

# Using the Bees Algorithm to tune a fuzzy logic controller for a robot gymnast

D.T. Pham, A. Haj Darwish, E.E. Eldukhri, S. Otri

*Manufacturing Engineering Centre, Cardiff University, Cardiff CF24 3AA, UK*

---

## Abstract

This paper focuses on using the Bees Algorithm to tune the parameters of a fuzzy logic controller developed to stabilise and balance an under-actuated two-link acrobatic robot (ACROBOT) in the upright position. A linear quadratic regulator (LQR) was first developed to obtain the scaling gains needed to design the fuzzy logic controller. Simulation results confirmed that using the Bees Algorithm to optimise the membership functions and the scaling gains of the fuzzy system improved the controller performance.

**Keywords:** Bees Algorithm, optimisation, fuzzy logic, robot, LQR

---

## 1. Introduction

The nonlinear characteristics of ill-defined and complex modern plants make classical controllers inadequate for such systems. However, using fuzzy sets and fuzzy logic principles [1] has enabled researchers to understand better, and hence control, complex systems that are difficult to model. These newly developed fuzzy logic controllers [2, 3] have given control systems a certain degree of intelligence.

A fuzzy logic controller or fuzzy controller can be considered a fuzzy rule-based controller which consists of input and output variables with membership functions, a set of (IF ... THEN) rules and an inference system.

Designing fuzzy controllers involves deciding appropriate values for the parameters of the fuzzy membership functions and constructing the rule base in order to achieve the required performance.

This problem can be solved by tuning the controller parameters using an optimisation technique to obtain the best possible solution according to a given criterion or fitness function.

Many intelligent optimisation techniques such as the Genetic Algorithm [4], Simulated Annealing [5] and Particle Swarm Optimisation [6] have been proposed to tune fuzzy controllers.

A new optimisation technique called the Bees Algorithm [7] provides an alternative means for controller tuning. The Bees Algorithm is a swarm-based optimisation algorithm that mimics the food foraging behaviour of honey bees.

This paper discusses the use of the Bees Algorithm to tune a fuzzy controller for an under-actuated Robot Gymnast known as ACROBOT (ACRObatic roBOT) [8].

Controlling under-actuated robots is one of the most challenging problems in robotics because of the complex nonlinear dynamics of these robots and the lack of actuating torque in one or more of their joints.

The ACROBOT simulated in this study is a planar robot consisting of two links with two joints. The structure of the robot is modelled on the body of a human gymnast balancing on a high bar where the first joint of the robot represents the gymnast's fists gripping the bar, the first link his arms, head and torso, the

second joint his hips, and the last link his legs and feet. One actuator is connected directly to the second joint while the first joint is left unpowered.

Controllers for the ACROBOT can be divided into two types: (i) up-swing controllers (ii) balancing controllers.

The function of an up-swing controller is to move the ACROBOT from its stable state (in which it hangs vertically below the bar) to the inverted position (where the robot stands vertically upright, on its first joint, above the bar) by pumping energy from the second joint to the first joint. Methods such as using neural oscillators to eliminate the phase shift between the first and second joints [9], fuzzy control [10] and partial feed-back [8] have been adopted to achieve up-swinging. The main task of the up-swing controller is to force the ACROBOT to enter the attraction basin of the inverted state with minimum velocity so as to enable the balancing controller to catch it and maintain it stably in the upright position [8, 10].

This paper focuses on tuning a balancing fuzzy logic controller for an ACROBOT using the Bees Algorithm. Section 2 presents the mathematical model of the robot. Section 3 gives the parameters of a linear quadratic regulator (LQR) developed for balancing the ACROBOT. The LQR gains were used to create the fuzzy logic controller discussed in section 4. The tuning of the fuzzy controller is detailed in section 5. Section 6 presents the simulation results obtained for the control system before and after tuning. Section 7 concludes the paper.

## 2. Dynamics model of ACROBOT

Figure 1 is a schematic diagram of the ACROBOT to be controlled. The second joint of the robot is actuated by a DC motor [8, 10]. The description and values of the parameters shown in Fig. 1 are given in [10].

