Università della Svizzera italiana
Institute of Computing CI

Numerical Computing 2023

Lecture 7: Advanced Topics on Least Squares

Dr. Edoardo Vecchi

Preliminary Notes

- This set of slides contains the material concerning lecture 7, which expands the material introduced in lecture 6 by presenting additional, more advanced topics on least squares.
- Specifically, we will address the following points:
  - Focusing our attention on the least squares solution of inconsistent systems of equations, we introduce QR factorization—through different algorithms—as an alternative method to the normal equations. This allows us to achieve higher accuracy in all those cases where the system is ill-conditioned.
  - We then focus our attention on nonlinear least squares and models with nonlinear parameters, by presenting some tools to deal with this kind of problems.
- Feel free to get in touch with me or the TAs in case you have any doubt or concern.

Overview

Accuracy of the Least-Squares Solution α^*

When computing the least-squares solution $\boldsymbol{\alpha}^*$ to the problem $A\boldsymbol{\alpha} = \mathbf{b}$, we always have to solve the system of normal equations $(A^\intercal A)\boldsymbol{\alpha}^* = A^\intercal \mathbf{b}$.

Problem: depending on the condition number of matrix $A^\intercal A$, we can obtain a solution which is not accurate in double precision arithmetic.

An example where this problem arises is reported in the following Exercise 1.

Exercise 1: Accuracy of Least Squares

You are given the following data set on the measurements of a response variable:

x_i	2.0	2.2	2.4	2.6	2.8	3.0	3.2	3.4	3.6	3.8	4.0
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}

where the values y_i are obtained in the following way:

(a) $y_i = 1 + x_i + x_i^2 + x_i^3$,

(b) $y_i = 1 + x_i + x_i^2 + x_i^3 + x_i^4 + x_i^5 + x_i^6 + x_i^7$.

Fit the data to a 3rd-order and 7th-order polynomial, by computing—in both cases—the least-squares solution $\boldsymbol{\alpha}^*$ through the solution of the system of normal equations. Compare your results with the exact solution you expected to obtain.

Exercise 1: Solution I

Preliminary note on the exact solutions

We can notice that, in both cases (a) and (b), we already know the exact solutions of the problem. Indeed, the values of y have been generated from the 11 equidistantly spaced values of $\mathbf{x}$ in the interval $[2, 4]$, by using a 3rd and 7th degree polynomial respectively.

For case (a) the exact solution will be:

$$\boldsymbol{\alpha}_{\text{exact}} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix},$$

while, for case (b), we will have that:

$$\boldsymbol{\alpha}_{\text{exact}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Exercise 1: Solution II

Case (a) – Step 1: Write the system in the form $A\boldsymbol{\alpha} = \mathbf{b}$

We start with case (a) and we try to fit our data to a 3rd-order polynomial:

$$y = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \alpha_4 x^3.$$

Since we have 11 points, this will result in a system with $m = 11$ equations and $n = 4$ unknowns, given by the coefficients α_i of the polynomial. The equations will be:

$$\begin{aligned} &\alpha_1 + 2\alpha_2 + 4\alpha_3 + 8\alpha_4 = 15 \\ &\vdots \\ &\alpha_1 + 4\alpha_2 + 16\alpha_3 + 64\alpha_4 = 85 \end{aligned}$$

with the matrix form of the system given by:

Exercise 1: Solution III

Case (a) – Step 1: (continued)

$$\underbrace{\begin{bmatrix} 1.0000 & 2.0000 & 4.0000 & 8.0000 \\ 1.0000 & 2.2000 & 4.8400 & 10.6480 \\ 1.0000 & 2.4000 & 5.7600 & 13.8240 \\ 1.0000 & 2.6000 & 6.7600 & 17.5760 \\ 1.0000 & 2.8000 & 7.8400 & 21.9520 \\ 1.0000 & 3.0000 & 9.0000 & 27.0000 \\ 1.0000 & 3.2000 & 10.2400 & 32.7680 \\ 1.0000 & 3.4000 & 11.5600 & 39.3040 \\ 1.0000 & 3.6000 & 12.9600 & 46.6560 \\ 1.0000 & 3.8000 & 14.4400 & 54.8720 \\ 1.0000 & 4.0000 & 16.0000 & 64.0000 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}}_{\boldsymbol{\alpha}} = \underbrace{\begin{bmatrix} 15.0000 \\ 18.6880 \\ 22.9840 \\ 27.9360 \\ 33.5920 \\ 40.0000 \\ 47.2080 \\ 55.2640 \\ 64.2160 \\ 74.1120 \\ 85.0000 \end{bmatrix}}_{\mathbf{b}},$$

Exercise 1: Solution IV

Case (a) – Step 2: Find the least-squares solution

In order to find the least-squares solution, we compute the matrix of normal equations:

$$A^\intercal A = \begin{bmatrix} 11 & 33 & 103.4 & 336.6 \\ 33 & 103.4 & 336.6 & 1131.7328 \\ 103.4 & 336.6 & 1131.7328 & 3907.992 \\ 336.6 & 1131.7328 & 3907.992 & 13790.55392 \end{bmatrix},$$

and the right-hand side, which is given by:

$$A^\intercal \mathbf{b} = \begin{bmatrix} 484 \\ 1604.7328 \\ 5479.7248 \\ 19166.87872 \end{bmatrix}.$$

Exercise 1: Solution V

Case (a) – Step 2: (continued)

The system of normal equations is then given by:

$$\underbrace{\begin{bmatrix} 11 & 33 & 103.4 & 336.6 \\ 33 & 103.4 & 336.6 & 1131.7328 \\ 103.4 & 336.6 & 1131.7328 & 3907.992 \\ 336.6 & 1131.7328 & 3907.992 & 13790.55392 \end{bmatrix}}_{A^\intercal A} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}}_{\boldsymbol{\alpha}} = \underbrace{\begin{bmatrix} 484 \\ 1604.7328 \\ 5479.7248 \\ 19166.87872 \end{bmatrix}}_{A^\intercal \mathbf{b}}.$$

We are interested in computing the solution of this system, and we can simply use the backslash operator in Matlab as follows: `alpha_star = (A'*A)\(A'*b)`.

