Genetic Programming of Full Knowledge Bases for Fuzzy Logic Controllers

Daryl Battle
Lucent Technologies
67 Whippany Road
Whippany, NJ 07981
dbattle@lucent.com

Abdollah Homaifar
Dept. of Electrical Engrg.
NASA ACE Center
NC A&T State University
Greensboro, NC 27411
homaifar@ncat.edu

Edward Tunstel
Jet Propulsion Lab
Robotic Vehicles Grp.
4800 Oak Grove Dr.
Pasadena, CA 91109
tunstel@jpl.nasa.gov

Gerry Dozier
Computer Science & Engrg.
109 Dunstan Hall
Auburn University
Auburn, AL 36849
gvdozier@eng.auburn.edu

Abstract

Genetic programming (GP) is applied to automatic discovery of full knowledge bases for use in fuzzy logic control applications. An extension to a rule learning GP system is presented that achieves this objective. In addition, GP is employed to handle selection of fuzzy set intersection operators (t-norms). The new GP system is applied to design a mobile robot path tracking controller and performance is shown to be comparable to that of a manually designed controller.

1 INTRODUCTION

In recent years, increased efforts have been centered on developing intelligent control systems that can perform effectively in real-time. These include the development of non-analytical methods of soft computing such as evolutionary computation and fuzzy logic. These methods have proven to be effective in designing intelligent control systems and handling real-time uncertainty, respectively. In this paper, our efforts are focused on combining these two paradigms to accommodate the development of intelligent controllers capable of performing in "real-world" applications. More specifically, genetic programming (GP) is employed for knowledge base learning in the fuzzy logic control domain. To date, many researchers have approached this problem using genetic algorithms (see (Cordon,1995)). The key difference is that our approach does not require a transformation from the evolved solution into a usable solution in the control code. The GP solution can be taken as is and executed it in the control code since its phenotype is already a C-coded subroutine.

This paper addresses the application of GP to the design of fuzzy logic controllers for mobile robot path tracking. It has already been demonstrated in (Tunstel, 1996) that genetic programming can be useful as an approach to learning fuzzy logic rules for mobile robot control and navigation. It has also proven useful for the classical cart-centering control problem by (Alba, 1996). However, its practical utility in such domains is weakened when using the common implementation in LISP (as in Tunstel (1996)) due to significant computational costs associated with simulation-based fitness evaluations. Therefore, we have employed a restructured version of the Simple Genetic Programming in C (SGPC) system (Tackett, 1993), which reduced the required evolution time and facilitated evolution of larger populations than attempted in (Tunstel, 1996). As an additional extension, we address the full design of fuzzy logic controllers using GP, i.e. evolution of both the membership functions *and* the rule base. In addition, we incorporate the random selection of fuzzy logic connectives (t-norms) into the evolution process.

2 OVERVIEW OF FUZZY CONTROL

A fuzzy logic controller (FLC) is an intelligent control system that smoothly interpolates between rules. A fuzzy set may be represented by a mathematical formulation known as a membership function. That is, associated with a given linguistic variable (e.g. speed) are linguistic values or fuzzy subsets (e.g. slow, fast, etc.) expressed as membership functions which represent uncertainty, vagueness, or imprecision in values of the linguistic variable. This function assigns a numerical degree of membership, in the closed unit interval [0,1], to a crisp (precise) number. Within this framework, a membership value of zero/one corresponds to an element that is definitely not/definitely a member of the fuzzy set. Partial membership is indicated by values between 0 and 1.

Implementation of a fuzzy controller requires assigning membership functions for inputs and outputs. Inputs are usually measured variables, associated with the state of the controlled plant that are assigned membership values before being processed by an inference engine. The heart of the controller inference engine is a set of if-then rules whose antecedents and consequents are made up of linguistic variables and associated fuzzy membership functions. Fuzzy set intersection, or conjunction, operators in the antecedent are generally referred to as t-norms. They commonly employ algebraic **`min`** or **`product`** operations on fuzzy membership values. Consequents from different rules are numerically aggregated by fuzzy set union and then defuzzified to yield a single crisp output as the control for the plant.

