FUZZY LOGIC BASED ROBOTIC CONTROLLER

F. ATTIA*, M. UPADHYAYA**
* Associate Professor, ** Research Associate

University of Houston, College of Technology
4800 Calhoun Road, Houston, Texas 77204 - 4083

Abstract

Existing Proportional-Integral-Derivative (PID) robotic controllers rely on an inverse kinematic model to convert user-specified cartesian trajectory coordinates to joint variables. These joints experience friction, stiction and gear backlash effects. Due to lack of proper linearization of these effects, modern control theory based on state space methods cannot provide adequate control for robotic systems. In presence of loads, the dynamic behavior of robotic systems is complex and nonlinear, especially where mathematical modeling is evaluated for real-time operations. Fuzzy Logic Control is a fast emerging alternative to conventional control systems in situations where it may not be feasible to formulate an analytical model of the complex system.

Fuzzy logic techniques track a user-defined trajectory without having the host computer to explicitly solve the nonlinear inverse kinematic equations. The goal is to provide a rule-based approach, which is closer to human reasoning. The approach used expresses endpoint error, location of manipulator joints, and proximity to obstacles as fuzzy variables. The resulting decisions are based upon linguistic and non-numerical information.

This paper presents a solution to the conventional robot controller which is independent of computationally intensive kinematic equations. Computer simulation results of this approach as obtained from software implementation are also discussed.

Introduction

Fuzzy set theory was developed in 1965 by Zadeh [1], and permits the treatment of vague, uncertain, imprecise, and ill-defined knowledge and concepts in an exact mathematical way. This theory addresses the uncertainty that results from boundary conditions as opposed to Probability theory of mathematics. It allows one to express the operational and control laws of a system, linguistically in words such as "too cold", "cool", "warm", "very hot" etc., which is a generalization of the classical set theory. Fuzzy arithmetic differs from classical Boolean arithmetic as it allows a variable to be partially included in any given set as opposed to being fully included or excluded in Boolean algebra. This is known as Crisp set theory. Fuzzy logic is multivalued and varies from maximum to minimum as a function of the input. Fuzzy sets are subjective as compared to standard crisp sets which are objective and are viewed as exceptional cases of fuzzy sets [2].

Fuzzy controllers offer some practical advantages over conventional controllers like increased robustness in spite of high ambient noise levels or sensor failures, an ability to handle nonlinearities without control system degradation, and easy formulation of fuzzy rules. This makes the understanding, modification and maintenance of a fuzzy logic based controller much easier than is possible with conventional controllers. This method can be used when a specific rule base or expert is available who can specify the rules underlying the system behavior and the fuzzy set that represents the characteristics of each variable. The drawbacks of the inverse kinematic equations have posed significant limitations on the robot controller since it is difficult to move the end-effector to a specified position and computing joint variables.

This paper discusses a novel approach in designing a fuzzy logic controller for the robotic arm which replaces the traditional controller and lays the foundation for a new generation of robotic controllers with a simpler architecture.

Conventional Controller Design of Manipulators

The most common controller for robotic manipulators in feedback systems is the Proportional-Integral-Derivative (PID) controller, which is implemented as a secondary controller. This controller corrects errors by means of trajectory tracking [3]. A PID controller performs Proportional amplification...
values of input are then converted to corresponding
to the domain of the variable. These regions are
given problem. Next, each control and solution variable
in the system, the basic
in the system. The first step
and the data elements output from the system. The first step consists of analyzing the system and understanding the
given problem. Next, each control and solution variable
in the fuzzy model is decomposed into a set of fuzzy
represents the weighted average of all fuzzy rules that
by the fuzzy rule evaluator to a physical variable is performed
inverse process of converting the fuzzy outputs of the
in the form of IF (some event) THEN (perform some
fuzzy set variables [6]. The conditional
in the interpretation of the real world data. It utilizes
fuzzy set labels and performs an appropriate reasoning
using Compositional Rule of Inference (CRI) [5]. The
CRI represents the core of the deduction mechanism of
the controller. It performs the composition of fuzzy sets
and matrices of fuzzy rules using the max-min operator.
One of the main advantages of using fuzzy approach is
that it provides the best technique for knowledge
representation that could be possibly devised for encoding
knowledge about continuous (analog) variables.