A state-space model of the ACROBOT was obtained by linearising its dynamics around the inverted position ( $q_1 = \pi/2, q_2 = 0, \dot{q}_1 = \dot{q}_2 = 0$ ) as

$$\dot{x} = Ax + B\tau \quad (1)$$

$$y = Cx + D\tau \quad (2)$$

where  $x$ , the state vector, is defined as:

$$x = [q_1 - \pi/2 \quad q_2 \quad \dot{q}_1 \quad \dot{q}_2]^T$$

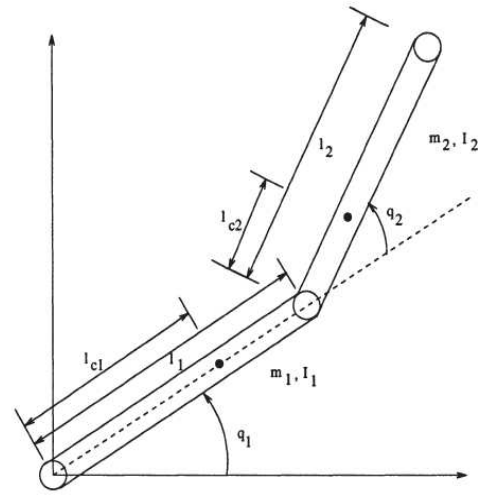


Fig. 1. ACROBOT representation.

In equations (1) and (2),  $\tau$  is the input torque applied to the actuator located at the second joint and  $y = x$  is the output vector. With the robot parameters chosen as defined in [10], the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  are [10]:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 49.4782 & -5.5038 & 0 & 0 \\ -50.0109 & 66.2336 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -23.9348 \\ 175.7326 \end{bmatrix}, \quad C = I_{4 \times 4}, \quad D = 0_{4 \times 1}$$

## 3. LQR controller

The linearised ACROBOT model in equations (1) and (2) has been used to develop a linear quadratic regulator (LQR) to maintain the robot balanced in a stable inverted position [8, 10]. Similar to the procedure followed in [10], the controller gains were reproduced using a MATLAB standard LQR solution with the weight matrices  $Q$  and  $R$  chosen as [10]:

$$Q = \begin{bmatrix} 1000 & -500 & 0 & 0 \\ -500 & 1000 & 0 & 0 \\ 0 & 0 & 1000 & -500 \\ 0 & 0 & -500 & 1000 \end{bmatrix} \quad R = [1000]$$

The obtained LQR is:

$$K_{LQR} = [-310.6372 \quad -26.3246 \quad -47.5231 \quad 5.3165]$$

The LQR was employed to give the scaling gains of the fuzzy input and output variables needed for designing the alternative fuzzy logic controller that was subsequently tuned using the Bees Algorithm.

#### 4. Fuzzy logic controller

The fuzzy controller consists of 4 input variables and 1 output variable. Each of the input variables has 3 membership functions defined in the universe of discourse  $[-1, 1]$  (see Fig. 2).

The output variable is composed of 9 triangular membership functions with universe of discourse  $[-1, 1]$  (see Fig. 3.).

The LQR feedback gains were used to evaluate the scaling gains for the fuzzy logic controller.

As in [10], if the scaling gains for the input variables are denoted as  $[g_0 \quad g_1 \quad g_2 \quad g_3]$  and the scaling gain for the output variable is  $h$ , then

$$K_{LQR} = [g_0 h \quad g_1 h \quad g_2 h \quad g_3 h]$$

Thus, a possible set of gains is:

$g_0 = 5.5555$ ,  $g_1 = 0.4708$ ,  $g_2 = 0.8500$ ,  $g_3 = 0.0951$ , and  $h = 55.9147$ . These are the same gains as those chosen for the controller reported in [10].

The Mamdani model [2] was used as the basis of the proposed controller with the Max-Min method of inferring and the Centroid method of defuzzification.

The rule base consists of 81 (IF ... THEN) rules.

