Homework 1 Robotics and Control 1

Christian Francesco Russo 2087922

Exercise 1

Analytical solution to inverse kinematics problem of a planar 3R robotic arm.
Direct kinematics equation:

$$x_e = \kappa(q) = \begin{bmatrix} \cos(\theta_1) + \cos(\theta_1+\theta_2) + \cos(\theta_1+\theta_2+\theta_3) \\ \sin(\theta_1) + \sin(\theta_1+\theta_2) + \sin(\theta_1+\theta_2+\theta_3) \\ \theta_1+\theta_2+\theta_3 \end{bmatrix}$$

with link lengths $a_1 = a_2 = a_3 = 1$, and $q = [\theta_1 \;\; \theta_2 \;\; \theta_3]^T$ vector of the coordinates.
Desired pose to be attained by the end-effector: $x_e^d = [2 \;\; 1 \;\; 0]^T$.

Solution:

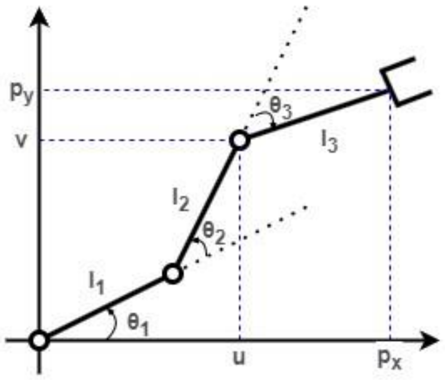
$$\begin{cases} p_x = \cos(\theta_1) + \cos(\theta_1+\theta_2) + \cos(\theta_1+\theta_2+\theta_3) \\ p_y = \sin(\theta_1) + \sin(\theta_1+\theta_2) + \sin(\theta_1+\theta_2+\theta_3) \\ \phi = \theta_1+\theta_2+\theta_3 \end{cases}$$

Substituting $\phi = \theta_1+\theta_2+\theta_3$ in the first two equations we obtain:

$$\begin{cases} p_x = \cos(\theta_1) + \cos(\theta_1+\theta_2) + \cos(\phi) \\ p_y = \sin(\theta_1) + \sin(\theta_1+\theta_2) + \sin(\phi) \end{cases} \rightarrow \begin{cases} p_x - c_\phi = c_1 + c_{12} \\ p_y - s_\phi = s_1 + s_{12} \end{cases}$$

where on the left side of the two equations we have the known terms and on the right side we have the unknown terms.
At this point, we can rename the known terms with two variables u, v:

$$\begin{cases} p_x - c_\phi = c_1 + c_{12} \\ p_y - s_\phi = s_1 + s_{12} \end{cases} \xrightarrow{\substack{u=p_x-c_\phi \\ v=p_y-s_\phi}} \begin{cases} u = c_1 + c_{12} \\ v = s_1 + s_{12} \end{cases}$$



Squaring and summing the two equations we obtain:

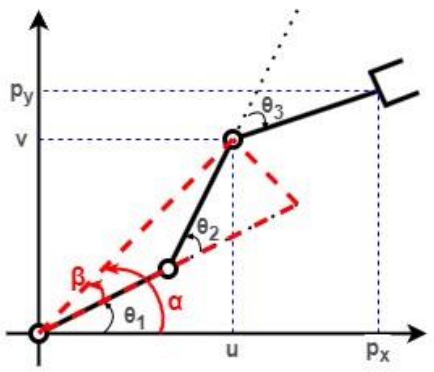
$$\begin{aligned} u^2 + v^2 &= c_1^2 + 2c_1c_{12} + c_{12}^2 + s_1^2 + 2s_1s_{12} + s_{12}^2 = \\ &= 2 + 2c_1c_{12} + 2s_1s_{12} = \\ &= 2 + 2c_1(c_1c_2 - s_1s_2) + 2s_1(c_2s_1 + c_1s_2) = \\ &= 2 + 2(c_1^2c_2 - c_1s_1s_2 + s_1^2c_2 + c_1s_1s_2) = \\ &= 2 + 2c_2 \end{aligned}$$

$$\Rightarrow \quad c_2 = \frac{u^2+v^2-2}{2}, \qquad s_2 = \pm\sqrt{1-c_2^2}$$

$$\Rightarrow \quad \theta_2 = \text{atan2}(s_2, c_2)$$

where must be $1 - c_2^2 \geq 0 \;\Rightarrow\; c_2 \in [-1,1]$.