Exercise 1: Solution VI

Case (a) – Step 2: (continued)

The solution of the least-squares problem $\boldsymbol{\alpha}^*$ is given by:

$$\boldsymbol{\alpha}^* = \begin{bmatrix} 0.999999999516122 \\ root - mean - square 1.000000000508656 \\ 0.999999999826834 \\ 1.000000000019140 \end{bmatrix}$$

and norm of the difference between the exact and the least-squares solution is given by:

$$\|\boldsymbol{\alpha}_{\text{exact}} - \boldsymbol{\alpha}^*\|_2 = 7.2334 \cdot 10^{-10}.$$

Exercise 1: Solution VII

Case (a) – Step 3: Comparison of the exact and least-squares solutions

As we can notice, the norm of the difference between the exact solution $\boldsymbol{\alpha}_{\text{exact}}$ and the least-squares solution $\boldsymbol{\alpha}^*$ is really small. The norm of the residual is given by:

$$\|\mathbf{r}\|_2 = \|\mathbf{b} - A\boldsymbol{\alpha}^*\| = 1.2087 \cdot 10^{-11}$$

while the root-mean-square error (RMSE) is:

$$\text{RMSE} = \sqrt{\frac{\text{SE}}{m}} = \frac{1.2087 \cdot 10^{-11}}{\sqrt{11}} = 3.6442 \cdot 10^{-12}.$$

The accuracy of the solution we computed is definitely satisfying and—thus—using least squares provides a good approximation of the original coefficients. Can we say the same for case (b)?

Exercise 1: Solution VIII

Case (b) – Step 1: Write the system in the form $A\alpha = \mathbf{b}$

We now consider case (b) and try to fit our data to a 7th-order polynomial:

$$y = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \alpha_4 x^3 + \alpha_5 x^4 + \alpha_6 x^5 + \alpha_7 x^6 + \alpha_8 x^7.$$

Since we have 11 points, this will result in a system with $m = 11$ equations and $n = 8$ unknowns, given by the coefficients α_i of the polynomial. The equations will be:

$$\begin{aligned}
&\alpha_1 + 2\alpha_2 + 4\alpha_3 + 8\alpha_4 + 16\alpha_5 + 32\alpha_6 + 64\alpha_7 + 128\alpha_8 = 255 \\
&\vdots \\
&\alpha_1 + 4\alpha_2 + 16\alpha_3 + 64\alpha_4 + 256\alpha_5 + 1024\alpha_6 + 4096\alpha_7 + 16384\alpha_8 = 21845
\end{aligned}$$

with the matrix form of the system given by:

Exercise 1: Solution IX

Case (b) – Step 1: (continued)

$$\underbrace{\begin{bmatrix} 1.00 & 2.00 & 4.00 & 8.00 & 16.00 & 32.00 & 64.00 & 128.00 \\ 1.00 & 2.20 & 4.84 & 10.65 & 23.43 & 51.54 & 113.38 & 249.44 \\ 1.00 & 2.40 & 5.76 & 13.82 & 33.18 & 79.63 & 191.10 & 458.65 \\ 1.00 & 2.60 & 6.76 & 17.58 & 45.70 & 118.81 & 308.92 & 803.18 \\ 1.00 & 2.80 & 7.84 & 21.95 & 61.47 & 172.10 & 481.89 & 1349.29 \\ 1.00 & 3.00 & 9.00 & 27.00 & 81.00 & 243.00 & 729.00 & 2187.00 \\ 1.00 & 3.20 & 10.24 & 32.77 & 104.86 & 335.54 & 1073.74 & 3435.97 \\ 1.00 & 3.40 & 11.56 & 39.30 & 133.63 & 454.35 & 1544.80 & 5252.34 \\ 1.00 & 3.60 & 12.96 & 46.66 & 167.96 & 604.66 & 2176.78 & 7836.42 \\ 1.00 & 3.80 & 14.44 & 54.87 & 208.51 & 792.35 & 3010.94 & 11441.56 \\ 1.00 & 4.00 & 16.00 & 64.00 & 256.00 & 1024.00 & 4096.00 & 16384.00 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix}}_{\boldsymbol{\alpha}} = \underbrace{\begin{bmatrix} 255.00 \\ 456.47 \\ 785.54 \\ 1304.54 \\ 2098.34 \\ 3280.00 \\ 4997.33 \\ 7440.39 \\ 10850.04 \\ 15527.47 \\ 21845.00 \end{bmatrix}}_{\mathbf{b}},$$