As in [1], the following equation was used to derive rules of the form :

If (q1 is  $i$ ) and (q2 is  $j$ ) and (dq1 is  $k$ ) and (dq2 is  $l$ ) then (action is  $m$ )

$$m = (i + j + k + l) \times \frac{2}{(N - 1)n} \quad (3)$$

where  $m$  is the index of the membership function of the output action.  $i, j, k$ , and  $l$  are the indices of the input membership functions ( $i, j, k, l = 1, 2$  or  $3$ ),  $N$  is the number of input membership functions and  $n$  is the number of inputs.

Note that indices were subsequently converted into linguistic variables (for example, NB, ZERO, PB etc) for ease of reading the rules.

The following are two examples of rules from the rule base. The first rule is for  $i=j=k=l=1$ . The second rule is for  $i=3, j=2, k=1$  and  $l=1$ .

Rule 1: If (q1 is NB) and (q2 is NB) and (dq1 is NB) and (dq2 is NB) then (action is MF1)

Rule 31: If (q1 is PB) and (q2 is Zero) and (dq1 is NB) and (dq2 is NB) then (action is MF4)

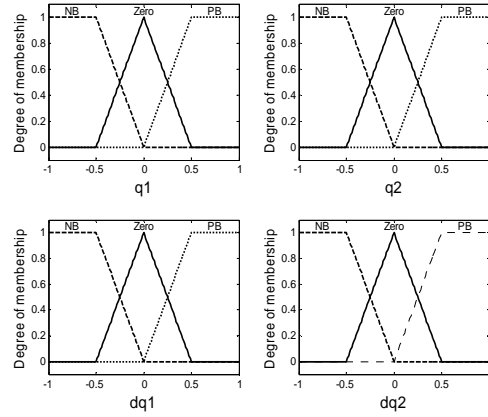


Fig. 2. Input membership functions.

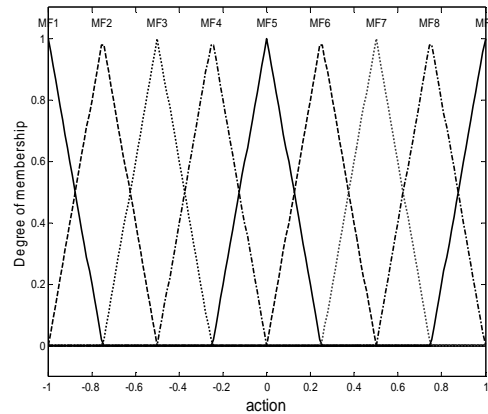


Fig. 3. Output membership functions.

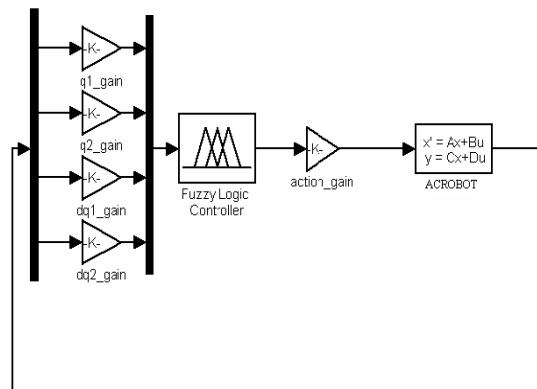


Fig. 4. SIMULINK representation of ACROBOT.

MATLAB and SIMULINK were used to implement the fuzzy controller and model the ACROBOT (see Fig. 4).

## 5. Tuning of fuzzy controller

### 5.1 The Bees Algorithm

The algorithm requires a number of parameters to be set, namely: number of scout bees ( $n$ ), number of elite bees ( $e$ ), number of patches selected out of  $n$  visited points ( $m$ ), number of bees recruited for patches visited by “elite bees” ( $nep$ ), number of bees recruited for the other ( $m-e$ ) selected patches ( $nsp$ ), and size of patches ( $ng$ ).

The pseudo code of the Bees Algorithm [7] is as follows.

