Homework 1

Giuseppe Labate 2095665

1) Analytical solution

The analytic solution for a planar robotic arm with 3 revolute joints (3R) can be determined because the three joint axes are consecutive and parallel to each other.

Since the links dimensions are $a_1 = a_2 = a_3 = 1$ the robot is built like in *Figure 1*

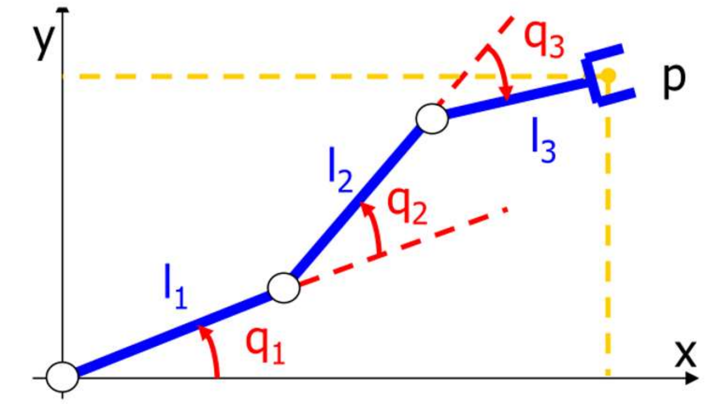


Figure 1: Schematic of the robot

The workspaces are:

- Reachable workspace: $WS_1 = \{p \in \mathbb{R}^2 : \|p\| \leq 3\} \subset \mathbb{R}^2$
- Dexterous workspace $WS_2 = \{p \in \mathbb{R}^2 : \|p\| \leq 1\} \subset \mathbb{R}^2$

Now the desired point is $x_e^d = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$ with $p_x = 2$ and $p_y = 1$.

This means that the norm is given by $\|p\| = \sqrt{{p_x}^2 + {p_y}^2} = \sqrt{5}$.

By looking at the WS_1 and WS_2, we can say that the robot is able to reach x_e^d but not in a dexterous way since $x_e^d \in WS_1$ and $x_e^d \notin WS_2$.

By looking at the k(q) and x_e^d, we can derive this system of equations $\begin{cases} c_1 + c_{12} + c_{123} = 2 \\ s_1 + s_{12} + s_{123} = 1 \\ \theta_1 + \theta_2 + \theta_3 = 0 \end{cases}$

Using $\theta_1 + \theta_2 + \theta_3 = 0$ we get this system of equations:

$\begin{cases} c_1 + c_{12} + c(0) = 2 \\ s_1 + s_{12} + s(0) = 1 \end{cases} \rightarrow \begin{cases} c_1 + c_{12} = 1 \\ s_1 + s_{12} = 1 \end{cases}$

1.1) Find θ_2

To find θ_2 we can square the two equations and sum the results.

$(s_1 + s_{12})^2 + (c_1 + c_{12})^2 = 2$

$s_1{}^2 + s_{12}{}^2 + 2s_1 s_{12} + c_1{}^2 + c_{12}{}^2 + 2c_1 c_{12} = 2$

By using the trigonometry fundamental identity $s_1{}^2 + c_1{}^2 = 1$

$\rightarrow \quad 2s_1 s_{12} + 2c_1 c_{12} = 0 \quad \rightarrow \quad s_1 s_{12} + c_1 c_{12} = 0$

$s_1(s_1 c_2 + c_1 s_2) + c_1(c_1 c_2 - s_1 s_2) \quad \rightarrow \quad c_2(s_1{}^2 + c_1{}^2) = 0$

Again, using the trigonometry fundamental identity:

$$c_2 = 0 \quad \rightarrow \quad s_2 = \pm\sqrt{1 - c_2{}^2} = \pm 1 \quad \rightarrow \quad \theta_2 = ATAN2\{s_2, c_2\}$$

So, we get two results for θ_2, which makes sense since we can reach the desired point with two different configurations.

