A Comparison of Selected Methods of Inverse Kinematics for Robot Manipulators

David Ugochukwu Asogwa

Faculty of Electronics, Photonics and Microsystems
Wrocław University of Science and Technology

July 14, 2022

Contents

Introduction

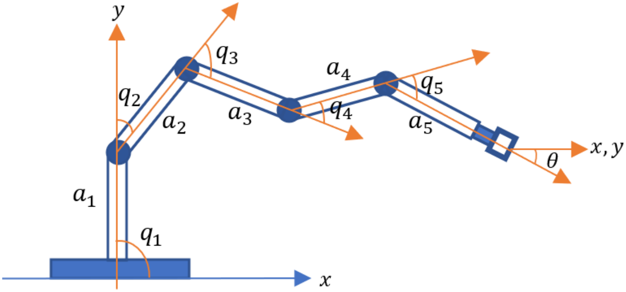


Figure: 5-DOF Manipulator Model

Objectives

- Solving the inverse kinematics problem using the Jacobi matrix methods based on the Newton-Gauss iterative formula.
- Implementation and testing of selected methods using different tasks.
- Evaluation of performance, and comparison of the methods.

Inverse Kinematic Methods

1. Analytical Method
2. Cyclic Coordinate Descent (CCD)
3. Dual Quaternions
4. Jacobian Transpose Method
5. Jacobian Pseudo-Inverse Method
6. Modified Levenberg-Marquardt Method

Jacobian Transpose Method

Definition

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \xi \cdot \boldsymbol{J}^T(\boldsymbol{q}_i)(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)) \tag{1}$$

Jacobian Pseudo-Inverse Method

Definition

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \xi \cdot \boldsymbol{J}^{\#}(\boldsymbol{q}_i)(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)) \tag{2}$$

$$\boldsymbol{J}^{\#}(\boldsymbol{q}_i) = \boldsymbol{J}^T(\boldsymbol{q}_i).(\boldsymbol{J}(\boldsymbol{q}_i)\boldsymbol{J}^T(\boldsymbol{q}_i))^{-1} = \boldsymbol{J}^T(\boldsymbol{q}_i).(\boldsymbol{M}(\boldsymbol{q}_i))^{-1} \tag{3}$$

Modified Levenberg-Marquardt Method

Definition

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \xi \cdot \boldsymbol{J}^T(\boldsymbol{q}_i)(diag(\boldsymbol{M}(\boldsymbol{q}_i))^{-1})(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)) \tag{4}$$

$$\boldsymbol{M}(\boldsymbol{q}_i) = (\boldsymbol{J}(\boldsymbol{q}_i)\boldsymbol{J}^T(\boldsymbol{q}_i))^{-1} \tag{5}$$

Kinematics and Jacobian Matrix

Definition

forward kinematics of the n-dof planar pendulum:

$$\boldsymbol{k}(\boldsymbol{q}_i) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} C_1(\boldsymbol{q}_i) \\ S_1(\boldsymbol{q}_i) \end{bmatrix} \tag{6}$$

$(m \times n)$ Jacobian matrix:

$$\boldsymbol{J}(\boldsymbol{q}_i) = \frac{\partial \boldsymbol{k}(\boldsymbol{q}_i)}{\partial \boldsymbol{q}_i} = \begin{bmatrix} -S_1(\boldsymbol{q}_i) & -S_2(\boldsymbol{q}_i) & \dots & -S_n(\boldsymbol{q}_i) \\ C_1(\boldsymbol{q}_i) & C_2(\boldsymbol{q}_i) & \dots & C_n(\boldsymbol{q}_i) \end{bmatrix} \tag{7}$$

Algorithm Implementation

Initialization:
Select method: Jacobian Transpose, Jacobian Pseudo-Inverse or Modified Levenberg-Marquardt
$x_f \longrightarrow$ Goal point
$\epsilon \longrightarrow$ Error margin (tolerance)
$q_0 \longrightarrow$ Initial configuration
$a_{i:n} \longrightarrow$ Manipulator arm lengths

```
procedure MATRIX MANIPULATIONS(J(q_i)^T, J(q_i)^#, diag(M(q_i))^{-1}.I)
    tic                                                   ▷ Start timer
    for i = 1 : ∞ do
        calculate q_{i+1}, toc    ▷ Calculate next configuration using previous configuration, stop timer
        if (‖ x_f − k(q_i) ‖) ≤ ε then
            break;                ▷ Stop iteration when tolerance condition is met
        end if
    end for
    d = |(x2−x1)(y1−y0)−(x1−x0)(y2−y1)| / sqrt((x2−x1)^2+(y2−y1)^2)
                                  ▷ Calculate distance between points and straight line segment
    α_i = arccos(⟨w1, w2⟩ / (‖w1‖·‖w2‖)), w1 = x_{i+1} − x_i,   w2 = x_f − x_i
                                  ▷ Calculate angle quality measure
    Display results               ▷ Print t, number of iterations, k_f and q_f
    plots                         ▷ Display plots
end procedure
```

$$d = \frac{|(x_2-x_1)(y_1-y_0)-(x_1-x_0)(y_2-y_1)|}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}$$

$$\alpha_i = \arccos\left(\frac{\langle w_1, w_2\rangle}{\|w_1\|\cdot\|w_2\|}\right), \quad w_1 = x_{i+1} - x_i, \quad w_2 = x_f - x_i$$

Statement of Problem

Definition

$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}), \boldsymbol{x} \in \mathbb{X} \subset \mathbb{SE}$

