Applied Bionics and Biomechanics
Vol. 5, No. 1, March 2008, 33–46

Taylor & Francis
Taylor & Francis Group

Adaptive Fuzzy-Lyapunov controller using biologically inspired swarm intelligence

Alejandro Carrasco Elizalde* and Peter Goldsmith

Department of Mechanical and Manufacturing Engineering, University of Calgary, 2500 University Drive N.W., Canada

(*Received 22 December 2006; final version received 24 February 2008*)

The collective behaviour of swarms produces smarter actions than those achieved by a single individual. Colonies of ants, flocks of birds and fish schools are examples of swarms interacting with their environment to achieve a common goal. This cooperative biological intelligence is the inspiration for an adaptive fuzzy controller developed in this paper. Swarm intelligence is used to adjust the parameters of the membership functions used in the adaptive fuzzy controller. The rules of the controller are designed using a computing-with-words approach called Fuzzy-Lyapunov synthesis to improve the stability and robustness of an adaptive fuzzy controller. Computing-with-words provides a powerful tool to manipulate numbers and symbols, like words in a natural language.

Keywords: Lyapunov stability; fuzzy control; adaptive control; swarm intelligence; robotics

Introduction

Adaptive controllers have been used to address problems such as time-varying environments, plants with changing parameters and changing performance objectives. Several adaptive controllers have been proposed where the parameters are adjusted via optimisation, normally using gradient information (Ghorbel et al. 1989; Hill and Ysdtie 2004). However, gradient algorithms may exhibit slow convergence and are vulnerable to getting trapped at local minima of the function being optimised. Besides, gradient information is not always present in many situations. New algorithms for optimisation have been developed in the past years to overcome the drawbacks of gradient algorithms. These algorithms are based on stochastic approaches like genetic algorithms (Mnif and Ghommem 2005), tabu (Roesener and Barnes 2006), simulated annealing (Kirkpatrick et al. 1983), colony ant optimisation (Gambardella et al. 1999) and particle swarm optimisation (PSO) (Eberhart and Kennedy 1995).

Fuzzy logic has been around for 30 years now, and despite the controversy around it has been successful in many areas, including control systems (Hasan and Zekai, 1999; Carrasco and Goldsmith 2004; Maragos 2005). Fuzzy control incorporates the knowledge and experience of the operator to generate rules that dictate the behaviour of the controller. A difficulty in fuzzy control is determining the control parameters. In this research, we use PSO to select the control parameters of an adaptive fuzzy controller whose rules are created using Lyapunov synthesis. In the next sections we introduce a structure for adaptive fuzzy controller, create rules for the controller by Lyapunov method and select parameters using PSO.

Fuzzy controller

There are three main structures of adaptive controllers: model reference, gain scheduling and self-tuning. Our proposed controller is of the third type, which means the controller adapts its own parameters to maintain stability and good performance. First of all, we introduce the definition of fuzzy set.

Definition 1. *A fuzzy set A is a collection of elements $x \in X$ from some universe of discourse X, by a certain degree of membership.*

Definition 2. *A membership function μA of a fuzzy set A is a mapping from some universe of discourse X into the real interval* [0,1].

There exists several functions to define the membership functions; however the most commonly used are of the triangular shape defined by (1) and Gaussian shape defined by (2)

$$\mu A(x) = \begin{cases} 0 & x \le a,\ x \ge c \\ \frac{x-a}{b-a} & a < x < b \\ \frac{x-b}{c-b} & b \le x < c \end{cases} \quad (1)$$

$$\mu A(x) = exp\left(-\frac{(x-a)^2}{2b^2}\right) \quad (2)$$

*Corresponding author: Email: ecarrasc@ucalgary.ca; carreli@yahoo.com

ISSN: 1176-2322 print / 1754-2103 online

DOI: 10.1080/11762320802027869
http://www.informaworld.com

Definition 3. *For any fuzzy sets A and B, $C = A \cup B$, means that $\mu C(x) = max(\mu A(x), \mu B(x))$.*

Definition 4. *For any fuzzy sets A and B, $C = A \cap B$, means that $\mu C(x) = min(\mu A(x), \mu B(x))$.*

The representation of knowledge from systems can be encoded in different forms, one of which is a rule-based system. A rule consists of an IF part (antecedent or premise) and a THEN part (consequent or conclusion). The antecedent lists a set of conditions in some logical combination. A single if-then rule looks like:

$$\text{If } x_1 \in A \text{ and } x_2 \in B \text{ then } y \in C$$

where, if the conditions of the rule are satisfied then the conclusion is executed. A fuzzy rule follows the same structure, but with a difference: when the antecedents conditions are partially true, then the consequent is also true to the same degree. The formulation of the fuzzy controller consists of rules based on the knowledge of an expert to control a particular system. Figure 1 shows the fuzzy controller structure, which has two inputs (error e and derivative of error $\dot{e}$) and one output u.

The fuzzification part converts input data to a degree of membership for each fuzzy set specified in the universe of discourse. Figure 2 shows the universe of discourse of error with its linguistic variables corresponding to each membership function. The linguistic variables are the names (BN big negative, SP small positive, Z zero, etc.) given to the fuzzy sets.

The fuzzy inference engine computes all the antecedents of the rule-base by using either the union or intersection of the linguistic variables. This will give to the rule some degree of truth for the consequent. Finally, the defuzzification part combines the outputs of the rules to give a crisp output value u for the controller. The most effective method computationally is having a singleton in the consequents and using the weighted sum

$$u(e, \dot{e}) = \frac{\sum_j min[\mu A^j(e), \mu B^j(\dot{e})] O^j}{\sum_j min[\mu A^j(e), \mu B^j(\dot{e})]} \tag{3}$$

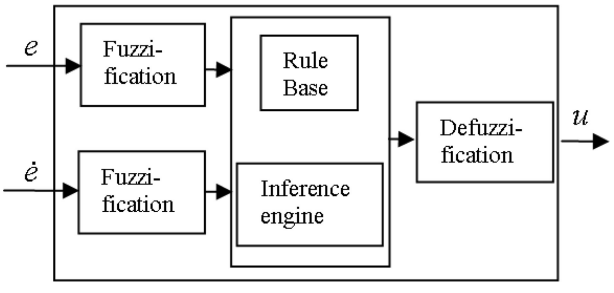

Figure 1. Fuzzy controller.

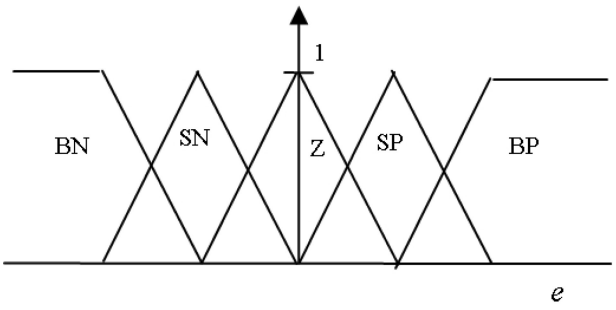

Figure 2. An example of universe of discourse and membership function.

where μA and μB are the membership functions to evaluate the jth rule, for the values given in e and $\dot{e}$, respectively, and O is the consequent part of the rule.

