subplot(3,1,2)
plot((0:k_max),X(2,:),"k","LineWidth",2)
ylabel("$\omega$ [rad/s]","Interpreter","Latex","FontSize",nfontslatex)

subplot(3,1,3)
plot((0:k_max),X(3,:),"k","LineWidth",2)
ylabel("$\varphi$ [rad]","Interpreter","Latex","FontSize",nfontslatex)
xlabel("iteration number","Interpreter","Latex","FontSize",nfontslatex)
```

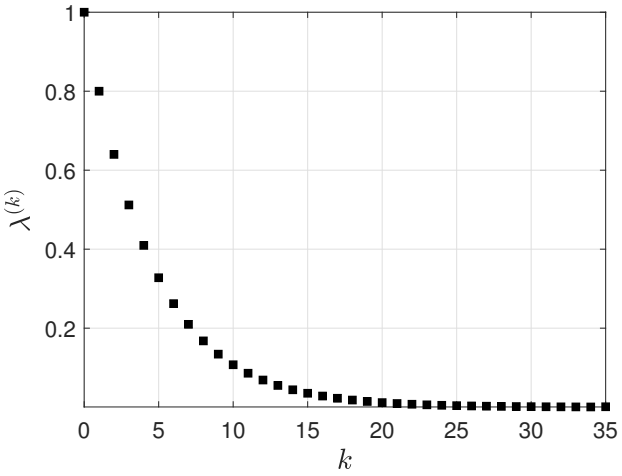
The evolution of parameters' values in consecutive iterations of LM method is depicted in Fig. 2



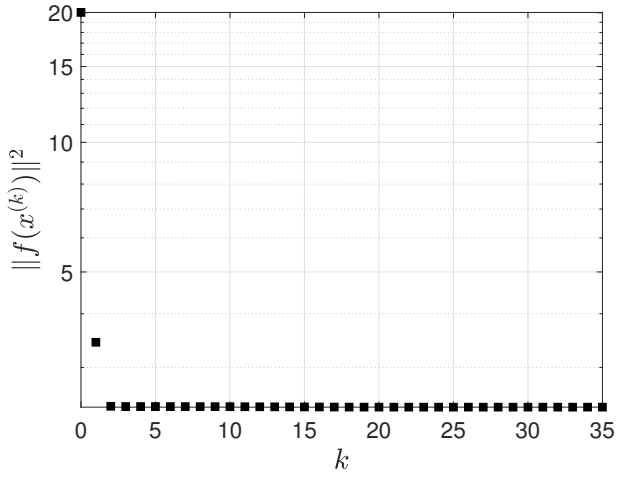
Rysunek 2: The evolution of parameters' values in consecutive iterations of LM method [Exercise 1].

5 Exercises

Zadanie 1. Using the LM algorithm, fit the model $y = A\sin(\omega t + \phi)$ into measurement data from file `LM01Data.mat`. Assume $\lambda^{(0)} = 1.0$. Write an appropriate script in the Matlab environment and generate the graphs shown in Fig. 1–4. (This exercise actually amounts to repeating Example 1).

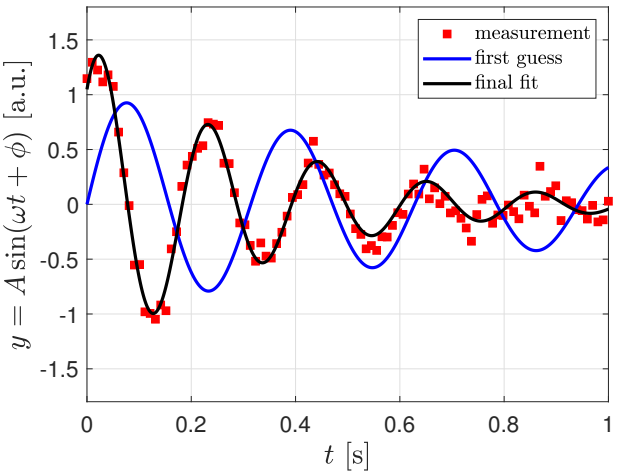


Rysunek 3: Trust parameter $\lambda^{(k)}$ change in consecutive iterations of LM algorithm [Exercise 1].

