Implementation of modified FABRIK for robot manipulators

Ram Ananya Tenneti
Chaitanya Bharathi Institute of Technology
Hyderabad, Telangana, India
tramananya@gmail.com

Abhishek Sarkar
International Institute of Information Technology
Hyderabad, Telangana, India
abhishek.sarkar@iiit.ac.in

ABSTRACT

Forward And Backward Reaching Inverse Kinematics (FABRIK) is a heuristic iterative method for calculating inverse kinematics (IK). But, while solving the IK problem FABRIK method considers joint axis about any random orientation, and only the link lengths are kept fixed. A serial robot manipulator is always fixed with its geometrical configuration and it is represented with Denavit and Hartenberg (DH) parameters. DH parameter method is well structured for representation of robot manipulators but it is difficult to do IK with the transformation matrices calculated through DH method. We have used DH convention to define the geometry of a serial robot manipulator and devised a method to solve the IK problem with modified FABRIK method. We have simulated for both planar and non-planar manipulators and showed that the modified FABRIK can solve the IK for planar and non-planar manipulators (with only revolute joints).

• Computer systems organization~Robotic components

KEYWORDS

Inverse Kinematics, FABRIK, Serial Manipulator

1 INTRODUCTION

Inverse kinematics is the problem of determining the values of joint variables of a robot manipulator, given the end-effector position and orientation [1]. There are many methods that have been designed to solve the IK problem, based on different algorithms. These methods are evaluated on the basis of their speed, accuracy and the number of iterations taken (in case of heuristic methods).

One of the most commonly used approach is the inverse of the Jacobian matrix. There are many methods that have been proposed to calculate the Jacobian Inverse, which include Jacobian Transpose, Damped Least Square (DLS), Pseudoinverse DLS, etc. [2-5]. The Inverse Jacobian methods involve complex matrix calculations, which sometimes create singularity issues. Cyclic Coordinate Descent (CCD) algorithm [6-7] is an iterative method that has many advantages over the Inverse Jacobian methods for its simplicity, ease of implementation, speed and lack of singularities. However, CCD has some disadvantages as well. It can sometimes produce unrealistic solutions and when joint movements are constrained, and also fails to solve the IK problem. Triangulation method [8] is another algorithm that uses the cosine rule to calculate each joint angle. It reaches the target in single iteration and is as accurate as CCD. FABRIK [9-12] is a heuristic iterative method that outperforms many of the IK methods in terms of speed, time and computational complexity. FABRIK avoids the use of rotational angles or matrices and thus, it converges in few iterations and has low computational cost. But, the FABRIK method only incorporates link lengths and considers all joints as 3 DoF joints which can rotate about any 3D angle. The extension of FABRIK [10] covers the procedure to extend and/or adjust FABRIK with model constraints. Prismatic joints are included but it is irrelevant to the context of this paper as all the joints considered in this paper are revolute.

FABRIK algorithm involves a geometric approach to evaluate the joint positions. Each iteration consists of forward and backward stages. Each iteration starts with a forward stage. The calculation starts from the end effector and goes to the base joint, adjusting each joint along the way. Thereafter, it works backward, from the base joint to the end effector. The distance between the position of a target T and the position of end effector O_n for a n-link manipulator is considered as the error e. The FABRIK algorithm decreases the error after each iteration and hence the end effector moves nearer to the target position.

We are using modified FABRIK algorithm to solve the IK problem for robot manipulators for which the notations are used as DH parameter conventions. $Joint_i$ connects link[i] to link[i-1] for i=1,2,…,n, where the base link is referred as link[0] [Figure 1]. A frame $\{X_{i-1}, Y_{i-1}, Z_{i-1}\}$ is attached to each $joint_i$ of the robot manipulator. For an n link manipulator, the coordinates of origins of frames (w.r.t inertial frame) start from O_0,O_1, … O_n, where O_n corresponds to the position of end effector. A 3 link serial manipulator is shown in Figure 2(a) with link lengths a_1, a_2 and a_3. Figure 2 illustrates the implementation of unconstrained FABRIK [9] for a randomly chosen target T inside the workspace of the manipulator. At the beginning of forward stage, assume the new position of end effector at O_3'= T and the position of O_2' on the connecting line O_3'-O_2 and at a distance of a_3 from O_3' [Figure 2(a)]. Similarly the position of O_1' from O_2' is estimated [Figure 2(b)]. Finally the position of O_0' from O_1' is estimated [Figure 2(c)]. Backward stage begins by estimating the position of O_1'' from the base joint O_0 [Figure 2(d)]. The position of O_2'' is estimated [Figure

AIR '19, July 2–6, 2019, Chennai, India
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6650-2/19/07..$15.00
https://doi.org/10.1145/3352593.3352605

2(e)] and finally the position of $O_3^{"}$ is estimated [Figure 2(f)]. The procedure is then repeated, for as many iterations as needed, until the error e goes below a threshold value e_{limit}.

