Adaptive Neuro-Fuzzy Inference System logic to improve on Inverse Kinematic computations for planar robot arm

Jidnyesha Vijaykumar Patil
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
jpatil@wpi.edu

Jonathan Nieves Melendez
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
jnieves2@wpi.edu

Lucas Buermeyer
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
ldbuermeyer@wpi.edu

Gaurav Bhosale
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
gbhosale@wpi.edu

***Abstract*—This document presents an analysis of the performance difference between a prediction control algorithm based on traditional inverse kinematics(IK) and one based on adaptive neuro-fuzzy inference system(ANFIS). The study was done using a MATLAB environment to simulate randomized projectiles tossed at a 2-link planar arm. Both algorithms where tested at different projectile flight times that generated randomized deviation from original predicted trajectory. ANFIS model had trouble accounting for configurations that generated singularities in the robot. Processing time for ANFIS was much shorter than IK at the cost of a large decrease in accuracy. Low resolution systems would benefit the most from using ANFIS algorithm.**

***Index Terms*—ANFIS, inverse, kinematics, robotics, prediction, dynamics, controls, fuzzy, neural networks**

I. Introduction

Computing robot motion is a core aspect of any robotic system. As technology develops, more complex problems are able to be solved with robotic solutions. These advanced objectives often require more complex kinematic models, higher precision motion, and faster processing time, which increase the computational strain for modern robotic systems.

A key aspect of a robot is that it must be able to sense its environment and adapt to the environment. In many cases, the robot must have a short response time to a rapidly changing environment in order to complete its objective. If the response time is too long,there are several approaches that may be taken to remedy the problem. If possible, the computer processor can be upgraded such that the same computations are performed at a faster rate, thus decreasing the response time. But faster computers are exponentially more expensive and there are physical limitations to the rate of computation, as described by Zhirnov in response to Moore's Law [1], so alternative optimization approaches must be taken to further reduce computation time. Such optimization include non-linear estimations and function estimates [2].

Calculation of forward and inverse kinematics is a fundamental concept when dealing with manipulators. Forward kinematics determines the position of the end-effector given the position of each joint. Conversely, inverse kinematics uses the desired position of the end-effector to calculate the positions of each joint.

The traditional method for solving the inverse kinematics involves non-linear mathematical equations [3]. Soft computing methods which rely on approximating the solution rather than calculating the exact solution are proven to perform better in real-time environments [4]. The use of neural networks and fuzzy logic separately has been discussed and demonstrated to work with few pitfalls. This paper focuses on the approach based on the work done by Vasile Duka [5]. In this paper, he proposes that an Adaptive Neuro-Fuzzy Inference System model is effective for solving configurations of 2 DOF and 3 DOF systems, albeit with a degree of error that is attributed to the soft computing method. Fuzzy models work well with uncertain systems but need standardization and effective methods to tune the membership functions being used.As explained in [6], the Adaptive-Network-based Fuzzy Inference System used adaptive networks to tune the membership functions in order to minimize the error in models. Neural networks are not easy to interpret or modify, while Fuzzy logic is explicit and easy to understand. In ANFIS, fuzzy logic and neural networks are applied together to help overcome the shortcoming of each approach. A general overview of how the ANFIS network corresponding to the Fuzzy system is shown in Figure 1.

This paper focuses on how the performance of ANFIS models compares to the traditional, analytical inverse kinematics approach in a simulated, realistic and real-time environment. In this research, the considered scenario is a robotic arm attempting to catch a ball by intersecting the ball's trajectory. The robot predicts the ball's trajectory using a simplified

