Optimization Algorithms for Inverse Kinematics of Robots with MATLAB Source Code

Hazim Nasir Ghafil[1](✉) and Károly Jármai[2]

[1] University of Miskolc, Miskolc, Egyetemváros 3515, Hungary
vegyhnr@uni-miskolc.hu
[2] University of Kufa, Najaf, Iraq

Abstract. This work presents a methodology to solve the inverse kinematic problem for any kind of robot arm using optimization algorithms. Forward kinematic is usually a straightforward analysis for any robot while inverse kinematic is hard to be solved for many cases. Thus, depending on a set of the forward kinematic equation, the objective function can be formulated to be minimized to find the inverse position. This methodology makes the inverse kinematic very simple operation for all types of the robot, even for those who are complicated with a high degree of freedom. A particle example of 5DOF revolute joint arm was used to present this methodology with source code written in MATLAB for the objective function. Dynamic differential optimization algorithm DDAO was used to minimize the objective. DDAO has promising usage for embedded systems when prototyping a controller that estimates the inverse kinematic as per user request.

Keywords: Inverse kinematics · Optimization algorithms · Robotics · Dynamic differential optimization algorithm · Particle swarm optimization

1 Introduction

Inverse kinematics [1] is the cornerstone for articulated robots in all daily life applications because all the rest of the robotic processes depends on its output. Articulated arm robot moves by giving joint input variables to the actuators [2], and accordingly, the tip of the arm moves in Cartesian space. This is called forward kinematic, and it is a straightforward operation that does not need serious computations or optimizations. We deal with the motion in Cartesian space because that what is desired for most of the application while the movement in the joint space still in the shadow. The most difficult process is when the inputs are the Cartesian coordinates, and the desired is to find the corresponding joint variables. This is called inverse kinematics which is what we usually deal with most of the robotic applications. In the automotive industry, robots follow a specific trajectory of Cartesian points to do some achievement like welding, cutting, grinding, painting, etc. [3]. Thus, inverse kinematics maps the motion of the tooltip (or just the extreme tip of the arm) from Cartesian space where the tip swims to the joint space where the actuators perform. It is worth mentioning that for simple topology robots, inverse

K. Jármai and K. Voith (Eds.): VAE 2020, LNME, pp. 468–477, 2021.
https://doi.org/10.1007/978-981-15-9529-5_40

kinematics can be simple and solved by many methods like geometric or analytical solutions. For a complicated high degree of freedom robots, the process will be hard or even impossible to be solved by traditional methods [4]. In this work, we are presenting the formulation of the objective function for inverse kinematics to be solved by any of the optimization algorithms. Dynamic annealed optimization algorithm (DDAO) [5], specifically, proposed to find the inverse solutions for robot arms. What is interesting in this algorithm is that it is independent of the population size, and that makes DDAO perfect for embedded systems applications as we will express in the respective sections.

2 DDAO

Dynamic differential optimization algorithm (DDAO) is a physically inspired optimization algorithm that mimics the process of production dual-phase steel. The mathematical model of the algorithm is expressed as follows:

$$S^k = \left(Sc_i - Sc_j\right) + Sr.f \tag{1}$$

$$f = \begin{cases} 1 & if\ rem(iteration, 2) = 1 \\ random\,[0, 1] & if\ rem(iteration, 2) = 0 \end{cases}, \tag{2}$$

where rem is the remonder after division on 2, we suggest the same procedure depending on the probability formula described by SA algorithm