1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met)
  - // Forming new population.
4. Select elite bees and elite sites for neighbourhood search.
5. Select other sites for neighbourhood search.
6. Recruit bees around selected sites and evaluate their fitness.
7. Select the fittest bee from each site.
8. Assign the remaining bees to search randomly and evaluate their fitness.
9. End while.

### 5.2 Applying the Bees algorithm

With the rule base fixed, the Bees Algorithm was used to tune the parameters of the input and output membership functions and the scaling gains for the input and output variables.

In theory, each bee was a vector comprising 60 real numbers. Five of those numbers were reserved for scaling gains, 11 for the parameters of each input variable membership function (3 to represent a triangular function and 4 to represent a trapezoidal function) and 27 numbers to represent the triangular membership functions of the output variable.

However, due to symmetry and by appropriate design of the membership functions, a bee only needed to represent 12 numbers, one for each of the four input variables ( $X_1, X_2, X_3, X_4$ ), 3 for the three output variables ( $X_5, X_6, X_7$ ), and 5 for the scaling gains ( $X_8, X_9, X_{10}, X_{11}, X_{12}$ ).

The search space was different for the numbers

mentioned above.

The search space for  $X_1, X_2, X_3$  and  $X_4$  to represent the membership functions of the input variables was  $[0, 1]$ , and the spaces for the output variable membership functions  $X_5, X_6$  and  $X_7$  were as follows:

- $X_5: [0, 1]$
- $X_6: [X_5, 1]$
- $X_7: [X_6, 1]$

The values of  $X_1, X_2 \dots X_7$  were used to construct the fuzzy controller. For example, the left-most input variable trapezoidal membership function for  $q_1$ ,  $[-1 -1 -X_1 0]$ , can be seen in Figure 5 which also shows the triangular function  $[-X_1 0 X_1]$  and the right-most (trapezoidal) function  $[0 X_1 1 1]$ .

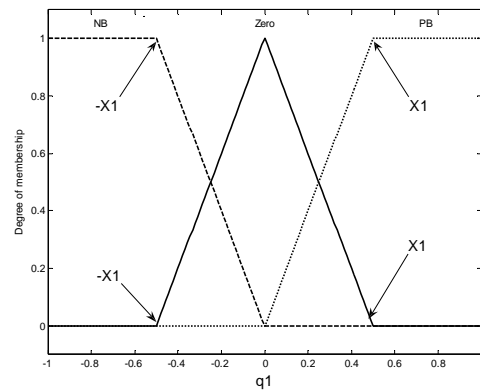


Fig. 5. Construction of membership function

Table 1

The Bees Algorithm parameters

The Bees Algorithm parameters	Symbol	Value
Population size	$n$	20
Number of selected sites	$m$	10
Number of elite sites	$e$	5
Number of bees around other selected points	$nsp$	3
Number bees around elite	$nep$	10
Patch size	$ng$	0.025

The first input variable gain X8 was in the range [0.2, 0.8], the second input gain X9 belonged to [0.1, 2.5], the third input gain X10 was in the range [0.1, 2.0] and the fourth input gain X11 belonged to [0.01, 2.0]. The range of the output gain X12 was [4.0, 99.0].

The ranges for the X8, X9, X10, X11, X12 scaling gains were the same as those used in [10].

The fuzzy controller derived from the LQR parameters given in section 3 was employed as a seed for the Bees Algorithm. The values of the parameters of the Bees Algorithm are shown in Table 1.

### 5.3 Fitness function

The fitness function was the same as that proposed in [10]. It is based on a weighted sum of parameters chosen to minimise the input torque  $\tau$ , angular displacement  $(q_1 - \pi/2)$  of the first link away from the vertical inverted position and angular displacement  $q_2$  away from the first link.

Let  $w$  be the vector of weights  $[w_1 w_2 \dots w_9]^T$  and  $S$  the vector of parameters  $[S_1 S_2 \dots S_9]^T$ . The fitness function is given by:

$$f = \frac{1}{w^T S}$$

$f$  was calculated over a simulated control run of 10 seconds with the ACROBOT starting from an almost fully inverted position.