Exercise 1: Solution X

Case (b) – Step 2: Find the least-squares solution

In order to find the least-squares solution, we compute the matrix of normal equations:

$$A^\intercal A = \begin{bmatrix} 1.10 \cdot 10^1 & 3.30 \cdot 10^1 & 1.03 \cdot 10^2 & 3.37 \cdot 10^2 & 1.13 \cdot 10^3 & 3.91 \cdot 10^3 & 1.38 \cdot 10^4 & 4.95 \cdot 10^4 \\ 3.30 \cdot 10^1 & 1.03 \cdot 10^2 & 3.37 \cdot 10^2 & 1.13 \cdot 10^3 & 3.91 \cdot 10^3 & 1.38 \cdot 10^4 & 4.95 \cdot 10^4 & 1.80 \cdot 10^5 \\ 1.03 \cdot 10^2 & 3.37 \cdot 10^2 & 1.13 \cdot 10^3 & 3.91 \cdot 10^3 & 1.38 \cdot 10^4 & 4.95 \cdot 10^4 & 1.80 \cdot 10^5 & 6.65 \cdot 10^5 \\ 3.37 \cdot 10^2 & 1.13 \cdot 10^3 & 3.91 \cdot 10^3 & 1.38 \cdot 10^4 & 4.95 \cdot 10^4 & 1.80 \cdot 10^5 & 6.65 \cdot 10^5 & 2.47 \cdot 10^6 \\ 1.13 \cdot 10^3 & 3.91 \cdot 10^3 & 1.38 \cdot 10^4 & 4.95 \cdot 10^4 & 1.80 \cdot 10^5 & 6.65 \cdot 10^5 & 2.47 \cdot 10^6 & 9.28 \cdot 10^6 \\ 3.91 \cdot 10^3 & 1.38 \cdot 10^4 & 4.95 \cdot 10^4 & 1.80 \cdot 10^5 & 6.65 \cdot 10^5 & 2.47 \cdot 10^6 & 9.28 \cdot 10^6 & 3.50 \cdot 10^7 \\ 1.38 \cdot 10^4 & 4.95 \cdot 10^4 & 1.80 \cdot 10^5 & 6.65 \cdot 10^5 & 2.47 \cdot 10^6 & 9.28 \cdot 10^6 & 3.50 \cdot 10^7 & 1.33 \cdot 10^8 \\ 4.95 \cdot 10^4 & 1.80 \cdot 10^5 & 6.65 \cdot 10^5 & 2.47 \cdot 10^6 & 9.28 \cdot 10^6 & 3.50 \cdot 10^7 & 1.33 \cdot 10^8 & 5.08 \cdot 10^8 \end{bmatrix},$$

and the right-hand side, which is given by:

Exercise 1: Solution XI

Case (b) – Step 2: (continued)

$$A^{\intercal}\mathbf{b} = \begin{bmatrix} 6.88 \cdot 10^4 \\ 2.49 \cdot 10^5 \\ 9.14 \cdot 10^5 \\ 3.39 \cdot 10^6 \\ 1.27 \cdot 10^7 \\ 4.77 \cdot 10^7 \\ 1.81 \cdot 10^8 \\ 6.88 \cdot 10^8 \end{bmatrix}$$

Once again, we are interested in computing the solution of the system $(A^{\intercal}A)\boldsymbol{\alpha} = A^{\intercal}\mathbf{b}$ by using Matlab backslash operator as follows: `alpha_star = (A'*A)\(A'*b)`.

Exercise 1: Solution XII

Case (b) – Step 2: (continued)

The solution of the least squares problem $\boldsymbol{\alpha}^*$ is given by:

$$\boldsymbol{\alpha}^* = \begin{bmatrix} 5.114092487784963 \\ -9.164107427261779 \\ 11.653362333469046 \\ -5.141646622972061 \\ 3.103585436454946 \\ 0.571844285088259 \\ 1.047961023611489 \\ 0.997718453778757 \end{bmatrix}$$

and the norm of the difference between the exact and the least-squares solution is given by:

$$\|\boldsymbol{\alpha}_{\text{exact}} - \boldsymbol{\alpha}^*\|_2 = 16.615.$$

Exercise 1: Solution XIII

Case (b) – Step 3: Comparison of exact and least-squares solution

As we can notice, the norm of the difference between the exact solution $\boldsymbol{\alpha}_{\text{exact}}$ and the least-squares solution $\boldsymbol{\alpha}^*$ has increased significantly. The norm of the residual is now:

$$\|\mathbf{r}\|_2 = \|\mathbf{b} - A\boldsymbol{\alpha}^*\| = 7.4406 \cdot 10^4$$

while the root-mean-square error (RMSE) is:

$$\text{RMSE} = \sqrt{\frac{\text{SE}}{m}} = \frac{7.4406 \cdot 10^4}{\sqrt{11}} = 2.2434 \cdot 10^4.$$

The solution we found is definitely not acceptable. **How can we fix this problem?**

Least Squares by QR Factorization

We can improve the accuracy of the solution found in Exercise 1 by using QR factorization inside the least-squares algorithm. In other words, instead of solving the ill-conditioned system of equations $(A^\intercal A)\boldsymbol{\alpha} = A^\intercal \mathbf{b}$, we start by finding the QR factorization of $A = QR$ and we define the following quantities:

- $\overline{R}$, the upper $n \times n$ sub-matrix of R
- $\overline{\mathbf{c}}$, the upper n entries of vector $\mathbf{c} = Q^\intercal \mathbf{b}$.

The least-squares solution is then computed by solving the system:

$$\overline{R}\alpha^* = \overline{\mathbf{c}}$$

which, in Matlab, corresponds to the command: `alpha_star = R_bar\c_bar`.

Exercise 1: Alternative Solution of Part (b) I

Case (b) – Step 2: Compute the solution α^* through the full QR factorization of A

The QR factorization can be computed, in Matlab, with the `qr()` function. After defining the quantities $\overline{R}$ and $\overline{\mathbf{c}}$, and solving the corresponding systems of equations, we obtain the following least-squares solution based on QR factorization:

$$\boldsymbol{\alpha}^* = \begin{bmatrix} 1.000000020865378 \\ 0.999999941058347 \\ 1.000000069137147 \\ 0.999999956154886 \\ 1.000000016289759 \\ 0.999999996445136 \\ 1.000000000422810 \\ 0.999999999978819 \end{bmatrix}.$$