By geometric intuition (see lesson PDF: “23-InverseKinematics_Analytical_methods”):



θ₁ = α - β

$\theta_1 = \text{atan2}(v,u) - \text{atan2}(s_2, 1+c_2)$

$\phi = \theta_1 + \theta_2 + \theta_3 \quad \Rightarrow \quad \theta_3 = \phi - \theta_1 - \theta_2$

At this point we have found all the relations for $\theta_1, \theta_2, \theta_3$, so we can do the computations to solve the inverse kinematics problem in order to make the end-effector reach the pose $x_e^d = [2 \;\; 1 \;\; 0]^T$:

$u = p_x - c_\phi = 2 - \cos 0 = 1$

$v = p_y - s_\phi = 1 - \sin 0 = 1$

$c_2 = \dfrac{u^2+v^2-2}{2} = \dfrac{1^2+1^2-2}{2} = 0$

$s_2 = \pm\sqrt{1-c_2^2} = \pm\sqrt{1-0^2} = \pm 1$

$\theta_2' = \text{atan2}(s_2, c_2) = \text{atan2}(1{,}0) = \dfrac{\pi}{2}$

$\theta_2'' = \text{atan2}(s_2, c_2) = \text{atan2}(-1{,}0) = -\dfrac{\pi}{2}$

$\theta_1' = \text{atan2}(v,u) - \text{atan2}(s_2, 1+c_2) = \text{atan2}(1{,}1) - \text{atan2}(1{,}1+0) = \dfrac{\pi}{4} - \dfrac{\pi}{4} = 0$

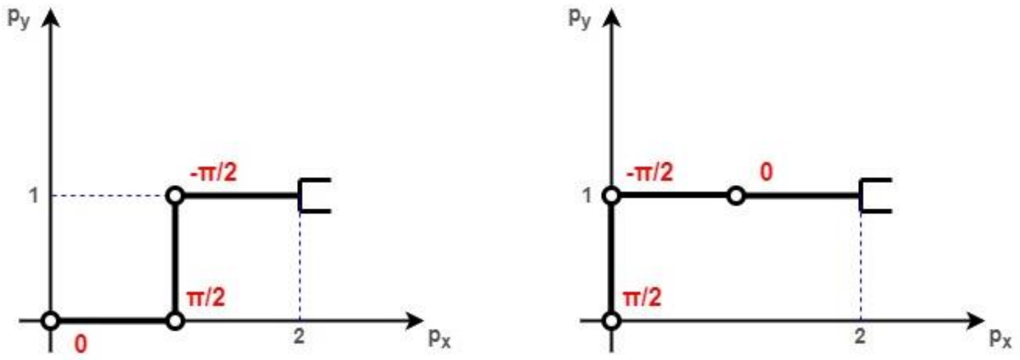
$\theta_1'' = \text{atan2}(v,u) - \text{atan2}(s_2, 1+c_2) = \text{atan2}(1{,}1) - \text{atan2}(-1{,}1+0) = \dfrac{\pi}{4} + \dfrac{\pi}{4} = \dfrac{\pi}{2}$

$\theta_3' = \phi - \theta_1 - \theta_2 = 0 - \dfrac{\pi}{2} - 0 = -\dfrac{\pi}{2}$

$\theta_3'' = \phi - \theta_1 - \theta_2 = 0 + \dfrac{\pi}{2} - \dfrac{\pi}{2} = 0$

Therefore, the two solutions for this inverse kinematics problem are:

- $\theta_1' = 0, \theta_2' = \pi/2, \theta_3' = -\pi/2$
- $\theta_1'' = \pi/2, \theta_2'' = -\pi/2, \theta_3'' = 0$



Exercise 2

Implementation of the Gradient method for inverse kinematics.
Solution:
First, we need to compute the Jacobian of $\kappa(q)$:

$$J_\kappa(q) = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} & \frac{\partial p_x}{\partial \theta_3} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} & \frac{\partial p_y}{\partial \theta_3} \\ \frac{\partial \phi}{\partial \theta_1} & \frac{\partial \phi}{\partial \theta_2} & \frac{\partial \phi}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ c_1 + c_{12} + c_{123} & c_{12} + c_{123} & c_{123} \\ 1 & 1 & 1 \end{bmatrix}$$

which is used in the MATLAB program **`gradient_method.mlx`** that I have implemented (see code attached) to compute the Gradient method update equation:

$$q(i+1) = q(i) + \alpha\, J_\kappa^T\big(q(i)\big)\left(x_e^d - \kappa\big(q(i)\big)\right)$$