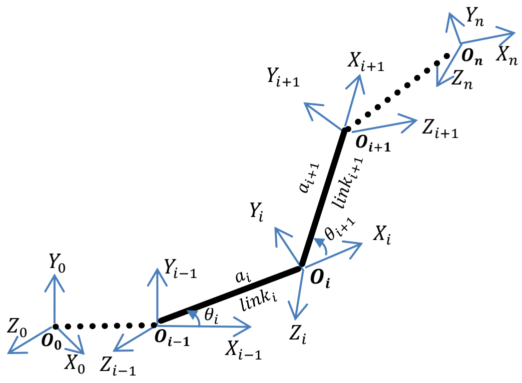

Figure 1: n-link manipulator.

2 FABRIK WITH DH PARAMETERS

In this paper, we have extended the use cases of FABRIK algorithm to solve the IK problem for robot manipulators, which are described using DH parameters. The unconstrained FABRIK algorithm can solve the IK problem for any manipulator, given that the joints are free to move in any direction and the target is within reach.

In case of robot manipulators, the orientation of frames, especially the Z-axis, plays a very important role in the DH notation. A serial manipulator generally has single DoF joints defined by Z_i axis which is perpendicular to the plane of movement of succeeding link[i+1]. For a planar manipulator, the axes of rotation of the joints are parallel, but for a non-planar manipulator, the movement of links are restricted to planes that are not parallel to each other. Apart from that, there might also be some link offset, depending on the DH parameters. Thus, both the position and orientation of frames attached to each link is very important and must be taken care of during the analysis of non-planar manipulators. The FABRIK algorithm does not consider all the parameters, which define the manipulator structure. In this paper, we have modified the FABRIK algorithm to take DH parameters as the inputs to solve the IK problem for both planar and non-planar manipulators. All the joints considered throughout this paper are revolute joints.

2.1 Modified constrained FABRIK for planar manipulator

Generally, for any robot manipulators the joints are constrained with joint limits $[\theta_{max}, \theta_{min}]$. When the joint movement is limited, the modified constrained FABRIK algorithm is applied. The black dotted semi-circle in Figure 3 represents the rotational plane of the successive link and joint limit is $[\theta_{max}, \theta_{min}]$. After finding the configuration using the unconstrained FABRIK, the algorithm checks whether the new succeeding joint position is reachable by the rotation of preceding joint within its rotational limit. If it is reachable, then the precedent joint rotates by required angle (as shown in Figure 3). In Figure 3(a) O_1' (as evaluated from unconstrained FABRIK) is reachable from $joint_3$ (i.e. O_2') during forward iteration. Hence, the value of O_1' remains the same. If the succeeding joint position is not reachable then the preceding joint rotates to the maximum possible angle either in clockwise (θ_{min}) or anti-clockwise direction (θ_{max}), whichever side the new succeeding joint position is present (as shown in Figure 4). This takes the succeeding joint ($joint_3$) to the nearest possible position from the new succeeding joint position ($O_2^{"}$), hence indirectly taking the further joints nearer to the target. Figure 4 represents a part of the forward stage, where the joint position $O_2^{"}$ (evaluated from unconstrained FABRIK) is not reachable from $joint_2$. Hence, the $joint_2$ bends the maximum, i.e. +90-degrees towards $O_2^{"}$.

