Available online at www.sciencedirect.com
ScienceDirect
Mathematical and Computer Modelling 46 (2007) 24–37
MATHEMATICAL AND COMPUTER MODELLING
www.elsevier.com/locate/mcm

Fuzzy control of inverted pendulum and concept of stability using Java application

Yasar Becerikli[a,*,1], B. Koray Celik[b]

[a] *Kocaeli University, Department of Computer Engineering, 41040, Kocaeli, Turkey*
[b] *Kütahya Porselen Industry, Information Technology Department, 43001 Kütahya, Turkey*

Received 18 May 2006; accepted 15 December 2006

Abstract

In this paper, a fuzzy controller for an inverted pendulum system is presented in two stages. These stages are: investigation of fuzzy control system modeling methods and solution of the "Inverted Pendulum Problem" by using Java programming with Applets for internet based control education. In the first stage, fuzzy modeling and fuzzy control system investigation, Java programming language, classes and multithreading were introduced. In the second stage specifically, simulation of the inverted pendulum problem was developed with Java Applets and the simulation results were given. Also some stability concepts are introduced.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Fuzzy control; Java; Stability; Multithreading; e-learning

1. Introduction

As we move into the information area, human knowledge becomes increasingly important. So a theory is necessary to formulate human knowledge and heuristics in a systematic manner and put them into engineering systems, together with other information such as mathematical models and sensory measurements. This aspect is a justification for fuzzy systems in the literature and characterizes the unique feature of fuzzy systems theory. For many practical systems, important information comes from two sources: one source is human experts who describe their knowledge about the system in natural languages; the other is mathematical models that are derived according to physical laws and sensory measurements [2]. Therefore, we are faced with an important task of combining these two types of information into systems design. To manage this combination, we should answer the question of how to transform human knowledge and heuristic base into a mathematical model. Essentially, a fuzzy system performs this transformation [1,13–15].

Fuzzy systems are knowledge-based or rule-based systems that contain descriptive IF-THEN rules that are created from human knowledge and heuristics. Also fuzzy systems are multi-input–single-output mappings from a real-valued vector to a real-valued scalar, but for large scale nonlinear systems the multi-output mapping can be decomposed into a collection of single-output mappings as shown in Fig. 1 [5].

* Corresponding author. Tel.: +90 262 3351168; fax: +90 262 3351150.
E-mail addresses: ybecerikli@kou.edu.tr, ybecerikli@ieee.org (Y. Becerikli), celik@kutahyaporselen.com.tr (B.K. Celik).

[1] Also part time member of Halic University, Department of Computer Engineering, and Electronics and Telecommunication Engineering, Istanbul, Turkey.

0895-7177/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2006.12.004

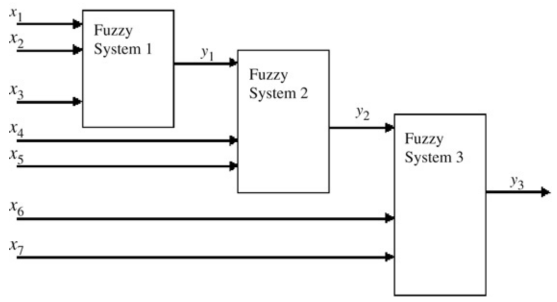


Fig. 1. Decomposing the multi-output mapping into a collection of a single-output mappings.

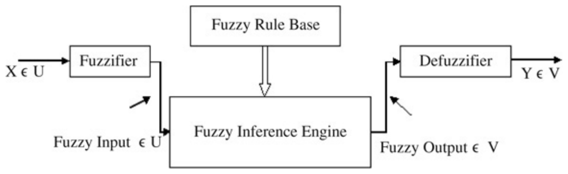


Fig. 2. General configuration of fuzzy system.

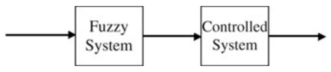


Fig. 3. Open-loop controller.

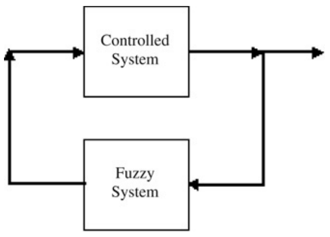


Fig. 4. Closed-loop controller.

An important contribution of fuzzy systems theory is that it provides a systematic procedure for transforming a knowledge base into a nonlinear mapping. So we can use this transformation in engineering systems (control) in the same manner as we use mathematical models and sensory measurements.

Consequently, by means of fuzzy systems, we can perform analysis and design of engineering systems in a mathematically rigorous manner [3].