$$\theta_2 = \pm\frac{\pi}{2}$$

1.2) Find θ_1

To find θ_1, we can replace θ_2 in $\begin{cases} c_1 + c_{12} = 1 \\ s_1 + s_{12} = 1 \end{cases}$

$$\begin{cases} c_1 + c_1 c_2 - s_1 s_2 = 1 \\ s_1 + s_1 c_2 + c_1 s_2 = 1 \end{cases}$$

- $If\ \theta_2 = +\frac{\pi}{2}$

  $\begin{cases} c_1 - s_1 = 1 \\ s_1 + c_1 = 1 \end{cases} \quad \rightarrow \quad s_1 + c_1 + c_1 - s_1 = 2 \quad \rightarrow \quad c_1 = 1$

  Replace $c_1 = 1$ in one of the two equations:

  $1 + s_1 = 1 \quad \rightarrow \quad s_1 = 0$

  $$\theta_1 = ATAN2\{s_1, c_1\} = 0$$

- $If\ \theta_2 = -\frac{\pi}{2}$

  $\begin{cases} s_1 + c_1 = 1 \\ s_1 - c_1 = 1 \end{cases} \quad \rightarrow \quad s_1 + c_1 + s_1 - c_1 = 2 \quad \rightarrow \quad s_1 = 1$

  Replace $s_1 = 1$ in one of the two equations:

  $1 + c_1 = 1 \quad \rightarrow \quad c_1 = 0$

  $$\theta_1 = ATAN2\{s_1, c_1\} = +\frac{\pi}{2}$$

1.3) Find θ_3

Finally, use the $\theta_1 + \theta_2 + \theta_3 = 0$ to find θ_3.

- $If\ \theta_1 = +\frac{\pi}{2}\ and\ \theta_2 = -\frac{\pi}{2}$

$$\theta_3 = 0$$

- $If\ \theta_1 = 0\ and\ \theta_2 = +\frac{\pi}{2}$

$$\theta_3 = -\frac{\pi}{2}$$

1.4) Results

The solutions for the inverse kinematic problem are:

- $\theta_1 = 0, \theta_2 = +\frac{\pi}{2}, \theta_3 = -\frac{\pi}{2}$

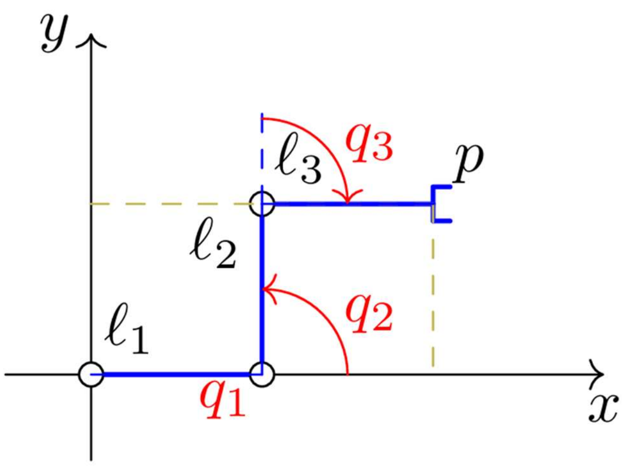


Figure 2a: First configuration

- $\theta_1 = +\frac{\pi}{2}, \theta_2 = -\frac{\pi}{2}, \theta_3 = 0$

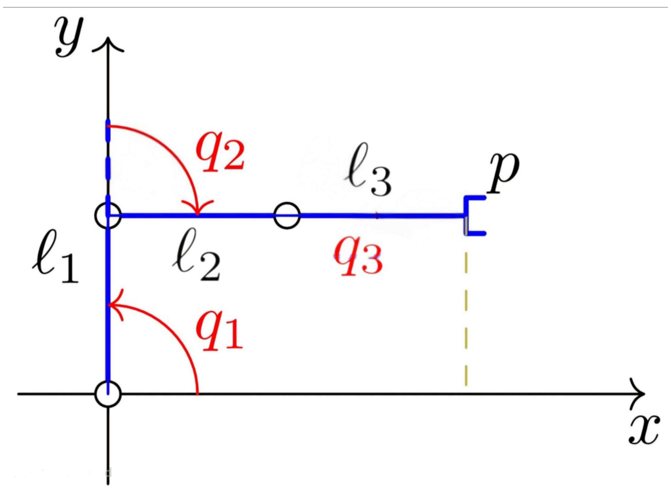


Figure 2b: Second configuration

2) Gradient Method

To solve the inverse kinematic problem using the gradient method, I've implemented a MATLAB script that applies the following formula until the error drops below a threshold (in my case I've set it to 1e-6).

$$q^{k+1} = q^k + \alpha J_k^T\,(q^k)\,[x_e^d - k(q^k)]$$

If the error fails to fall below the threshold within 1000 iterations, we consider it a divergence.