Exercise 1: Alternative Solution of Part (b) II

Case (b) – Step 3: Comparison of exact and least-squares solution

The norm of the difference between the exact and the new least-squares solution is given by:

$$\|\boldsymbol{\alpha}_{\text{exact}} - \boldsymbol{\alpha}^*\|_2 = 1.0436 \cdot 10^{-07}$$

As we can notice, the norm of the difference between the exact solution $\boldsymbol{\alpha}_{\text{exact}}$ and the least-squares solution $\boldsymbol{\alpha}^*$ has decreased significantly. The norm of the residual is now:

$$\|\mathbf{r}\|_2 = \|\mathbf{b} - A\boldsymbol{\alpha}^*\| = 3.5680 \cdot 10^{-12}$$

while the root-mean-square error (RMSE) is:

$$\text{RMSE} = \sqrt{\frac{\text{SE}}{m}} = \frac{7.4406 \cdot 10^4}{\sqrt{11}} = 1.0758 \cdot 10^{-12}.$$

Definitely better than the results obtained in the previous case!

QR Factorization

As we have observed in the previous example, QR factorization is extremely useful to compute the least-squares solution when the matrix of normal equations $A^\intercal A$ is strongly ill-conditioned.

While in Exercise 1 we observed the application of QR factorization through the Matlab function `qr()`, we now want to take a closer look at how it is computed.

In order to do so, we will start by recalling how to compute Gram-Schmidt orthogonalization. After some examples, we will also introduce Householder reflectors, as an alternative and more stable method to compute the matrices Q and R.

Gram-Schmidt Orthogonalization I

Gram-Schmidt orthogonalization is a technique that—starting from a set of linearly independent vectors—tries to find a set of orthonormal (i.e., perpendicular and of length 1) vectors spanning the same space of the original set of vectors.

In other words, starting from a set of n vectors $\mathbf{v_1}, \ldots, \mathbf{v_n} \in \mathbb{R}^m$, it finds a new set of n vectors $\mathbf{q_1}, \ldots, \mathbf{q_n} \in \mathbb{R}^m$, such that all vectors are pairwise-orthogonal and of length 1.

It is worth recalling the fact that, in order for the vectors $\mathbf{v_1}, \ldots, \mathbf{v_n} \in \mathbb{R}^m$ to be linearly independent, the condition $n \leq m$ must hold. If we had $n > m$, the $n - m$ additional vectors could all be written as a linear combination of the first n vectors.

Gram-Schmidt Orthogonalization II

Let us now see how Gram-Schmidt orthogonalization works, both in theory and through an example exercise. We can start by setting $\mathbf{u_1}$ equal to the first vector of our set, namely $\mathbf{v_1}$, and then divide it by its norm, in order to make it of length 1:

$$\mathbf{u_1} = \mathbf{v_1} \longrightarrow \mathbf{q_1} = \frac{\mathbf{u_1}}{\|\mathbf{u_1}\|_2}. \tag{1}$$

After finding the first vector of the new set $\mathbf{q_1}$, we want to find a second unit vector $\mathbf{q_2}$ which is orthogonal to the previous one. We know that two vectors $\mathbf{x}$ and $\mathbf{y}$ are orthogonal if:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle = 0.$$

How do we then compute the vector $\mathbf{q_2}$?

Gram-Schmidt Orthogonalization III

In order to find $\mathbf{q_2}$, we subtract from $\mathbf{v_2}$ its projection in the direction of $\mathbf{q_1}$ and, then, normalize the obtained result:

$$\mathbf{u_2} = \mathbf{v_2} - \mathbf{q_1}(\mathbf{q_1}^\intercal \mathbf{v_2}) \longrightarrow \mathbf{q_2} = \frac{\mathbf{u_2}}{\|\mathbf{u_2}\|_2}. \tag{2}$$

We can verify that the orthogonality condition between the two vectors is satisfied, since:

$$\mathbf{q_1}^\intercal \mathbf{u_2} = \mathbf{q_1}^\intercal(\mathbf{v_2} - \mathbf{q_1}(\mathbf{q_1}^\intercal \mathbf{v_2})) = \mathbf{q_1}^\intercal \mathbf{v_2} - \underbrace{\mathbf{q_1}^\intercal \mathbf{q_1}}_{=1}(\mathbf{q_1}^\intercal \mathbf{v_2}) = 0,$$

and—thus—also $\mathbf{q_1}$ and $\mathbf{q_2}$ (which is just $\mathbf{u_2}$ normalized) are pairwise orthogonal.

Gram-Schmidt Orthogonalization IV

We can proceed in the same way till we reach the nth vector, which is simply given by:

$$\mathbf{u_n} = \mathbf{v_n} - \mathbf{q_1}(\mathbf{q_1}^\intercal \mathbf{v_n}) - \mathbf{q_2}(\mathbf{q_2}^\intercal \mathbf{v_n}) - \cdots - \mathbf{q_{n-1}}(\mathbf{q_{n-1}}^\intercal \mathbf{v_n}) \longrightarrow \mathbf{q_n} = \frac{\mathbf{u_n}}{\|\mathbf{u_n}\|_2}.$$

By proceeding analogously at what we did before for the product $\mathbf{q_1}^\intercal \mathbf{u_2}$, we can show that $\mathbf{q_n}$ is perpendicular to all the previously defined vectors $\mathbf{q_1}, \ldots, \mathbf{q_{n-1}}$.

$\star$ **Additional exercise:** Starting from the three following linearly independent vectors:

$$\mathbf{v_1} = \begin{bmatrix} 3 \\ 4 \\ 2 \end{bmatrix}, \quad \mathbf{v_2} = \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}, \quad \mathbf{v_3} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix},$$

use Gram-Schmidt orthogonalization to find the orthonormal vectors $\mathbf{q_1}$, $\mathbf{q_2}$ and $\mathbf{q_3}$.