Fuzzy systems have been applied to a wide variety of fields ranging from control, signal processing, communication, medicine, expert systems to business, etc. However, most significant applications have concentrated on control problems. The fuzzy systems that are shown in Fig. 2 can be used either as closed-loop controllers or open-loop controllers. As shown in Fig. 3, when the fuzzy system is used as an open-loop controller, the system usually sets up control parameters and then the system operates according to these parameters. When it is used as a closed-loop controller as shown in Fig. 4, the fuzzy system takes the outputs of the controlled system and applies the control action on the controlled system continuously. In this figures, the controlled system can be considered as an application process [3–5].

The goal of this text is to show how transformation of a knowledge base into a nonlinear mapping is done, and how analysis and design are performed on control systems. As a nonlinear system, the inverted pendulum system is often used as a benchmark to achieve the goal of verifying the performance and effectiveness of a control method because

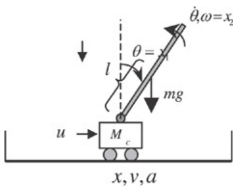


Fig. 5. Schematic diagram of inverted pendulum system.

of its simple structure. Recently, a lot of research on control of the inverted pendulum system by using fuzzy control systems containing fuzzy inference have been done.

Margaliot [6] showed a new approach to determining the structure of fuzzy controllers for inverted pendulums by fuzzy Lyapunov synthesis. Yamakawa [7,6] demonstrated a high-speed fuzzy controller hardware system and used only seven fuzzy rules to control the angle of the inverted pendulum system in 1989. Although stabilization control of an inverted pendulum system should also include the position control of the cart besides the angular control of the pendulum because of the limited length of the rail, the above stated approaches only took into consideration the angular control of the pendulum. Yubazaki [9,10] built a new fuzzy controller for inverted pendulum systems. The fuzzy controller has four input items, each with a dynamic importance degree [8].

In this paper, a fuzzy controller for the inverted pendulum system that needs two input items, of which one is the angle between the pendulum and the vertical position, and the other is the derivation of the angle (angular velocity) of the pendulum, is presented simply for educational purposes. The fuzzy controller takes the angle and angular velocity of pendulum from the inverted pendulum system, aggregates inputs with defined IF-THEN rules and derives the obtained force as an output item by means of inference methods.

However, recently, obtaining information resources quickly has become increasingly important. So, by using internet technologies (Java Applets), designing a simulation program for a fuzzy controller of an inverted pendulum system is unavoidable. The reason for choosing Java Applet technology arises from supporting all features necessary for extending the Web in ways previously impossible and Java is based on object-oriented technology that has evolved in diverse segments of computer science as a means of managing the complexity inherent in many different kinds of system [18].

2. Inverted pendulum system

The inverted pendulum system defined here is shown in Fig. 5, which is formed from a cart, a pendulum and a rail for defining the position of the cart. The pendulum is hinged in the center of the top surface of the cart and can rotate around the pivot in the same vertical plane with the rail. The cart can move right or left on the rail freely. It is given that no friction exists in the system between the cart and the rail or between the cart and the pendulum [5].

The dynamic equation of the inverted pendulum system can be expressed as [2,11,12],

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 = \ddot{\theta} = \dot{\omega} = \alpha &= \frac{g\sin\theta - \frac{m_p l\dot{\theta}^2\cos\theta\sin\theta}{m_c+m_p} + \frac{\cos\theta}{m_c+m_p}u}{l\left(\frac{4}{3} - \frac{m_p\cos^2\theta}{m_c+m_p}\right)}\end{aligned} \tag{1}$$

where the parameters, m_c and m_p are, respectively, the mass of the cart and the mass of the pendulum in the unit (kg), and $g = 9.8\ \text{m/s}^2$ is the gravity acceleration. The parameter l is the half length of the pendulum in the unit (m). The variable u means the control force in the unit (N) applied horizontally to the cart. The variables θ, ω, α represent the angle between pendulum and upright position, the angular velocity and the angular acceleration of the pendulum, respectively. Also, the clockwise direction is positive. The variables x, v, a denote the position of the cart from the rail origin, its velocity, its acceleration and the right direction is positive. The variables θ, ω, x, v are the four state variables to describe the dynamic system, but we use the variables θ and $\omega(\dot{\theta})$ to control the inverted pendulum system. However, we use the variable a to control the position of the cart in the java application but do not it use in the fuzzy control.