2.1) Results for $\boldsymbol{\alpha = \frac{1}{2}}$

2.1.1) $q(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

Here the algorithm can't find a proper solution since the value of α is too big.
As evidence of this we can see the q-values (*Figure 3a*) and the error (*Figure 3b*) keep bouncing back and forth.

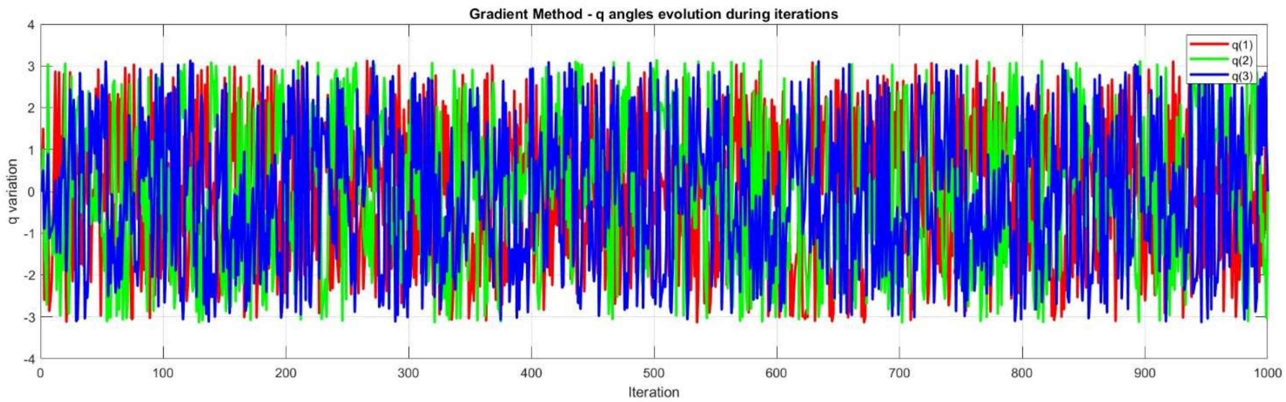


Figure 3a: the values of q oscillate continuously, indicating that the chosen alpha is too large and we keep missing the solution eternally

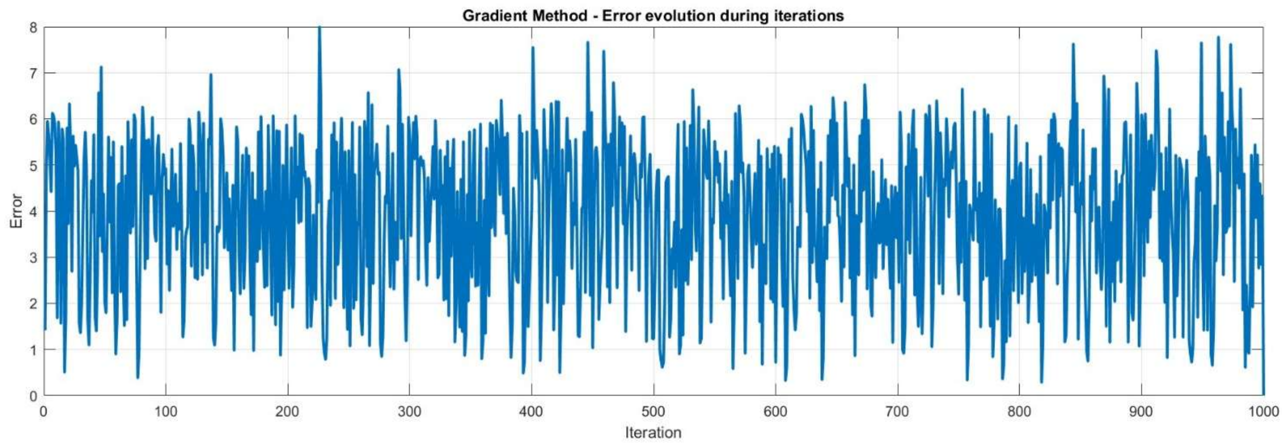


Figure 3b: the error oscillates continuously, indicating that the chosen alpha is too large and we keep missing the solution eternally