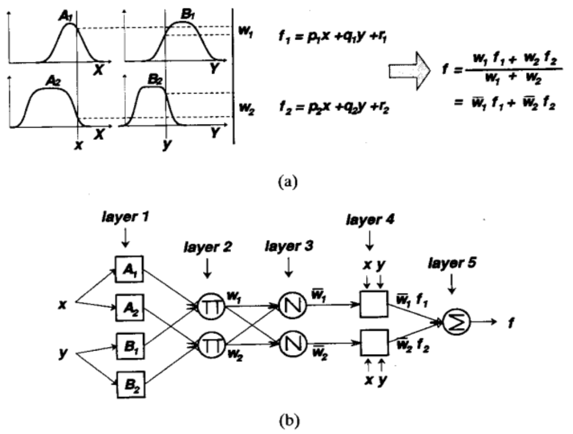

Fig. 1. (a) Fuzzy reasoning. (b) Equivalent ANFIS

model with a known deviation from the actual, and more realistic, trajectory. As the ball approaches the robot, the predicted intersection point will become more accurate. The robot will approach the ball using both an analytical inverse kinematics approach and ANFIS approach to determine the optimal joint angles.

II. Significance and Innovation

Fast processing of changing conditions is important for managing dynamic and real-time environments. Modern robotics require systems that can figure out solutions for increasingly complex problems. Robots are more frequently being used in high-risk environments, such as intercepting space debris near the international space station.

Solving for inverse kinematic models is a fundamental step in any robotic application, especially those with multiple degrees of freedom. This paper compares the performance of traditional inverse kinematics approach with the Adaptive Neuro-Fuzzy system as a way to reduce processing time and improve reaction time in robot manipulators.

III. Project Scope

The objective of this study to analyze the performance of the ANFIS system and traditional inverse kinematics approach to determine how the performance of each approach compares to the other. The simulated use case is a robot arm catching a ball with an unpredictable trajectory and varying flight time.

The success of an attempted catch will only consider one distance between origin of the ball and robot's end effector. The flight times will vary between 0.1s and 0.9s to reduce processing requirements and maintain realistic conditions. Different lengths of arms or dynamic lengths are not considered in this study, nor is a mobile robot base. The simulated robot has two degrees of freedom and operates in the YZ plane. It is assumed that the position and velocity of the ball can be perfectly measured at any point in time on an external sensor (as is the robot joint controller), such that those calculations do not influence the calculation time measurements of either algorithm.

IV. Methodology

This section will look at how the simulation for the test is setup and how each part of the program contributes to the study.

A. Manipulator Design

The robot design for this study is a 2-link planar arm where each link is 0.9144m long. The end-effector is not modeled but is assumed to be a perfect catcher, meaning that if the projectile is within a certain tolerance, it will be caught.

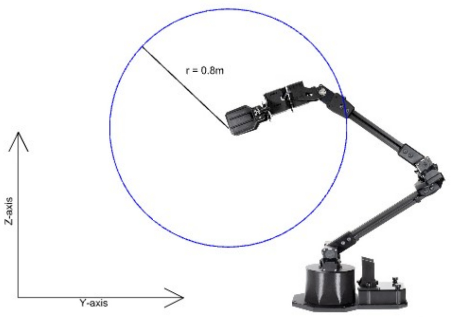

Fig. 2. Established work-space for robot during the test.

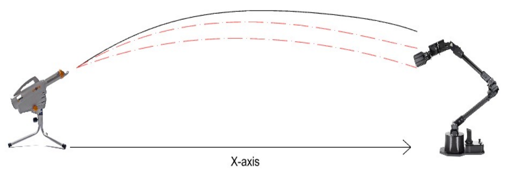

Fig. 3. Example on initial predicted trajectory changing as the ball travels. Dashed lines represent previous predictions.

B. Trajectory of Projectile

The trajectory of the projectile was modeled assuming the influence of two forces: gravity and air resistance. Gravity was assumed to have a constant acceleration of 9.81m/s. Air resistance was calculated using equation 1. By using given initial conditions, the trajectory of the ball was able to be modeled in a realistic simulation.

$$F_{AR} = 1/2 * \rho_{air} * V_{relative}^2 * c_d * A \tag{1}$$

In order to calculate the trajectory, the position of the ball was incrementally updated using a fixed time interval of 0.0005s. At every time interval, the new position and velocity were calculated. The first step in these calculations was to determine the forces acting on the ball. Once the forces were summed together to find the net force, the acceleration of the ball was easily determined by dividing the net force by the mass of the ball. As a result, the velocity could be easily incremented by adding the acceleration multiplied by the time interval to the previous velocity. In a similar fashion, the position was updated by increasing the position by the product of the velocity and time interval. This process was repeated until the ball exited the robotic arm's workspace, as denoted by the ball's x position being less than or equal to 0.