Fuzzy-Lyapunov synthesis

Consider the nonlinear system

$$\dot{x} = f(x(t)) \tag{4}$$

where $x \in \Re^n$ is an n vector and $f : D \to \Re^n$ with $D \in \Re$. Let the equilibrium point of the system be $\bar{x}$, so $f(\bar{x}) = 0$. We say that $\bar{x}$ is stable in the sense of Lyapunov if for every $\epsilon > 0$ there exists a $\delta > 0$ such that

$$|x(t_0) - \bar{x}| < \delta \Rightarrow | x(t) - \bar{x} | < \epsilon \tag{5}$$

for all $t > t_0$. We say that $\bar{x}$ is asymptotically stable if it is stable and,

$$|x(t) - \bar{x}| \to 0 \text{ as } t \to 0. \tag{6}$$

To determine the stability of $\bar{x}$, Lyapunov introduced two main methods. The first is called Lyapunov's first or indirect method: the linearisation technique. Consider the linearisation of (4) about $\bar{x}$

$$\dot{x} = Ax \tag{7}$$

where

$$A = \left.\frac{\partial f(x)}{\partial x}\right|_{\bar{x}}$$

is the Jacobian matrix of $f(x)$ evaluated at $\bar{x}$, and x has been redefined as $x - \bar{x}$. Then, the nonlinear system (4) is asymptotically stable if the linear system (7) is; i.e., if all eigenvalues of A have negative real parts.

One disadvantage of the method is that if some eigenvalues of A are zero and the rest have negative real parts, then we cannot draw any conclusions on the nonlinear system: the equilibrium $\bar{x}$ can be either stable or unstable.

Lyapunov's second or direct method: this is a generalisation of Lagrange's concept of stability of minimum potential energy. Consider the nonlinear system (4) and suppose that there exists a function $V(x)$, called a Lyapunov function, with the following properties:

- $V(\bar{x}) = 0$
- $V(x(t)) > 0$, for $x \neq \bar{x}$
- $\dot{V}(x(t)) < 0$, along trajectories of $\dot{x} = f(x(t))$

Then (4) is asymptotically stable. Classical Lyapunov synthesis suggests the design of a controller that should guarantee $\dot{V}(x) < 0$ for a Lyapunov function $V(x)$. Fuzzy-Lyapunov synthesis follows the same idea, making $\dot{V}(x)$ *Negative*, using the linguistic description of the plant and controller.

For example, consider the nonlinear affine system

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x_1, x_2) + g(x_1, x_2)u. \end{aligned} \tag{8}$$

Let the Lyapunov function candidate be

$$V(x) = \frac{1}{2}\left(x_1^2 + x_2^2\right) \tag{9}$$

differentiating $V(x)$ yields

$$\begin{aligned} \dot{V}(x) &= \frac{\partial V}{\partial x}\dot{x} \\ \dot{V}(x) &= x_1 x_2 + x_2 \dot{x}_2 \\ \dot{V}(x) &= x_2(x_1 + f(x) + g(x)u). \end{aligned} \tag{10}$$

The linguistic description is given as

$$\begin{aligned} LV(\dot{V}(x)) = LV(x_2)[&LV(x_1) + \\ &LV(f(x)) + LV(g(x))LV(u)] \end{aligned} \tag{11}$$

where $LV(\cdot)$ indicates the linguistic variable. From Zhou (2002) we have,

Theorem 1. *If $V(x)$ is a Lyapunov function and the linguistic variable $LV(\dot{V}(x)) = Negative$, where we have $Supp(Negative) \subset (-\infty, 0]$, then the fuzzy controller designed by Fuzzy-Lyapunov synthesis is locally stable. Furthermore, if $Supp(Negative) \subset (-\infty, 0)$ then the stability is asymptotic*

If $f(x)$ and $g(x)$ are unknown we can approximate them by $\tilde{f}(x)$ and $\tilde{g}(x)$ using a fuzzy model. The following rules are examples of rules acquired from (11) to design the fuzzy controller and achieve a stable system:

R1: IF x_2 is P and x_1 is P and $\tilde{f}(x)$ is P and $\tilde{g}(x)$ is P THEN u is N
R2: IF x_2 is N and x_1 is P and $\tilde{f}(x)$ is P and $\tilde{g}(x)$ is P THEN u is P
R3: IF x_2 is P and x_1 is P and $\tilde{f}(x)$ is P and $\tilde{g}(x)$ is N THEN u is P
R4: IF x_2 is N and x_1 is N and $\tilde{f}(x)$ is P and $\tilde{g}(x)$ is N THEN u is BN.

Here P, N and BN are the linguistic variables *Positive*, *Negative* and *BigPositive*, respectively. It can be said that the logic of our controller is correct, but there exists a drawback: how to design the parameters of the membership functions. The proposed method is to use a PSO to adapt the parameters off-line and a fuzzy adapter for on-line adaptation. The adaptation of the controller is based on a desired Lyapunov function.

Swarm intelligence

Optimisation theory is the branch of applied mathematics and numerical analysis that deals with the optimisation of single or multiple, possibly even conflicting, criteria. These criteria are expressed as a set of mathematical functions $F = [f_1, f_2, \ldots, f_n]$, the so-called objective functions. The result of an optimisation process is the set of inputs for which these objective functions return optimal values (Weise 2007).

Definition 5. *Local maximum $\hat{x} \in D$ of an objective function $f : D \leftarrow \Re$ is an input element with $f(\hat{x}) > f(x)$ for all x in a neighborhood of $\hat{x}$. Thus*

$$\hat{x} : \exists \epsilon > 0 : f(\hat{x}) > f(x) \forall \in D, |x - \hat{x}| < \epsilon.$$

Definition 6. *Global maximum $x^* \in D$ of an objective function $f : D \leftarrow \Re$ is an input element with $f(x^*) > f(x)$ for all x. Thus*

$$x^* : f(x^*) > f(x) \forall \in D.$$

Theorem 2. *A point $x \in D$ is a maximum of f on D if and only if x is a minimum of $-f$ on D.*

In optimisation theory there are two major categories: deterministic and probabilistic. In the deterministic category, the most common technique uses the gradient information to obtain the searching direction. In the multivariable problem the Jacobian of the function indicates the search direction, but the calculation of the Jacobian is computational expensive and could lead towards a local minimum. In the probabilistic category there is a trade-off between finding the optimal solution and the search

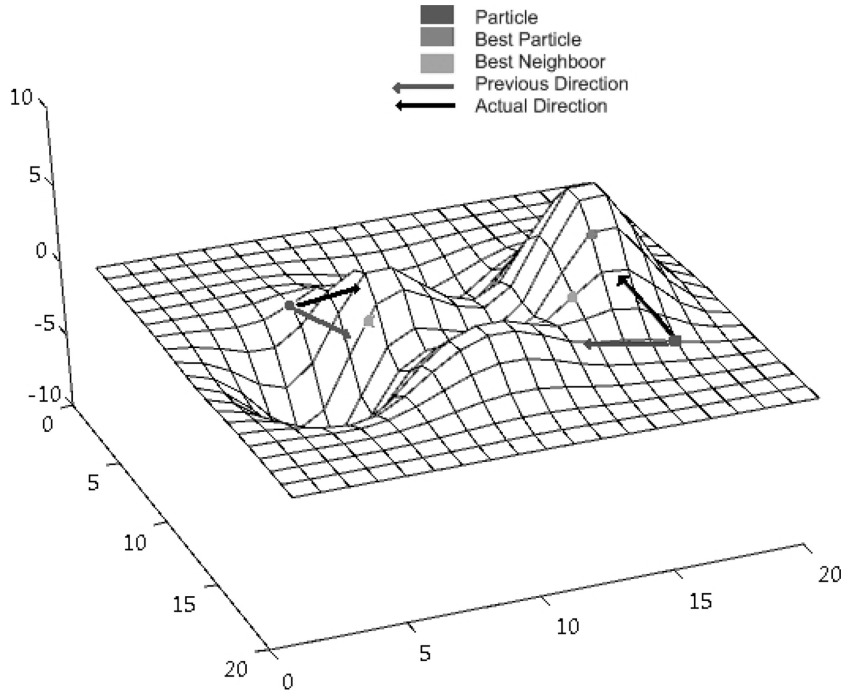

Figure 3. PSO model.

time. Many of these methodologies use a metaheuristic to improve the speed of the algorithm. Examples of these categories are genetic algorithms and simulated annealing searches.