The Reduced QR Factorization

Let us now express the results obtained in matrix form. Eq. (1) and (2) can be rewritten as:

$$\mathbf{v_1} = \mathbf{q_1}\|\mathbf{u_1}\|_2,$$
$$\mathbf{v_2} = (\mathbf{q_1}^\intercal \mathbf{v_2})\mathbf{q_1} + \mathbf{q_2}\|\mathbf{u_2}\|_2,$$

and the same procedure can be applied to all subsequent vectors, up to $\mathbf{v_n}$. By renaming $r_{ii} = \|\mathbf{u_i}\|_2$ and $r_{ij} = \mathbf{q_i}^\intercal \mathbf{v_j}$, we can rewrite Gram-Schmidt orthogonalization in matrix form:

$$\underbrace{\begin{bmatrix}\mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n}\end{bmatrix}}_{V \in \mathbb{R}^{m\times n}} = \underbrace{\begin{bmatrix}\mathbf{q_1} & \mathbf{q_2} & \dots & \mathbf{q_n}\end{bmatrix}}_{Q \in \mathbb{R}^{m\times n}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & r_{nn} \end{bmatrix}}_{R \in \mathbb{R}^{n\times n}}.$$

The formulation above corresponds to the **reduced QR factorization** of matrix V.

The Full QR Factorization I

The **full QR factorization** can be achieved by expanding matrix Q such that it achieves full rank (or, in other words, it spans all $\mathbb{R}^m$). We can simply do this by adding $m-n$ additional vectors through the means of Gram-Schmidt orthogonalization to matrix Q and by adding the same amount of zero vectors as additional rows of matrix R. We then obtain:

$$\underbrace{\begin{bmatrix}\mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n}\end{bmatrix}}_{V\in\mathbb{R}^{m\times n}} = \underbrace{\begin{bmatrix}\mathbf{q_1} & \mathbf{q_2} & \dots & \mathbf{q_n} & \mathbf{q_{n+1}} & \dots & \mathbf{q_m}\end{bmatrix}}_{Q\in\mathbb{R}^{m\times m}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \\ 0 & \dots & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 \end{bmatrix}}_{R\in\mathbb{R}^{m\times n}}.$$

The Full QR Factorization II

It is important to stress the dimensions of matrices Q and R in the two cases:

- In the **reduced QR factorization**, matrix $Q \in \mathbb{R}^{m \times n}$ and matrix $R \in \mathbb{R}^{n \times n}$.
- In the **full QR factorization**, matrix $Q \in \mathbb{R}^{m \times m}$ and matrix $R \in \mathbb{R}^{m \times n}$. In other words, matrix Q presents $m - n$ additional columns, while matrix R will have an additional matrix of zeros of dimension $\mathbb{R}^{(m-n) \times n}$ in the lower part.

In the full QR factorization, matrix Q is then an orthogonal square matrix. This implies that:

$$Q^\intercal = Q^{-1}.$$

Furthermore, Q multiplied for a generic vector $\mathbf{x} \in \mathbb{R}^{m \times 1}$, simply yields that:

$$\|Q\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2.$$

⋆ Exercise 2: The Reduced QR Factorization

Using Gram-Schmidt orthogonalization, find the reduced QR factorization of these matrices:

$$V_1 = \begin{bmatrix} 4 & 0 \\ 3 & 1 \end{bmatrix},$$

$$V_2 = \begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 2 & 1 \end{bmatrix}.$$

⋆ Exercise 3: The Full QR Factorization

Using Gram-Schmidt orthogonalization, find the full QR factorization of these matrices:

$$V_1 = \begin{bmatrix} -4 & -4 \\ -2 & 7 \\ 4 & -5 \end{bmatrix},$$

$$V_2 = \begin{bmatrix} 4 & 8 & 1 \\ 0 & 2 & -2 \\ 3 & 6 & 7 \end{bmatrix}.$$

⋆ Exercise 4: QR Factorization and Least Squares

Consider the following least squares problem $A\boldsymbol{\alpha} = \mathbf{b}$, given by:

$$\begin{bmatrix} 2 & 4 \\ 0 & -1 \\ 2 & -1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Compute the full QR factorization of matrix A and use it to find the least squares solution $\boldsymbol{\alpha}^*$.

Classical Gram-Schmidt Orthogonalization Algorithm

The classical Gram-Schmidt orthogonalization method can be summarized as follows:

Algorithm Classical Gram-Schmidt Orthogonalization

Let $\mathbf{v_j}$, $j = 1, \ldots, n$ be n linearly independent vectors
for $j = 1$ to n **do**
  $\mathbf{u} = \mathbf{v_j}$
  for $i = 1$ to $j - 1$ **do**
    $r_{ij} = \mathbf{q_i}^\mathsf{T} \mathbf{v_j}$
    $\mathbf{u} = \mathbf{u} - r_{ij}\mathbf{q_i}$
  end for
  $r_{jj} = \|\mathbf{u}\|_2$
  $q_j = \frac{\mathbf{u}}{r_{jj}}$
end for

Modified Gram-Schmidt Orthogonalization Algorithm

The accuracy of the computations can be improved with a small modification (check line 5):

Algorithm Modified Gram-Schmidt Orthogonalization

1: Let $\mathbf{v_j}$, $j = 1, \ldots, n$ be n linearly independent vectors
2: **for** $j = 1$ to n **do**
3: $\quad \mathbf{u} = \mathbf{v_j}$
4: $\quad$ **for** $i = 1$ to $j - 1$ **do**
5: $\quad\quad r_{ij} = \mathbf{q_i}^\intercal \mathbf{u}$
6: $\quad\quad \mathbf{u} = \mathbf{u} - r_{ij}\mathbf{q_i}$
7: $\quad$ **end for**
8: $\quad r_{jj} = \|\mathbf{u}\|_2$
9: $\quad q_j = \frac{\mathbf{u}}{r_{jj}}$
10: **end for**

⋆ Exercise 5: Comparison of Gram-Schmidt Algorithms

Consider the matrix V defined as follows:

$$V = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}.$$

Implement the classical and modified Gram-Schmidt algorithms in Matlab and compare the results, computed in double precision, in the following cases:

(a) $\varepsilon = 10^{-3}$,

(b) $\varepsilon = 10^{-10}$.