The process of choosing initial conditions proved more difficult than anticipated. The primary goal of the initial condition generation process was to determine a wind velocity, initial ball position, and initial ball velocity to create a trajectory with the desired independent variables: time of flight and deviation from the first predicted position to the actual position where the ball intersected the robot's workspace. In order to achieve this, the first approach chose arbitrary ball parameters and changed them in a direction to minimize the error between the goal and actual trajectory. While this worked well for the x initial velocity, the y velocity was dependent on the z velocity, and the z velocity was less straightforward. The x velocity was simple because the ball was always moving forward in the positive x direction, and the only factor that determines initial x velocity is time of flight. The z-velocity, however, is derived from the deviation from the initially predicted position and the actual final position of the ball. Because the ball can either travel only upward, only downward, or change z directions mid-flight, the simple minimize error approach was insufficient at finding the optimal result. Due to these directional opportunities, the error could reach a local minimum and result in error without having found the optimal parameters.

The next approach was typically able to solve this issue by solving for the predicted error over a given range of parameters and using MATLAB to find the local minimum errors. Each one of the errors was then iterated over again, with a smaller time step, to more efficiently focus on the relevant parameters and increase solution accuracy. Although this provided sufficient data, the processing time was too long, nearing 3 minutes per data point. The next, and final, approach was simpler, aiming to only control the x velocity, leave y and z velocities as a random factor. This approach, while more error-prone, was ultimately chosen due to it's high speed per iteration, under 1 ms per trajectory generation. The final step in generating the trajectory was to adjust the initial position of the ball such that the first predicted position was 0.4m above the base of the arm. This standardized the trajectory of the ball such that the ball's deviation would always be from the same spot, with the hope of producing more comparable data.

C. Data Collection

The first step in the data collection process was to initialize the ball and robot. For each attempt to catch the ball, the a new ball-catching robot was created with the same initial conditions. Additionally, the initial conditions of the ball and trajectory (initial ball velocity, initial ball position, and wind velocity) were randomly generated to meet the desired specifications: deviation from parabolic path and ball flight time. Using these conditions, the full trajectory of the ball was calculated. Starting from the instant of the ball being thrown, the robot controller would query the ball trajectory object to get a predicted position of where the ball would enter the robot's workspace. This prediction neglected air resistance, resulting in a simple parabola. Using one of the prediction models (ANFIS or analytical inverse kinematics), the desired joint angles to catch the ball were calculated and the robot moved toward the desired pose, while maintaining a maximum allowable joint velocity. This process was repeated for one algorithm until the ball passed through the workspace, then repeated with the same ball trajectory with the other algorithm. The ball was determined to have been caught if the distance between the end effector and ball was below 0.1m. This distance was chosen as it is approximately the radius of the ball. In order to adequately measure the impact of the ball's deviation from the initial prediction position and the time of flight, each algorithm attempted to catch the ball 15 times per flight time, with the flight time varying from 0.1s to 0.9s in 0.005 second intervals, thus producing 2715 data points per algorithm.

D. Prediction Models

Both models were generated using MATLAB and require the Robotic Systems Toolbox(RST) and Fuzzy Logic Toolbox(FLT).

1) Analytical Inverse Kinematics Model: The IK model follows the standard RST function that uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) gradient projection algorithm. As per the documentation [7], this method is a "quasi Newton method that uses the gradients of the cost function from past iterations to generate approximate second-derivative information." and ensures that the direction of each search is in the direction of the target. This method is considered more "robust" than the alternative [7].

2) Adaptive Neuro-Fuzzy Inverse System Model: A simple ANFIS architecture is represented in the Figure 1-b. As shown, there are 5 layers in this architecture. The first layer performs fuzzification according to the Takagi-Sugeno if-then rules based on membership functions. There can be triangular, trapezoidal, Gaussian or bell-shaped membership functions having variable parameter which will be trained by the neural network. The second layer is the Product Layer which performs the multiplication or the logical 'AND' operation on the weights. The Product layer generates the 'firing strength' for each each rule. The third layer is the normalization layer that normalizes the weights received from the previous layer. Then, the de-fuzzification layer calculates the weight corresponding to each rule based on the normalised weights and output functions. The final layer gives a sum of all incoming values as a overall output. The learning phase allows changing of the parameter if represented as squares in the Fig 4. A variety of training methods can be used with ANFIS like backpropogation gradient descent and least square estimates.