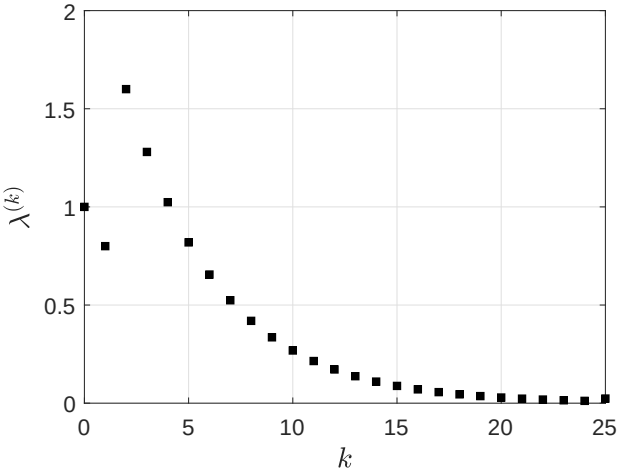


Rysunek 4: Objective function value in consecutive iterations of LM algorithm [Exercise 1].

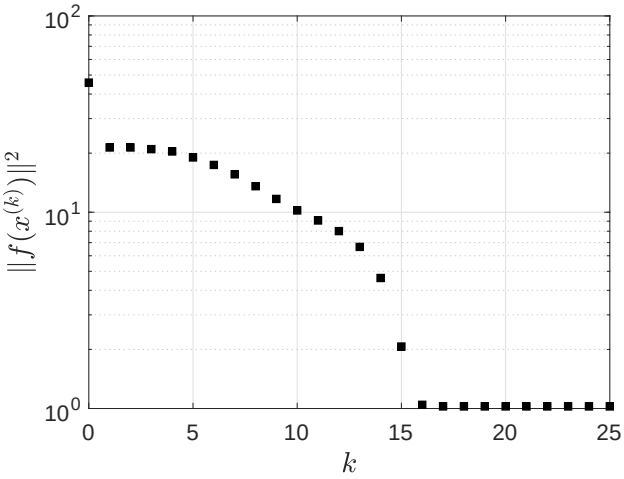
Zadanie 2. Using LM algorithm fit the model $y = Ae^{-at}\sin(\omega t + \phi)$ into measurement data from file `LM04Data.mat`. Write an appropriate script in the Matlab environment and generate the graphs shown in Fig. 5–7.



Rysunek 5: Damped sinusoid fitting $y = Ae^{-at}\sin(\omega t + \phi)$ [Exercise 2].



Rysunek 6: Trust parameter $\lambda^{(k)}$ change in consecutive iterations of LM algorithm [Exercise 2].



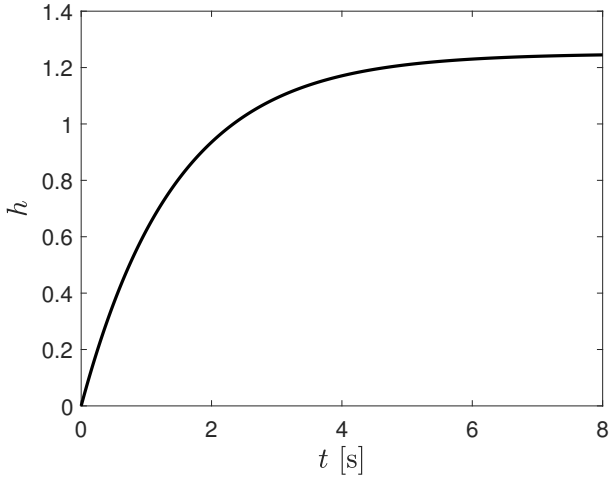
Rysunek 7: Objective function value in consecutive iterations of LM algorithm [Exercise 2].