2.1.2) $q(0) = \begin{pmatrix} \pi/2 \\ \pi/2 \\ \pi/2 \end{pmatrix}$

What said above is true also for this initialization.
Usually we can start from different initializations to reach all possible solutions, but in this case, even if we're starting from a different point q (0), alpha is still too large to fall into the minimum error and find the actual solutions to the problem.
As you can see, the solution (*Figure 4a)* and the error (*Figure 4b)* keep changing as before.

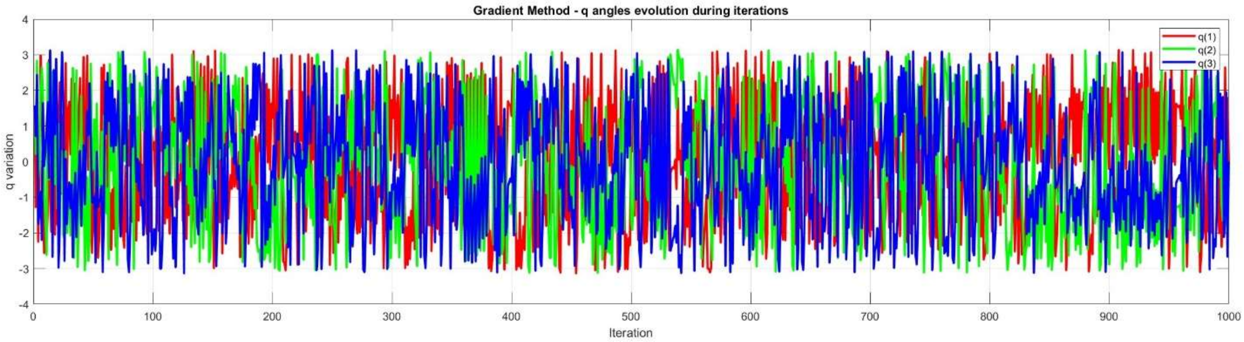


Figure 4a: q-values

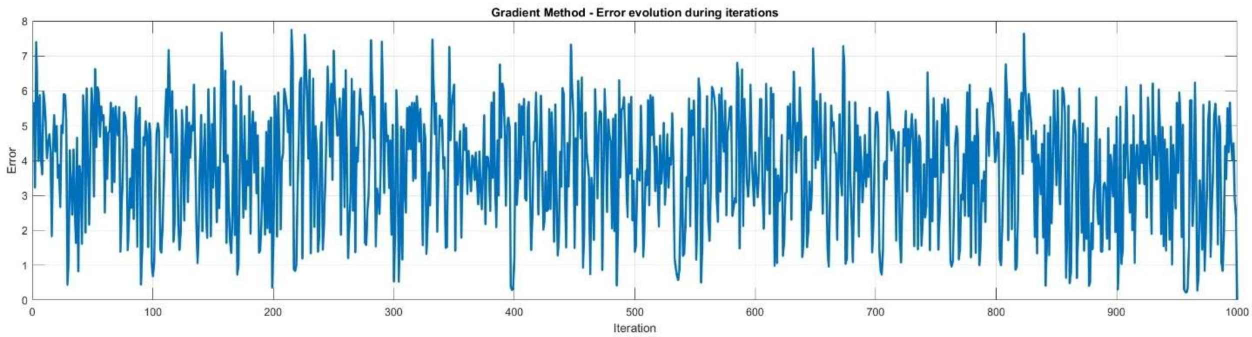


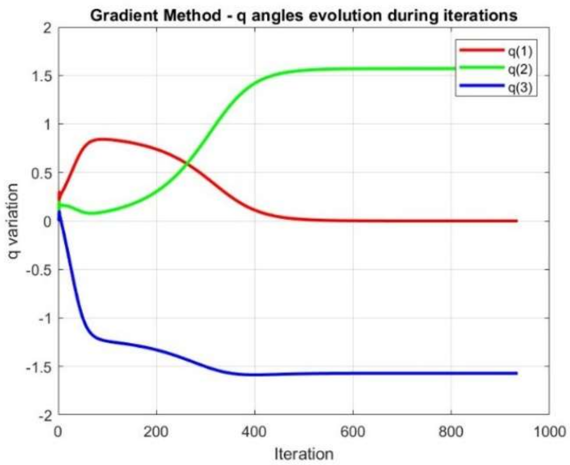
Figure 4b: Error

2.2) Results for $\boldsymbol{\alpha} = \frac{\mathbf{1}}{\mathbf{10}}$

Since we've now lowered the alpha value, we can converge to a solution by approaching it gradually, unlike before, when the alpha was too large and the gradient method couldn't converge to the minimum because it kept bouncing around it.