The elements  $S_1$  to  $S_9$  are defined as follows:

$S_1, S_2$  and  $S_3$  = mean value of  $(q_1 - \pi/2)$ ,  $q_2$  and  $\tau$ , respectively, over the time period 5-10 s, assuming the simulated control experiment started at time  $t=0$  s.

$S_4, S_5$  and  $S_6$  = normalised sum of squares  $(q_1 - \pi/2)^2$ ,  $q_2^2$  and  $\tau^2$ , respectively, over the time period 0-5 s.

$S_7, S_8$  and  $S_9$  = standard deviation of  $(q_1 - \pi/2)$ ,  $q_2$  and  $\tau$ , respectively, over the time period 5-10 s.

The weight vector used in [10],  $w = [1 \ 1 \ 1 \ 100 \ 80 \ 5 \ 100 \ 80 \ 0.5]$ , was also adopted as it had been found to reduce variations in  $q_1$  and  $q_2$  and to minimise the required input torque  $\tau$ .

The Bees Algorithm was run for 10 iterations and the parameters obtained at the end of the tenth iteration were taken as the tuned parameters of the control system.

## 6. Results

The simulation of the fuzzy controller was carried out under the MATLAB and SIMULINK environment with a fuzzy logic toolbox and the Runge-Kutta (ODE45) solver. The time step for the simulation was 0.01 s. The state vector  $[\pi/2 + 0.04, -0.05, -0.2, 0.04]^T$  was used as the initial condition of the ACROBOT.

Table 2 shows the new values for the parameters of the fuzzy logic controller for a typical run of the Bees Algorithm.

Results before and after tuning are illustrated in Figs 6, 7, 8 and 9 respectively. Figure 6 and Fig 7 show the behaviour of the ACROBOT with the controller not tuned, whereas Fig 6 illustrates the variations in the values of angles  $q_1$  and  $q_2$ .

Figure 7 shows the required control input to keep the ACROBOT balanced.

Table 2  
Tuned parameters

	Before tuning	After tuning
X1	0.5	0.5214
X2	0.5	0.4832
X3	0.5	0.5007
X4	0.5	0.5051
X5	0.25	0.2676
X6	0.5	0.5451
X7	0.75	0.7174
X8	5.5555	5.5691
X9	0.4708	0.4494
X10	0.8500	0.8737
X11	0.0951	0.1038
X12	55.9147	55.9472

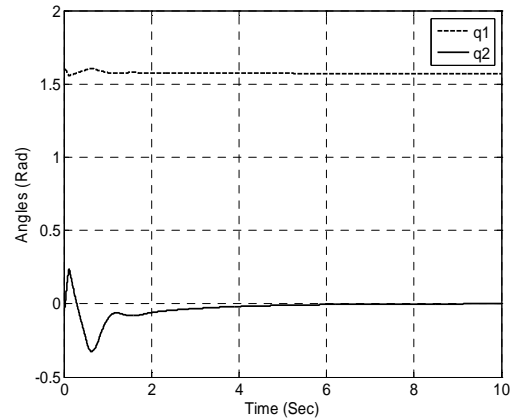


Fig. 6.  $q_1, q_2$  angles of balanced ACROBOT with the controller before tuning

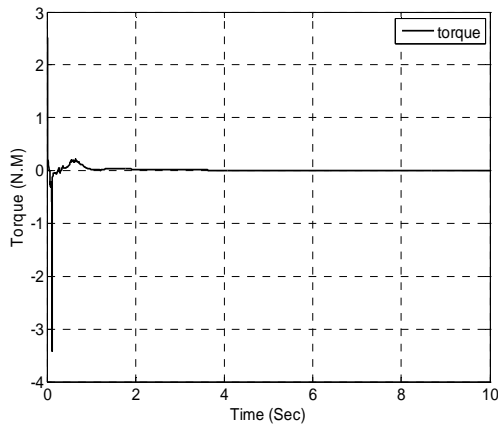


Fig. 7. Control signal from the controller before tuning

The behaviour of the ACROBOT as regards angles  $q_1, q_2$  and control torque  $\tau$  with a tuned controller is shown in Fig 8 and Fig 9.