Zadanie 3. The inertial system is described by the following transfer function

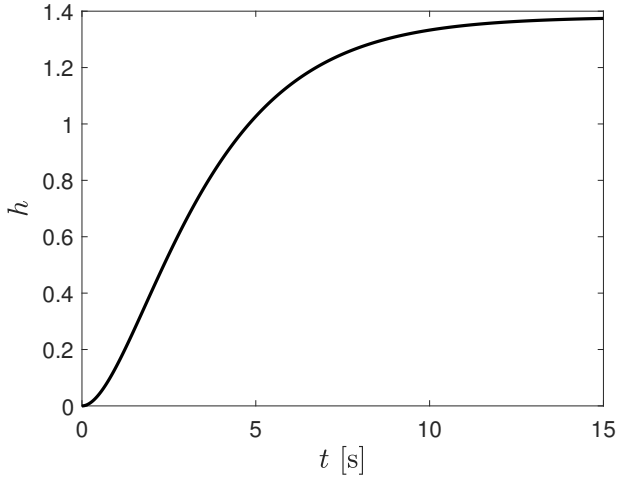
$$G(s) = \frac{k}{Ts+1}. \tag{46}$$

Its step response is

$$h(t) = k\left(1 - \mathrm{e}^{-\frac{t}{T}}\right). \tag{47}$$



Rysunek 8: Step response $h(t)$ of inertial system. [Exercise 3].



Rysunek 9: Step response $h(t)$ of double inertial system. [Exercise 4].

Let us assume that we have samples of step response $y_1, \ldots, y_N$ for corresponding time instants $t_1, \ldots, t_N$ and we want to estimate gain k time constant T by fitting (47) with LM method. In other words, using LM method we want to minimize, with respect to k and T, the expression

$$\sum_{i=1}^{N}[h(t_i) - y_i]^2 = \sum_{i=1}^{N}\left[k\left(1 - \mathrm{e}^{-\frac{t_i}{T}}\right) - y_i\right]^2. \tag{48}$$

This is equivalent to (1) for

$$f_i = k\left(1 - \mathrm{e}^{-\frac{t_i}{T}}\right) - y_i, \quad i = 1, \ldots, N. \tag{49}$$

We have

$$\frac{\partial f_i}{\partial k} = 1 - \mathrm{e}^{-\frac{t_i}{T}}, \tag{50a}$$

$$\frac{\partial f_i}{\partial T} = -\frac{k}{T^2}\mathrm{e}^{-\frac{t_i}{T}}, \tag{50b}$$

for $i = 1, \ldots, N$.

Using data from file `inertialData.mat` and LM method, estimate k and T, and make appropriate plots.

(Answer: $k = 1.45$, $T = 1.25$.)

Zadanie 4. Double-inertial system is described by transfer function

$$G(s) = \frac{k}{(T_1 s + 1)(T_2 s + 1)}. \tag{51}$$

Its step response is

$$h(t) = k\left[1 - \frac{1}{T_1 - T_2}\left(T_1\mathrm{e}^{-\frac{t}{T_1}} - T_2\mathrm{e}^{-\frac{t}{T_2}}\right)\right]. \tag{52}$$

Let us assume that we have samples of step response $y_1, \ldots, y_N$ for corresponding time instants $t_1, \ldots, t_N$ and we want to estimate gain k and time constants T_1 and T_2 by fitting (52) with LM method. In other words, using LM method we want to minimize, with respect to k, T_1 and T_1 the expression

$$\sum_{i=1}^{N}[h(t_i) - y_i]^2. \tag{53}$$

that is

$$\sum_{i=1}^{N}\left\{k\left[1 - \frac{1}{T_1 - T_2}\left(T_1\mathrm{e}^{-\frac{t_i}{T_1}} - T_2\mathrm{e}^{-\frac{t_i}{T_2}}\right)\right] - y_i\right\}^2. \tag{54}$$

It is equivalent to (1) for

$$f_i = k\left[1 - \frac{1}{T_1 - T_2}\left(T_1\mathrm{e}^{-\frac{t_i}{T_1}} - T_2\mathrm{e}^{-\frac{t_i}{T_2}}\right)\right] - y_i \tag{55}$$

and $i = 1, \ldots, N$.