Note that we sacrifice computation time to ensure that the machine actually converges to a solution.
With alpha = 1/2, convergence would be much faster, but we wouldn't be sure of reaching a solution at all.

2.2.1) $q(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

The solution obtained is: $q = \begin{pmatrix} 0 \\ \pi/2 \\ -\pi/2 \end{pmatrix}$, so the configuration assumed by the arm is the one that we can see in *Figure 2a*



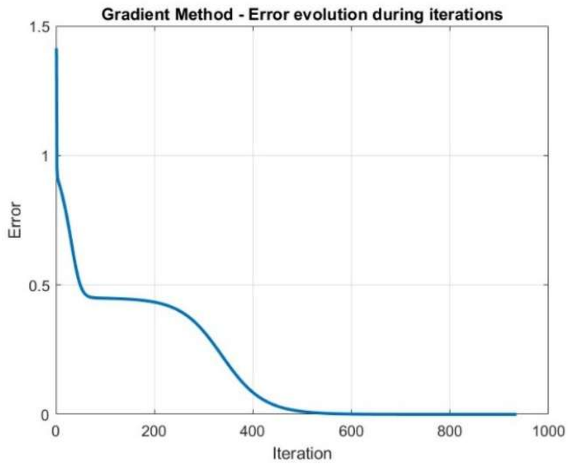


Figure 5: q-values on the left and the error on the right

2.2.2) $q(0) = \begin{pmatrix} \pi/2 \\ \pi/2 \\ \pi/2 \end{pmatrix}$

Because of the small alpha, even when starting from a different value of q, we're still able to reach a solution.
Usually, multiple different initializations are used to find the various solutions to the problem. In this case, however, we end up finding the same solution as before.