3 MOBILE ROBOT PATH TRACKING

The control problem examined in this paper is a path tracking problem, which was formulated in (Hemami, 1994) for a class of low speed tricycle-model vehicles. Essentially, the control objective is to successfully navigate a mobile robot along a desired path in a two-dimensional environment. We wish to design a fuzzy controller that will achieve this objective. The inputs consist of a measurable position error, ε_d, and a measurable orientation error, ε_θ, associated with path following in the plane (see Fig. 1). The output is the steering angle, δ, which is the corrective control action that would cause the errors to approach zero and, thus, force the robot to follow the desired path. The position error is taken as the deviation of the center of gravity, C, or any other desired point of the robot from the nearest point on the path. The orientation error is the angular deviation of the robot from the tangent of the desired path.

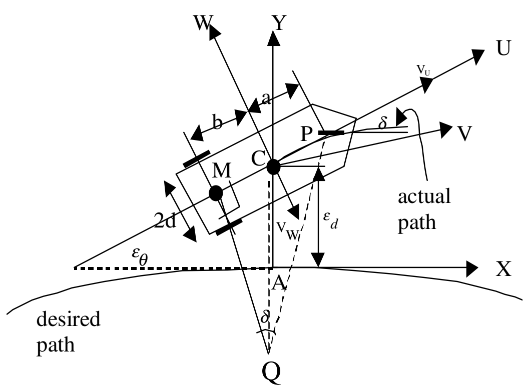


Figure 1: Tracking control and error variables

Hemami et al derived a state-space kinematic model for this robot where the state vector was comprised of the pose errors described. The resulting kinematic model is repeated herein for clarity in the discussion that follows. The reader is referred to (Hemami, 1994) for details of the derivation, which culminates in the following:

$$\begin{bmatrix} \dot{\varepsilon}_d \\ \dot{\varepsilon}_\theta \end{bmatrix} = \begin{bmatrix} 0 & V_u \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon_d \\ \varepsilon_\theta \end{bmatrix} + \begin{bmatrix} MC/MP \\ 1/MP \end{bmatrix} V_u \tan\delta \pm \begin{bmatrix} \dot{\eta}_d \\ \dot{\eta}_\theta \end{bmatrix} \quad (1)$$

where V_u is forward linear velocity of the robot, and η_d and η_θ are rates of change of the effects of path curvature. In (Hemami, 1994) it is concluded (based on dynamic analysis) that for small steering angle, δ ($\tan \delta = \delta$), (1) approximates the slow dynamics of the vehicle when its forward velocity is low. In the simulations presented later, we have simplified the kinematic model by taking this approximation into account. Furthermore, we apply the controller to straight-line path following and, therefore, neglect the model effects of path curvature.

To allow for control of the mobile robot, some means of measuring the input information is needed to feed into the system in order to generate a desired output. Thus the system under control is assumed to have some suitable sensory apparatus. For our implementation, we assume that the robot has odometry sensors that provide access to the error states at all times, or permit calculations thereof. This sensory input data is then mapped to control outputs according to the desired control policy.

4 GP FORMULATION

In this work, the primary focus is on applying GP for simultaneous evolution of fuzzy membership functions and rule bases. We also examine the performance of the evolved controllers relative to that of a fuzzy controller designed by engineers through the usual manual process of trial-and-error with iterative refinement. One such controller was presented in (Tunstel, 1996) and is used here as a basis for comparison.