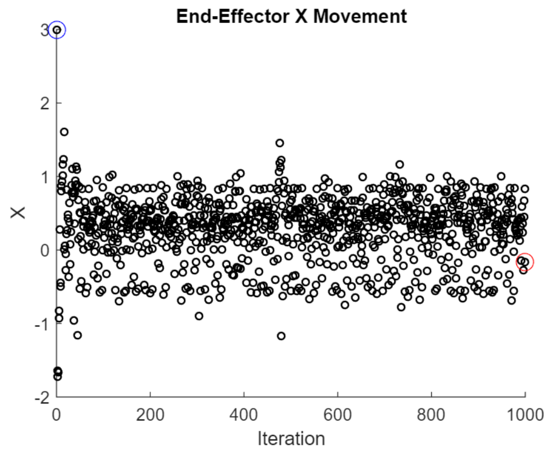
Case 1a:

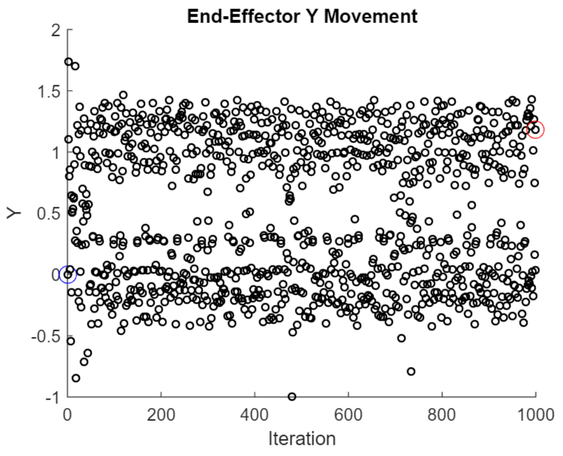
Initial condition:

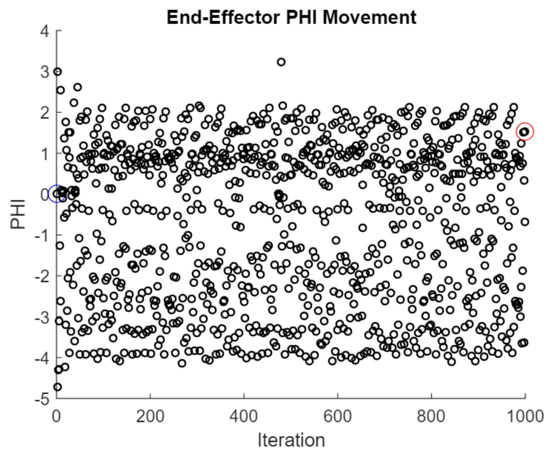
$$q(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

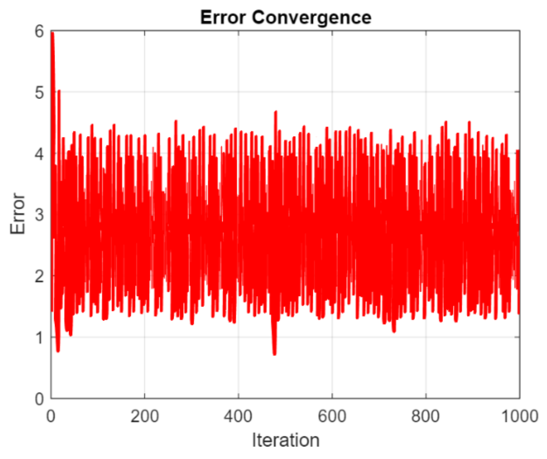
assuming $\alpha = 1/2$.

Results:
It does not converge:









Discussion:
From the theory, we know that the only cases in which the Gradient algorithm can get stuck (and then does not converge) are when the following two conditions are true:

- q is a singular point
- The corresponding error $e \in \ker(J_\kappa^T)$

Let's find out which are the singular points for this inverse kinematics problem.
First of all, let's compute the determinant of the Jacobian matrix.

$$
\begin{aligned}
\det(J_\kappa(q)) &= \begin{vmatrix} -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ c_1 + c_{12} + c_{123} & c_{12} + c_{123} & c_{123} \\ 1 & 1 & 1 \end{vmatrix} = \\
&= 1[\cancel{(-s_{12} - s_{123})(c_{123})} - \cancel{(-s_{123})(c_{12} + c_{123})}] + \\
&\quad -1[(-s_1 \cancel{- s_{12} - s_{123}})(c_{123}) - (-s_{123})(c_1 + \cancel{c_{12} + c_{123}})] + \\
&\quad +1[(-s_1 - s_{12} - s_{123})(c_{12} + c_{123}) - (-s_{12} - s_{123})(c_1 + c_{12} + c_{123})] = \\
&= \cancel{s_1 c_{123}} - \cancel{s_{123} c_1} - s_1 c_{12} - \cancel{s_1 c_{123}} - \cancel{s_{12} c_{12}} - \cancel{s_{12} c_{123}} - \cancel{s_{123} c_{12}} - \cancel{s_{123} c_{123}} \\
&\quad + s_{12} c_1 + \cancel{s_{12} c_{12}} + \cancel{s_{12} c_{123}} + \cancel{s_{123} c_1} + \cancel{s_{123} c_{12}} + \cancel{s_{123} c_{123}} = \\
&= c_1 s_{12} - s_1 c_{12}
\end{aligned}
$$