To avoid the problem or searches being trapped at local minima, recent research in optimisation has led to swarm intelligence techniques. These algorithms are based on population interaction with cooperation among individuals or particles, rather than competition. Three interesting swarm intelligence techniques currently in existence are: ant colony optimization (ACO), stochastic diffusion search (SDS) (Myatt et al. 2003) and particle swarm optimisation (PSO).

PSO is based on the social behaviour of bird flocking, and thus the potential solutions in PSO are said to fly through the problem space (objective function) looking for the best solution, based on three basic steps:

(1) Evaluation
(2) Comparison
(3) Imitation

PSO initialises a population of particles with random positions $\vec{p}$ and velocities $\vec{v}$, and then an objective function is used to evaluate each particle position. After this it compares them with the best particle global position $\vec{p}_b$ and best particle neighbour position $\vec{p}_{bn}$ and finally it adjusts the position of each particle, trying to imitate the best by a random amount $\alpha_i \in [0, 1]$. The following equations are principal components of the PSO,

$$\vec{v}_{k+1} = w\vec{v}_k + \alpha_1(\vec{p}_b - \vec{p}_k) + \alpha_2(\vec{p}_{bn} - \vec{p}_k) \quad (12)$$

$$\vec{p}_{k+1} = \vec{p}_k + \vec{v}_{k+1} \quad (13)$$

Figure 3 gives a visual understanding of the interaction between particles with the best global particle and best neighbour. The inertia weight, w, is employed to control the impact of the previous history of velocities on the current velocity (Eberhart and Shi 1998), thus to influence the trade-off between global (wide-ranging) and local (nearby) exploration abilities of the "flying points". A larger inertia weight, w, facilitates global exploration (searching new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Transforming (12) and (13) to a simplified dynamic system, we can obtain

$$P_{k+1} = AP_k$$

where

$$P_k = \begin{bmatrix} v_k \\ y_k \end{bmatrix} \quad A = \begin{bmatrix} w & \alpha \\ -w & 1-\alpha \end{bmatrix} \quad y_k = p_b - p_k$$

These equations show that a suitable selection of the inertia weight, w and α can provide a balance between global and local exploration abilities and thus require fewer iterations on average to find the optimum. If the eigenvalues

of A are complex, the system is cyclic, and these oscillations prevent getting trapped in a local minimum, while the real negative eigenvalues of A allow a rapid convergence towards the best position. The analysis of stability and convergence of PSO has been presented in other works like Clerc and Kennedy (2002) and Eberhart and Shi (1998). Their algorithm in pseucode is:

```
Loop
    For i= 1 to Population size
        if f(p_i) < f(p_b) then
          For d = 1 to dimension
              p_b = x_id // p_b is best so far
          Next d
        End if
        p_bn = min(pneighbours)
        For d = 1 to dimension
          v_id = w * v_id + α1(p_b − p_id) + α2(p_bn − p_id)
          p_id = p_id + v_id
        Next d
    Next i
Until criterion
```

Particle swarm optimisation is easy to implement and computationally efficient. In our case, the particles contain the parameters of membership functions, such as centers, widths and weights. The number of parameters is the dimension of the particle. The PSO is applied off-line to tune the membership functions. The performance index to stop the algorithm could be given by:

$$\sum_{t=0}^{t_f} \|y(t) - y_d(t)\| + \beta \sum_{t=0}^{t_f} \|u(t)\| < \epsilon \tag{14}$$

where $y_d(t)$ is the desired trajectory, $y(t)$ is the system output, $u(t)$ is the input control, β is a constant and ϵ is a small constant. The second term of Equation (14) is to obtain the minimum input control $u(t)$, and is controlled by β.

Adaptive fuzzy controller

After the design of a controller, it may be tuned to improve its performance when it is applied to a real system. Systems that learn from their environment have the ability to improve their performance over time. Like fuzzy controllers have the capability to include the experience of expert into the fuzzy rules in a linguistic manner, similarly fuzzy adapters are constructed, improving the performance of our fuzzy controller.

Choosing a Lyapunov function is not a simple task, and in complex systems is even harder. If the rules require $\dot{V}(x)$ to be just *Negative* and the approximations of the system are not accurate enough, the fuzzy controller could yield a different result.

Figure 4 shows the diagram of the fuzzy controller and adapter. The fuzzy adapter evaluates the error obtained after the fuzzy controller is applied to the plant and adapts the consequents of the fuzzy controller rules following desired criteria. Defining the error for the fuzzy adapter as $e_{adp} = x_{k+1} - y_d$, we could define the rules with two membership functions as:

Ra1: IF e_{adp} is P and $\dot{e}_{adp}$ is P THEN $\triangle u$ is N
Ra2: IF e_{adp} is P and $\dot{e}_{adp}$ is N THEN $\triangle u$ is Z
Ra3: IF e_{adp} is N and $\dot{e}_{adp}$ is P THEN $\triangle u$ is Z
Ra4: IF e_{adp} is N and $\dot{e}_{adp}$ is N THEN $\triangle u$ is P

The construction of the rules is based on expert knowledge. If the error and its derivative have the same sign, we have to correct the behaviour of the controller for that condition, and if they have the different signs, then the controller is well-behaved and no correction is made to the controller. The adaptation will be given only to the consequents of the fuzzy controller rules, which were applied for input control in the process. Therefore, the adaptation rule for the consequent of rule R_i is given by

$$O_i(t+1) = O_i(t) + \triangle u * d(e, \dot{e}) \tag{15}$$

where $d(\cdot)$ is the contribution of rule R_i to the consequent of the controller rule-base. This is given into (3) as $\frac{min[A_i(e), B_i(\dot{e})]O_i}{\sum_j^i min[A^j(e), B^j(\dot{e})]}$. The membership function parameters will dictate the degree of adjustment needed in our controller. Here the idea is to obtain the parameters of the adapter and controller so that the PSO yields a negative derivative of Lyapunov function $\dot{V}_d(x)$. One way to choose $\dot{V}_d(x)$ is $\dot{V}_d(x) \leq -\phi V(x)$, with $\phi > 0$, that will give an exponential convergence rate.

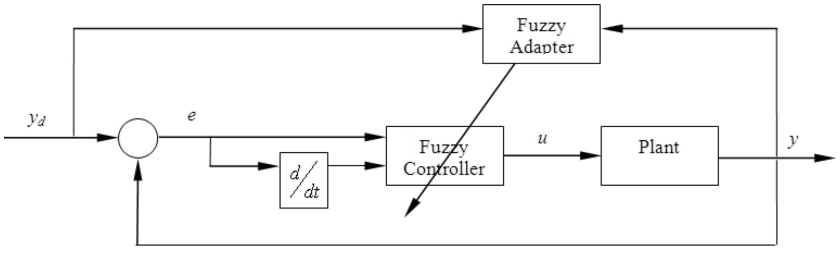

Figure 4. Schematic representation of adaptive fuzzy controller.

The new performance index criteria is given by

$$\sum_{t=0}^{t_f} \|y(t) - y_d(t)\| + \beta \sum_{t=0}^{t_f} e_v(t) < \epsilon \tag{16}$$

where $e_v(t)$ is the deviation of the desired derivative Lyapunov function, given by

$$e_v(t) = \begin{cases} 0 & \dot{V}(x) \le \dot{V}_d(x) \\ \left\| \dot{V}(x) - \dot{V}_d(x) \right\| & \dot{V}(x) > \dot{V}_d(x) \end{cases} \tag{17}$$

Although we cannot explicitly solve for $\dot{V}(x)$ in (17), we can approximate it by

$$\dot{V}(x) \approx \triangle V(x) = \frac{V_{k+1}(x) - V_k(x)}{\triangle t}$$

Since $\dot{V}_d(x)$ is less than $-\phi V(x)$, the closed-loop system will be stable. We can also make use of a fuzzy constraint instead of a crisp constraint. After applying the PSO, the adapter learns how to modify the fuzzy controller to keep the closed-loop system stable and to enable tracking a reference.