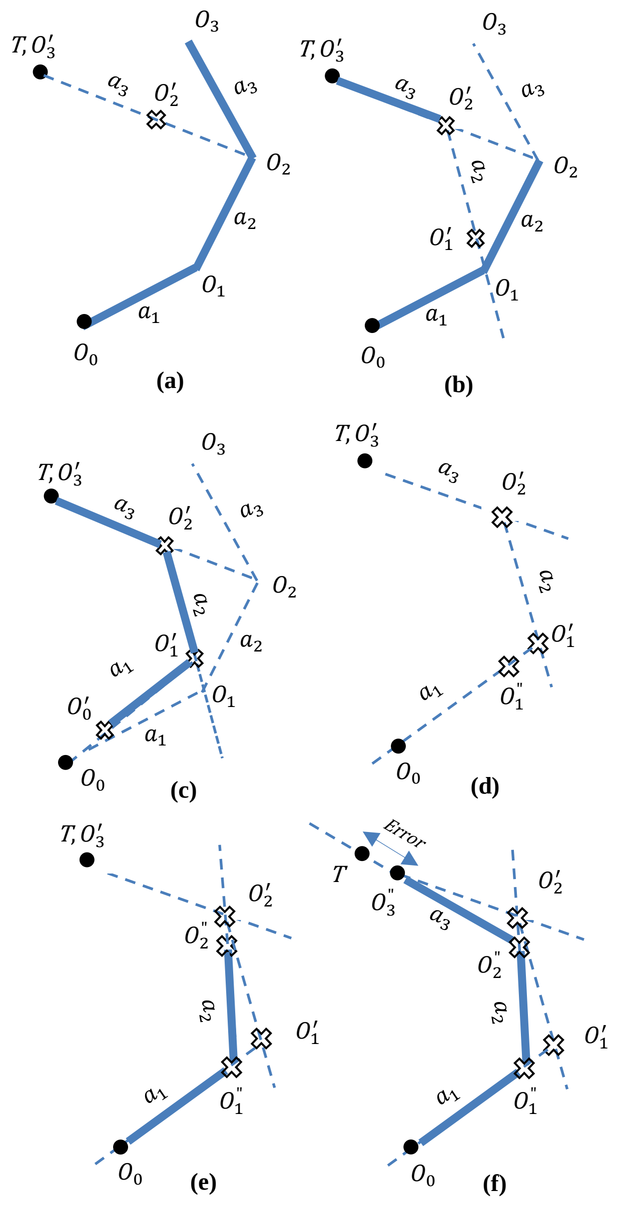

Figure 2: Implementation of unconstrained FABRIK for three joint manipulator.

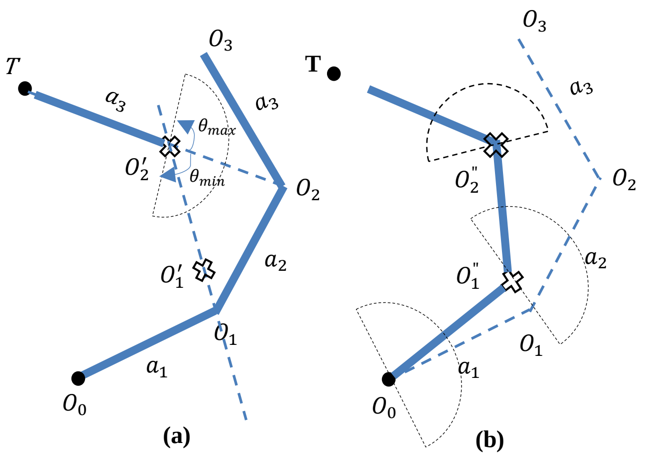

Figure 3: (a) Implementation of constrained FABRIK during (a) forward stage and (b) backward stage.

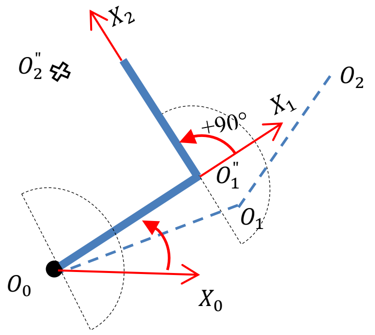

Figure 4: A situation during backward stage where the succeeding joint position is not reachable by preceding joint.

2.2 Modified Constrained FABRIK for Non-planar Manipulator

Working with a non-planar manipulator is more complex than a planar manipulator, mainly because of the additional DH parameters like 'link offset' and 'link twist'. These parameters play a crucial role in determining the orientation of frames attached to each link. In the case of a non-planar manipulator, the shape formed is a sector of circle, but the planes on which these sectors lie have different orientations, depending on the DH parameters and the joint axes. For a non-planar manipulator, to decrease the complexity, we implement the constrained FABRIK only during the backward iteration. This, in no way, affects the structure of the manipulator because all the joint constraints are considered during the backward stage, i.e. going from $\boldsymbol{O_0}$ to $\boldsymbol{O_n}$.