Let's now find out which are the singular points of the Jacobian, i.e., those points for which the Jacobian loses its rank:

$$\det(J_\kappa(q)) = c_1 s_{12} - s_1 c_{12} = 0$$

this happens when:

$$c_1 s_{12} = s_1 c_{12} \;\Leftrightarrow\; \frac{s_{12}}{c_{12}} = \frac{s_1}{c_1} \;\Leftrightarrow\; \tan(\theta_1 + \theta_2) = \tan(\theta_1) \;\Leftrightarrow\; \theta_2 = 0, \pi$$

so, the singular points are all those points which have $\theta_2 = 0$ or $\theta_2 = \pi$.
By this result, we can see that the initial point $q(0) = [0 \quad 0 \quad 0]^T$ from which the Gradient algorithm starts in this case is a singular point.
Let's finally check whether the algorithm can get stuck in this singular point.
To do this, first, we compute the error for $q(0)$:

$$e = x_e^d - \kappa(q(0)) = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

then we check whether $e \in \ker(J_\kappa^T)$:

$$J_\kappa^T\, e = \begin{bmatrix} -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ c_1 + c_{12} + c_{123} & c_{12} + c_{123} & c_{123} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \neq 0$$

this means that: $e \notin \ker(J_\kappa^T)$, hence, the Gradient algorithm will not get stuck in this point.
Moreover, we notice that the algorithm did not converge/get stuck in any local minima, but instead it just oscillated without never converging, and at the same time it did not diverge!
This means that, no other singular points were found during the iterations of the algorithm.
Hence, the only reason why the algorithm is not able to converge is due to the step size which is too high.

<u>Note</u>: an improvement for the program **`gradient_method.mlx`** I have proposed, could be to add a check to see whether $e \in \ker(J_\kappa^T)$ in order to find out if it gets stuck in some local minima.

Case 1b:

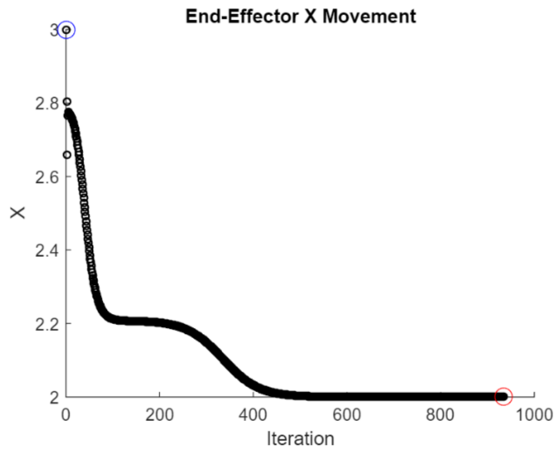
Initial condition:

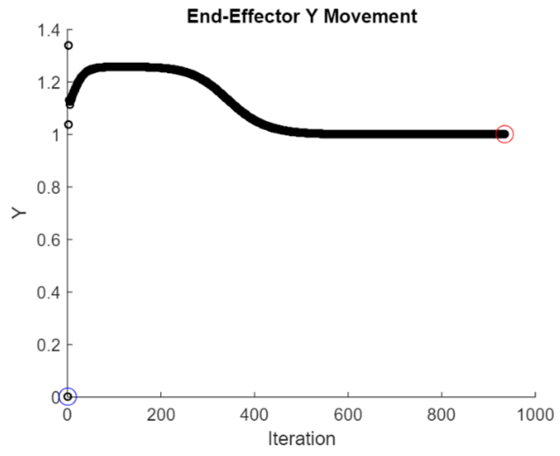
$$q(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

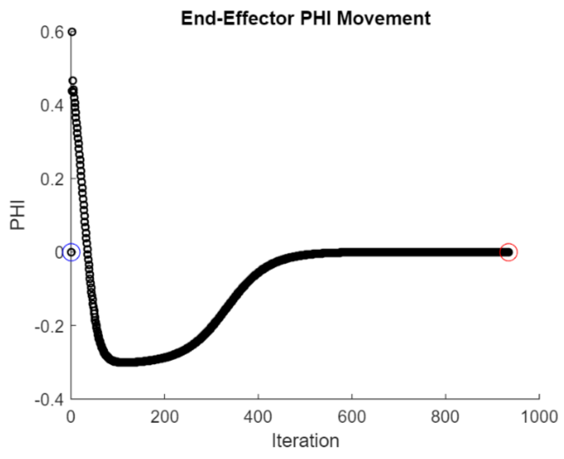
assuming $\alpha = 1/10$.