**Implementation of Fuzzy Logic**

A fuzzy logic controller can be considered as a
control expert system which simulates human thinking
in the interpretation of the real world data. It utilizes
fuzzy set labels and performs an appropriate reasoning
using Compositional Rule of Inference (CRI) [5]. The
CRI represents the core of the deduction mechanism of
the controller. It performs the composition of fuzzy sets
and matrices of fuzzy rules using the max-min operator.
One of the main advantages of using fuzzy approach is
that it provides the best technique for knowledge
representation that could be possibly devised for encoding
knowledge about continuous (analog) variables.

The components of the conventional and fuzzy
systems are similar. They differ mainly in fuzzy
systems containing the Fuzzifier which maps the input
physical variables measured by an external sensor to
fuzzy set variables [6]. The conditional rules expressed
in the form of IF (some event) THEN (perform some
action) are contained within the rule evaluator. The
inverse process of converting the fuzzy outputs of the
fuzzy rule evaluator to a physical variable is performed
by the Defuzzifier. The value produced by the defuzzifier
represents the weighted average of all fuzzy rules that
were fired within the fuzzy rule evaluator.

The fuzzy system designer's task lies in defining
the data points flowing in the system, the basic
transformations performed on the data and the data
elements output from the system. The first step consists of analyzing the system and understanding the
given problem. Next, each control and solution variable
in the fuzzy model is decomposed into a set of fuzzy
regions. These regions are given unique names, called
labels within the domain of the variable. The measured
values of input are then converted to corresponding
degrees of membership in fuzzy sets. This is done by
applying the definition of membership functions for each
input variable. Rules that tie the input values to the
output model are written as follows: "if < fuzzy
proposition A >, then < do fuzzy proposition B >".
Generally, the number of rules a system requires is
related to the number of control variables. The last step
would be to select a method of "defuzzification". There
are several ways to convert an output fuzzy set into a
crisp solution variable, but the most commonly used one
is the centroid technique. Thus the real complexity in
developing a fuzzy system is in creating and testing both
the degree of membership functions and the rule base,
rather than implementing the run-time environment.

**Proposed Fuzzy Logic Controller Model**

The two basic problems encountered when attempting to apply a fuzzy control in real systems are:

- Choice of primary fuzzy sets to be used together
  with the rules that constitute the control law or
  algorithm for a fuzzy control structure.

- Numerical description of the linguistics to
  implement a fuzzy control algorithm in a
  computer, which is a nonfuzzy machine.

The typical robot control problem consists of
moving the end-effector to a user-specified position
(x,y,z) and orientation (roll, pitch, yaw) [7]. To achieve
this, the robot joint motors must be driven to specific
angular positions. The task of computing these specific
joint angles is referred to as the inverse kinematic
problem. In general, inverse kinematic equations are
highly coupled and involve nonlinear differential
equations, whose closed form solutions are often
undefined. This poses a computational bottleneck. The
block diagram of the proposed Fuzzy Logic Controller is
shown in Figure 1.