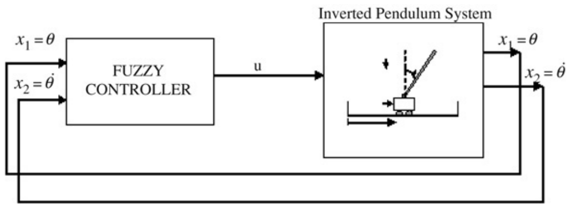


Fig. 6. Inverted pendulum fuzzy control system.

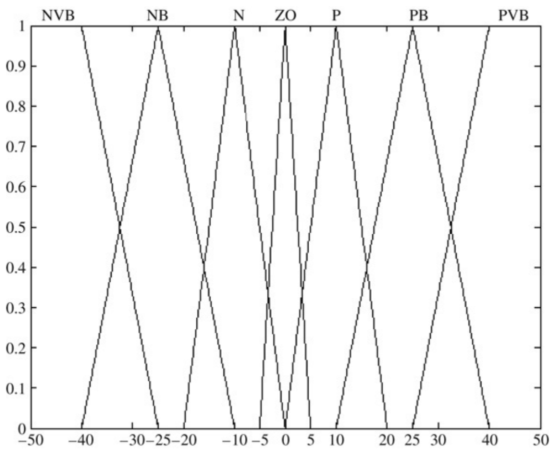


Fig. 7. Membership functions for state variable x_1.

3. Fuzzy control system of inverted pendulum

The inverted pendulum fuzzy control system is based on the closed-loop fuzzy system shown in Fig. 6 and designed as shown in Fig. 6. The stages of the control method was designed and constructed step by step.

3.1. Constructing the membership functions

To control the system, the state variable vector is chosen as $\begin{pmatrix}\theta\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}x_1\\ x_2\end{pmatrix}$ and then specifically, we assume the universe of discourse for the two state variables to be $-40^\circ \leq x_1 \leq 40^\circ$ and -8 dps $\leq x_2 \leq 8$ dps (degree per second). First we construct seven membership functions for x_1 on its universe, that is, for the values positive very big (PVB), positive big (PB), positive (P), zero (ZO), negative (N), negative big (NB), negative very big (NVB) as shown in Fig. 7. Then, we construct five membership functions for x_2 on its universe, that is, for the values positive big (PB), positive (P), zero (ZO), negative (N), negative big (NB), negative very big (NVB) as shown in Fig. 8.

To partition the control space as output, we construct nine membership functions for u on its universe which is -32 N $\leq u \leq 32$ N as shown in Fig. 9.

3.2. Constructing rule base

After the stage of constructing membership functions, we construct a Fuzzy Associative Memory (FAM) Table for the rule base. According to the FAM table our rule base can be expressed as,

R1: IF (x_1 = PVB) and (x_2 = PB) THEN u = PVVB
R2: IF (x_1 = PVB) and (x_2 = P) THEN u = PVVB
$\vdots$
R34: IF (x_1 = NVB) and (x_2 = N) THEN u = NVVB
R35: IF (x_1 = NVB) and (x_2 = NB) THEN u = NVVB

The rule base is formed by thirty five rules to control the inverted pendulum system (see Table 1).

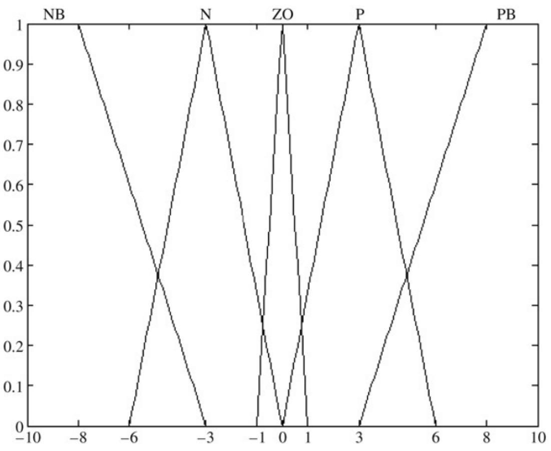


Fig. 8. Membership functions for state variable x_2.

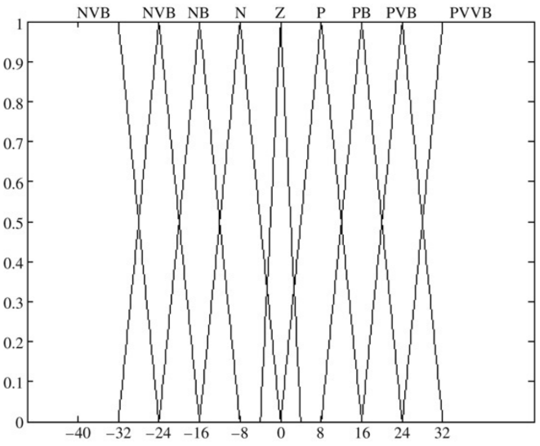


Fig. 9. Membership functions for output variable u.