$$P = e^{\frac{-\Delta E}{T}}, \tag{3}$$

$$\Delta E = \frac{Cost(S^k) - Cost(S_L)}{Cost(S_L)}, \tag{4}$$

where S^k is a new solution proposed for the iteration number (k), $k = 1 \ldots n$ where n is the number of iterations, and Sc_i and Sc_j, are randomly chosen solutions from the population with random (i) and (j) indices. Sr is a randomly generated solution within the search space of the problem out of the population. P is the probability of accepting a new solution, ΔE is the difference between the objective value of the proposed solution from Eq. (1) and the objective value of the solution S_L, which is a solution of index L in the population, $L = 1,\ldots,$ population size. T is the temperature variable, which should start with high value and be updated during iterations constantly to a lower value. The proposed solution can be accepted if $P >$ random number $\in [0, 1]$. At the beginning of the search, T starts with high value; consequently, P will be close to one, according to Eq. (3). This means that a wide range of random numbers can be less that one and the solution will be selected. At the low value of T, the probability P will be close to zero; according to Eq. (3), this means that a very narrow range of random numbers could be less than P and the solution is less likely to be selected. The pseudocode illustrated below.

```
Initialize population X_i (i = 1,2,…,n)
Initialize parameter T, cooling rate
Calculate the cost of each solution
X_b= The best solution
While (t < Max iteration)
Initialize sub-population S
Calculate the cost of the sub-population
Sort sub-population
S_r= Best solution in sub-population
Choose two random solutions X_m and X_n from population
Calculate S_k from Eq. (1)
Sort population X
foreach solution in population X
if there is an improvement
X_i= S_k
otherwise, replace the worst solution in population X using
Eqs. (3) and (4)
endif
endfor
Update X_b
T = T*cooling rate
t = t+1
endwhile
return X_b
```

DDAO has a unique characteristic which is that it is independent of population size, this means that it uses the minimum size of the RAM when considering the population size of three individuals. Of course, other algorithms like particle swarm optimization [6], genetic algorithm [7], grey wolf optimization [8]. These algorithms can be used by setting population size as minimum as possible and what is the best algorithm for the inverse kinematic problem is left to the reader for future works.

3 Practical Example

In this study, LabVolt 5150 robot manipulator [9] is used to apply the proposed methodology of calculating the inverse kinematics using optimization algorithms. It is a 5 DOF manipulator; its rotational axes are base, shoulder, elbow, pitch and roller rotation, the manipulator equipped with a griper and all the revolute joints actuated by fife stepper motors. Figure 1 shows this robot manipulator, Fig. 2 expresses the configuration pace of the robot, and Fig. 3 reveals the frame assignment. According to the model shown in Fig. 2, the spatial parameters are estimated for each link, as shown in Table 1.