The Southwestern Research Institute (SWRI)
[6] at San Antonio, Texas applied fuzzy logic to control
a robot without having to explicitly solve inverse
kinematic equations. This controller, mimics intelligent
human-like decision-making via a fuzzy rule base, which
is essentially a collection of varying degrees of cause-
and-effect relationships. The fuzzy rule base is the most
critical element within the novel robot controller. The
performance of the controller is directly dependent on the
quality of fuzzy rules. The approach taken to realize the
optimum set of rules which would track enabling control
was to linearize the robot model and then apply the
principle of superposition to the resulting linearized
equations. First, the x and y components of the
individual locations of robot joints and the observed
tracking error of the robot end-effector need to be
represented in fuzzy terms such as: Positive Big (PB), Positive Medium (PM) etc. up to Negative Big (NB). Next, simple fuzzy rules were formulated to evaluate the individual joint axis contributions to reduce the tracking errors of the robot end-effector. For example, if the tracking error in the x direction is PM and the y component of the end-effector is PB, then move the first joint by PM. If robot end point is Negative Medium (NM) and tracking error is Positive Big (PB), change joint angle 1 by NM.

A Simple 2-Degree of Freedom Manipulator

The problem of designing a manipulator controller stems from the basic idea of the simplest known biological controller which is the human arm [8]. When we reach for an object, we determine the approximate error (distance from our hand to the object), and move in a way to reduce the error. We do not precompute the path or the elbow or shoulder angles which is required to grasp the object. Our motions continuously aim at reducing the distance between the hand and the target. In fact, we are successful at reaching and grasping both stationary and moving objects and accomplish these feats without an accurate mathematical model of the kinematics involved. Thus, the fuzzy logic approach allows an initial control system to be derived from fundamental concepts without the need for extended training sets. There are several approaches that achieve this objective. One such approach is discussed in this paper.

The coordinates of the manipulator of the desired point, or target (the end-effector is assumed to be located at the tip of the second link, or at the second joint) are \((x_d, y_d)\), \((x_0, y_0)\) the coordinates of the manipulator of the initial point, \(e(r)\) is the error of the manipulator between the initial and the end points, \(r_d\) and \(r_0\) are the desired and initial arm lengths (distance from the base joint to the manipulator), angles 180-C, 180-D and E are the initial and final angles between the links respectively and the error angle \(E = C - D\), we have:

\[
e(r)^2 = r_d^2 - r_0^2 = 2L_1 L_2 (\cos C - \sin C \cdot E - \cos C)\]

if angle \(E\) is small

\[
e(r)^2 = 2L_1 L_2 \sin C \cdot E\]

where \(\sin E = E\) for \(E << 0\).

Here, \(e(r)^2\) is used as the input signal to the fuzzy set rules. Actually, \(e(r)^2 = [(x_d^2 + y_d^2) - (x_0^2 + y_0^2)]\), which reduces the error [9]. After achieving the desired \(r_d\) through the change in angle \(C\) to angle \(D\), angle \(A\) is changed to \(A'\) to rotate the robot arm to reach the desired position. The pictorial representation is given in Figure 2.

Here, the rules are arranged as follows:

- For the position of Fig 2(a):
  If (robot arm length needs to be changed by <fuzzy set 1>, and current joint angle is <fuzzy set 2>), then (change second link angle \(C\) by \(E\)).

- For the position of Fig 2(b):
  If (change in angle \(C\) is \(E\), and desired angular change of robot arm length T'-T is <fuzzy set 3>), then (change angle \(A\) to \(A'\)) where <fuzzy set i> (i = 1, 2, ...) is of the form "positive big", "small" etc.

The developed fuzzy rule sets reside within the fuzzy controller, which outputs an incremental joint command to the individual joints of the robot based on the configuration and the deviations of the actual end point to the desired end point. The actual Cartesian end point is determined by applying the forward kinematic equations on joint angles [10,11]. The same procedure can be extended to 3 or higher DOF manipulators.

Simulation

The simulation of the proposed algorithm of the above algorithm, was done on a Mach operating system running NExT machine. The trajectory of a robot tracking a user specified straight line and partial configurations are shown in Figs. 3 and 4. The configurations of the robot are all in reasonable good positions, in the sense that those positions keep all joints away from their singular points. It also shows that the robot has passed one of its singular points, which usually causes an overflow in the conventional mathematical algorithm. The error between the actual and the desired trajectory are between specified limits. A computer simulation program is included in the Appendix.