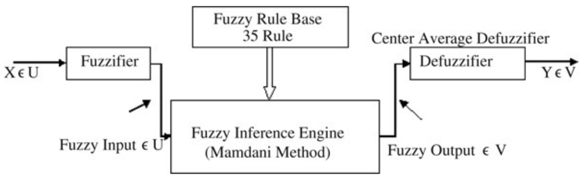


Fig. 10. Methods of fuzzy system.

Then we determine the fuzzification method with a Singleton Fuzzifier, inference system with Mamdani Method, and defuzzification method with Center Average for this fuzzy control system as shown in Fig. 10.

3.3. Product inference engine

In PIE we use an algebraic product for all t-norm operators and max for all the s-norm operators. So the product inference engine is expressed as

$$\mu_{B'}(y) = \max_{l=1}^{M} \left[\sup_{x \in U} \left(\mu_{A'}(x) \prod_{i=1}^{n} \mu_{A_i^l}(x_i) \mu_{B^l}(y) \right) \right]. \tag{2}$$

That is, given fuzzy set A' in U, the product inference engine gives the fuzzy set B' in V according to the expression.

Table 1
FAM table for inverted pendulum fuzzy control system rule base

x_1	x_2				
	PB	P	Z	N	NB
PVB	PVVB	PVVB	PVB	PB	P
PB	PVVB	PVB	PB	P	Z
P	PVB	PB	P	Z	N
Z	PB	P	Z	N	NB
N	P	Z	N	NB	NVB
NB	Z	N	NB	NVB	NVVB
NVB	N	NB	NVB	NVVB	NVVB

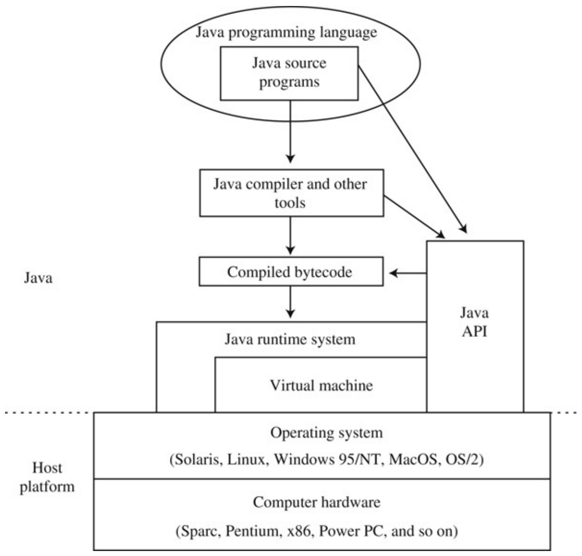

Java programming language
Java source programs
Java compiler and other tools
Compiled bytecode
Java
Java runtime system
Virtual machine
Java API
Operating system
(Solaris, Linux, Windows 95/NT, MacOS, OS/2)
Host platform
Computer hardware
(Sparc, Pentium, x86, Power PC, and so on)

Fig. 11. Java unveiled.

3.4. *Mamdani's minimum inference engine*

In MMIE we use Mamdani's minimum implication and min for all the t-norm operations and max for all the s-norm operators. So the MMIE is expressed as

$$\mu_{B'}(y) = \max_{l=1}^{M}\left[\sup_{x\in U}\left(\min(\mu_{A'}(x), \mu_{A_1^l}(x_1), \ldots, \mu_{A_n^l}(x_n), \mu_{B^l}(y))\right)\right]. \tag{3}$$

4. Java programming language

Java is a programming language, a runtime system, a set of development tools, and an application programming interface (API). The relationships between these elements are depicted in Fig. 11 [18].

As illustrated in Fig. 11, a software developer writes programs in the Java language that use predefined software packages of the Java API. The developer compiles his or her programs using the Java compiler. This results in what is known as compiled bytecode. Bytecode is in a form that can be executed on the Java virtual machine, the core of the Java runtime system. The Java runtime system consists of the virtual machine plus additional software, such as dynamic link libraries that are needed to implement the Java API on operating systems and hardware.