We have

$$\frac{\partial f_i}{\partial k} = 1 - \frac{1}{T_1 - T_2}\left(T_1\mathrm{e}^{-\frac{t_i}{T_1}} - T_2\mathrm{e}^{-\frac{t_i}{T_2}}\right), \tag{56a}$$

$$\frac{\partial f_i}{\partial T_1} = \frac{k}{T_2 - T_1}\left[\frac{t_i}{T_1}\mathrm{e}^{-\frac{t_i}{T_1}} + \frac{T_2}{T_1 - T_2}\left(\mathrm{e}^{-\frac{t_i}{T_2}} - \mathrm{e}^{-\frac{t_i}{T_1}}\right)\right], \tag{56b}$$

$$\frac{\partial f_i}{\partial T_2} = \frac{k}{T_1 - T_2}\left[\frac{t_i}{T_2}\mathrm{e}^{-\frac{t_i}{T_2}} + \frac{T_1}{T_2 - T_1}\left(\mathrm{e}^{-\frac{t_i}{T_1}} - \mathrm{e}^{-\frac{t_i}{T_2}}\right)\right], \tag{56c}$$

for $i = 1, \ldots, N$.
Using data from`duobleInertialData.mat` and LM method estimate parameters k and T, make appropriate plots.
(Answer: $k = 1.38$, $T_1 = 1.45$, $T_2 = 2.32$.)

Zadanie 5. In [2] the following transfer function is considered

$$G(s) = -M\omega_\mathrm{s}\frac{Ks + 2\Delta}{s^4 + Ks^3 + 2\Sigma s^2 + K\Sigma s + \Delta^2}, \tag{57}$$

where

$$M = \frac{2\sqrt{V_\mathrm{in}^2 - V_\mathrm{out}^2}}{\pi|1 - \omega_\mathrm{s}^2 LC|}, \tag{58a}$$

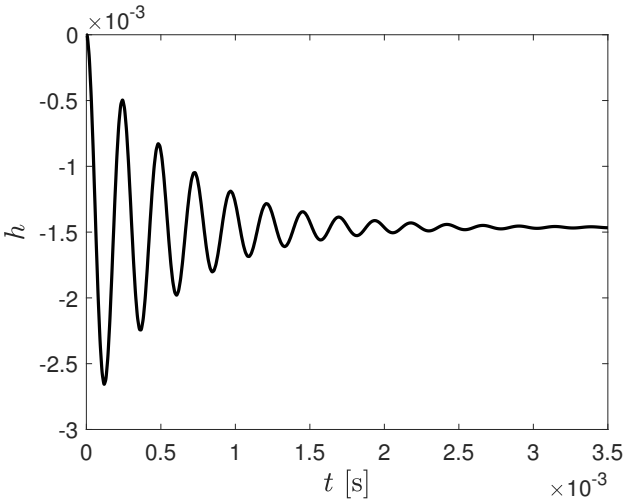
$$K = \frac{2\omega_0^2 V_\mathrm{out}}{\pi M\omega_\mathrm{s}}, \tag{58b}$$

$$\omega_0^2 = \frac{1}{LC}, \tag{58c}$$

$$\Sigma = \omega_\mathrm{s}^2 + \omega_0^2, \tag{58d}$$

$$\Delta = \omega_\mathrm{s}^2 - \omega_0^2 \tag{58e}$$

and $L = 197 \times 10^{-6}$ H, $C = 100 \times 10^{-9}$ F, $V_{\text{in}} = 14.0$ V, $V_{\text{out}} = 1.95$ V, $\omega_{\text{s}} = 40 \times 2\pi \times 10^3$ rad/s. Figure 10 depicts the step response of this transfer function.



Rysunek 10: Step response of transfer function (57) [Exercise 5].