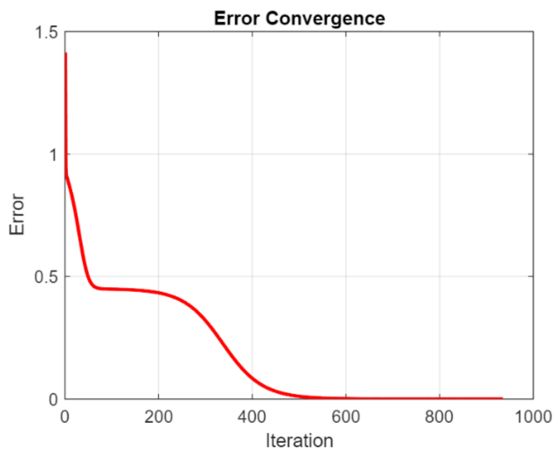
Results:

It converges after 935 iterations to the global minimum $q = [0, \pi/2\,, -\pi/2]^T$:









Discussion:

As proof of what discussed in the previous case, re-running the algorithm for a smaller step size: $\alpha = 1/10$ (since we know that, for α small enough, the Gradient algorithm is always able to converge to a local minima), as expected, the algorithm converges to a global minimum.

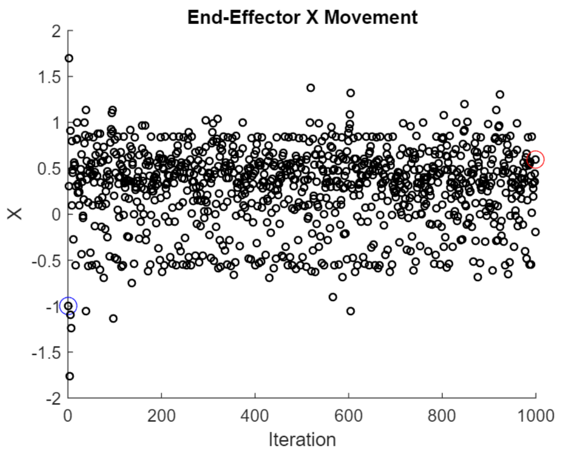
Case 2a:

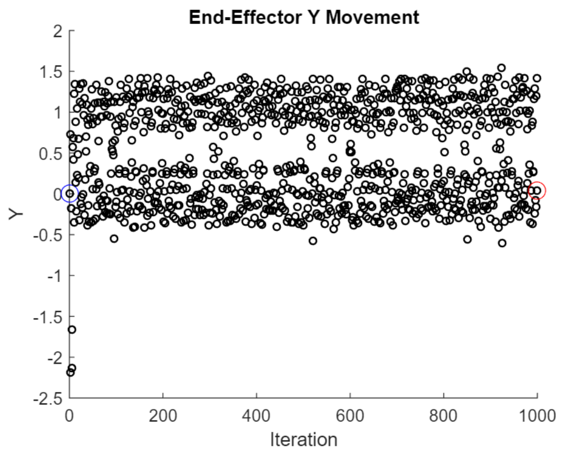
Initial condition:

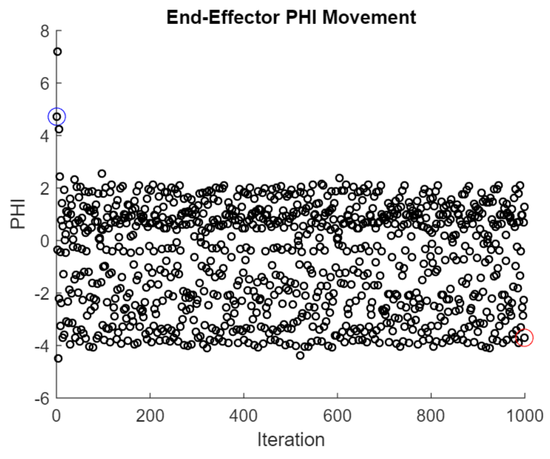
$$q(0) = \begin{bmatrix} \pi/2 \\ \pi/2 \\ \pi/2 \end{bmatrix}$$

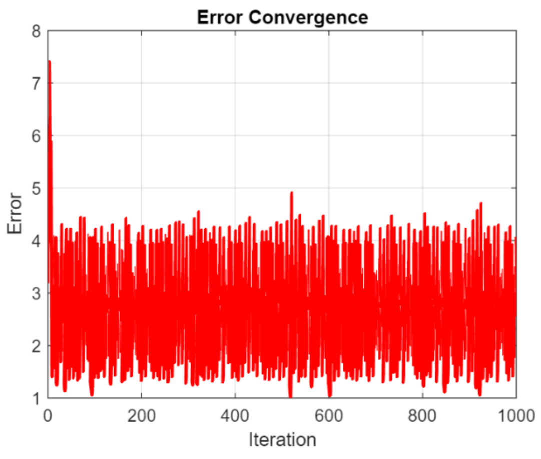
assuming $\alpha = 1/2$.