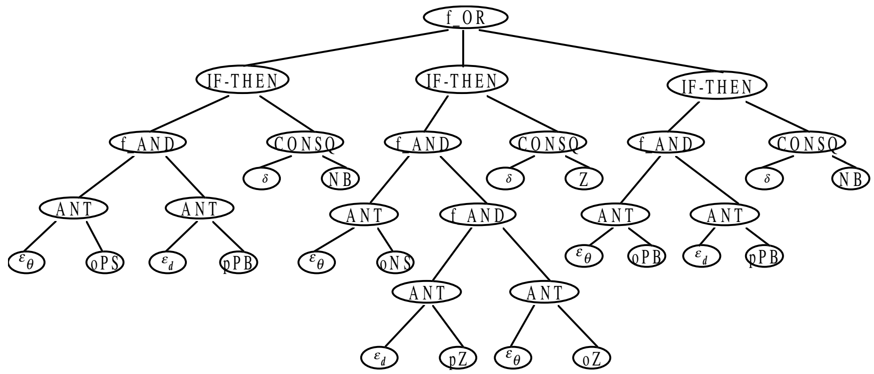
4.1 FUNCTION AND TERMINAL SETS

One of the advantages of the GP paradigm, in this context, is that it allows the elements of the function and terminal sets to be taken directly from the terminology of the manipulated problem. More specifically, the same fuzzy linguistic terms and operators that comprise the genes and chromosome persist in the phenotype (fuzzy rule base). As such, encoding/decoding of numerical representations of chromosomes is not required. The following function and terminal sets are used for the path following problem:

$$F = \{\texttt{f\_OR}, \texttt{IF\_THEN}, \texttt{ANT}, \texttt{CONSQ}, \texttt{f\_AND}, \texttt{MEM\_FUNC}, \texttt{INPUT\_1}, \texttt{INPUT\_2}\} \quad (2)$$

$$T = \{\varepsilon_d, \varepsilon_\theta, \delta, \texttt{pNB}, \texttt{pNS}, \texttt{pZ}, \texttt{pPS}, \texttt{pPB}, \texttt{oNB}, \texttt{oNS}, \texttt{oZ}, \texttt{oPS}, \texttt{oPB}, \texttt{NB}, \texttt{NS}, \texttt{Z}, \texttt{PS}, \texttt{PB}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}\} \quad (3)$$

The functions represent different parts of rule bases and membership functions. **f_OR** represents an entire individual or rule base and serves as an aggregation operator. It occupies the root node of every parse-tree in the population of rule bases. Each rule that fires in a fuzzy rule base returns an output fuzzy set as a result of the rule consequence. The **f_OR** function operates on the output fuzzy sets by taking their fuzzy union to produce a resultant fuzzy set representing the overall output of the rule base. The **IF_THEN** function represents individual rules within the rule base. For a given rule, it returns the rule firing strength and the fuzzy set of the output. More specifically, it returns an output pair containing the **ANT** value and the **CONSQ** value. The **ANT** function represents a single fuzzy proposition in the antecedent portion of individual rules. It returns a numerical value in the closed unit interval [0,1] representing the membership value, or degree of truth, of the proposition. The **CONSQ** function represents the consequent portion of individual rules, and returns the output fuzzy set designated in the rule consequence. The **f_AND** function represents a conjunction operator (fuzzy set intersection) that can be defined using any t-norm. It is limited to the conjunction of two propositions with the idea that conjunctive forms of higher order can be constructed by recursive calls to the function (the level of recursion is bound by a specified maximum depth of a rule base parse tree).



Rule 1: IF ε_θ is oPS and ε_d is pPB THEN δ is NB
Rule 2: IF ε_θ is oNS and ε_d is pZ and ε_θ is oZ THEN δ is Z
Rule 3: IF ε_θ is oPB and ε_d is pPB THEN δ is NB

Figure 2: Example of a syntactically valid program when evolving rule bases

The **`MEM_FUNC`** function represents the membership function definitions for both inputs of the FLC. The **`INPUT_1`** and **`INPUT_2`** functions represent any of the individual fuzzy set definitions for position error and orientation error respectively. The terminals within the terminal set represent the actual inputs and the output, their respective membership functions, and integers used to designate the fuzzy set overlap for the evolving membership functions (explained in more detail below). In (3), notations **`NB`**, **`NS`**, **`Z`**, **`PS`**, and **`PB`** represent fuzzy linguistic terms of "negative big", "negative small", "zero", "positive small", and "positive big", respectively. Terms describing the position error and orientation error are preceded with the prefix "**`p`**" and "**`o`**" respectively. Terms describing the steering control are labeled without prefixes.

Selection of t-norms is automated, thereby, giving the GP system greater control of the evolutionary design. Since the two most commonly used t-norms for fuzzy control are **`min`** and **`product`**, we focus on selection of these two alone. One of the t-norms, for each conjunctive rule, is selected at random by GP for rule bases in the initial population, and propagated based on fitness through successive generations.