This response resembles the step response of an oscillating element. Note that the oscillation term is of the second order, while the transfer function (57) is of the fourth order. It is generally easier to design a controller for a second-order term than for a fourth-order term, so we would like to find a second-order term whose step response is as close as possible to the step response of the first-order term (57). We know that the step response of the oscillatory term

$$G_{\text{osc}}(s) = \frac{k}{1 + 2\xi T s + T^2 s^2}, \tag{59}$$

is given by

$$h_{\text{osc}}(t) = k\left[1 - \mathrm{e}^{-\gamma t}\left(\cos(\beta t) + \frac{\gamma}{\beta}\sin(\beta t)\right)\right], \tag{60}$$

where

$$\gamma = \frac{\xi}{T}, \qquad \beta = \frac{\sqrt{1-\xi^2}}{T}. \tag{61}$$

From (61) it follows that

$$\xi = \frac{\gamma}{\sqrt{\beta^2 + \gamma^2}}, \qquad T = \frac{1}{\sqrt{\beta^2 + \gamma^2}}, \tag{62}$$

in other words

$$T = \frac{1}{\sqrt{\beta^2 + \gamma^2}}, \qquad \xi = \gamma T. \tag{63}$$

Having values of the step response $h(t)$ of the transfer function (57) and denoting, $y_i = h(t_i)$, $i = 1, \ldots, N$, we can compute the values of parameters γ and β minimizing expression

$$\sum_{i=1}^{N}\left\{k\left[1 - \mathrm{e}^{-\gamma t_i}\left(\cos(\beta t_i) + \frac{\gamma}{\beta}\sin(\beta t_i)\right)\right] - y_i\right\}^2, \tag{64}$$

which, taking into account that

$$f_i(k, \gamma, \beta) = k\left[1 - \mathrm{e}^{-\gamma t_i}\left(\cos(\beta t_i) + \frac{\gamma}{\beta}\sin(\beta t_i)\right)\right] - y_i, \tag{65}$$

for $i = 1, \ldots, N$ and $k = x_1$, $\gamma = x_2$, $\beta = x_3$ boils down to (1). Let us note that

$$\frac{\partial f_i}{\partial k} = 1 - \mathrm{e}^{-\gamma t_i}\left(\cos(\beta t_i) + \frac{\gamma}{\beta}\sin(\beta t_i)\right), \tag{66a}$$

$$\frac{\partial f_i}{\partial \gamma} = k\mathrm{e}^{-\gamma t_i}\left(t_i\cos(\beta t_i) - \frac{1 - t_i\gamma}{\beta}\sin(\beta t_i)\right), \tag{66b}$$

$$\frac{\partial f_i}{\partial \beta} = k\mathrm{e}^{-\gamma t_i}\left[\left(t_i + \frac{\gamma}{\beta^2}\right)\sin(\beta t_i) - \frac{\gamma}{\beta}t_i\cos(\beta t_i)\right], \tag{66c}$$

for $i = 1, \ldots, N$. Thus, taking into account the aforementioned notation $k = x_1$, $\gamma = x_2$, $\beta = x_3$ we have

$$\frac{\partial f_i}{\partial x_1} = 1 - \mathrm{e}^{-x_2 t_i}\left(\cos(x_3 t_i) + \frac{x_2}{x_3}\sin(x_3 t_i)\right), \tag{67a}$$

$$\frac{\partial f_i}{\partial x_2} = x_1\mathrm{e}^{-x_2 t_i}\left(t_i\cos(x_3 t_i) - \frac{1 - t_i x_2}{x_3}\sin(x_3 t_i)\right), \tag{67b}$$

$$\frac{\partial f_i}{\partial x_3} = x_1\mathrm{e}^{-x_2 t_i}\left[\left(t_i + \frac{x_2}{x_3^2}\right)\sin(x_3 t_i) - \frac{x_2}{x_3}t_i\cos(x_3 t_i)\right] \tag{67c}$$

for $i = 1, \ldots, N$.