Inverted pendulum application

The inverted pendulum (see Figure 5) is frequently used as a benchmark dynamic nonlinear plant for evaluating a control algorithm or a combination of control algorithms. The dynamic equations of inverted pendulum (Slotine and Li 1991) are described by (8), where

$$f(x_1, x_2) = \frac{9.81 \sin x_1 - \frac{mlx_2^2 \cos x_1 \sin x_1}{mc+m}}{l\left(\frac{4}{3} - \frac{m \cos^2 x_1}{mc+m}\right)} \tag{18}$$

$$g(x_1, x_2) = \frac{\frac{\cos x_1}{mc+m}}{l\left(\frac{4}{3} - \frac{m \cos^2 x_1}{mc+m}\right)} \tag{19}$$

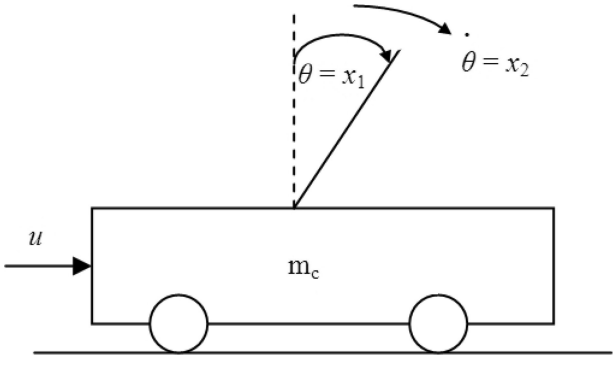

Figure 5. Inverted pendulum.

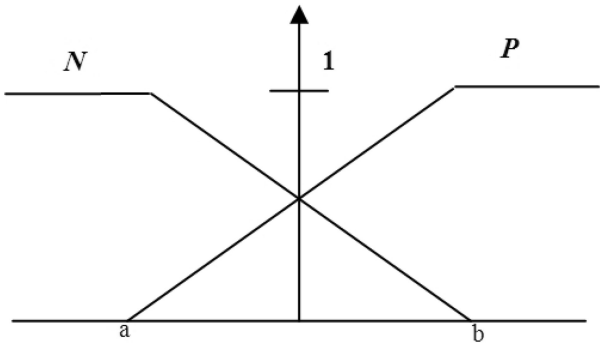

Figure 6. Membership functions.

where $2l$ is the pole's length, mc is the mass of the cart, m is the mass of the pole and u is the control. In this case $mc = 0.75$, $m = 0.25$ and $l = 0.75$. First, we construct a reduced fuzzy controller by defining only two membership functions for each variable x_1 and x_2. Figure 6 shows the shape of membership functions *Negative* and *Positive*, which are the same for both variables, x_1 and x_2.

The equations of membership function *Negative* and *Positive* are

$$\mu N(x) = \begin{cases} 0 & x \ge b \\ \frac{b-x}{b-a} & a < x < b \\ 1 & x \le a \end{cases} \tag{20}$$

$$\mu P(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a < x < b \\ 1 & x \ge b \end{cases} \tag{21}$$

Using Lyapunov synthesis to obtain the control law of the system, with (9) as the candidate Lyapunov function we obtain

$$\dot{V}(x) = x_2(x_1 + f(x) + g(x)u) \tag{22}$$

defining the $\dot{V}_d(x)$ as

$$\dot{V}_d(x) = -\phi\left(x_1^2 + x_1^2\right) \tag{23}$$

from Equations (22) and (23) we have

$$x_2(x_1 + f(x) + g(x)u) = -\phi\left(x_1^2 + x_1^2\right) \tag{24}$$

Choosing $a_1 = -0.25$ and $b_1 = 0.25$ radians for x_1 and $a_2 = -1$ and $b_2 = 1$ rad/s for x_2, we can obtain the Table 1 even though we have just an approximation of our system, $\tilde{f}(x)$ and $\tilde{g}(x)$

Therefore the rules for the fuzzy control are

R1: IF x_1 is P and x_2 is P THEN u is -10.5
R2: IF x_1 is P and x_2 is N THEN u is -3.2
R3: IF x_1 is N and x_2 is P THEN u is 3.2

Table 1. Derivation of control law

x_1	x_2	ϕ	$\tilde{f}(x)$	$\tilde{g}(x)$	u
P	P	10	3.2	1.3	−10.5
P	N	1	3.2	1.3	−3.2
N	P	1	−3.2	1.3	3.2
N	N	10	−3.2	1.3	10.5

R4: IF x_1 is N and x_2 is N THEN u is 10.5

Using Takagi–Sugeno inference the output of the fuzzy controller is given by

$$u(x_1, x_2) = \frac{\sum_j h_j(x_1, x_2) O_j}{\sum_j h_j(x_1, x_2)} \tag{25}$$

where h_j is the premise of the rule R_j and O_j its conclusion. Figure 7 shows that the control u successfully stabilises the pole for initial conditions $x_1(0) = -0.1$, $x_1(0) = 0.25$ and $x_1(0) = 0.6$, with $x_2(0) = 0$ for each of them.

The purpose of the controller is not only to maintain the pole in the upright position, but also to track a desired trajectory. Figure 8 shows the pole's angle with same fuzzy controller tracking a sinusoidal reference $r = 0.25\sin(t)$ and initial condition $x_1(0) = -0.1$ rad $x_2(0) = 0$ rad/s and Figure 9 shows the error between the reference r and x_1. It can be seen that the controller is not close to tracking the desired reference. This is because the controller is not designed for it.

Now we introduce the fuzzy adapter to adjust on-line the parameters O_j of the conclusions of the rules from the fuzzy controller. Defining the error and derivative for the fuzzy adapter as

$$e_{adp} = x_1^{k+1} - y_d^k \dot{e}_{adp} = x_2^{k+1} - \dot{y}_d^k$$

and choosing the membership functions as in Figure 6 for both of them, with parameters of e_{adp} as $a_1^{apd} = -0.01$, $b_1^{apd} = 0.01$ and parameters of $\dot{e}_{adp}$ as $a_2^{apd} = -0.05, b_2^{apd} = 0.05$, gives the rule-base as

Ra1: IF e_{apd} is P and $\dot{e}_{apd}$ is P THEN $\triangle u$ is -2
Ra2: IF e_{apd} is P and $\dot{e}_{apd}$ is N THEN $\triangle u$ is -1
Ra3: IF e_{apd} is N and $\dot{e}_{apd}$ is P THEN $\triangle u$ is 1
Ra4: IF e_{apd} is N and $\dot{e}_{apd}$ is N THEN $\triangle u$ is 2

The output of the fuzzy adapter has the same form as (25), and the adaptation rule is given by (15). The parameters for the fuzzy adapter were chosen by the knowledge of the expert. Figure 10 shows the performance of the system tracking the same sinusoidal reference with the fuzzy controller and fuzzy adapter working together.