Fig. 1. Lab-volt 5150 manipulator

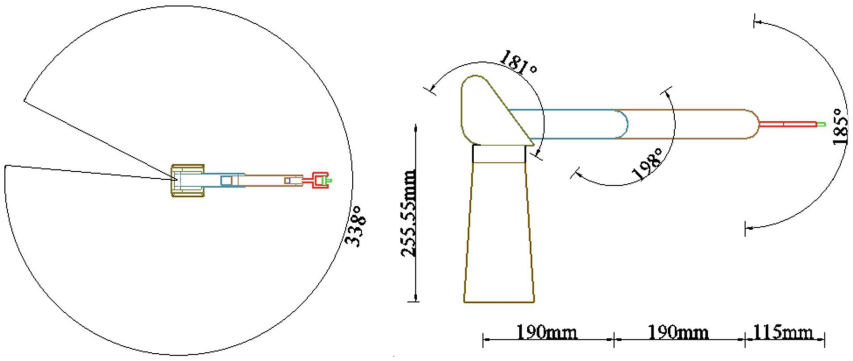

Fig. 2. Operative ranges and specifications of lab-volt 5150

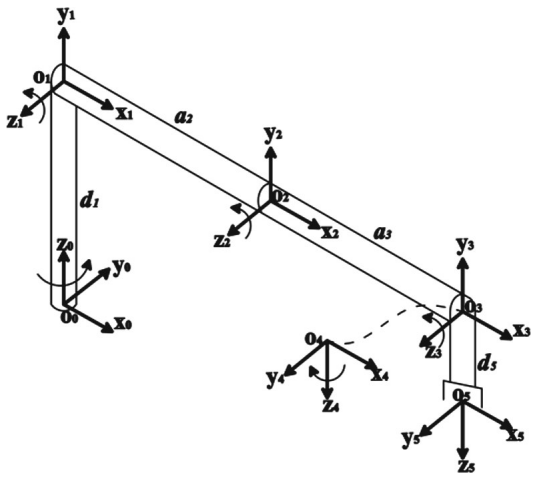

Fig. 3. Operative ranges and specifications of lab-volt 5150

Table 1. Spatial parameters of the lab-volt 5150 manipulator

Link ID	Frame	ϕ	α	a	d	*limits*
1	$o_0x_0y_0z_0$–$o_1x_1y_1z_1$	0	90	0	d_1	−185, 153
2	$o_1x_1y_1z_1$–$o_2x_2y_2z_2$	0	0	a_2	0	−32, 149
3	$o_2x_2y_2z_2$–$o_3x_3y_3z_3$	0	0	a_3	0	−147, 51
4	$o_3x_3y_3z_3$–$o_4x_4y_4z_4$	0	90	0	0	−5, 180
5	$o_4x_4y_4z_4$–$o_5x_5y_5z_5$	0	0	0	d_5	−360, 360

By using Denvit-Hartenberg convention [10], the homogenous transformation matrix HTM for the links are calculated as follows:

$$H_1^0 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$H_2^1 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2.C_2 \\ S_2 & C_2 & 0 & a_2.S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$H_3^2 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3.C_3 \\ S_3 & C_3 & 0 & a_3.S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$H_4^3 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & -C_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$H_5^4 = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$H = H_1^0 \times H_2^1 \times H_3^2 \times H_4^3 \times H_5^4 \tag{10}$$

equation (10) is 4×4 matrix holds the orientation and position vector of the end-effector with respect to the base frame, as revealed in Eq. (11) and Eq. (12).

$$R_5^0 = \begin{bmatrix} S_1.S_5 + C_1.C_5.C_{234} & S_1.C_5 - C_1.S_5.C_{234} & C_1.S_{234} \\ -C_1.S_5 + S_1.C_5.C_{234} & C_1.S_5 + S_1.C_5.C_{234} & S_1.S_{234} \\ C_5.S_{234} & S_5.S_{234} & -C_{234} \end{bmatrix} \tag{11}$$

$$P_5^0 = \begin{bmatrix} x = d_5C_1S_{234} + a_2C_1C_2 + a_3C_1C_{23} \\ y = d_5S_1S_{234} + a_2C_2S_1 + a_3C_{23} \\ z = -d_5C_{234} + a_2S_2 + d_1 + a_3S_{23} \end{bmatrix} \tag{12}$$

4 Objective Function

This is the problem of finding the joint variables from the given position and orientation of the end-effector. While forward kinematics is detecting the position and orientation of the end effector from the given set of joint variables, inverse kinematics is the inverse operation, but it is somewhat complicated.

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \Rightarrow \\ \Leftarrow \end{matrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

While forward equations are a straightforward process, we will rely on these equations to establish the objective function for the inverse problem.

Here we looking for an optimum set of joint variables that can lead to the minimum of a cost function, the only thing to do is developing cost function for the inverse Kinematic. Consider Fig. 4, for a specific robot configuration, the current position vector of the end-effector can be represented by the distance from the base of the end-effector of the manipulator while the desired position vector represents the task point. Obviously, if the difference between these two vectors is zero, then the tooltip will be in the right position at the task point, and this is the objective function f of the inverse problem

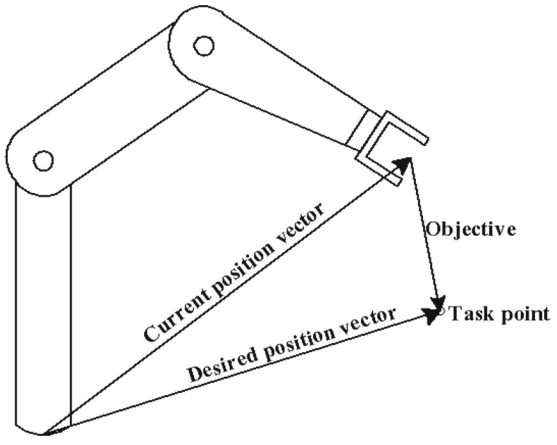

Fig. 4. Representation of the objective function for inverse kinematic problem

$$f = \|Ci - De\| \tag{14}$$

Where *Ci* denotes the instantaneous position vector, and *De* is the desired position vector. In other words, Eq. (14) is the function that has to be minimized as much as possible, and it just the distance between the end-effector and task point.

$$f = \sqrt{(x_{Ci} - x_t)^2 + (y_{Ci} - y_t)^2 + (z_{Ci} - z_t)^2} \tag{15}$$

Where *t* refers to the task point coordinates which is given for the inverse kinematic problem.