4.2 SYNTAX CONSTRAINTS: FULL DESIGN

To achieve the goal of evolving fuzzy rule bases, the GP system must conform to strong syntactic constraints when breeding individuals. Special rules of construction were introduced by (Tunstel, 1996) to ensure that the function and terminal sets satisfied the closure property of GP defined in (Koza, 1992). The syntactic rules for evolving fuzzy rule bases are:

- The aggregation function must occupy the root node of an individual.
- The rule function must occupy the nodes immediately below the root node.
- A left child of the rule function must be the conjunction or antecedent function.
- A right child of the rule function must be the consequent function.
- A child of the conjunction function must be the conjunction or antecedent function.
- Children of the antecedent function must be input linguistic variables & fuzzy sets.
- Children of the consequent function must be output linguistic variables & fuzzy sets.

An individual that could potentially evolve from the designated function and terminal sets in accordance with these rules can be expressed as a rooted, point-labeled tree with pre-ordered branches. An example of a syntactically valid rule tree is illustrated in Fig. 2. Individuals constructed using these rules are subject to an additional constraint on the depth of their tree structures, i.e. the longest sequence of branches from the root node to a terminal (leaf node). The individual in Fig. 2, for example, has a depth of five. The maximum allowable depth is specified as a parameter of the GP run.

In order to accommodate evolution of membership functions, in addition to the rule base, the allowable syntax must be extended. Genes representing membership functions must be added to the chromosomes of individuals in the population. This is achieved by adding three more syntactic rules and modifying the second rule from the previous set. The additional rules are stated below, and an example of a syntactically valid knowledge base conforming to these rules is illustrated in Fig. 3; the absent rule portion of the tree can be taken from Fig. 2.

- (Modified Rule) The function that specifies a membership function definition must occupy the left-most child of the nodes immediately below the root node. The rule function must occupy the remaining nodes immediately below the root node.
- The function that specifies the fuzzy sets for the first input must occupy the left child of the function specifying membership function definitions.

- The function that represents the fuzzy sets for the second input must occupy the right child of the function specifying membership function definitions.
- Children of functions that specify input membership function definitions must be fuzzy set values.

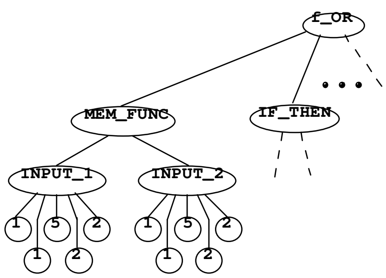


Figure 3: Parse tree for full FLC design

4.3 GENETIC OPERATORS

All rule bases in an initial population are randomly created using the constrained syntax but descendant generations are created using the reproduction, crossover, and mutation operators. Fitness-proportionate reproduction is used and offspring modified by crossover and mutation conform to the syntactic structure. Structure-preserving crossover prohibits the mating individuals from transferring information at different non-root points. The only exception is the valid crossover between **ANT** and **f_AND** function nodes, provided violations do not occur with respect to specifications on the maximum depth of the parse trees of resulting offspring. Selecting (at random) a non-root point and eliminating all the information succeeding it constitutes structure-preserving mutation. The discarded information is replaced by a random subtree that maintains conformance to the syntactic regulations.

5 MEMBERSHIP FUNCTIONS

The GP system presented in (Tunstel, 1996) evolved fuzzy rule bases only, assuming that the membership functions were fixed. Herein, we address full design of fuzzy controllers by extending the former approach using an algorithm introduced in (Homaifar, 1995). In this case, the membership functions are also allowed to vary dynamically during the evolution process. We utilize a unique integer based subtree structure to represent the membership functions. Consider the first branch of the **f_OR** function in Fig. 3. This subtree denotes the membership function specifications for that particular individual. The terminals of this subtree represent the values that will be converted to the base lengths (supports) of the triangular fuzzy sets describing FLC inputs. The peaks of the triangles (height=1) representing membership functions are fixed and taken from the nominal set in (Tunstel, 1996); thus we are only concerned with evolving the supports of the fuzzy sets. The calculation of the supports is done using a modified version of the algorithm proposed in (Homaifar, 1995). The modified algorithm proceeds as follows:

1. Subtract 1 from the allele (terminal) value and divide by 10 (making the range 0-0.4).
2. Subtract this value from 1 (The value of 1 is the distance between the peaks in the original, thus the use of a scaling factor must be incorporated.)
3. Double each of the resulting values to provide a product between 1.2 to 2.0.
4. Multiply each value by independent scaling factors (Note: The scaling factors refer to distances between peaks of a nominal set of membership functions). Multiply by 0.3m for position error fuzzy sets. For orientation error, multiply by 0.2618 radians for fuzzy sets 1 and 5, 0.3927 radians for fuzzy sets 2 and 4, and 0.5236 radians for fuzzy set 3. These values represent the base lengths for each of the individual triangles.
5. Divide the 1^{st} and 5^{th} allele values by 2.
6. Add the new 1^{st} allele value to the peak of fuzzy set 1. Subtract the new 5^{th} allele value from the peak of fuzzy set 5. This designates the inner end-points for the **NB** and **PB** fuzzy sets.

Following this method, the remainder of the branches of the **f_OR** function, which represent the actual rule base, are evaluated using the translated input membership function definitions. Output membership functions are fixed as defined for the hand-derived controller.

6 SIMULATION

We apply GP to evolve fuzzy controllers that will direct the robot's motion from initial locations near the desired path to final locations on the path such that steady state and final pose errors are minimized. This involves frequent simulation of robot motion throughout the evolution process. The simulated robot is based on (1) with dimensions taken from a Hero-1 mobile robot, which has a tricycle wheel configuration in which the front wheel is driven by a DC motor and steered by a stepper motor. Its two rear wheels are passive. Dimensions are 0.3*m* for the wheelbase, and 0.2*m* for the offset from the rear axle to the front wheel. Referring to Fig. 1, the wheelbase refers to the constant length 2d and the offset refers to the constant length MP. All simulations were conducted assuming a controller sampling rate of 20 *Hz* and run for a maximum of ten seconds. In each case, the robot travels at a constant nominal forward speed of 1.5 *m/s*.

6.1 CONTROLLER FITNESS EVALUATION

During the GP process, each rule base is evaluated by simulating the robot's motion from each of a finite number of initial conditions (fitness cases) until either the goal state is achieved or the allotted time expires. For this problem we use eight different initial conditions based on the pair-wise symmetry of the possible error categories:

(a) $\varepsilon_d = 0,\ \varepsilon_\theta < 0$; (b) $\varepsilon_d < 0,\ \varepsilon_\theta = 0$;

(c) $\varepsilon_d < 0,\ \varepsilon_\theta < 0$; (d) $\varepsilon_d > 0,\ \varepsilon_\theta < 0$.

Consider error category (d), which represents a case where the robot is located on the left of the desired path with a negative heading orientation. There also exists a symmetric case where the robot is located on the right of the desired path with a positive heading orientation. These symmetric cases are each represented by category (d). The same holds for category (a), (b) and (c), yielding a total of eight fitness cases that fully describe the possible combinations of errors with respect to the path. In (Hemami, 1994), it was shown that error category (d) is the most general for studying path tracking by tricycle-type vehicles.

We compute path tracking performance by summing the Euclidean norms (normalized) of the final error states plus the average control effort ($\bar{\delta}$) over all eight fitness cases. The following fitness function drives the evolution process

$$Raw\ Fitness = \sum_{i=1}^{8} \sqrt{(\varepsilon_d^2 + \varepsilon_\theta^2 + \bar{\delta}^2)_i} \qquad (4)$$

where ε_d and ε_θ are the position error and orientation error existing at the end of each fitness case simulation. The objective of this fitness function is to minimize final path tracking errors as well as the control effort expended. A perfect fitness score is zero and, in general, lower fitness values are associated with better controllers. Path tracking success is also based on ability to minimize the error states to within the following specified tolerances, $|\varepsilon_d|<0.15$m and $|\varepsilon_\theta|<0.26$ rad., for each fitness case. A fitness case simulation in which these tolerances are satisfied is considered a hit, or successful trial. Thus, each individual has the potential of receiving a total of eight hits during fitness evaluation for this path tracking problem.