It is clear that the parameters of the fuzzy adapter are not appropriate to correct the controller. In this part we have two alternatives: (a) increase the rule-base of the adapter to perform dynamically focused learning (DFL) (Passino and Yurkovich 1998), or (b) optimise the parameters of our adapter and controller. The second option requires an optimisation method to find the parameters, but swarm

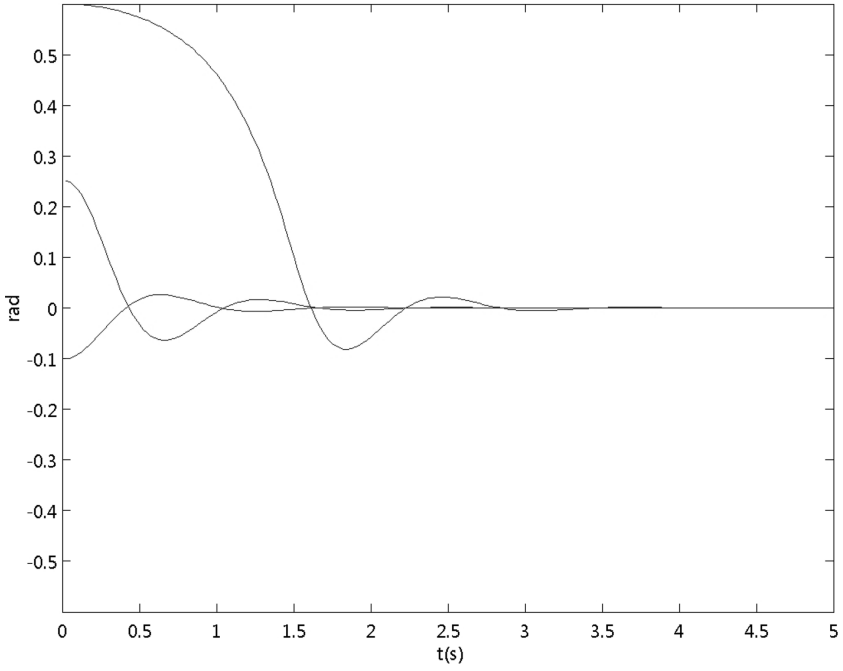

Figure 7. Pole's angle x_1 using different initial conditions.

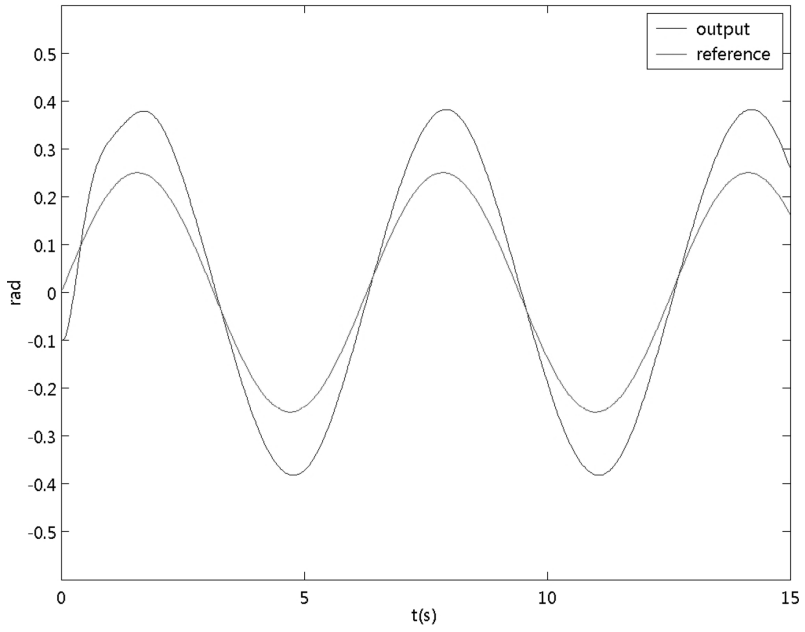

Figure 8. Pole's angle x_1 and reference r using fuzzy controller.

intelligence optimisation, particularly PSO, is easy to implement and can be used to avoid local minima. Also, the controller and adapter stay computationally efficient, which is desirable for real time control of a fast process.

The variables for PSO are: particles $\vec{p}$, velocities $\vec{v}$, dimension *dim*, number of particles n_p global best particle $\vec{p}_b$, best neighbour particle $\vec{p}_{bn}$, performance index p_{ind}, max. iterations *iter* and inertia weight w. The initialisation is as follows,

$$
\begin{aligned}
dim &= 12 \\
n_p &= 10 \\
\vec{p}_b &= [\,-10.5\ -3.2\ 3.20\ 10.5\ 0.25\ 1.00 \\
&\qquad -2.0\ -1.0\ 1.00\ 2.00\ 0.01\ 0.05]
\end{aligned}
$$

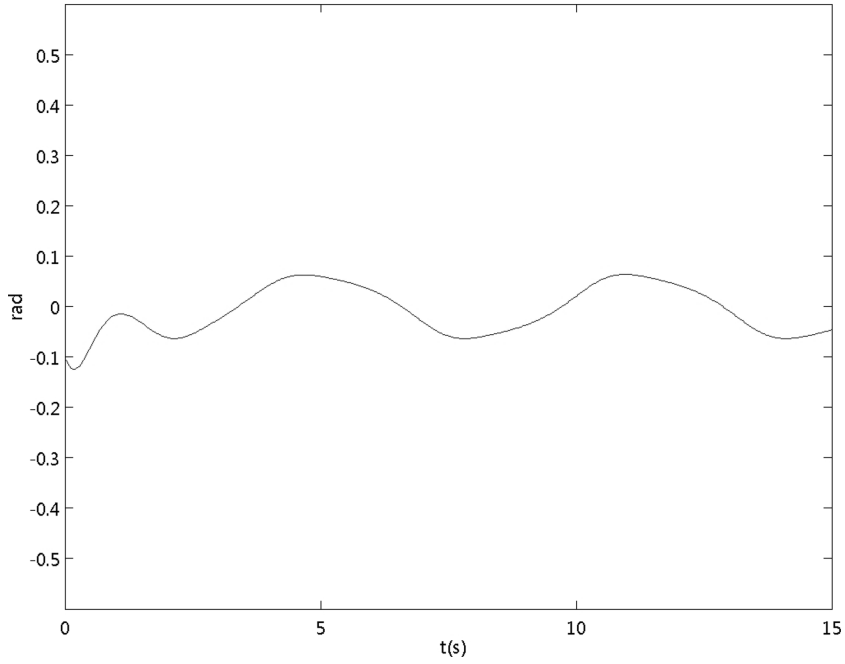

Figure 9. Tracking error using fuzzy controller.

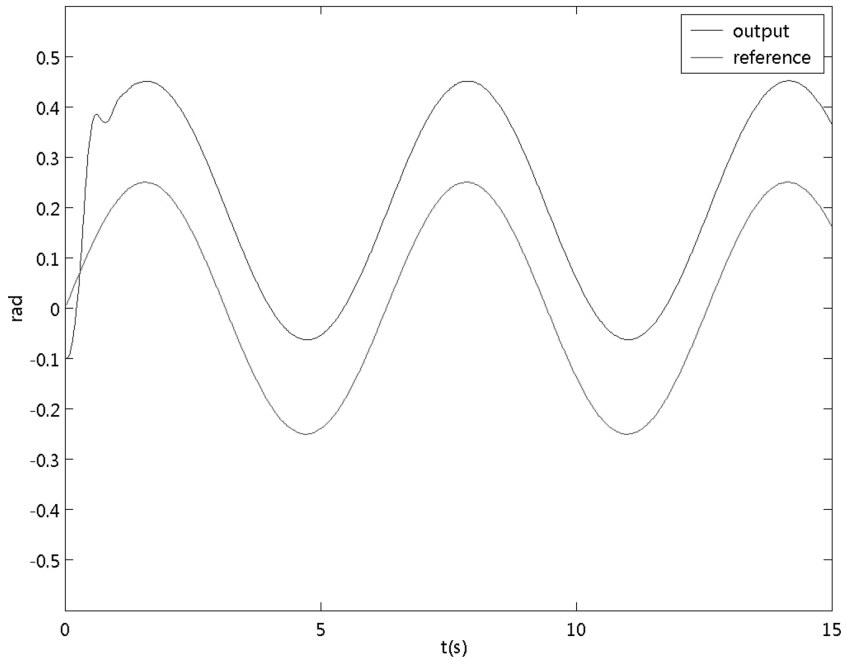

Figure 10. Pole's angle and reference using fuzzy controller and fuzzy adapter before PSO.