Thus, we focus more on the backward stage. Figure 5 represents the implementation of modified FABRIK algorithm for a non-planar manipulator. During the backward stage, first $O_1^{''}$ is evaluated using unconstrained FABRIK. Then, all the positions that O_1 can reach by rotating θ_1 are evaluated, represented by the arc of the dotted sector in Figure 5 (a). From the points of the arc, the point with the least distance to $O_1^{''}$ is considered as the new position of O_1 (we call it O_1^*), and the preceding joint is rotated to achieve that configuration. Hence, for non-planar manipulator we introduce O_i^* which is the position of $joint_{i+1}$ after backward iteration. Figure 5 (b) represents how the backward stage continues to move the end effector O_2 nearer to the target T.

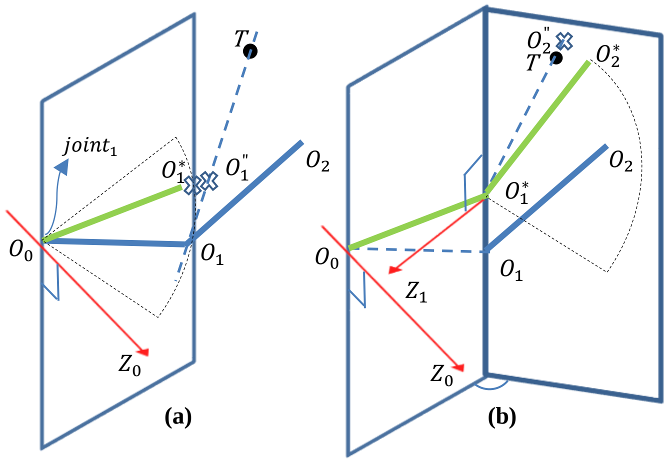

Figure 5: Implementation of modified FABRIK for non-planar manipulator.

3 SIMULATION AND RESULTS

MATLAB is used to implement the algorithm and solve the IK problem for serial planar 3, 6, 8 and 10 DoF manipulators for 1000 target positions respectively, and serial non-planar 2 DoF manipulator for 100 target positions. The algorithm is iterated repeatedly until the current error (e) becomes less than a threshold value e_{limit} or the number of iterations itr $\leq$ 200. A part of raw code from [13] was utilized to calculate the joint positions based on DH parameters for non-planar manipulators.

3.1 Planar Manipulator

The DH parameters of serial planar 3 DoF RRR manipulator are given in Table 1. The value chosen for e_{limit} is 0.1 units. Asymptotic nature of error minimization for modified FABRIK algorithm is shown in Figure 6 with 3 random goal positions T_1, T_2 and T_3 (mentioned in Figure 6). Out of the 1000 targets, the modified FABRIK solves the IK problem for 998 targets, with a mean final error of 0.0511 units. Figure 7 represents the total number of targets solved for different number of iterations and shows that majority of the targets took less than five iterations. The algorithm fails to solve for the two targets [(10.92,0,0) and (11.87,0,0)] because of the lock condition as explained in the next section.

Table 1: DH parameters of planar manipulator

Link	a_i	d_i	α_i	θ_i	Joint Limit
1	4	0	0	θ_1	[90,-90]
2	4	0	0	θ_2	[90,-90]
3	4	0	0	θ_3	[90,-90]

3.1.1 THE LOCK CONDITION

There are a few situations where the target is in reach, but because of the structure of algorithm, the algorithm is stuck after few iterations and program gives the same result continuously. This case happens specifically when the joint constraints are present. The condition to test lock situation at any iteration itr is when the previous error $e_{(itr-1)}$equals the current error $e_{(itr)}$ (i.e. e), which means the error becomes constant. Figure 8 shows the lock condition. Once the algorithm gets stuck, O_1' is always reachable by rotating $joint_1$. Hence, O_1'' is directly evaluated using the unconstrained version of FABRIK. It can clearly be seen that the target is not reachable due to the joint limit of $joint_2$ (+90 degrees). Because O_1'' is not close enough to reach the target, it leads to a lock condition. If the error repeats for two consecutive iterations, then θ_1 can be rotated by a small angle of Δθ in either clockwise or anti-clockwise direction based on the position of O_2'. Figure 9 shows how rotating θ_1 by a small angle solved the lock condition.