7 EVOLVED CONTROLLER RESULTS

All GP runs for the path tracking problem were executed using a restructured version of SGPC (Tackett, 1993) on a 260 MHz MIPS DECstation. The GP system was executed for five consecutive runs on a population of 200 individuals for a maximum of 50 generations. Reproduction, crossover, and mutation probabilities used for these runs were 0.399, 0.6, and 0.001 respectively. Maximum depths for new trees, trees after crossover, and mutated trees were set at 5, 7, and 4 respectively. About one hour of computation time was required for a run of this magnitude. A rule base of 25 rules emerged as the fittest among all five runs. The co-evolved membership functions associated with the best rule base are shown in Fig. 4, and the rules are listed in Table 1 with t-norms indicated for conjunctive rules. Support widths of the evolved input membership functions differ slightly from those specified for the hand-derived controller. Although the rule base size is identical, the evolved rules differ significantly. Observation of the evolved rules suggests possible coherence problems in the rule base. Such outcomes are possible since the system does not guarantee intuitive (much less, coherent) GP solutions. Rather, it allows such possibilities in order to avoid non-discovery of innovative solutions that may be counter-intuitive to the designer and, consequently, overlooked.

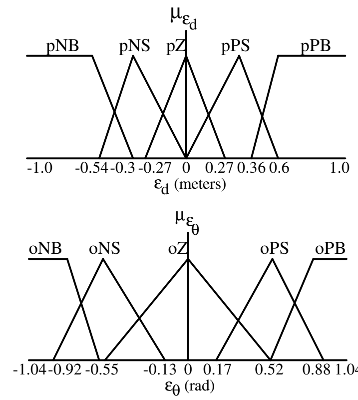


Figure 4: Co-evolved membership functions

The fully-designed controller received a raw fitness of 0.1091 with 8 hits. In comparison, the original hand-derived complete rule base received a raw fitness of 0.08 with 8 hits. Figure 5 illustrates the temporal responses of position error, orientation error, and control effort for the evolved controller and for the hand-derived controller. This result corresponds to category (d), the most general error category. The resulting controller achieved comparable response characteristics to those of the hand-derived controller in the remaining error categories as well.

Table 1: Co-evolved Rule Base with T-norm Selection

1	IF **oZ** THEN **NS**
2	IF **pPB** THEN **Z**
3	IF **pNB** THEN **Z**
4	IF **pPS** THEN **NB**
5	IF **pNS** and **oPS** THEN **NS** (**min**)
6	IF **pNB** THEN **PB**
7	IF **oNS** THEN **Z**
8	IF **oNB** THEN **PS**
9	IF **pNS** THEN **NS**
10	IF **pNS** and **oZ** THEN **PB** (**prod**)
11	IF **oPB** THEN **NB**
12	IF **pNS** and **oPB** THEN **NB** (**prod**)
13	IF **pPS** THEN **NS**
14	IF **oNS** THEN **PB**
15	IF **pPB** THEN **NB**
16	IF **oZ** THEN **PS**
17	IF **oNB** THEN **PB**
18	IF **pNS** and **oNS** THEN **PB** (**min**)
19	IF **pNS** THEN **Z**
20	IF **oPS** THEN **NB**
21	IF **pZ** THEN **PS**
22	IF **pPB** and **oZ** THEN **Z** (**min**)
23	IF **pPB** THEN **PS**
24	IF **oPS** THEN **PS**
25	IF **oNS** THEN **PS**