If Eq. (15) has been used alone as an objective function, we may get the end-effector in the task point but with many choices of orientations.

5 The Procedure of the Objective Function

In this section, we shall model the objective function for the inverse kinematic of any robot manipulator. Figure 5 shows a schematic diagram for the inverse problem; it is more descriptive to explain the procedure by a set of notes as follows.

1. Optimization algorithm sent the candidate solution, which is a set of possible joint variables, to the cost function to evaluate its fitness.
2. Cost function contains the desire task point coordinates; it sends the possible solution to Forward function to get x, y, and z coordinates of the tooltip.
3. Forward function contains all the forward kinematic equations of the robot arm, by substituting the candidate solution to that equations we can get the overall homogenous transformation matrix by a repeated call for HTM function. The output of the Forward function is the position vector of the total transformation matrix.
4. Cost function will receive the position vector and apply Eq. (15) to the candidate vector and the desired task position vector. The result is the fitness of the solution that will be back to the main optimization algorithm.

This process is constant for all types of manipulators; the only thing to change is the forward kinematic equations and the task position vector. It is worth mentioning that not all optimization methods can return a guaranteed solution; this depends on the efficiency of the algorithm itself.

The implementation of this methodology has a great significance on robot autonation, consequently, facilate and increase the productivity especially in aumotive engineeering. The automotive enginnering has wide applications for robots where they can be used in many cases and many operations [11, 12].

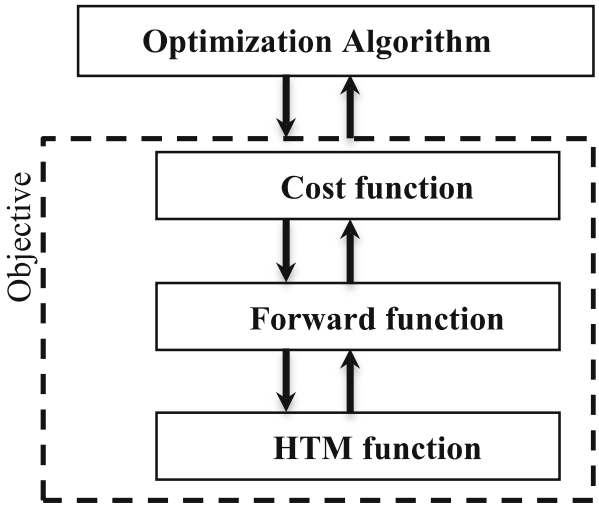

Fig. 5. Objective function scheme for inverse kinematic problem

6 Implementation of the Objective Function

The problem for LabVolt 5150 consists of five variables with lower and upper limits shown in Table 1, for a given point in space $v \in R^3$ the objective is finding the best corresponding joint angles that drive the end-effector to that point. From optimization algorithm, the candidate solution (`sol`) is transferred to the cost function (`cost`):

```
function obj = cost(sol)
t1 = sol(1);
t2 = sol(2);
t3 = sol(3);
t4= sol(4);
t5= sol(5);
[x,y,z] = Forward(t1,t2,t3,t4,t5);
% Cartesian point coordinate [x,y,z]
v = [495,0,255.55];
% Objective function; equation 11
obj = sqrt((x-v(1))^2+(y-v(2))^2+(z-v(3))^2);
end
```