The modified algorithm solves 986 targets with a mean final error of 0.0606 units for 6 DoF manipulator, 994 targets with mean final error of 0.0403 units for 8 DoF manipulator and 1000 targets with mean final error of 0.0315 units for 10 DoF manipulator. Figure 10 shows the total number of targets solved for different number of iterations for 3 manipulators. It is clear from Figure 10 that majority (~70%) of the targets took less than 3 iterations.

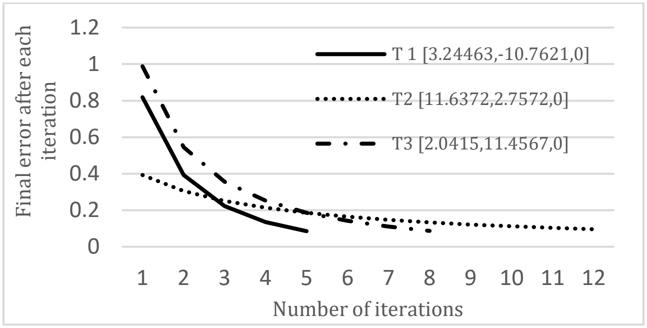

Figure 6: Error vs number of iterations for 3 targets

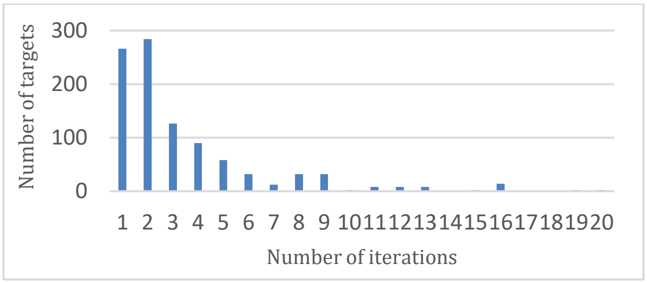

Figure 7: Plot of number of targets solved vs the number of iterations those targets took for 3 DoF planar manipulator.

3.2 Non-Planar Manipulator

The DH parameters of the manipulator are given in Table 2. The value e_{limit} used here is 0.15 units because of the increased number of failure cases when e_{limit} was set to 0.1 units. Figure 11 shows the total number of targets solved for different number of iterations. It can be seen that most of the targets (~70) took less than 10 iterations.

Table 2: DH parameters of non-planar manipulator

Link	a_i	d_i	α_i	θ_i	Joint Limit
1	8	0	pi/6	θ_1	[90,-90]
2	6	0	pi/2	θ_2	[90,-90]

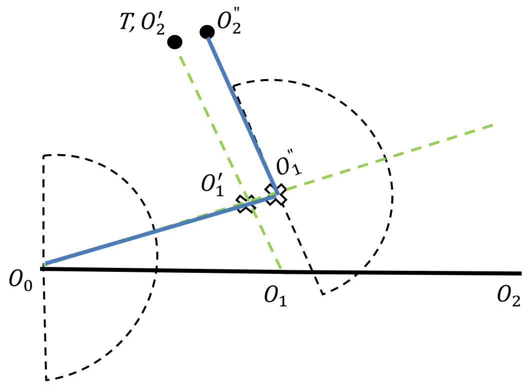

Figure 8: Represents the lock condition

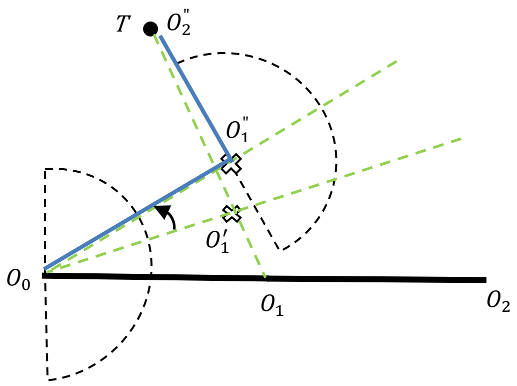

Figure 9: Solving the lock condition.