$$\vec{p}_i = \vec{p}_b + 0.1 * randn$$
$$\vec{p}_{bn} = \vec{p}_{10}$$
$$\vec{v}_i = 0.1 * \vec{p}_b$$
$$iter = 50$$
$$w = 0.7$$
$$p_{ind} = 30{,}000$$

The initial particles are small deviations of the best particle, which in this case include the parameters of the fuzzy controller and fuzzy adapter previously obtained. The 5th and 6th elements of the particles are the center for

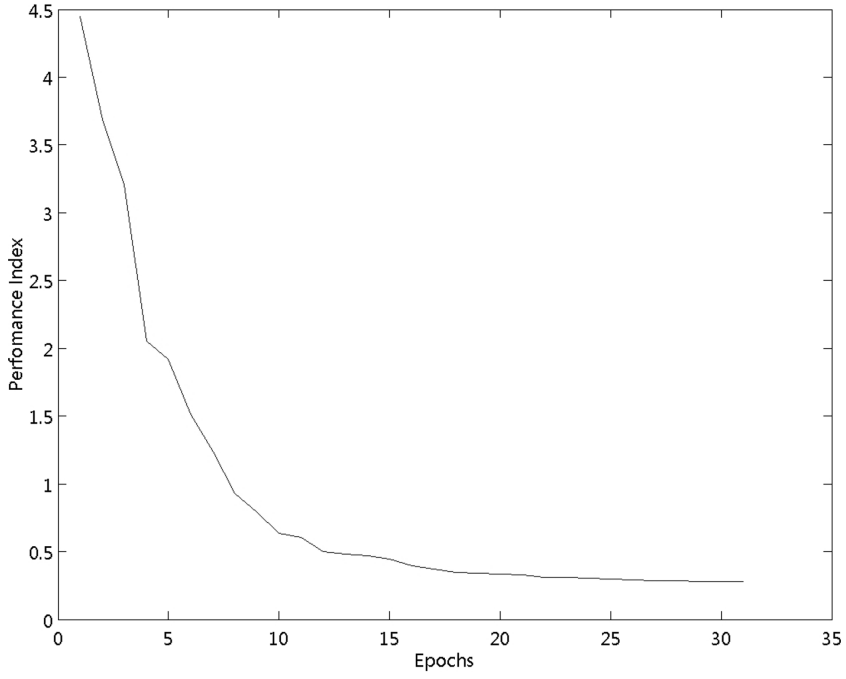

Figure 11. Convergence of PSO.

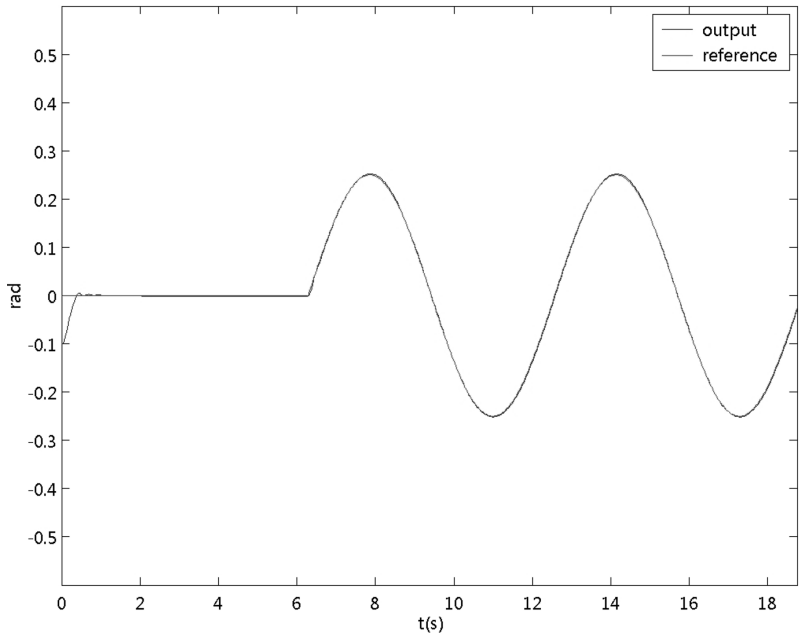

Figure 12. Tracking a reference using fuzzy controller and adapter after PSO.

the membership functions, in this case $a_1 = -b_1$ and $a_2 = -b_2$, and similarly for the 11th and 12th elements.

Figure 11 shows the convergence of PSO when training the controller and adapter using the performance index in (16) with $\beta = 0.1$ and the next reference signal

$$r = \begin{cases} 0 & t < 6.25seg \\ 0.25\sin(t) & t \geq 6.25seg \end{cases}$$

Figure 12 shows the performance of the system tracking the reference, and Figure 13 shows the tracking error of the best particle found by PSO. The best particle after optimisation is

$$\vec{p}_b = [\,-32.28\ -3.749\ 3.945\ 29.83\ 0.281\ 0.877\ 0.72\ -0.086\ -1.56\ -2.69\ 0.017\ 0.089]$$

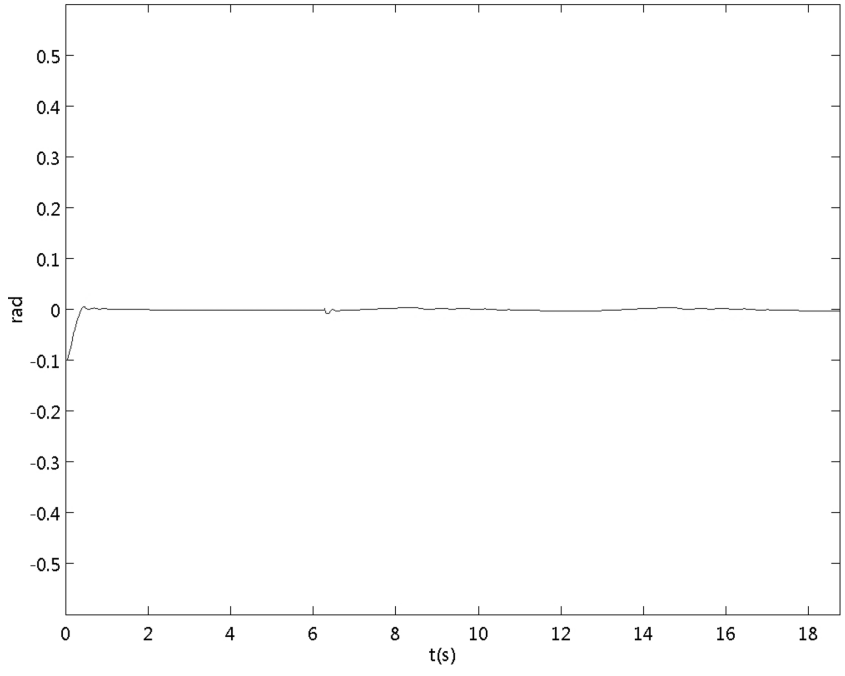

Figure 13. Tracking error after PSO.