In the scenario considered, two ANFIS networks are used: one for each joint angle. The MATLAB function anfis is used

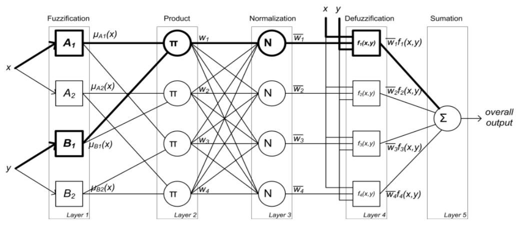

Fig. 4. ANFIS network for two input system.

with a set of training data to generate a model. For two joint angles, two separate models are created. These models are used directly to evaluate joint angle for a given end-effector position.

E. Parameters Measured

Mean target error: For each flight time of the ball, 10 attempts will be made to catch the ball with random deviation. The errors for each run will be averaged out for that run.

Mean interval time: For each attempt to catch the ball, time taken by the algorithm to complete calculations for each time step will be recorded and averaged out for that run.

Catch success: If the distance of the ball to the end-effector for the attempt is equal or less than 0.1m it is considered caught by the robot. Distances between 0.1m and 0.5m are near misses. Distances higher than 0.5m are complete misses.

V. Results

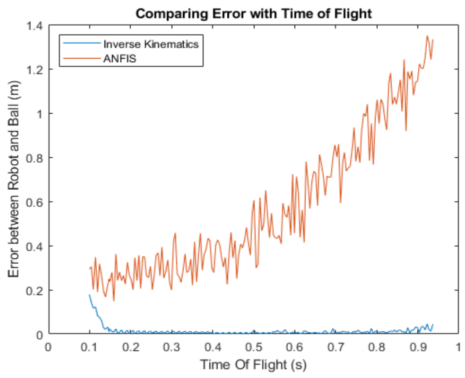

Fig. 5. Comparison of distance from ball to end-effector between traditional inverse kinematics and Adaptive Neuro-Fuzzy Inverse System Model with respect to time of flight

In figure 5, the difference between IK and ANFIS models can be observed as the time of flight changes. With very short flight times (less than 0.15s), both models struggle to catch the ball (when error between the end effector and ball is less than 0.1 m). Using the analytical inverse kinematics approach, the error quickly decreases from 0.1 to 0.15 s time of flights, then the error stabilizes such that the ball is consistently caught for the remaining time intervals. In contrast, the error resulting from the ANFIS algorithm tends to remain between 0.2m and 0.4m until the time of flight surpasses 0.4s, at which point the error appears to linearly increase with the time of flight.

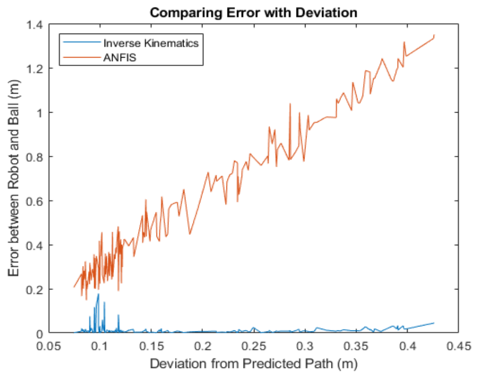

Fig. 6. Comparison of distance from ball to end-effector between traditional inverse kinematics and Adaptive Neuro-Fuzzy Inverse System Model with respect to deviation from the initially predicted path

Figure 6 describes the relationship between the ball's deviation and the robot's error. The error resulting from the ANFIS approach appears to linearly increase with the deviation, while the analytical inverse kinematics approach results in an error spike when the deviation is around 0.1m, and the ball is caught over the rest of the data.