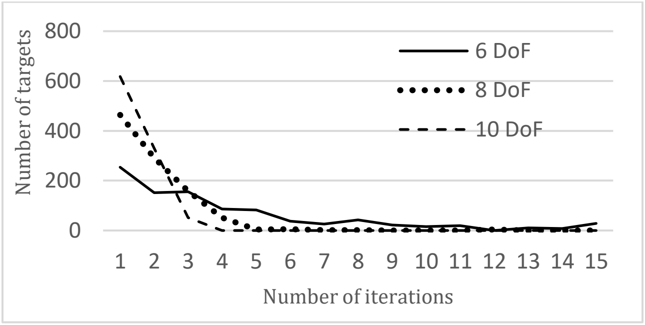

Figure 10: Plot of number of targets solved vs number of iterations those targets took for planar manipulators.

3.2.1 LOCK CONDITION

Similar to a planar manipulator, a non-planar manipulator also enters a lock situation (as shown in Figure 12). Figure 12 (a) represents the lock condition. Figure 12 (b) shows how the lock condition is solved by rotating $joint_1$ in CCW direction making O_1 move and hence making the end effector reach the target.

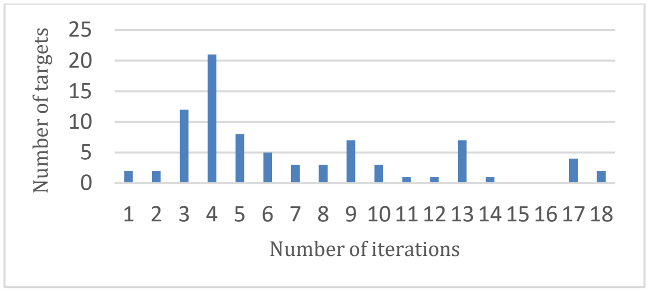

Figure 11: Plot of the number of targets solved for the number of iterations for non-planar manipulator.

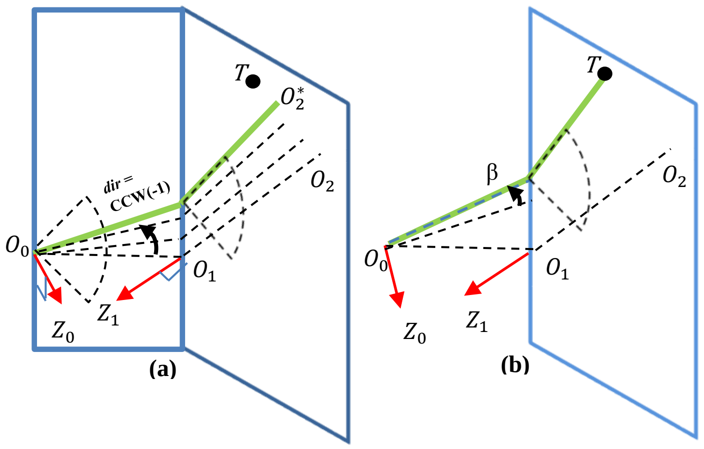

Figure 12: (a) Lock condition (b) Solving the lock condition

Out of the 100 targets, 94 targets were locked after an average of 2.575(~3) iterations. Once the lock condition was reached, the average error was 1.7462 units. Hence, to solve the issue, we manually rotated $joint_1$ by a very small angle. To do so, the direction, *dir* [CW(+1) or CCW(-1)] in which $joint_1$ was initially rotating is evaluated. The joint is then manually rotated by a small angle, say β (β=1°) in the same direction to see if the new joint position of $joint_1$ is sufficient to reach the target [Figure 12 (b)]. If not, then the process continues. The algorithm sometimes goes into the lock condition after the first iteration. This means that there are no data points to evaluate the direction *dir* in which $joint_1$ was rotating. To solve this kind of a situation, $joint_1$ is rotated in both CW and CCW direction by β (β=1°) and two positions of $O_1^{''}$ are evaluated. From the two positions of $O_1^{''}$, two positions of O_2^* are evaluated. For both the positions of O_2^* , the error *e* is calculated. The direction, which gives lesser error, is chosen as *dir*. Once the direction *dir* is evaluated, the algorithm resumes. It was observed that this approach is successful for 85 out of 94 targets that were stuck in the lock condition. Therefore, in total the modified FABRIK algorithm solved the IK problem for 91 targets positions with a mean final error of 0.0641 units.