Results:
It does not converge:









Discussion:
Analogously to case 1a, the reason why in this case the algorithm does not converge is that the step size α is too big.

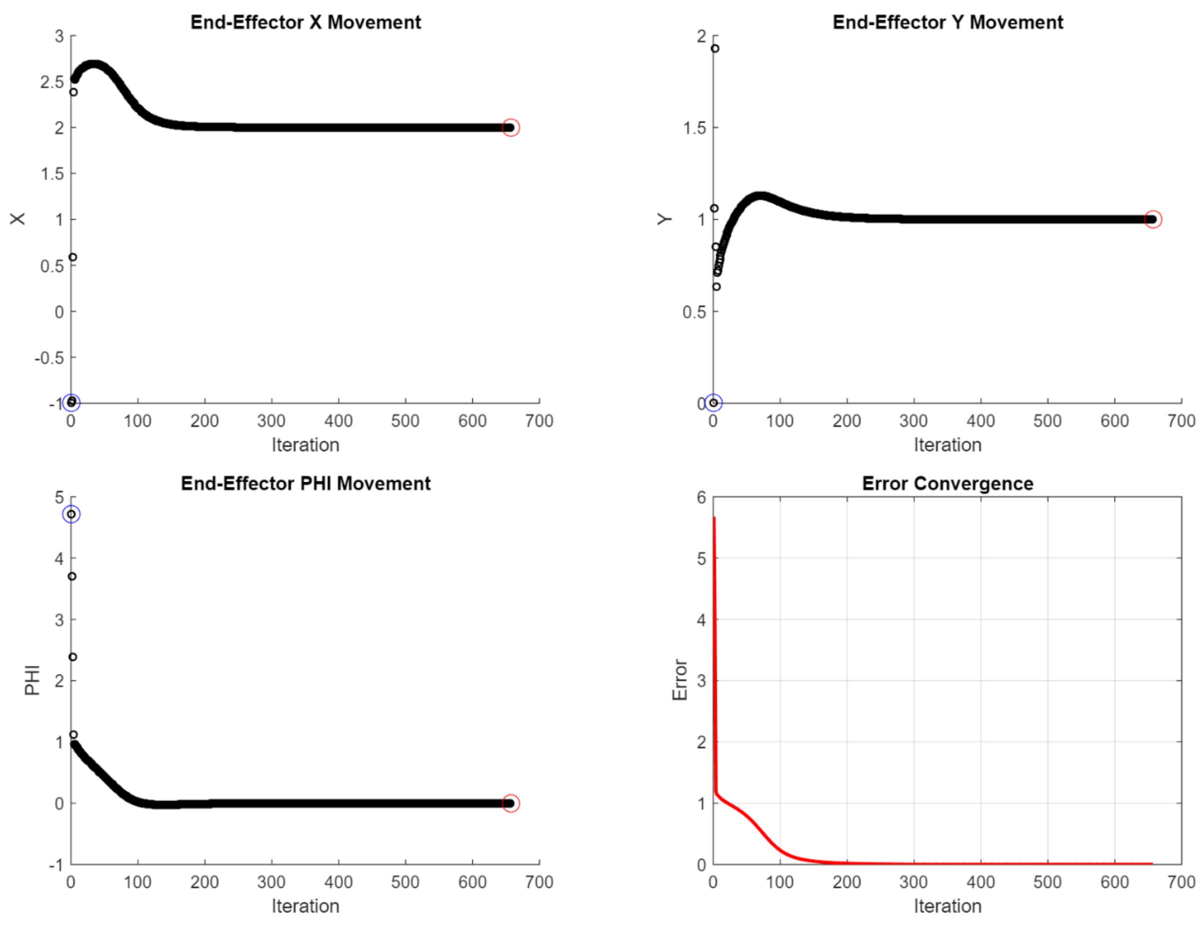
Case 2b:

Initial condition:

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

assuming $\alpha = 1/10$.

Results:
It converges after 657 iterations to the global minimum $q = [0, \pi/2\,, -\pi/2]^T$:



Discussion:
In this case, the algorithm slowly converges to a global minimum, where the speed of convergence is given by the value of the step size α.