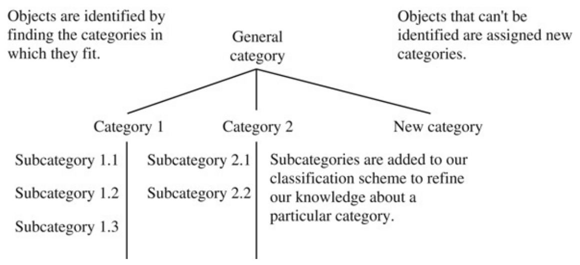

Objects are identified by finding the categories in which they fit.
General category
Objects that can't be identified are assigned new categories.
Category 1
Category 2
New category
Subcategory 1.1
Subcategory 1.2
Subcategory 1.3
Subcategory 2.1
Subcategory 2.2
Subcategories are added to our classification scheme to refine our knowledge about a particular category.

Fig. 12. Hierarchical classification knowledge.

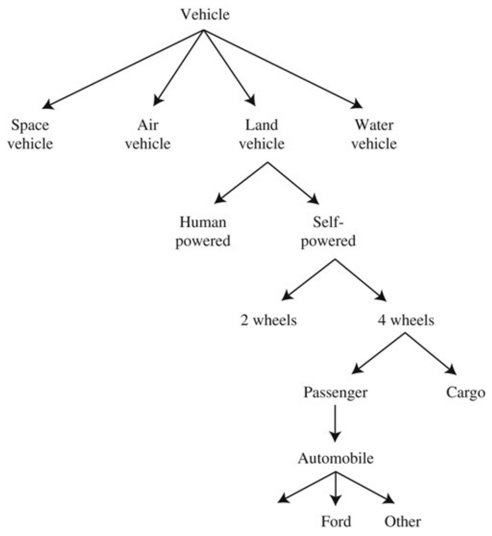


Fig. 13. Vehicle classification tree.

4.1. Classes

Java, C++, Smalltalk, and some other object-oriented languages follow a class-based approach. This approach allows declaring classes that serve as a template from which objects are created. As you would expect, a class defines the type of data that is contained in an object and the methods that are used to access this data. A class also defines one or more methods to be used to create objects that are instances of the class. An instance of a class is a concrete manifestation of the class in a computer's memory.

4.2. Classification and inheritance

Classification is a common way that we organize knowledge. When we encounter a new object in our daily experience, we try to fit that object into our hierarchical classification scheme. If it fits in an existing category, we know what kind of object it is. If it doesn't fit, we add a new category. Fig. 12 describes how we use classification to represent knowledge [18].

When we classify objects in this hierarchical fashion, the object categories at the top of the classification tree include all the object categories below them. If an object category appears in the classification tree, it satisfies the properties of all object categories above it in the tree.

Fig. 13 presents a classification tree for vehicles. All categories in the tree below the category automobile, for example, share the common characteristics of being four-wheeled, self-powered, and designed for passenger transportation. The fact that a lower-level category shares the characteristics of the categories above it on the classification tree is known as inheritance. The lower-level categories are said to inherit the characteristics of the categories above them on the tree.

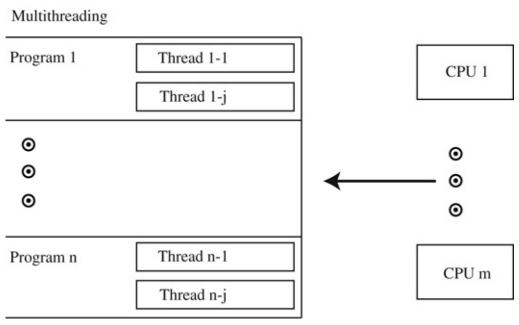


Fig. 14. Multithreading.

The classes we explained in this section can also be organized in a hierarchical fashion. A class X is said to extend another class Y if it contains all the data contained in class Y and implements all the methods implemented by class Y. Class X is said to be a subclass of class Y, and class Y is said to be a superclass, or parent class, of class X.

Classes form a hierarchical classification tree under the subclass relationship. If a class X is a subclass of a class Y, it inherits the properties of Y. This means that all of the data and methods defined for class Y are available to class X [17,18].

4.3. Multithreading

Multithreaded programs are similar to the single-threaded programs that we have been studying. They differ only in the fact that they support more than one concurrent thread of execution–that is, they are able to simultaneously execute multiple sequences of instructions. Each instruction sequence has its own unique flow of control that is independent of all others. These independently executed instruction sequences are known as threads (Fig. 14).