Householder Reflectors

The modified Gram-Schmidt orthogonalization can be used to compute the QR factorization of a matrix in a more efficient way than with the classical algorithm. However, there are other techniques that can reduce even further the propagation of numerical errors in the algorithm.

Among these, we can mention the Householder reflection, a linear transformation consisting in a reflection about a plane or hyperplane which contains the origin. The Householder reflector consists in an orthogonal matrix H that relocates a given vector to another vector of the same length. In other words, given two vectors $\mathbf{u}$ and $\mathbf{v}$ we have that $H\mathbf{u} = \mathbf{v}$ and $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2$.

Householder Reflectors

Consider two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ such that $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2$. We can define the following quantity:

$$\mathbf{w} = \mathbf{v} - \mathbf{u},$$

and construct the projection matrix P (i.e., a matrix for which $P^2 = P$) as:

$$P = \frac{\mathbf{w}\mathbf{w}^\mathsf{T}}{\mathbf{w}^\mathsf{T}\mathbf{w}}.$$

Then the Householder reflector H, which ensures that $H\mathbf{u} = \mathbf{v}$, is given by

$$H = I_n - 2P$$

and is a symmetric orthogonal matrix. Please notice that I_n is the identity matrix in $\mathbb{R}^n$.

$\star$ **Additional exercise:** Show that P is a symmetric projection matrix and that $Pw = w$. Verify that also the householder reflector H is a symmetric matrix.

Exercise 6: Householder Reflectors

Consider the following two vectors in $\mathbb{R}^2$:

$$\mathbf{u} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}.$$

Find the Householder reflector H, such that $H\mathbf{u} = \mathbf{v}$.

Exercise 6: Solution I

Step 1: Check that $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2$ and compute $\mathbf{w} = \mathbf{v} - \mathbf{u}$

We can easily verify that $\|\mathbf{u}\|_2 = \sqrt{9 + 15} = 5$ and $\|\mathbf{v}\|_2 = 5$, so the first condition is satisfied.

We can now compute $\mathbf{w}$, which is given by:

$$\mathbf{w} = \mathbf{v} - \mathbf{u} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} - \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \end{bmatrix}.$$

Starting from this intermediate result, we can then compute the projection matrix P and the Householder reflector H.

Exercise 6: Solution II

Step 2: Compute the projection matrix P

We can compute the intermediate quantities:

$$\mathbf{w}\mathbf{w}^\mathsf{T} = \begin{bmatrix} 2 \\ -4 \end{bmatrix} \begin{bmatrix} 2 & -4 \end{bmatrix} = \begin{bmatrix} 4 & -8 \\ -8 & 16 \end{bmatrix},$$

$$\mathbf{w}^\mathsf{T}\mathbf{w} = \begin{bmatrix} 2 & -4 \end{bmatrix} \begin{bmatrix} 2 \\ -4 \end{bmatrix} = 20.$$

The projection matrix P is then given by:

$$P = \frac{\mathbf{w}\mathbf{w}^\mathsf{T}}{\mathbf{w}^\mathsf{T}\mathbf{w}} = \begin{bmatrix} \frac{1}{5} & -\frac{2}{5} \\ -\frac{2}{5} & \frac{4}{5} \end{bmatrix}.$$

Exercise 6: Solution III

Step 3: Compute the Householder reflector H

The Householder reflector H can be computed as follows:

$$H = I_2 - 2P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - 2 \begin{bmatrix} \frac{1}{5} & -\frac{2}{5} \\ -\frac{2}{5} & \frac{4}{5} \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & \frac{4}{5} \\ \frac{4}{5} & -\frac{3}{5} \end{bmatrix}.$$

We can also easily verify that the condition $H\mathbf{u} = \mathbf{v}$ is satisfied, by computing:

$$H\mathbf{u} = \begin{bmatrix} \frac{3}{5} & \frac{4}{5} \\ \frac{4}{5} & -\frac{3}{5} \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} = \mathbf{v}.$$

⋆ Exercise 7: QR Factorization with Householder Reflectors

Consider the following matrix in $\mathbb{R}^{2\times 2}$:

$$V = \begin{bmatrix} 3 & 1 \\ 4 & 3 \end{bmatrix}$$

Compute the QR factorization by using the Householder reflectors.

⋆ **Hint:** You can start by moving the first column $\mathbf{v_1} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ to $\begin{bmatrix} \|\mathbf{v_1}\|_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$.

Nonlinear Least Squares

In the previous slides we found the least-squares solution of a system of equations $A\mathbf{x} = \mathbf{b}$ with two different methods: the normal equations and the QR factorization. We then introduced some other algorithms and modifications aimed at improving the numerical stability and the quality of the results obtained. However, in some real-world scenarios we could face systems of nonlinear equations and we need other tools capable of dealing with this kind of problems.

In this last part of the lecture, we will consider two methods capable of dealing with nonlinear least squares: the Gauss-Newton and the Levenberg-Marquardt methods. In particular, we will present two examples, in which we consider both an application to the solution of nonlinear least-squares problems and the applications to models with nonlinear parameters.