Results of this simulation were graphed, and the performance for the position of the x and y co-ordinates and the error of the arm with respect to time were plotted (Figures 3 & 4). From Figure 3 one can see that the arm was successful in tracking the desired trajectory. Figure 4 shows that the error progressively decreases to zero in the least possible time.

Conclusions

A non algorithmic, model free approach has been developed that relies on a fuzzy rule base to evaluate the required axis motion for the robot. This scheme does not require solution to the inverse kinematic equation to arrive at the joint set points. The fuzzy rule base provides fast execution speed because the fuzzy rules
perform simple integer additions and multiplications to evaluate the required axis motion. It can be shown that only a maximum of 15 rules are required to evaluate individual joint axis motion and that a linear relationship exists between the number of rules and the degree of freedom of the robot. The fuzzy logic controller approach is found to be 33% faster than traditional controller methods that require solution to the inverse kinematic equation. However, the fuzzy rule approach cannot achieve the tracking accuracies of the PID controller, since a single fuzzy rule describes a patch in the state space rather than an exact single point.

References


Appendix

/* C program to compute the trajectory of the 2 DOF manipulator when the arm is constrained to move in a st. line of the form y = -X + 4. */
#define m -1 /* define the slope of the st. line */
#define y_intercept 4
#define A 2 /* define a random x value */
#define B 2 /* define a y value for the first link */
#define C 4 /* define the initial arm position */
#include <stdio.h>
#include <string.h>
double x_final, y_final;
double x_A, y_B, x_C[500], y_D[500];
double dist1, dist2, arm1_len, D, arm1_len;
FILE *fp;
main()
{
    double time[500];
    double arm2_len, angle_2;
    int i=0, j;
    double error[500];
    D = (m*C) + y_intercept;
    fp = fopen("datafile","w");
    dist1 = sqrt(pow(A,2) + pow(B,2));
    dist2 = sqrt(pow((A-C),2) + pow((B-D),2));
    arm1_len = sqrt(pow(C,2) + pow(D,2));
    puts("give the co-ordinates of final arm position");
    scanf("%d %d", &x_final, &y_final);
    arm1_len = arm1_len;
    x_C[i] = C; time[0] = 0; y_D[i] = D;
    error[i] = 0;
    do
     { x_C[i+1] = x_C[i] + ( C/abs(x_final - C)));
      y_D[i+1] = m * x_C[i+1] + y_intercept;
      arm2_len = sqrt(pow(x_C,2) + pow(y_D,2));
      time[i+1] = time[i] + 0.1;
      ++i;
    }while((error[i-1] = abs(x_C[i-1] - x_final)) < 0.01 && abs(y_D[i-1] - y_final) < 0.01);for(j=0; j<i; j++)
    { fprintf(fp, "%d \n", time[j]);
      fprintf(fp, "%d\n", error[j]);
      fprintf(fp, "%d\n", x_C[j]);
      fprintf(fp, %d\n", y_D[j]);
    }
    fclose(fp);
    printf("Final arm position: \n");
    printf("x: %d, y: %d\n", x_C[i-1], y_D[i-1]);
    printf("Error: %f\n", error[i-1]);
    printf("Arm1 length: %f\n", arm1_len);
    printf("Arm2 length: %f\n", arm2_len);
    printf("Time: %f\n", time[i]);
}
Figure 1. Block Diagram of Proposed Fuzzy Logic Controller

Figure 2. Stretching and Rotating the Robot Arm to Obtain Desired Position
Fig 3: Graph of the x Vs y co-ordinates of the manipulator, as it moves along the preset trajectory path, \( y = -x + 4 \) (a straight line).

\[
y = 4.0325 - 1.0071x \quad R^2 = 0.999
\]

Fig 4: Graph of the error in the position of the manipulator at various time intervals in inches.