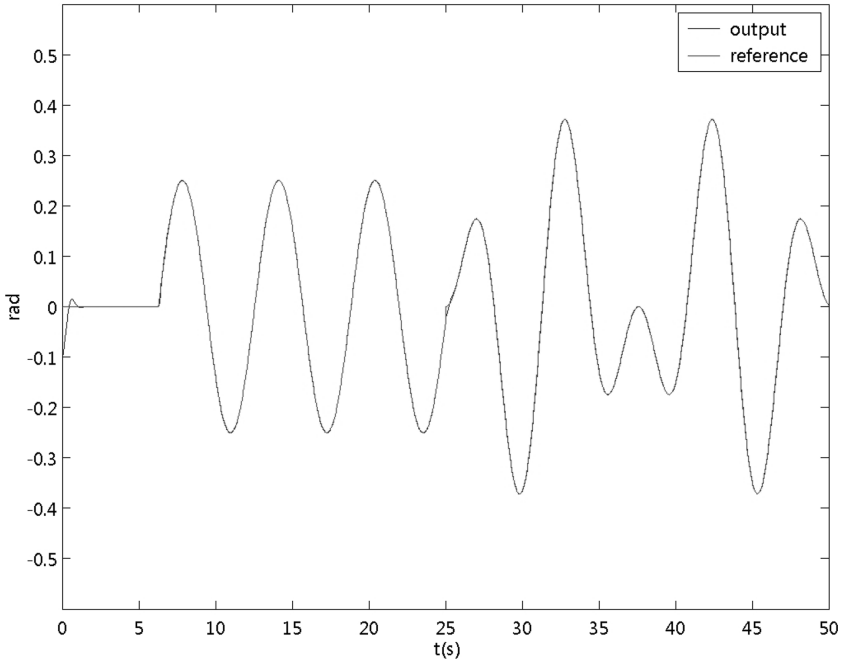

Figure 14. Adaptive fuzzy control for different settings.

Some aspects of PSO implementation deserve mention. The consequents of the fuzzy controller are only allowed to vary up to four times the original values obtained for the Fuzzy-Lyapunov synthesis. This is because when the gains reach the high values in real systems, chattering in the control signal occurs, similar to what happens in a variable structure system with a sliding mode. This chattering causes stress in the actuator, shortening its span life. The rule-base must be such that the centers of the membership functions do not have to cross each other; otherwise the logic of the rule-base changes undesirably.

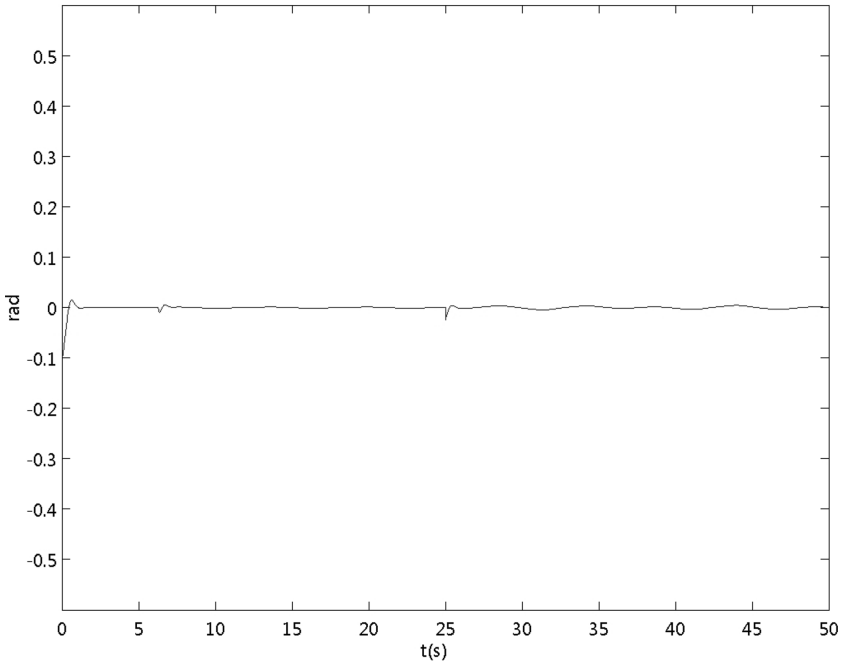

Figure 15. Tracking error for different settings.

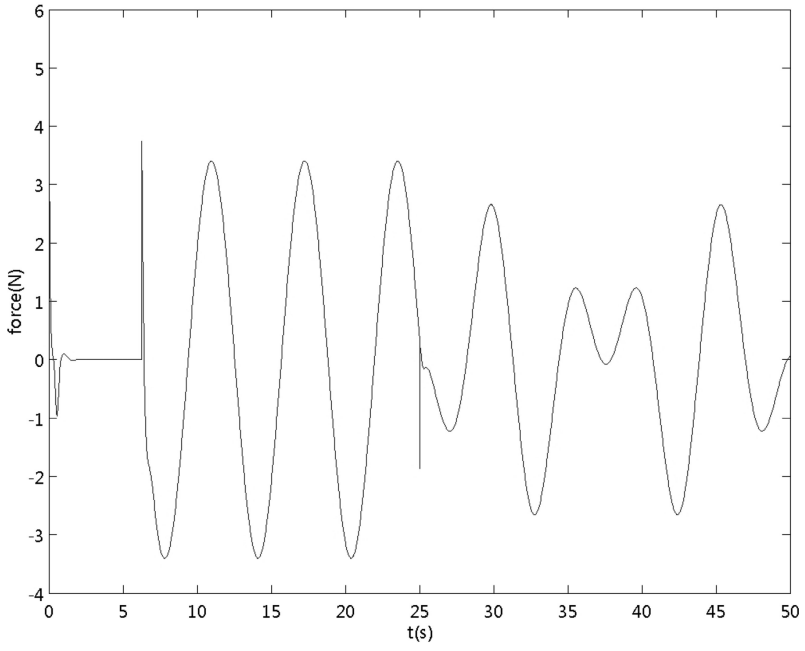

Figure 16. Control law for different settings.

Figure 14 shows the adaptive fuzzy controller output with a different reference and different values of pole length, pole mass and cart mass. The reference signal is given by

$$r = \begin{cases} 0 & t \leq 6.25seg \\ 0.25\sin(t) & 26seg > t > 6.25seg \\ 0.4\sin(t)\cos(0.25t) & t \geq 26 \end{cases} \quad (26)$$

The settings for the first part of the reference are, $l = 0.75$, $mc = 0.75$ and $m = 0.25$, for the second part of the reference, $l = 0.5$, $mc = 0.5$ and $m = 0.1$, and for the last part, $l = 0.75$, $mc = 0.8$ and $m = 0.5$. The adaptive controller is able to track the reference over a larger range than it was trained for and maintains good performance. Figure 15 shows that tracking error is not heavily penalised in spite of switching the parameters in the system and working out-

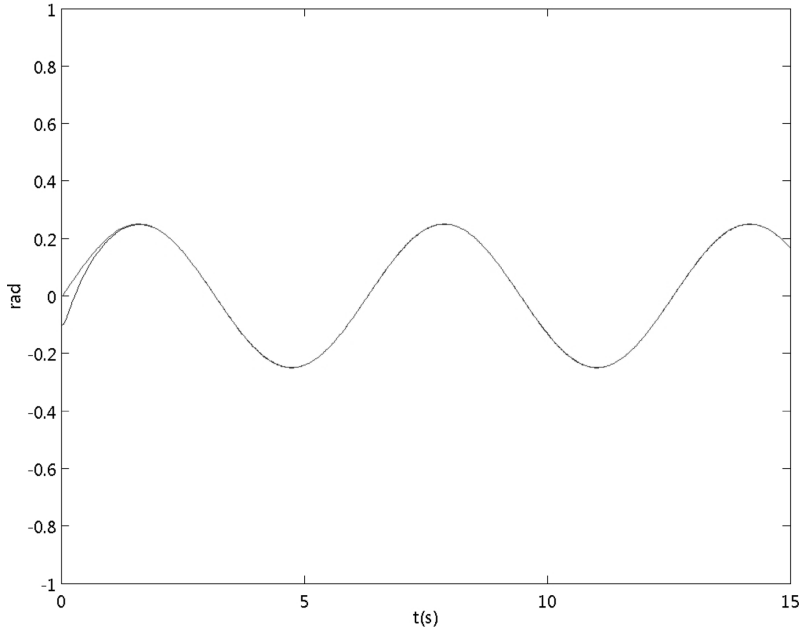

Figure 17. LQR with feedforward controller tracking a sinusoidal reference.