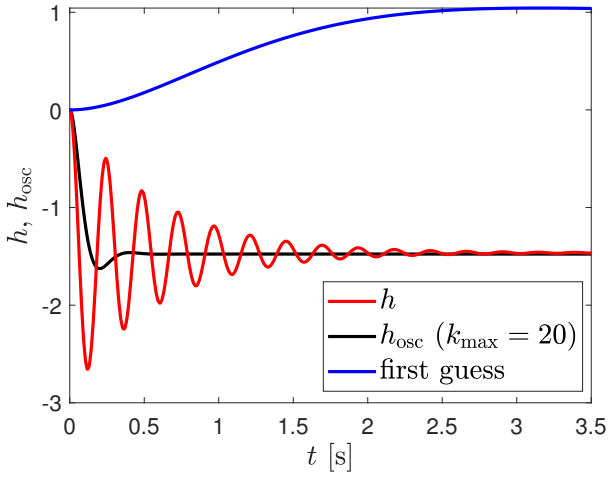
Using the LM method, determine the second-order equivalent in the form of an oscillation term for the transfer function (57). The step response of (57) can be computed (simulated) or taken from the file `reductionData.mat`. Make corresponding plots. As a starting point take $x^{(0)} = [1, 1, 1]^{\text{T}}$. More information on transfer function and step response may be found in, for instance, [3] and [4].

Uwaga: Form Fig. 10 it follows that $y_1, y_2, \ldots, y_N$ and $t_1, t_2, \ldots, t_N$ are relatively small, of order 10^{-3}. It is convenient to carry out the computations for rescaled values $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_N$ i $\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_N$ and later regain "true" values rescaling them back. If $\tilde{y}_i = 10^3 \times y_i$ and $\tilde{t}_i = 10^3 \times t_i$ for $i = 1, \ldots, N$, then finally one has to multiply parameters k, γ and β by 10^{-3}. Finally, identified values of the parameters, after rescaling, are

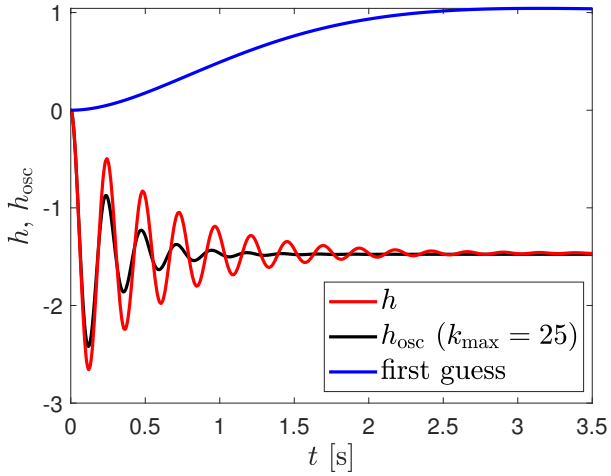
$$k = -1.4639 \times 10^{-3}, \tag{68a}$$

$$\gamma = 1.7303 \times 10^{-3}, \tag{68b}$$

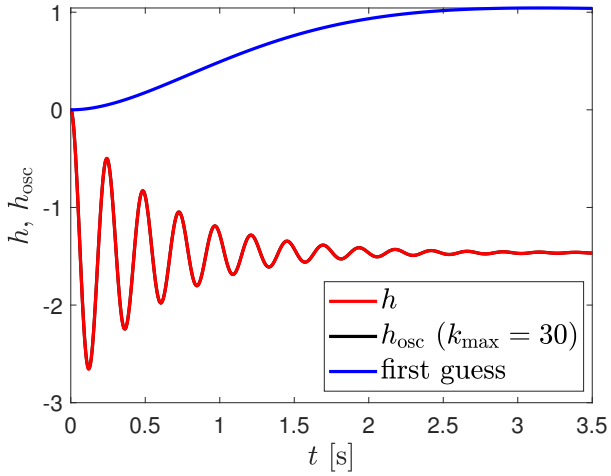
$$\beta = -25.9807 \times 10^{-3}. \tag{68c}$$



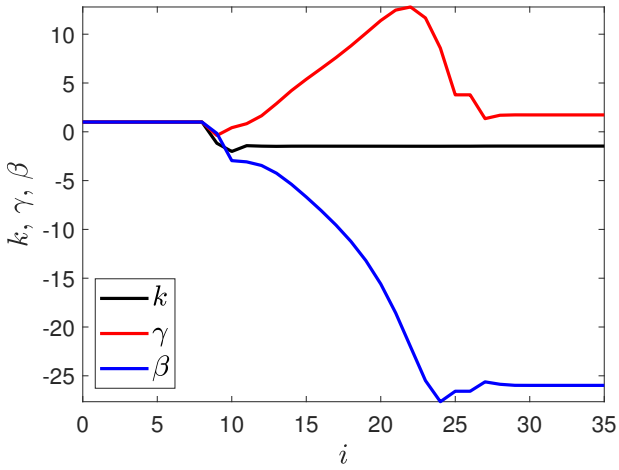
Rysunek 11: Step response $h(t)$ plot (red) of the transfer function (57) and step response $h_{\text{osc}}(t)$ of oscillatory term described by transfer function (59) for initial values of the parameters (blue) and after 20 iterations (black). [Exercise 5].