Gauss-Newton Method

Starting from a system of m residual equations in n unknowns (i.e., $\mathbf{x} \in \mathbb{R}^n$) of the form:

$$r_j(\mathbf{x}) = 0,$$

with $j = 1, \ldots, m$, the Gauss-Newton method aims at minimizing the sum of squared residuals:

$$E(\mathbf{x}) = \sum_{j=1}^{m} \left[r_j(\mathbf{x})\right]^2 = \mathbf{r}(\mathbf{x})^\intercal \mathbf{r}(\mathbf{x}),$$

where $\mathbf{r}(\mathbf{x}) \in \mathbb{R}^m$ is the vector of residual functions. Given an initial guess $x^{(0)}$, the algorithm proceeds iteratively by computing, at every iteration i, the following quantity:

$$x^{(i+1)} = x^{(i)} - \left[J_{\mathbf{r}}\left(\mathbf{x}^{(i)}\right)^\intercal J_{\mathbf{r}}\left(\mathbf{x}^{(i)}\right)\right]^{-1} J_{\mathbf{r}}\left(\mathbf{x}^{(i)}\right)^\intercal \mathbf{r}\left(\mathbf{x}^{(i)}\right),$$

with $J_{\mathbf{r}}\left(\mathbf{x}^{(i)}\right)$ indicating the Jacobian matrix of the vector-valued function $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$.

⋆ Vector Calculus Reminder: The Jacobian Matrix

Given a vector-valued function of several variables, the Jacobian matrix is the matrix consisting of all its first-order partial derivatives. More formally, let us consider a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, taking as input n variables, represented by a vector $\mathbf{x} \in \mathbb{R}^n$, and returning as output a vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$. The Jacobian matrix $J_{\mathbf{f}} \in \mathbb{R}^{m \times m}$ of function $\mathbf{f}$ is given by:

$$J_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}, \tag{3}$$

where the entry in position (i, j) simply corresponds to the first-order partial derivative of $f_i(\mathbf{x})$ with respect to x_j, and every row corresponds to the transpose of the gradient of the component function $f_i(\mathbf{x})$, i.e., $\nabla f_i(\mathbf{x})$.

Gauss-Newton Algorithm

The Gauss-Newton method can be summarized as follows:

Algorithm Gauss-Newton Method

1: Given a random initial guess $\mathbf{x}^{(0)}$
2: **for** $i = 0, 1, 2, \dots$ **do**
3: $\quad A = J_{\mathbf{r}}(\mathbf{x}^{(i)})$
4: $\quad \mathbf{v}^{(i)} = -(A^\intercal A)^{-1} A^\intercal \mathbf{r}(\mathbf{x}^{(i)})$
5: $\quad \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \mathbf{v}^{(i)}$
6: **end for**

In other words, the Gauss-Newton algorithm solves for a root of the gradient of the squared residual, since we know that a necessary condition for having a minimum is that the gradient must be equal to 0 in the given point. However, the fact that the gradient is equal to 0 does not necessary imply that the point is a minimum (we could converge, e.g., to a maximum): thus, the results obtained must be interpreted with due care.

Exercise 8: Distance from three Circles

Consider the following three circles in the plane:

1. center $(x_1, y_1) = (-4, -1)$, radius $R_1 = 3$,
2. center $(x_2, y_2) = (1, -2)$, radius $R_2 = 3$,
3. center $(x_3, y_3) = (1, 3)$, radius $R_3 = 2$.

Use the Gauss-Newton method to find the coordinates of the point minimizing the squared distance to the three circles defined above.

Exercise 8: Solution I

Given a circle centred in the origin $x^2 + y^2 = R$, where R indicates the radius, and a point A with coordinates (x_A, y_A), we know that the distance between the two is given by:

$$d(x_A, y_A) = \left| \sqrt{x_A^2 + y_A^2} - R \right|.$$

In case the circle is not centred in the origin but in (x_C, y_C), the distance equation becomes:

$$d(x_A, y_A) = \left| \sqrt{(x_C - x_A)^2 + (y_C - y_A)^2} - R \right|. \tag{4}$$

By using Eq. (4), we can formulate the problem as a system of equations, given by the residuals (or distances) between the unknown point (x, y) and the centers of the three circles.

Exercise 8: Solution II

Step 1: Define the vector of the residuals $\mathbf{r}(\mathbf{x})$

Our system of equations will then be given by:

$$
\begin{aligned}
r_1(x, y) &= \sqrt{(x - x_1)^2 + (y - y_1)^2} - R_1, \\
r_2(x, y) &= \sqrt{(x - x_2)^2 + (y - y_2)^2} - R_2, \\
r_3(x, y) &= \sqrt{(x - x_3)^2 + (y - y_3)^2} - R_3,
\end{aligned}
$$

and it can be condensed in the residual vector $\mathbf{r}(x, y)$:

$$
\mathbf{r}(x, y) = \begin{bmatrix} ((x - x_1)^2 + (y - y_1)^2)^{\frac{1}{2}} - R_1 \\ ((x - x_2)^2 + (y - y_2)^2)^{\frac{1}{2}} - R_2 \\ ((x - x_3)^2 + (y - y_3)^2)^{\frac{1}{2}} - R_3 \end{bmatrix}.
$$

Exercise 8: Solution III

Step 2: Compute the Jacobian matrix $J_{\mathbf{r}}(\mathbf{x})$ and apply Gauss-Newton method