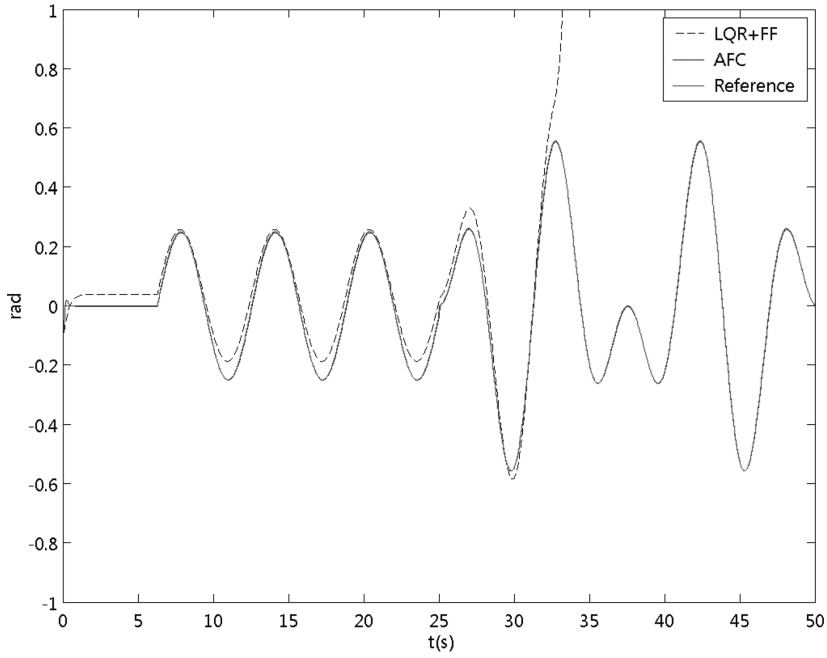

Figure 18. Tracking a reference in several conditions with LQR plus feedforward and adaptive fuzzy controller.

side the training range. Finally, Figure 16 shows the control signal.

We compare the adaptive fuzzy controller with the linear quadratic regulator (LQR) plus feedforward controller. Consider the linearisation of the inverted pendulum

$$\begin{aligned} \dot{x} &= Ax + Bu + d \\ y &= Cx \end{aligned} \tag{27}$$

with

$$A = \begin{bmatrix} 0 & 1 \\ 13.1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 1.3 \end{bmatrix}$$

$$C = [1\ 0]$$

The proportional and derivative gains for the LQR are $K_p = -43.26 \quad K_d = -12.9$ and for feedforward, $K_r = -10.8 \quad K_{dr} = -.25$ using

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix} \quad r = 0.01$$

Figure 17 shows the LQR plus feedforward controller tracking a sinusoidal reference. Although it shows a good performance, this will degrade if the uncertainty in the plant increases, the reference changes or the disturbance increases. Figure 18 shows the LQR plus feedforward and the adaptive fuzzy controller tracking the reference as in (26), including a disturbance $d^T = [0.05 \quad 0]$ and parameters shift. Figure 18 shows the limitation of the LQR controller. In contrast, the simplified adaptive fuzzy controller maintains good performance in the presence of uncertainty and a disturbance in the plant.

Conclusions

This work presented a methodology to design an adaptive Fuzzy-Lyapunov controller using biologically inspired swarm intelligence. Fuzzy controllers have been shown in different works to be a good option for many applications. A main characteristic of fuzzy controllers is the introduction of expert rule-based knowledge into the controller. This expert knowledge can come from different areas; in this paper, Lyapunov stability theory and particle swarm optimisation were applied to the fuzzy controller and fuzzy adapter. The design was divided into two stages: the controller design based on Lyapunov synthesis and the adapter optimised using PSO.

Although it is difficult to choose the Lyapunov function candidate for a particular system, our approach based on a desired Lyapunov function for the adaptive fuzzy controller is flexible. Our approach is suitable for complex systems, as it provides a simple manner to choose Lyapunov functions that yield good performance.

The paper also presented how PSO can be used in the optimisation of the parameters of fuzzy controller and fuzzy adapter. PSO shows a fast rate of convergence: in our exam-

ple, the parameters of the adaptive controller are optimised within 30 epochs. PSO is very easy to implement in practice since it does not require gradient computation, unlike gradient techniques. As explained in the section entitled 'Fuzzy adaptive controller', optimising the performance index using gradient techniques requires finding a derivative that cannot be solved explicitly.

In the last part of our work, we presented simulation results supporting the feasibility of the proposed approach. Adaptive fuzzy controller delivers stability and good response to drastic changes in the system parameters, unlike the LQR controller tested. Optimising the parameters of our adaptive fuzzy controller keeps the controller in a reduced form and computationally efficient, unlike other adaptive controllers. Future work related to our adaptive fuzzy controller includes determining how to select the smallest rule set and the appropriate number of fuzzy sets.

References

Carrasco A, Goldsmith P. 2004. Adaptive neuro-fuzzy control using swarm intelligence. In International Symposium on Robotics and Automation, Queretaro, Mexico, pp. 220–227.

Chen C-L, Chen W-C. 1994. Fuzzy controller design by using neural network techniques. IEEE Trans Fuzzy Syst. 2(3):235–244.

Clerc M, Kennedy J. 2002. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evolut Comput. 6(1):58–73.

Eberhart RC, Kennedy JA. 1995. A new optimizer using particle swarm theory. In Proceedings of Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, IEEE Service Center, Piscataway, NJ, pp. 39–43.

Eberhart RC, Shi YH. 1998. Comparison between genetic algorithms and particle swarm optimization. In 7th Annual Conference on Evolutionary Programming, San Diego, pp. 611–616.

Gambardella LM, Taillard E, Dorigo M. 1999. Ant colonies for the quadratic assignment problem. OPSEARCH. 50(2):167–176.

Ghorbel F, Hung JY, Spong MW. 1989. Adaptive control of flexible-joint manipulators. IEEE Control Syst Mag. 9(7):9-13.

Hasan T., Sen Z. 1999. A new fuzzy modelling approach for predicting the maximum daily temperature from a time series. Tr J Eng Environ Sci. 173–180.

Hill JH, Ydstie BE. 2004. Adaptive control with selective memory. Internat J Adapt Control Signal Process. 18:571–587.

Kirkpatrick S, Gelatt CD , Vecchi MP. 1983. Optimization by simulated annealing. Science. 220:671–680.

Maragos P. 2005. Lattice image processing: An unification of morphological and fuzzy algebraic systems. J Math Imaging Vision. 22:333–353.

Mnif F, Ghommem J. 2005. Genetic algorithms adaptive control for an underactuated system. Internat J Comput Cognition. 3(1):12–20.

Myatt DR, Bishop JM, Nasuto SJ. 2003. Minimum stable convergence criteria for stochastic diffusion search. Electron Lett. 40(2):112–113.

Passino K, Yurkovich S. 1998. Fuzzy control. Berkeley, California: Addison-Wesley Longman.

Roesener A, Barnes JW. 2006. An advanced tabu search approach to the airlift loading problem. In MORS Conference, US Air Force Academy, Colorado Springs, CO.

Slotine JJE, Li W. 1991. Applied nonlinear control. Englewood Cliffs, NJ: Prentice Hall.

Spooner JT, Passino JT. 1996. Stable adaptive control using fuzzy systems and neural networks. IEEE Trans Fuzzy Syst. 4:339–359.

Weise T. 2007. Global optimization algorithms: Theory and application. Online as e-book, - - - - http://www.it-weise.de.

Zhou C. 2002. Fuzzy-arithmetic-based Lyapunov synthesis in the design of stable fuzzy controllers: A computing-with-words approach. Int J Appl Math Comput Sci. 12(3):411–421.