In single-processor systems, only a single thread of execution occurs at a given instant. The CPU quickly switches back and forth between several threads to create the illusion that the threads are executing at the same time. Logical concurrency is the characteristic exhibited when multiple threads execute with separate, independent flows of control. The important feature of multithreaded programs is that they support logical concurrency, not whether physical concurrency is actually achieved. Multithreading provides concurrency within the context of a single process and multiprogramming provides concurrency between processes [17,18].

In Java, we can create threads in a couple of ways. The simplest way is to take an existing class and turn it into a thread. We do this by modifying the class so that it implements the Runnable interface, which declares the run() method required by all types of threads. (The run() method contains the code to be executed by a thread.)

The second way to create a thread is to write a completely separate class derived from Java's Thread class. Because the Thread class itself implements the Runnable interface, it already contains a run() method. However, Thread's run () method doesn't do anything. We usually have to override the method in our own class in order to create the type of thread we want [19].

5. Designing Java application

We use Java Applets to create an inverted pendulum system and fuzzy controller for this system. In this section we give the tricks for designing java applications.

5.1. Designing inverted pendulum system

We benefit from (1) to develop java code to create inverted pendulum system dynamics. The code is defined in Listing.1. When we develop the code, we use the Euler Method to solve the differential equation [20].

The dynamic system equation is given as,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t), \qquad y(0) = y_0. \tag{4}$$

Table 2
Developed java code of inverted pendulum system dynamic

```
sinangle=Math.sin(angle);
cosangle=Math.cos(angle);
angleDotSq = angleDot*angleDot;
angleDDot=(9.8*sinangle-
                        (poleMassLength*angleDotSq*
                        cosangle*sinagnle/totalMass)+
                        (force*cosangle/totalMass))/
      l*(4/3-poleMass*cosangle*cosangle/totalMass);
posDDot=(force + poleMassLength *angleDotSq*
                        sinangle+
                        poleMassLength*angleDDot*cosangle)/
                        totalMass;
   pos += posDot * h;
   posDot += posDDot * h;
   angle += angleDot * h;
   angleDot += angleDDot * h
```

From the Euler approach,

$$\frac{y_{i+1} - y_i}{\Delta t} = f(y_i, t_i) \quad y_{i+1} = y_i + \Delta t f(y_i, t_i). \tag{5}$$

The code shown in Listing.1 defines the dynamics of the inverted pendulum system. In the code, the variables poleMassLength, poleMass, totalMass, l, h are constants and explained as the multiplication of the pole mass and the pole half length, the pole mass, the total mass of the cart and the pole and iteration number for the Euler Method, respectively, and the variables angle, angleDot, angleDDot, pos, posDot, posDDot are the system state variables and explained as the angle between the pendulum and the upright acceleration of the pendulum, the position of the cart, the angular velocity and the angle from the rail origin, its velocity, and its acceleration, respectively.

We create a thread called *AnimationThread*. We add the code shown in Table 2 in the run() method of this thread. Then by using the start() method of the thread we execute the inverted pendulum system and investigate the behavior of the system.

5.2. Membership function class

To implement the membership functions explained in Section 3.1, we develop Java code to create a trapezoidal membership function class. As shown in Table 3.

We create the fuzzifier function for the fuzzy controller first stage and the code shown in Table 4 is added to the MembershipFunction class as a method.

A trapezoidal membership function has four support points, we implement the points as an array and we set the value of elements of the array called range [] according to the constructed membership function in Section 0. The variable X used in the fuzzy function Java code is a crisp value of an input variable and the fuzzyresult the output of the fuzzification process.

As shown in Fig. 15 we implement the goal of the application: as the pendulum starts to fall, the cart on the surface moves in the same direction trying to counteract the pendulum's motion; but the unit has made an overcompensated movement and the pendulum has changed direction so the unit will follow the pendulum again until the fall motion is close to zero; in the last figure, the pendulum has been balanced. The unit will remain motionless until the pendulum starts to fall.

We use the graphics methods in the paint() method of Java to develop the interface of the application as shown in Fig. 16.