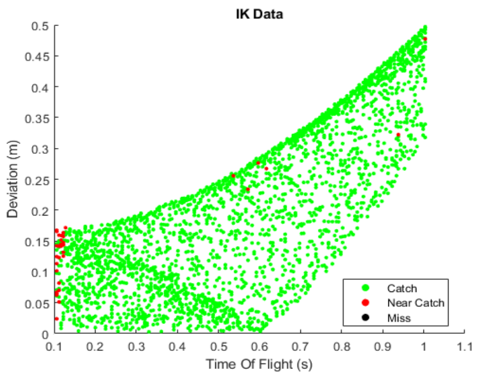

Fig. 7. All catch attempts for IK algorithm. Distance $<$ 0.1 = successful catch.

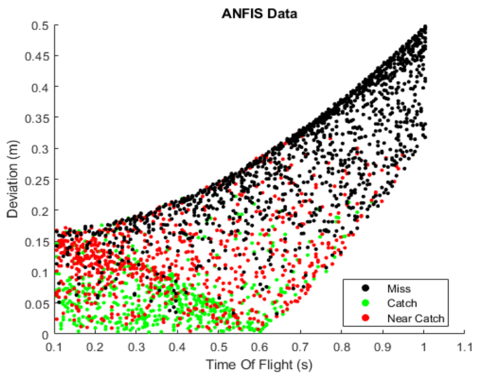

Fig. 8. All catch attempts for ANFIS algorithm. Distance $<$ 0.1 = successful catch.

Using the established ”successful catch” conditions, each attempt to catch the ball was determined to be either a catch, near catch, or miss. The miss condition is met when the error between the end effector and ball is greater than 0.5m, implying that the algorithm was unable to successfully determine what joint positions would result in a catch. Figures 7 and 8 describe the properties of the trajectory which are most influential in determining the success rate of each algorithm. It can be seen that the primary cluster of near-catches using analytical inverse kinematics occurs when the time of flight is very short, approximately 0.1s. Additionally, there are no complete misses present in this data set, and few near misses outside the short time of flight cluster.

In contrast, figure 8 shows how the ANFIS algorithm resulted in numerous complete misses (failure of the algorithm) and the majority of the catches occur when both the time of flight is short, and the ball deviation is minimal. As the deviation and time of flight increase, the ANFIS algorithm becomes less accurate, replacing catches with near catches, and near catches with misses.

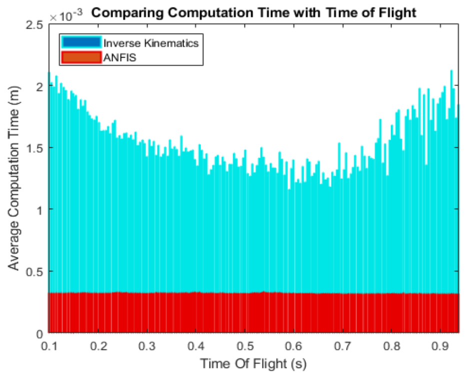

Fig. 9. Mean iteration time for each algorithm based on the flight time of the ball

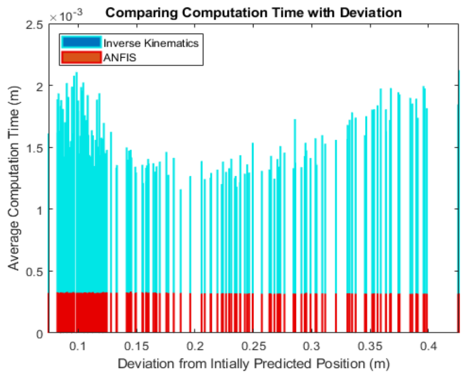

Fig. 10. Mean iteration time for each algorithm based on the deviation in the ball’s trajectory

Figures 9 and 10 relate the computation time (average time to calculate set of joint positions to reach target) of the inverse kinematics and ANFIS algorithms to that of the ball’s time of flight and deviation, respectively. Across all data points, the ANFIS algorithm maintains a consistent computation time of approximately 0.21s, while the computation time of the analytical inverse kinematics approach varies with both time of flight and ball deviation. As the time of flight increases, the IK computation time tends to decrease until 0.6s, at which point the computation time increases again. Additionally, as the time of flight increases, the variation in the computation time also increases. A similar relationship is present with respect to the ball’s deviation, but is less defined.