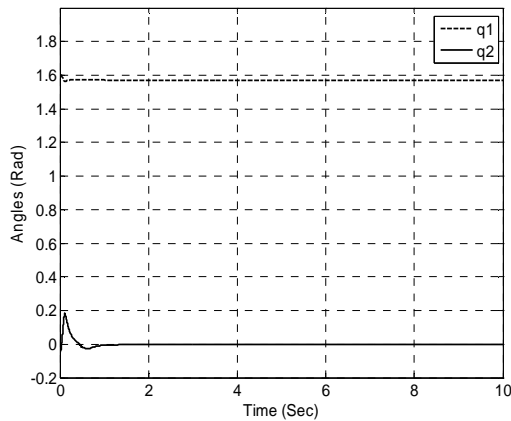


Fig. 8.  $q_1, q_2$  angles of balanced ACROBOT with tuned controller

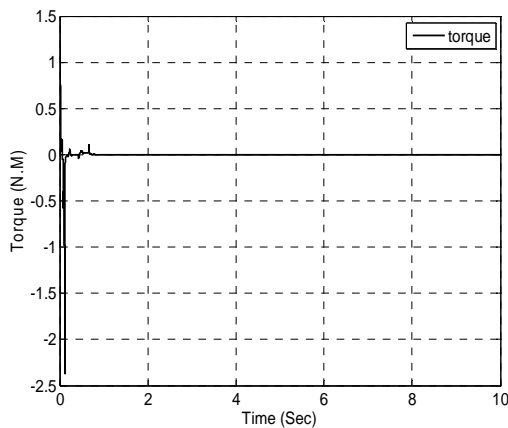


Fig. 9. Tuned control signal

## 7. Conclusion and further work

From the simulation results, it is evident that the controller tuned using the Bees Algorithm gave a smooth performance with fewer variations in the values of  $q_1, q_2$  and smaller input control signals  $\tau$  than in the case of the untuned controller. Hence, the Bees Algorithm is a useful tool for tuning fuzzy logic controllers to achieve better performance.

Further work includes employing the Bees Algorithm for the swinging up phase with a view to developing a combined controller for both the swinging up and balancing of the ACROBOT.

## References

- [1] Passino K.M. and Yurkovich S. *Fuzzy Control*. 1998, Menlo Park, CA: Addison-Wesley Longman. 475.
- [2] Mamdani E.H. and Assilian S. *An experiment in linguistic synthesis with a fuzzy logic controller*. International Journal of Man-Machine Studies, 1975(7): pp. 1-13.
- [3] Sugeno M. *Industrial Applications of Fuzzy Control* 1985: Elsevier Science Ltd 269.
- [4] Herrera F., Lozano M., and Verdegay J.L. *Tuning Fuzzy Logic Controllers by Genetic Algorithms* International Journal of Approximate Reasoning, 1995(12): pp. 299-315.
- [5] Liu G. and Yang W. *Learning and tuning of fuzzy membership functions by simulated annealing algorithm*. 2000 pp. 367-370.
- [6] Feng M. *Particle swarm optimization learning fuzzy systems design*. 2005 pp. 363-366 vol.1.
- [7] Pham D.T., Ghanbarzadeh A., Koç E., Otri S., Rahim S., and Zaidi M. *The Bees Algorithm – A Novel Tool for Complex Optimisation Problems*. Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems, D.T. Pham, E.E. Eldukhri and A. J. Soroka (eds), Elsevier (Oxford) (2006), ISBN-10:08-045157-8, pp. 454-459.
- [8] Spong M.W. *The Swing up Control Problem for the Acrobot* IEEE Control System Magazine, 1995. **15**(2): pp. 49-55.
- [9] Matsuoka K., Ohyama N., Watanabe A., and Ooshima M. *Control of a giant swing robot using a neural oscillator*. International Congress Series, 2006. **1291**: pp. 153-156.
- [10] Brown S.C. and Passino K.M. *Intelligent Control for an Acrobot*. Journal of Intelligent and Robotic Systems, 1997. **18**: pp. 209-248.