6. Simulation results

As shown in Fig. 16, inverted pendulum fuzzy control system Java Applet was developed. This Java Applet can be used on Fuzzy Logic for educational goals. The mass of the pendulum is $m_p = 0.1$ kg; the half length of the

Table 3
The membership function class

```
public class MembershipFunction
{
  private String name;
  private double[] range;
    public String getname()
  {return name; }
public double[] getrange()
{return range; }
//CONSTRUCTOR
public MembershipFunction(String name_i,double[] range_i)
{         super();
        this.name=name_i;
        this.range=range_i; }}
```

Table 4
Fuzzify function

```
public double fuzzifier(double X)
  {
    double fuzzyresult=0.0;
  if(X>=range[1] && X<=range[2])
    fuzzyresult=1;
  if (X>=range[0] && X<range[1])
    fuzzyresult=(X-range[0])/(range[1]-range[0]);
  if (X>range[2] && X<=range[3])
    fuzzyresult=(range[3]-X)/range[3]-range[2];
    return fuzzyresult;
  }
```

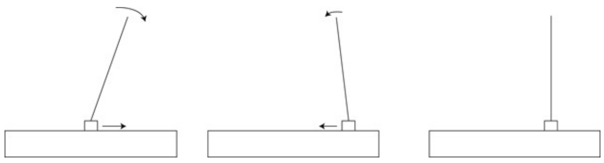


Fig. 15. The goal animation of application.

pendulum is $l = 1$ m; and the mass of the cart is $m_c = 1.0$ kg. The angle and the angular velocity of the pendulum is limited to [−40°, +40°], [−8 dps, +8 dps] respectively. Also in the program for the angle and angular velocity, the initial conditions are determined as $x_1 = -30°$, $x_2 = 2$ dps.

In the first test, we selected a mass of the cart 2 N and we compared the results as shown in Fig. 17. As appears, increasing the mass of the cart makes the inverted pendulum system more difficult to control.

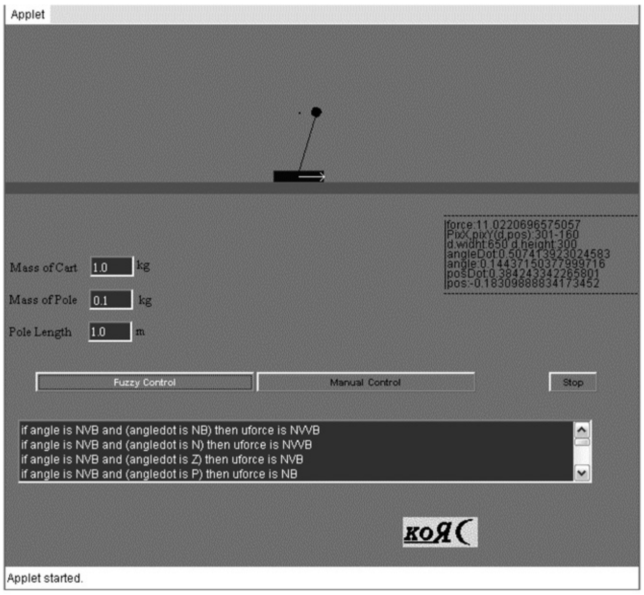
In the second test, we selected a pole length of 0.5 m and we compared the results as shown in Fig. 18. Here, increasing the pole length makes the inverted pendulum system more difficult to control.

According to the results of the two tests, we can control the inverted pendulum by defining shorter pole length values and lower weight cart mass values.

Also, as shown in Fig. 19, we can learn approximate stability of inverted pendulum system although it is difficult to determine the stability. According to Fig. 19, we can say the inverted pendulum is asymptotically stable [16]. Asymptotic stability means that the equilibrium is stable, and that in addition, states started from some initial values actually converge to 0 as time t goes to infinity (see the Appendix).

Fig. 19 shows that system trajectories starting from within the initial conditions converge to the origin.

If we narrow the range of Z membership functions that are implemented for the $x1$ and $x2$ states, convergence of the system to the origin will be fast. So the stability of the system will increase.



Mass of Cart 1.0 kg
Mass of Pole 0.1 kg
Pole Length 1.0 m

```
force:11.0220696575057
PixX,pixY(d,pos):301-160
d.widht:650 d.height:300
angleDot:0.507413923024583
angle:0.144371503779997 16
posDot:0.384243342265801
pos:-0.183098888341 73452
```

```
if angle is NVB and (angledot is NB) then uforce is NVVB
if angle is NVB and (angledot is N) then uforce is NVVB
if angle is NVB and (angledot is Z) then uforce is NVB
if angle is NVB and (angledot is P) then uforce is NB
```

Applet started.

Fig. 16. The Java application for inverted pendulum fuzzy control system.