VI. Discussion

Comparing the accuracy of ANFIS and analytical inverse kinematics as implemented in this paper, figures 7 and 8 display how analytical kinematics has a much higher rate of catching the ball between both algorithms. This is demonstrated through the overwhelming percentage of catches (98.75%), as compared to the few catches using ANFIS (20.75%). The largest grouping of misses for analytical inverse kinematics occurred when flight time was low which was likely due to a the arm not being fast enough to account for the shorter available time for movement. As flight time is reduced the system quickly approaches its physical limitations and can be become impossible to account for regardless of algorithm.

ANFIS results, as seen in figure 8, do not present a clear grouping for successful catches. The majority of identifiable catches are located where the conditions for deviation are less the 0.1m. However, this condition also produced many near catches and several complete misses. As deviation increases from 0.1m to 0.2m, the rate of successful catches decreases, producing more near misses.The behavior shown is that, the rate of failed attempts for the ANFIS model increased with the increase in deviation, much more than for the analytical inverse kinematics.

One of the initial predictions was that longer flight times would lead to better results, as the algorithms had more time to calculate and adjust to the trajectory. The results for the simulations do not favor this prediction because, for both cases, longer flight times did not generate more successful catches. The cause of this behavior is that longer flight times are linked to larger deviation which has a more direct impact on success rate of each algorithm. This is clearly demonstrated in figures 5 and 6. By comparing these two figures, it can be easily seen that the ANFIS error has a direct, linear relationship with the ball’s variation, while the relationship to the flight time is more complex. The time of flight and deviation are related, as the longer the ball flies, the longer it experiences air resistance, resulting in a greater deviation. Thus, longer flight times are linked to larger deviations, and because air resistance does not have a direct correlation with flight time (force is proportional to velocity squared and the force is a multiple of the acceleration, which is the second derivative of position), the time of flight does not have a linear relationship to the ball’s deviation. This explains why the same general trends are shown between the time of flight figures and trajectory deviation figures while the shape of the data varies significantly. Analytical inverse kinematics, on the other hand, appears to be more dependent on time of flight than deviation, as the only significant spike in the error occurs when the time of flight is less than 0.15s. While this spike does appear in figure 6, it occurs at a deviation of 0.1m (which does not have any significance like the boundary does in figure 5). This spike at 0.1m is not occurring at a boundary because the error below 0.1m is also near zero. Because of the linear relationship between ANFIS accuracy and ball deviation and the strong relationship between analytical inverse kinematics with time of flight, the ideal use case for ANFIS over IK would occur when the time of flight is short and deviation is minimal. Over the rest of the data collected, IK appears more likely to catch the ball.

Assuming that the low ANFIS catch rate is a result of the increased deviation, the next question is why the deviation is causing accuracy to decline. By looking at the visualization of the robot as it attempts to catch the ball, it can be seen that the arm approaches the ball's predicted position, but is unable to reach the ball given any amount of time. This indicates that deviation is not the only factor that affect the success rate of the algorithm. Based on that information it is likely that the method used to train the ANFIS algorithm had issues with specific value ranges, as implemented here, rather than a fundamental issue with the ANFIS approach.

The model appears to suffer most around singularities of the arm, near the center of the arm and around the edges of the workspace. As the arm gets closer to a singularity, the joint positions must be more precise to maintain an equal level of task space precision, but this increase in joint space precision comes at the cost of increased computation time, which is limited to approximately 0.4s. As deviation increases, the final position of the ball gets farther from its initial position and closer to the robot base or to one of the workspace edges. Thus, higher deviations are more likely to position the ball close to the robots base or workspace edge, which greatly reduces the ANFIS model's accuracy.