We can now compute the Jacobian matrix of $\mathbf{r}(x, y)$, which is given by:

$$J_{\mathbf{r}}(x,y) = \begin{bmatrix} \frac{\partial r_1(\mathbf{x})}{\partial x} & \frac{\partial r_1(\mathbf{x})}{\partial y} \\ \frac{\partial r_2(\mathbf{x})}{\partial x} & \frac{\partial r_2(\mathbf{x})}{\partial y} \\ \frac{\partial r_3(\mathbf{x})}{\partial x} & \frac{\partial r_3(\mathbf{x})}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{x-x_1}{((x-x_1)^2+(y-y_1)^2)^{\frac{1}{2}}} & \frac{y-y_1}{((x-x_1)^2+(y-y_1)^2)^{\frac{1}{2}}} \\ \frac{x-x_2}{((x-x_2)^2+(y-y_2)^2)^{\frac{1}{2}}} & \frac{y-y_2}{((x-x_2)^2+(y-y_2)^2)^{\frac{1}{2}}} \\ \frac{x-x_3}{((x-x_3)^2+(y-y_3)^2)^{\frac{1}{2}}} & \frac{y-y_3}{((x-x_3)^2+(y-y_3)^2)^{\frac{1}{2}}} \end{bmatrix}.$$

By randomly choosing the initial guess as $\left(x^{(0)}, y^{(0)}\right)$, we now have all the quantities we need to use Gauss-Newton method. The rest of the solution is left as an exercise.

⋆ **Additional task:** Complete the exercise by applying Gauss-Newton algorithm.

Exercise 9: Exponential Model Revisited

Let us consider again the dataset introduced in Exercise 6 of tutorial 7:

x	1950	1955	1960	1965	1970	1975	1980
y	53.05	73.04	98.31	139.78	193.48	260.20	320.39

with x representing the year and y the value at that specific time. **Without** data linearization, find the least squares best fit to a simple exponential model, expressed as:

$$y = \alpha_1 e^{\alpha_2 x},$$

and compute the RMSE of the resulting model.

$\star$ **Hint:** For given x_i and y_i, we can define the residual as:

$$r_i = \alpha_1 e^{\alpha_2 x_i} - y_i.$$

Exercise 9: Solution I

Step 1: Define the vector of the residuals $\mathbf{r}(\boldsymbol{\alpha})$

As in the original exercise, we have 7 equations in 2 unknowns, with vector of the residuals:

$$\mathbf{r}(\alpha_1, \alpha_2) = \begin{bmatrix} \alpha_1 e^{\alpha_2 x_1} - y_1 \\ \alpha_1 e^{\alpha_2 x_2} - y_2 \\ \alpha_1 e^{\alpha_2 x_3} - y_3 \\ \alpha_1 e^{\alpha_2 x_4} - y_4 \\ \alpha_1 e^{\alpha_2 x_5} - y_5 \\ \alpha_1 e^{\alpha_2 x_6} - y_6 \\ \alpha_1 e^{\alpha_2 x_7} - y_7 \end{bmatrix}.$$

Please notice that the vector of the residuals is function of α_1 and α_2, since those are the two parameters we are trying to estimate (while the x and y values are given data).

Exercise 9: Solution II

Step 2: Compute the Jacobian matrix $J_{\mathbf{r}}(\boldsymbol{\alpha})$ and apply Gauss-Newton method

We can now compute the Jacobian matrix of $\mathbf{r}(\alpha_1, \alpha_2)$, which is given by:

$$J_{\mathbf{r}}(\alpha_1, \alpha_2) = \begin{bmatrix} \frac{\partial r_1(\boldsymbol{\alpha})}{\partial \alpha_1} & \frac{\partial r_1(\boldsymbol{\alpha})}{\partial \alpha_2} \\ \vdots & \vdots \\ \frac{\partial r_7(\boldsymbol{\alpha})}{\partial \alpha_1} & \frac{\partial r_7(\boldsymbol{\alpha})}{\partial \alpha_2} \end{bmatrix} = \begin{bmatrix} e^{\alpha_2 x_1} & \alpha_1 x_1 e^{\alpha_2 x_1} \\ e^{\alpha_2 x_2} & \alpha_1 x_2 e^{\alpha_2 x_2} \\ e^{\alpha_2 x_3} & \alpha_1 x_3 e^{\alpha_2 x_3} \\ e^{\alpha_2 x_4} & \alpha_1 x_4 e^{\alpha_2 x_4} \\ e^{\alpha_2 x_5} & \alpha_1 x_5 e^{\alpha_2 x_5} \\ e^{\alpha_2 x_6} & \alpha_1 x_6 e^{\alpha_2 x_6} \\ e^{\alpha_2 x_7} & \alpha_1 x_7 e^{\alpha_2 x_7} \end{bmatrix}.$$

By replacing every x_i and y_i with the data of the problem and by randomly choosing the initial guess as $\left(\alpha_1^{(0)}, \alpha_2^{(0)}\right)$, we now have all the quantities we need to use Gauss-Newton method. Once again, the rest of the solution is left to the reader as an additional exercise.

Levenberg-Marquardt Algorithm

To conclude, it is worth mentioning the Levenberg-Marquardt method, which tries to tackle the eventual ill-conditioning of the Jacobian matrix by introducing a regularization term λ.

Algorithm Levenberg-Marquardt Method

1: Given a random initial guess $\mathbf{x}^{(0)}$ and a regularization parameter λ
2: **for** $i = 0, 1, 2, \dots$ **do**
3: $\quad A = J_{\mathbf{r}}(\mathbf{x}^{(i)})$
4: $\quad \mathbf{v}^{(i)} = -(A^\intercal A + \lambda \operatorname{diag}(A^\intercal A))^{-1} A^\intercal \mathbf{r}(\mathbf{x}^{(i)})$
5: $\quad \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \mathbf{v}^{(i)}$
6: **end for**

We can notice that, in case we consider $\lambda = 0$, we end up with the Gauss-Newton method explained above. If, on the other end, we decide to increase the value of the regularization parameter λ, we increase the influence of the diagonal of $A^\intercal A$ and, consequently, the condition number of the resulting matrix. This modification allows the Levenberg-Marquardt method to achieve convergence with a broader set of starting values $x^{(0)}$ than Gauss-Newton method.