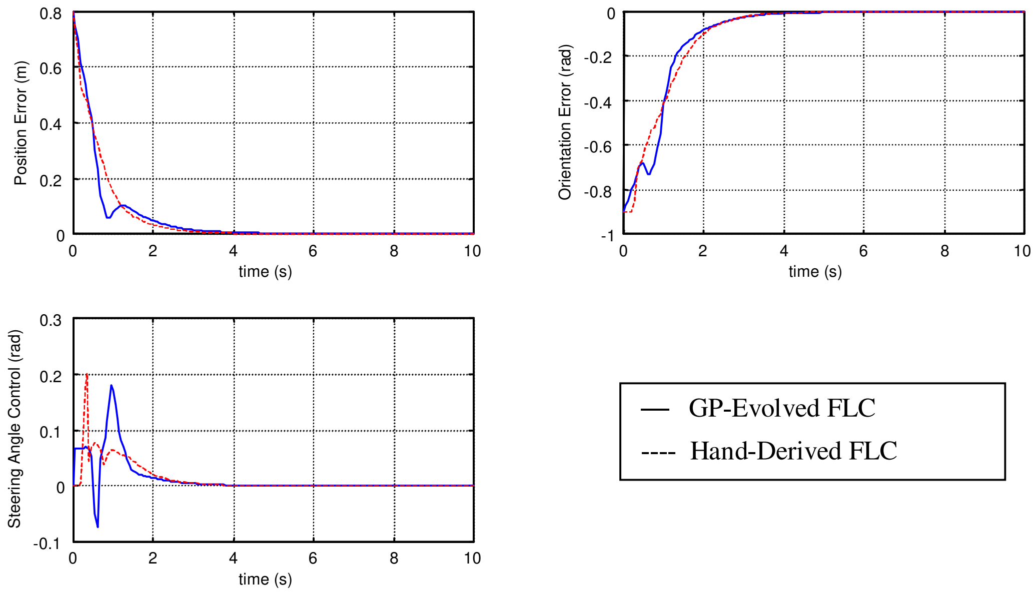


Figure 5: Temporal Path Tracking Response of Fuzzy Controller

8 CONCLUSIONS

GP was successfully applied to discover FLCs capable of steering a mobile robot to track straight-line paths in the plane. Instances of simultaneous evolution of membership functions and rules showed that GP was capable of evolving a FLC that demonstrated satisfactory responsiveness to various initial conditions while utilizing minimal human interface. Suboptimal solutions, with respect to the employed fitness function, were consistently found that compared favorably against manually-derived solutions, suggesting a strong basis for practical application of GP in the controller design process. Further automatic improvement towards optimal solutions could be made by synthesizing a hybrid between GP and a localized search method such as hill-climbing (O'Reilly, 1995, Dozier, 1997). The implementation proposed herein provides a means for full design of FLCs that can be directly applied to a physical system. Alternatively, human experts can use the evolved FLCs as design starting points for further manual refinement as suggested in (Alba, 1996).

Acknowledgments

This work is partially funded by grants from NASA ACE at North Carolina A&T SU under grant # NAG2-1196 and NASA Dryden Flight Research Center under grant # NAG4-131. The authors wish to thank the ACE Center and NASA Dryden for their financial support.

References

Alba, E., Cotta, C. and Troyo, J. (1996). Type-constrained Genetic Programming for Rule-base Definition in Fuzzy Logic Controllers. *1st Annual Conference on Genetic Programming*, Stanford University, CA. Pages 28–31.

Cordon, O., Herrera, F. and Lozano, M. (1995). A Classified Review on the Combination Fuzzy Logic-Genetic Algorithms Bibliography. *Technical Report DECSAI 95129*, Dept. of Computer Science and AI, Univ. of Granada, Spain.

Dozier, G., Bowen, J., Homaifar, A., Esterline, A. (1997). Solving Randomly Generated Static and Dynamic Fuzzy Constraint Networks Using Microevolutionary Hill-Climbing. *International Journal of Intelligent Automation and Soft Computing*, 3(1) pp. 51–62.

Hemami, A., Mehrabi, M., and Cheng, R. (1994). Optimal Kinematic Path Tracking Control of Mobile Robots with Front Steering. *Robotica*, 12(6) pp. 563–568.

Homaifar, A. and McCormick, E. (1995). Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2) 129–139.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press.

O'Reilly, U-M. (1995). *An Analysis of Genetic Programming*. PhD Dissertation. School of Computer Science. Carleton University.

Tackett, W., Carmi, A. (1993). SGPC: Simple Genetic Programming in C. *Prime Time Freeware for AI*, Issue 1(1).

Tunstel, E. and Jamshidi, M. (1996). On Genetic Programming of Fuzzy Rule-Based Systems for Intelligent Control. *International Journal of Intelligent Automation and Soft Computing*, 2(3) 271–284.