$$q = \begin{pmatrix} 0 \\ \pi/2 \\ -\pi/2 \end{pmatrix}$$

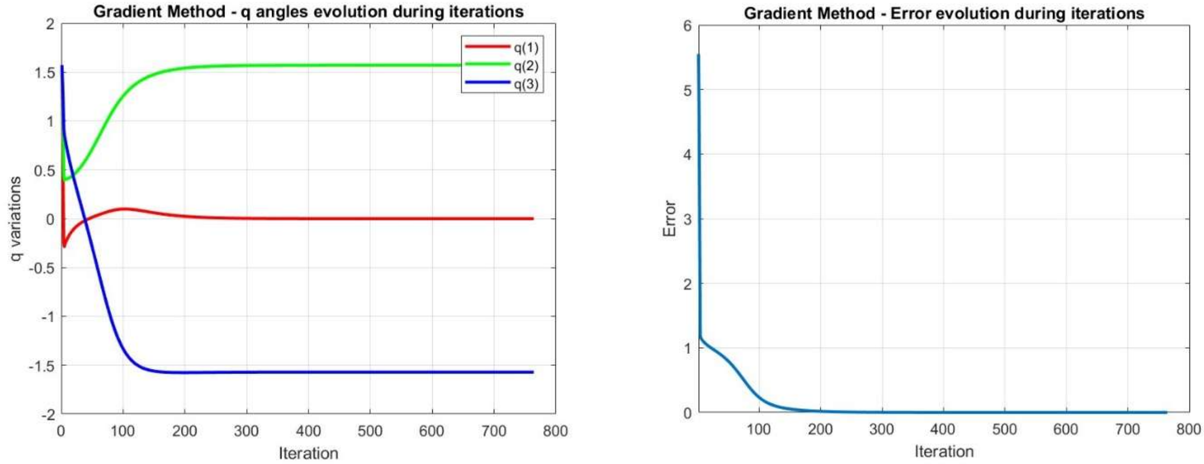


Figure 6: q-values on the left and the error on the right

3) Newton Method

The Newton Method has the advantage of converging much faster than the Gradient Method. However, it has a major drawback: if the robot starts from or passes through a configuration with a singular point, the method fails and cannot find a solution.
This happens because the method requires inverting the Jacobian J to compute the new q, as one can see in the formula below.

$$q^{k+1} = q^k + J_k^{-1}(q^k)[x_e^d - k(q^k)]$$

When the robot reaches a singular point, the Jacobian cannot be inverted since its determinant becomes zero.
In this case, $\det(\mathrm{J}) = 1/\sin(\theta_2)$, so when $\theta_2 = 0$ or $\theta_2 = \pi$, the Newton Method cannot be used.
To compute the solutions using the Newton Method, I wrote a MATLAB script that applies the formula, stops if it reaches a singular point, and accepts a solution only if the error is below a threshold of 1e-6.

3.1) $q(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

In this case, since $\theta_2 = 0$, the robot arm is at a singular point, and the Newton Method fails → no solution.

3.2) $q(0) = \begin{pmatrix} \pi/_2 \\ \pi/_2 \\ \pi/_2 \end{pmatrix}$

Here, the method successfully converges to this solution:

$$q = \begin{pmatrix} 0 \\ \pi/_2 \\ -\pi/_2 \end{pmatrix}$$

Note that it takes only 6 iterations to converge, compared to the 700/800 required by the Gradient Method.

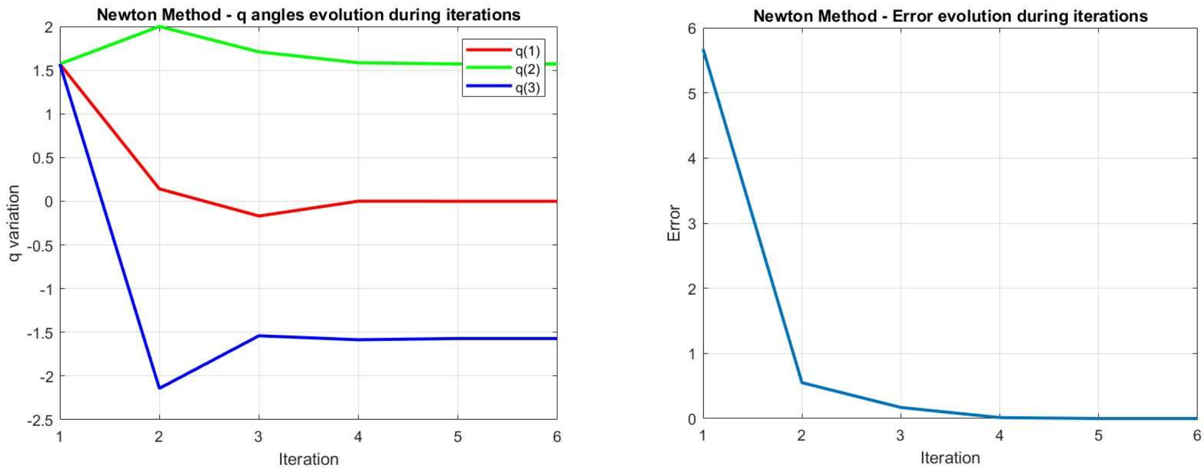


Figure 7: q-values on the left and the error on the right