The patterns of the average computation time also have interesting implications. The impressively stable computation times of the ANFIS implementation imply that a hard time limit was imposed on the algorithm, and the algorithm's accuracy could potentially be improved by increasing this time cap. Comparing average computational time in figures 9 and 10 its apparent that ANFIS has an advantage in the time to compute a solution. ANFIS uses a look-up table method that can provide results quickly with little change in processing time. There is a potential to increase viability of the algorithm by increasing the fines of the network and making more data points available for it to look up. This would increase processing time but can be balanced to provide higher accuracy for the system.

In contrast the analytical inverse kinematics algorithm has a more variability in its processing times. The initially high IK calculation time is likely due to the fact that with lower flight times, a greater percentage of the flight is devoted to moving the arm to a new position. This is because the predicted position of the ball is rapidly changing (predicted position stabilizes over time as the ball velocity approaches the wind velocity) and the arm is farther from its goal. The current implementation of analytical inverse kinematics utilizes the robot's current pose to determine the next pose. When these poses are similar, such as after the arm has a reached the desired position and the prediction has stabilized, the computation time is very short.

During the short flight times, a larger percentage of the flight is devoted to the arm's rapid motions, which result in the next predicted pose being more different than the current pose, thus increasing average computation time. In contrast with longer flight times, around 0.6s, the arm is given time to converge at the predicted position at which point processing time declines and decreases the average computation time. When the time of flight and deviation get too large, however, the ball approaches the edges of the workspace, which contain singularities, it is likely that the IK algorithm is slower to solve around singularities, thus increasing the computation time once again.

VII. Conclusion

Results using ANFIS to calculate the movements of the robot produced less accurate results than using IK. The benefit of ANFIS comes from its high speed of generating solutions. The lower processing time allows the robot to move along more intermediate way points, improving accuracy. The processing load from ANFIS is lower and more consistent than IK, making it possible to reduce the cost of the processor. Inverse kinematics provides higher accuracy at the cost computational time. The results of this paper suggest that the ANFIS approach is not recommended for high-precision tasks, however, it can be effective for faster operations with lower precision requirements.

Future research could attempt to re-implement an ANFIS algorithm such that it provides more accurate solutions near singularity positions and in environments with high deviation. Additionally, analytical inverse kinematics may have more frequently caught the ball in this case due to the simplicity of the robotic arm. Given another arm, one with more than 2 degrees of freedom, the analytical inverse kinematics algorithm require exponentially longer computation time while the ANFIS approach could maintains a similar time, demonstrating how the choice of algorithm may depend on the complexity of the arm. There are other aspects of ANFIS that can be examined in futures studies, such as the effect of retraining the algorithm, cost/benefit of increasing node fields for each variable or adjusting tolerance characteristics of defuzzyfication layer.

References

[1] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby and G. I. Bourianoff, ”Limits to binary logic switch scaling - a gedanken model,” in Proceedings of the IEEE, vol. 91, no. 11, pp. 1934-1939, Nov. 2003, doi: 10.1109/JPROC.2003.818324.

[2] Y. Favennec, P. Le Masson, Y. Jarny. ”Lecture 7: Optimization methods for non linear estimation or function estimation”. LTN – UMR CNRS 6607 – Polytetch’ Nantes – 44306 Nantes – France

[3] Adrian-Vasile Duka,ANFIS Based Solution to the Inverse Kinematics of a 3DOF Planar Manipulator,Procedia Technology,Volume 19,2015,Pages 526-533,ISSN 2212-0173, https://doi.org/10.1016/j.protcy.2015.02.075. (https://www.sciencedirect.com/science/article/pii/S2212017315000766)

[4] Sreenivas Tejomurtula, Subhash Kak, Inverse Kinematics in robotics using Neural Networks. Information Sciences 116, 1999, pp. 147-164

[5] Kucuk S, Bingul Z. Robot Kinematics: Forward and Inverse Kinematics. Industrial Robotics: Theory, Modelling and Control, Sam Cubero(Ed.),InTech.,2006

[6] J. -. R. Jang, ”ANFIS: adaptive-network-based fuzzy inference system,” in IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 3, pp. 665-685, May-June 1993, doi: 10.1109/21.256541.

[7] MATLAB 9.10.0.1739362 (R2021a) Update 5. Natick, Massachusetts: The MathWorks Inc.,2021