The 9 target positions that the modified algorithm failed to reach was because of either of the two reasons: Firstly, the direction of rotation *dir* was correct but the amount of rotation of precedent joint by 1 degree was high enough for the algorithm to miss the target. Secondly, the direction of rotation (*dir*) of precedent joint ($joint_1$) evaluated was wrong. Making β = 0.1° solved the first issue. This modification solved the issue for 7 target positions. The error when the algorithm reached the lock condition was very small (less than 0.25 units), and hence whenever a target reaches the lock condition with such a small error, the amount of rotation of precedent joint can be decreased to overcome this issue. After solving this issue, the mean final error was 0.0653 units for 98 targets. The second case happened for 2 targets positions. By inverting the direction of rotation, the issue was solved.

We found that most of the targets were stuck in lock condition with the errors ranging from 0.17 units to 3.33 units. To solve these lock conditions, we varied the value of β for $joint_1$ from [0.1° to 2.5°]. The data was analyzed to find out an equation that best fits the behavior of expected curve in Figure 13. Logarithmic curve, second order and third order polynomial equations matched the most (as seen in Figure 13). The least square method reflects that logarithmic equation fits the best (as seen in Table 3).

Table 3: square errors for different order polynomial equations

	Log	Order 2	Order 3
Error	0.524922	13.65866	4.901435

Figure 14 shows the number of iterations taken to solve the targets that were stuck in lock condition with different errors for 3 different β values. For targets with small errors (<0.25 units), a small value of β (0.5°) is suitable and with higher value of β, the target is missed. A smaller value of β solves the lock, for targets stuck with larger errors but takes many iterations as seen in Figure 14. Thus, it is suitable to increase β when the error is large to save the computation time. The complete flowchart of modified FABRIK algorithm for non-planar manipulator is shown in Figure 15.

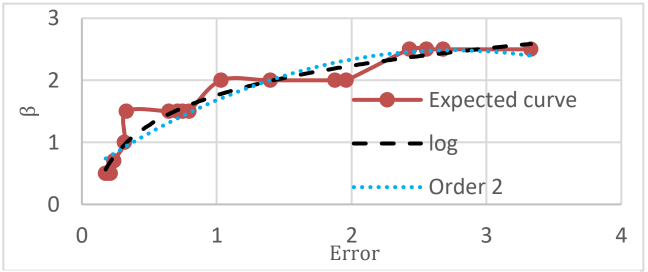

Figure 13: Represents the maximum value of β that solved the lock condition for targets with different errors

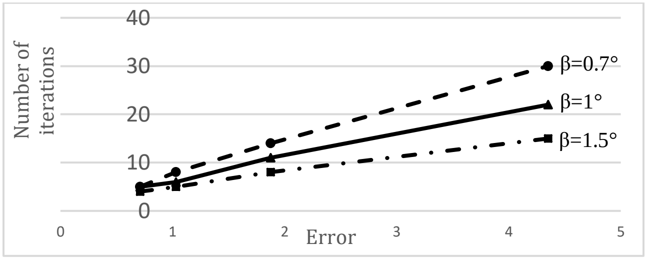

Figure 14: Plot of number of iterations taken to solve the targets vs the error for which they were stuck for 3 β values.