Exercise 3

Implementation of the Newton's method for inverse kinematics.
Solution:
First, we need to compute the Jacobian of $\kappa(q)$:

$$J_\kappa(q) = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} & \frac{\partial p_x}{\partial \theta_3} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} & \frac{\partial p_y}{\partial \theta_3} \\ \frac{\partial \phi}{\partial \theta_1} & \frac{\partial \phi}{\partial \theta_2} & \frac{\partial \phi}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ c_1 + c_{12} + c_{123} & c_{12} + c_{123} & c_{123} \\ 1 & 1 & 1 \end{bmatrix}$$

which is used in the MATLAB program **`newton_method.mlx`** that I have implemented (see code attached) to compute the Newton's method update equation:

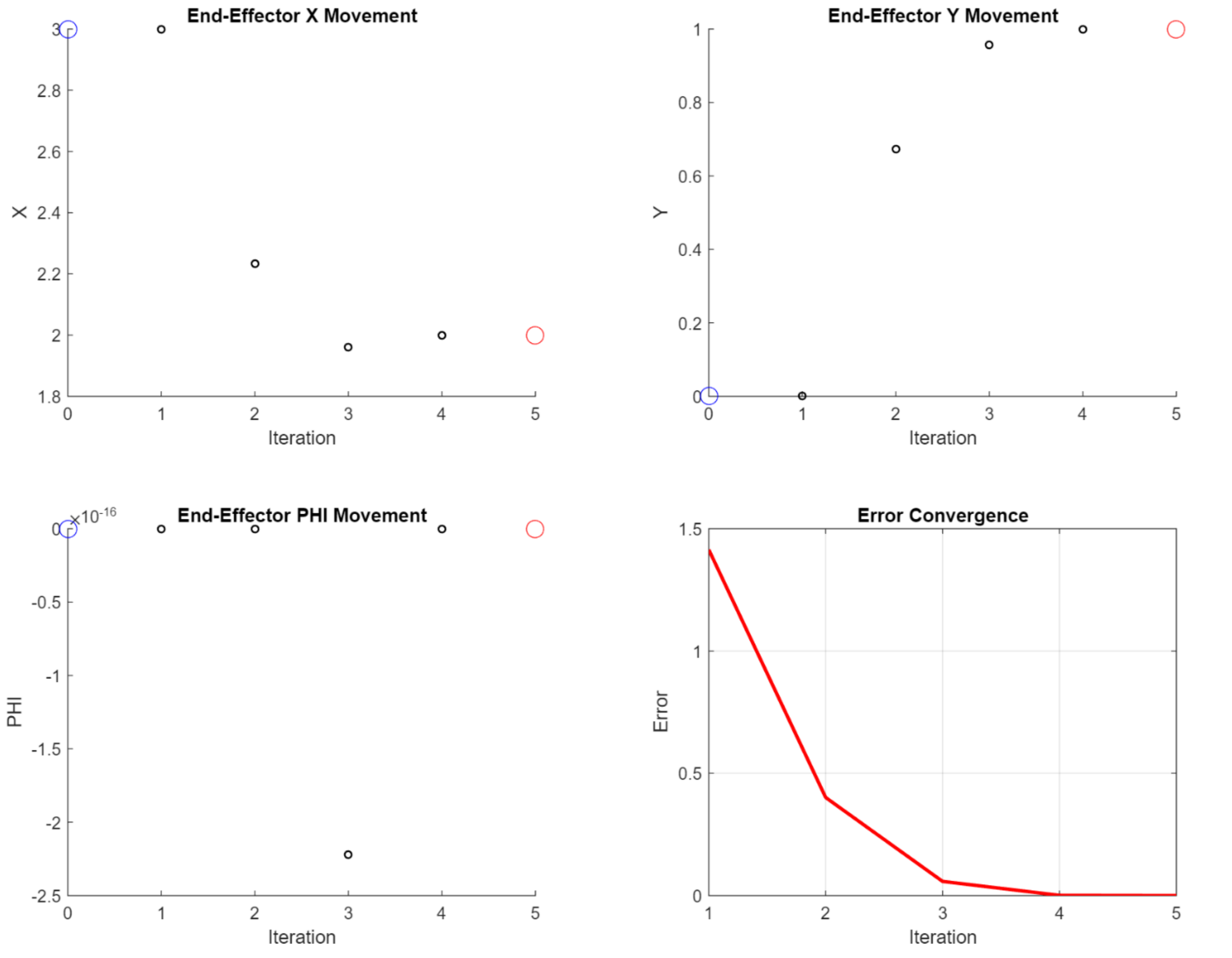
$$q(i+1) = q(i) + J_\kappa^{-1}\big(q(i)\big)\left(x_e^d - \kappa\big(q(i)\big)\right)$$