For each candidate solution, we have to calculate the corresponding forward kinematic to find the position vector in Eq. (12) to be used in Eq. (15) to estimate the objective. Thus, the candidate solution v is transferred to the forward kinematic equations function (`Forward`). In this function, the spatial parameters are defined for each link considering data in Table 1:

```
function [x,y,z] = Forward(t1,t2,t3,t4,t5)
alpha_1 = 90;          %twist angle for link 1
a_1 = 0;               %distance between z1 and z2
d_1 = 255.55;          % distance between x1 and x2

alpha_2 = 0;           %twist angle for link 2
a_2 = 190;             %distance between z2 and z3
d_2 = 0;               %distance between x2 and x3

alpha_3 = 0;           %twist angle for link 3
a_3 = 190;             %distance between z3 and z4
d_3 = 0;               % distance between x3 and x4

alpha_4 = 90;          %twist angle for link 4
a_4 = 0;               %distance between z4 and z5
d_4 = 0;               % distance between x4 and x5

alpha_5 = 0;       %twist angle for link 5
a_5 = 0;           %distance between z5 and z6
d_5 = 115;         % distance between x5 and x6

H_1= HTM(t1,alpha_1,a_1,d_1); % HTM 1
H_2= HTM(t2,alpha_2,a_2,d_2); % HTM 2
H_3= HTM(t3,alpha_3,a_3,d_3); % HTM 3
H_4= HTM(t4,alpha_4,a_4,d_4); % HTM 4
H_5= HTM(t5,alpha_5,a_5,d_5); % HTM 5
TH = H_1*H_2*H_3*H_4*H_5;      % over all HTM
x=TH(1,4);
y=TH(2,4);
z=TH(3,4);
end
```

For general usage, we have developed separated function (`HTM`) to return the homogenous transformation matrix described by Denavit matrix.

```
function [T] = HTM(t,alpha,a,d)
T = [cosd(t),-
1*sind(t)*cosd(alpha),sind(t)*sind(alpha),a*cosd(t);...
    sind(t),cosd(t)*cosd(alpha),-
1*cosd(t)*sind(alpha),a*sind(t);...
    0,sind(alpha),cosd(alpha),d;...
    0,0,0,1];
end
```

The source code described above is general, one can solve the inverse kinematic of any robot just by replacing the proper spatial parameters and define them in function (`Forward`). According to the number of links, less or more Denavit matrices can

be estimated `H_6,…, H_n`. The function (HTM) is still valid for all types of robot manipulators without changes.

7 Conclusion

Optimization algorithms are proposed to find the solution for the inverse kinematic problem for robots of any type by optimizing the minimization objective function. The proposed optimization algorithm is dynamic differential annealed optimization, which is simple, fast, and uses low space of the memory of host machines or target devices. A practical example of 5DOF revolute joints manipulator, LabVolt 5150, was considered for the inverse problem. The described methodology is quite simple and can simplify the hard problem of inverse kinematic greatly and make it simple, straightforward operation. The proposed DDAO does not promise a perfect solution, and many other optimization algorithms should be tested to find a solution for the inverse problem, and that is proposed for future works.

Acknowledgments. The research was supported by the Hungarian National Research, Development and Innovation Office - NKFIH under the project number K 134358.

References

1. Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot modeling and control (2006)
2. Craig, J.J.: Introduction to robotics: mechanics and control, 3/E. Pearson Education India (2009)
3. Ghafil, H.N., Jármai, K.: Optimization for Robot Modelling with MATLAB. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40410-9
4. Lee, C.S.G.: A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots–College of Engineering. The University of Michigan, Ann Arbor (1983)
5. Ghafil, H.N., Jármai, K.: Dynamic differential annealed optimization: new metaheuristic optimization algorithm for engineering applications. Appl. Soft Comput. **93**, 106392 (2020)
6. Jiang, Y., Hu, T., Huang, C., Wu, X.: An improved particle swarm optimization algorithm. Appl. Math. Comput. **193**(1), 231–239 (2007)
7. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)
8. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
9. Al-Saedi, F.A.T., Mohammed, A.H.: Design and implementation of chess-playing robotic system. Int. J. Sci. Eng. Comput. Technol. **5**(5), 90 (2015)
10. Bi, Z.M., Gruver, W.A., Zhang, W.-J., Lang, S.Y.T.: Automated modeling of modular robotic configurations. Rob. Auton. Syst. **54**(12), 1015–1025 (2006)
11. Jármai, K., Bolló, B.: Vehicle and Automotive Engineering 2: Proceedings of the 2nd VAE2018, Miskolc, Hungary. Springer (2018). ISBN 978-3-319-75677-6
12. Jármai, K., Bolló, B.: Vehicle and Automotive Engineering: Proceedings of the JK2016, Miskolc, Hungary. Springer (2018). ISBN 978-3-319-51189-4