Rysunek 12: Step response $h(t)$ plot (red) of the transfer function (57) and step response $h_{\text{osc}}(t)$ of oscillatory term described by transfer function (59) for initial values of the parameters (blue) and after 20 iterations (black). [Exercise 5].



Rysunek 13: Step response $h(t)$ plot (red) of the transfer function (57) and step response $h_{\text{osc}}(t)$ of oscillatory term described by transfer function (59) for initial values of the parameters (blue) and after 20 iterations (black). [Exercise 5].



Rysunek 14: Change of parameters k, γ and β in consecutive iterations of LM algorithm. [Exercise 5].

Zadanie 6. Repeat all previous exercises using Matlab function `lsqnonlin` or `lsqcurvefit`. When using `lsqcurvefit` take into account the possibility of providing explicitely Jacobian matrix as an input argument. For more information see `https://www.mathworks.com/help/optim/ug/lsqnonlin.html#buuhch7-problem`
and
`https://www.mathworks.com/help/optim/ug/lsqcurvefit.html?searchHighlight=lsqcurvefit&s_tid=doc_srchtitle`, in particular pay attention how the Jacobian matrix can be passed to the function `lsqcurvefit`.

Appendix

Derivation of formula (29) Formula (29) may be obtained as follows.

$$\begin{aligned}
\nabla\|f(x)\|_2^2 &= \nabla\left(\sum_{i=1}^{N} f_i(x)^2\right)\\
&= \sum_{i=1}^{N} \nabla\left(f_i(x)^2\right)\\
&= \sum_{i=1}^{N} \begin{bmatrix} \frac{\partial f_i^2}{\partial x_1}(x)\\ \frac{\partial f_i^2}{\partial x_2}(x)\\ \vdots \\ \frac{\partial f_i^2}{\partial x_n}(x)\end{bmatrix}\\
&= \sum_{i=1}^{N} \begin{bmatrix} 2f_i(x)\frac{\partial f_i}{\partial x_1}(x)\\ 2f_i(x)\frac{\partial f_i}{\partial x_2}(x)\\ \vdots \\ 2f_i(x)\frac{\partial f_i}{\partial x_n}(x)\end{bmatrix}\\
&= \sum_{i=1}^{N} 2f_i(x) \begin{bmatrix} \frac{\partial f_i}{\partial x_1}(x)\\ \frac{\partial f_i}{\partial x_2}(x)\\ \vdots \\ \frac{\partial f_i}{\partial x_n}(x)\end{bmatrix}\\
&= \sum_{i=1}^{N} 2f_i(x)\nabla f_i(x)\\
&= 2\begin{bmatrix}\nabla f_1(x) & \dots & \nabla f_1(x)\end{bmatrix}\begin{bmatrix} f_1(x)\\ \vdots \\ f_N(x)\end{bmatrix}\\
&= 2[\mathrm{D}f(x)]^{\mathrm{T}} f(x). \qquad (69)
\end{aligned}$$

Alternatively, yone can use a general formula

$$\nabla(f^{\mathrm{T}}g) = (\mathrm{D}f)^{\mathrm{T}}g + (\mathrm{D}g)^{\mathrm{T}}f, \qquad (70)$$

and substituting $g = f$. Formula (70) may be derived as follows. For differentaible, and in general case vector-valued, functions $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^m$,

$$f(x) = \begin{bmatrix} f_1(x)\\ \vdots \\ f_m(x)\end{bmatrix}, \qquad g(x) = \begin{bmatrix} g_1(x)\\ \vdots \\ g_m(x)\end{bmatrix}, \qquad (71)$$

we have,

$$\mathrm{D}f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n}\end{bmatrix}, \qquad (72)$$

hance it follows that

$$\mathrm{D}f = \begin{bmatrix} \mathrm{D}f_1\\ \mathrm{D}f_2\\ \vdots \\ \mathrm{D}f_m\end{bmatrix} \qquad (73)$$

and

$$(\mathrm{D}f)^{\mathrm{T}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla f_1 & \nabla f_2 & \ldots & \nabla f_m \end{bmatrix}, \tag{74}$$

and similarly for g.