Case 1:

Initial condition:

$$q(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Results:
It converges after 5 iterations to the global minimum $q = [0, \pi/2\,, -\pi/2]^T$, even if MATLAB shows a message to warn us that the Jacobian matrix is close to a singular point:



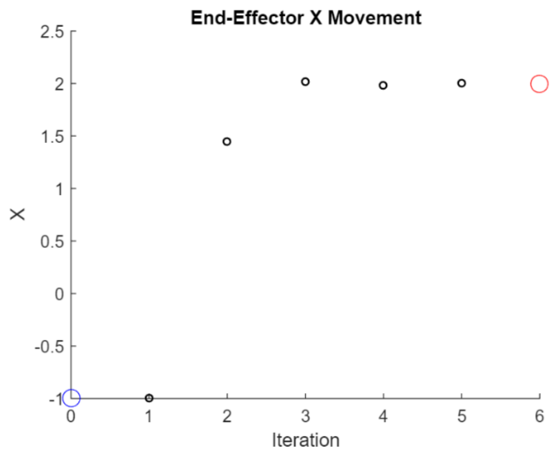
Discussion:
In this case, being $q(0) = [0 \quad 0 \quad 0]^T$ a singular point for the Jacobian, we expect to have numerical issues due to the computation of the inverse of the Jacobian.
However, surprisingly, the algorithm is able to run and to terminate finding out the correct global minimum solution in few iterations; although, at the first iteration of the algorithm, MATLAB shows a message to warn us that the Jacobian matrix is close to a singular point.
Finally, we notice that, as expected, the convergence in this case is appreciably much faster than the one we had with the Gradient method.

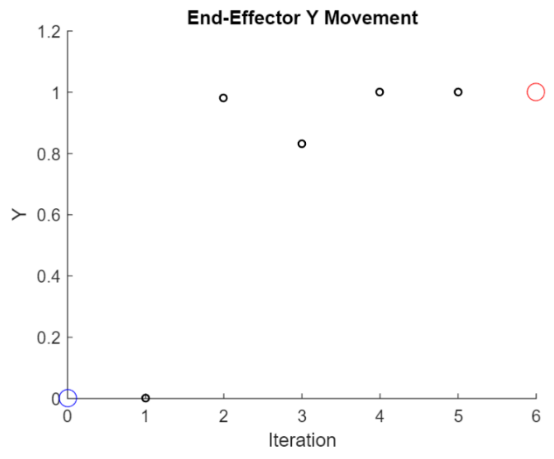
Case 2:

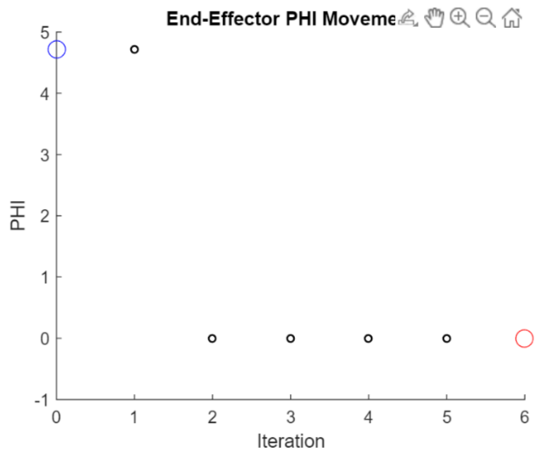
Initial condition:

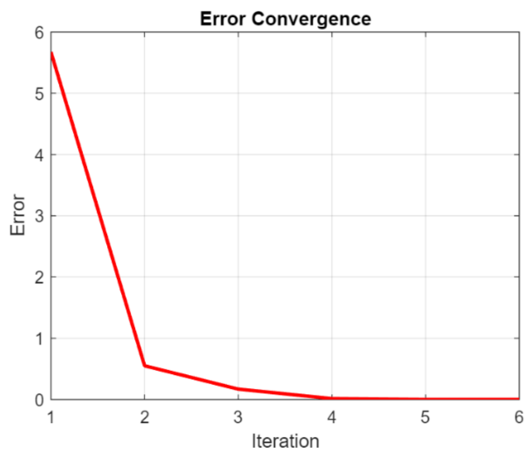
$$q(0) = \begin{bmatrix} \pi/2 \\ \pi/2 \\ \pi/2 \end{bmatrix}$$

Results:
It converges after 6 iterations to the global minimum $q = [0, \pi/2, -\pi/2]$:









Discussion:
In this case, the algorithm is able to find a global minimum in a much faster way w.r.t. the Gradient method.