$\boldsymbol{n} \in \{2, 3, ...\}$: Planar pendula of **n** DOF

$\boldsymbol{q}_0 = (q_1, ..., q_n)^T, q \in \mathbb{Q}$

$\boldsymbol{x}_f = (x, y)^T$: Goal point

$\boldsymbol{a}_n$: Manipulator arm length for $\boldsymbol{n}^{th}$ DOF

$\xi = 0.01$: Regulation Constant

$\epsilon = 0.01$: Error Margin

$\boldsymbol{m} = 2$

$dim(\boldsymbol{q}) = n \geq m = dim(\boldsymbol{x})$

Tasks simulated

Task 1: $2 - DOF, \boldsymbol{x}_f = (3, -2)^T, \;\; \boldsymbol{q}_0 = (15, -45)^T$

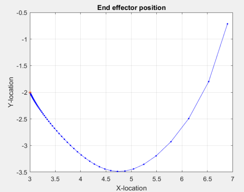
Task 2: $3 - DOF, \boldsymbol{x}_f = (-2, -3)^T, \;\; \boldsymbol{q}_0 = (30, -10, 45)^T$

Task 3: $4 - DOF, \boldsymbol{x}_f = (1, 3.6)^T, \;\; \boldsymbol{q}_0 = (0, -50, -10, -75)^T$

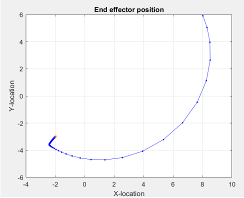
Task 4: $5 - DOF, \boldsymbol{x}_f = (4, 0)^T, \;\; \boldsymbol{q}_0 = (-60, 30, 0, 19, -55)^T$

Link Lengths: $a_1 = 4, \; a_2 = 3.5, \; a_3 = 3, \; a_4 = 3, \; a_5 = 3$

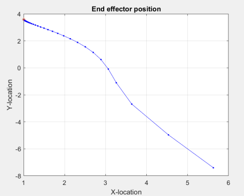
Jacobian Transpose Method



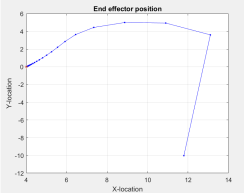
(a) Task 1: x-y position



(b) Task 2: x-y position

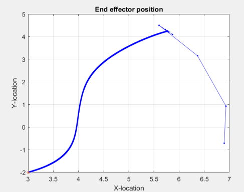


(c) Task 3: x-y position

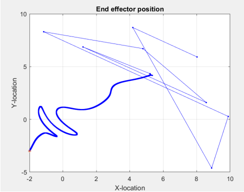


(d) Task 4: x-y position

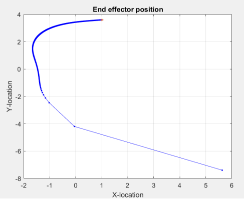
Modified Levenberg-Marquardt Method



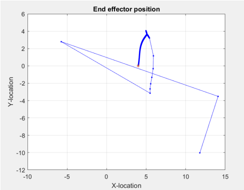
(a) Task 1: x-y position



(b) Task 2: x-y position

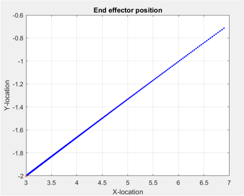


(c) Task 3: x-y position

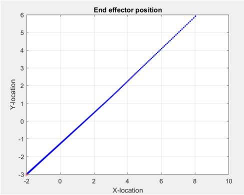


(d) Task 4: x-y position

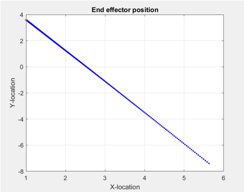
Jacobian Pseudo-Inverse Method



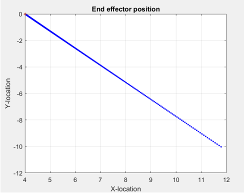
(a) Task 1: x-y position



(b) Task 2: x-y position



(c) Task 3: x-y position



(d) Task 4: x-y position

Execution Time

Table: Average Execution Time for Tasks 1 to 4:

Method	JPI	JT	MLM
Time (secs)	140.9075	5.3850	486.2975

Number of Iterations

Table: Average Number of Iterations for Tasks 1 to 4:

Method	JPI	JT	MLM
No. of Iteration	684	111	3,136

Distance from Straight Line Segment

Table: Average sum of distances from straight for Tasks 1 to 4:

Method	JPI	JT	MLM
Average (m)	0.0011535	0.1616535	0.4978198

Performance Characteristics

Table: Characteristics of the Jacobi methods discussed

Method	**Speed/Time**	**No. of iteration**	**Convergence**	**Straight Line**
JPI	—	—	Best	Best
JT	Fastest	Least	—	—
MLM	Takes much time	More iterations	least	least

Conclusion

- The number of iterations and execution time are not reliable for comparing the efficiency of these algorithms.
- The Levenberg-Marquardt method looses convergence at some point during the iteration process.
- Though completes the iterations faster and at less time, the Jacobian transpose method is still not the most efficient.
- A rare situation occurs when the Jacobian Transpose method runs into a continuous loop of iterations without convergence.
- the pseudo-inverse method is more effective and efficient in tracing a straight line to the goal point. Thus, this algorithm appears to be more intelligent compared to the other two.
- However, conclusions from the carried-out simulations should be treated with a due caution as they are based only on a few simulations and for arbitrary set of parameters.

Contributions

- Evaluation of Inverse Kinematics methods to determine optimal performance and efficiency.
- Algorithm and chart flow development, and implementation in MATLAB environment.
- Objective comparison of results based on data set obtained from algorithm implementation and simulation.

Thank You