We have

$$\begin{aligned} \mathrm{D}(f^{\mathrm{T}}g) &= \mathrm{D}\left(\sum_{i=1}^{m} f_i g_i\right) \\ &= \sum_{i=1}^{m} \mathrm{D}(f_i g_i) \\ &= \sum_{i=1}^{m} \begin{bmatrix} \frac{\partial (f_i g_i)}{\partial x_1} & \cdots & \frac{\partial (f_i g_i)}{\partial x_n} \end{bmatrix} \\ &= \sum_{i=1}^{m} \begin{bmatrix} g_i \frac{\partial f_i}{\partial x_1} + f_i \frac{\partial g_i}{\partial x_1} & \cdots & g_i \frac{\partial f_i}{\partial x_n} + f_i \frac{\partial g_i}{\partial x_n} \end{bmatrix} \\ &= \sum_{i=1}^{m} \begin{bmatrix} g_i \frac{\partial f_i}{\partial x_1} & \cdots & g_i \frac{\partial f_i}{\partial x_n} \end{bmatrix} + \sum_{i=1}^{m} \begin{bmatrix} f_i \frac{\partial g_i}{\partial x_1} & \cdots & f_i \frac{\partial g_i}{\partial x_n} \end{bmatrix} \\ &= \sum_{i=1}^{m} g_i \begin{bmatrix} \frac{\partial f_i}{\partial x_1} & \cdots & \frac{\partial f_i}{\partial x_n} \end{bmatrix} + \sum_{i=1}^{m} f_i \begin{bmatrix} \frac{\partial g_i}{\partial x_1} & \cdots & \frac{\partial g_i}{\partial x_n} \end{bmatrix} \\ &= \sum_{i=1}^{m} g_i \mathrm{D}f_i + \sum_{i=1}^{m} f_i \mathrm{D}g_i \\ &= g^{\mathrm{T}}\mathrm{D}f + f^{\mathrm{T}}\mathrm{D}g. \end{aligned} \tag{75}$$

Taking into account that

$$\nabla(f^{\mathrm{T}}g) = [\mathrm{D}(f^{\mathrm{T}}g)]^{\mathrm{T}} \tag{76}$$

we finally obtain

$$\nabla(f^{\mathrm{T}}g) = (\mathrm{D}f)^{\mathrm{T}}g + (\mathrm{D}g)^{\mathrm{T}}f. \tag{77}$$

Literatura

[1] S. Boyd and L. Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press, 2018.

[2] S.R Sanders, J.M Noworolski, X.Z Liu, and G.C Verghese. Generalized averaging method for power conversion circuits. *IEEE Transactions on Power Electronics*, 6(2):251–259, 1991.

[3] Gene F Franklin, J Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Pearson Education, global edition edition, 2014.

[4] Richard C Dorf. *Modern control systems*. Pearson Education, Harlow, 12th ed., new international ed. edition, 2014.