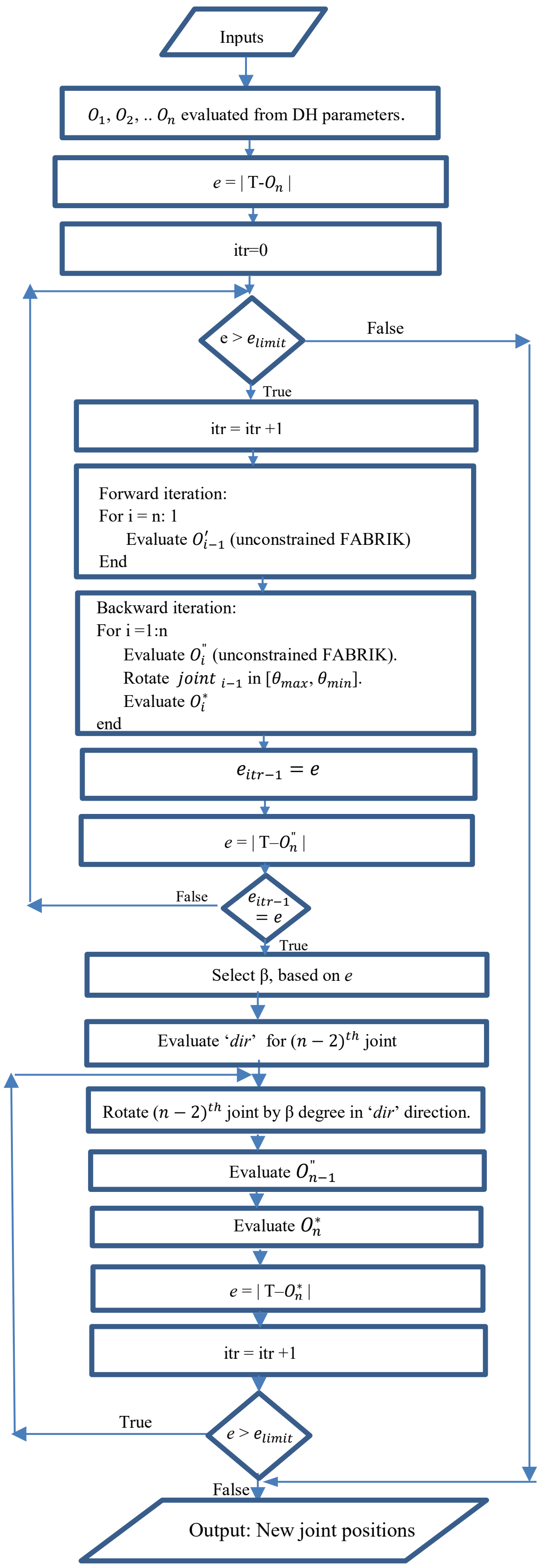

Figure 15: Flowchart of the algorithm for non-planar manipulator

4 CONCLUSION

This paper presents the possibility of implementing FABRIK algorithm to solve the IK problem for serial planar and non-planar robot manipulators, which are defined using DH parameters. The modified algorithm considers all the DH parameters, hence maintaining the structure of the manipulator. The analysis was restricted to serial planar and non-planar manipulators with revolute joints. As the analysis of multiple end effectors is not possible with DH parameters, the focus was on manipulators with single end effector. The modified FABRIK algorithm has solved the IK problem for planar manipulators with a success rate of 99.8% and a mean final error of 0.0510 units and with a success rate of 98% for non-planar manipulators and a mean final error of 0.0650 units. However, in this paper, the analysis of non-planar manipulator was limited to 2-link manipulator. In future, the implementation can be extended to higher DoF link manipulators. This paper does not consider prismatic joints, which will lead to variations only in link lengths and offsets.

REFERENCES

[1] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. 2018. Robot Dynamics and Control (2nd edition). Chapter 1-5.

[2] Wolovich, William A., and H. Elliott (1984). A computational technique for inverse kinematics. The 23rd IEEE Conference on Decision and Control. IEEE.

[3] Buss, Samuel R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation 17.1-19: 16.

[4] Nakamura, Yoshihiko, and Hanafusa Hideo (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. Journal of dynamic systems, measurement, and control 108.3: 163-171.

[5] Buss, Samuel R., and Jin-Su Kim (2005). Selectively damped least squares for inverse kinematics. Journal of Graphics tools10.3: 37-49.

[6] Canutescu, Adrian A., and Roland L. Dunbrack Jr. (2003). Cyclic coordinate descent: A robotics algorithm for protein loop closure. Protein science 12.5: 963-972.

[7] Wang, L-CT, and Chih-Cheng Chen (1991). A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. IEEE Transactions on Robotics and Automation 7.4: 489-499.

[8] R. Mü ller-Cajar, R. Mukundan (2007). Triangulation: a new algorithm for inverse kinematics, in: Proc. of the Image and Vision Computing New Zealand, New Zealand, December 2007, pp. 181–186.

[9] Andreas Aristidou and Joan Lasenby (2011). FABRIK: A fast, iterative solver for the Inverse Kinematics problem. Graphical Models 73.5: 243-260.

[10] Andreas Aristidou, Yiorgos Chrysanthou, and Joan Lasenby (2016). Extending FABRIK with model constraints. Computer Animation and Virtual Worlds 27.1: 35-57.

[11] Andreas Aristidou and Joan Lasenby (2009). Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. University of Cambridge, Department of Engineering.

[12] Poddighe, Renzo (2013). Comparing FABRIK and neural networks to traditional methods in solving Inverse Kinematics.

[13] Auralius Manurung, Implementation of the Denavit-Hartenberg (DH) parameters in MATLAB, https://github.com/auralius/my-matlab-robotics-toolbox.git.