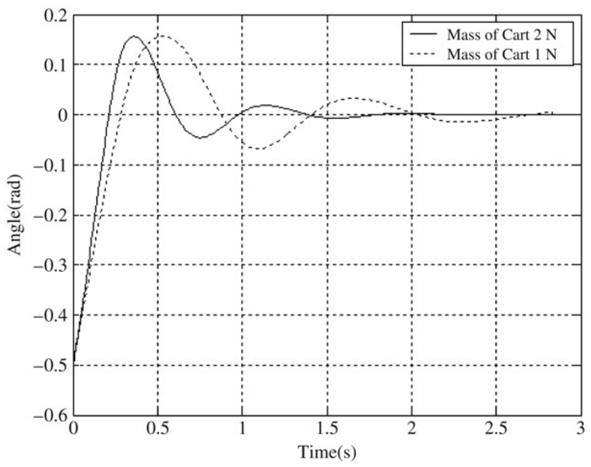


Fig. 17. The simulation result for change of cart mass value.

After these experimental results, we decided to compare the impacts of using in fuzzy control MMIE and PIE. This experiment is shown in Fig. 20. According to the results, using MMIE speeds up the pole behavior in reaching the equilibrium state. But PIE is not as successful as MMIE. Also, PIE and MMIE are the most commonly used fuzzy inference engines in fuzzy systems and fuzzy control. The main advantage of them is their computational simplicity; by watching the changes, we can say MMIE and PIE give very small membership values in all the cases and the engines are similar. Giving small membership values is a disadvantage for the inference engines and this may cause problems in implementation.

7. Conclusions

Recently, interest in fuzzy systems has become increasingly important. Therefore, the solution of "the inverted pendulum problem" has been chosen and presented on the Net by using Java Applets with educational goals. The simulation results are indicated as figures; the results show that systems with short poles and light carts can be controlled more easily and quickly.

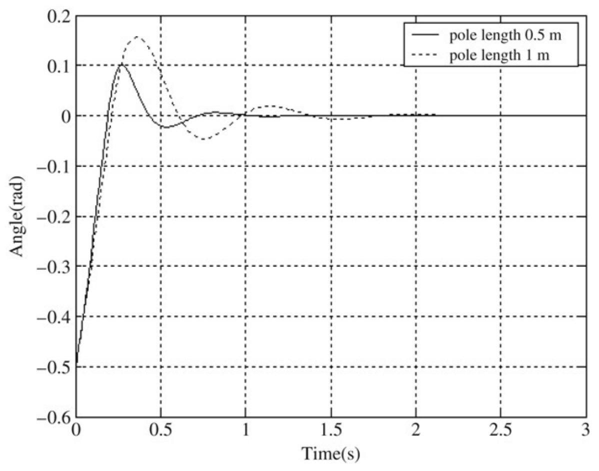


Fig. 18. The simulation result for change of pole length.

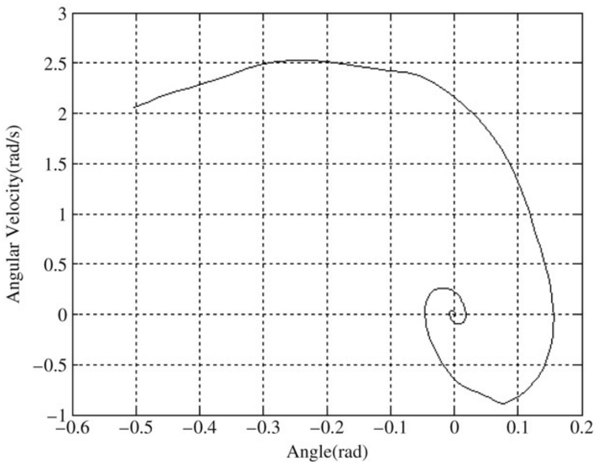


Fig. 19. State space trajectory of inverted pendulum system.

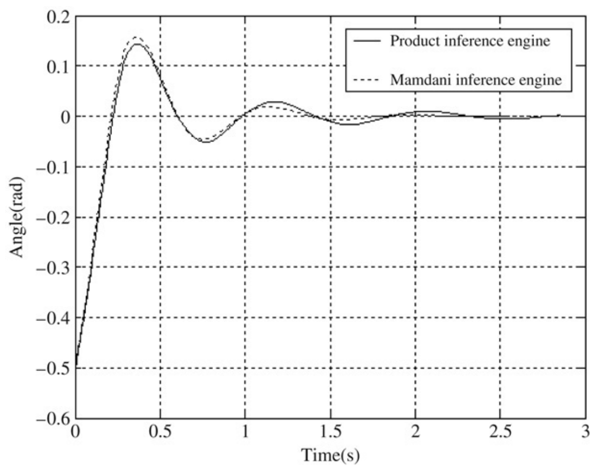


Fig. 20. The impacts of using PIE and MMIE.

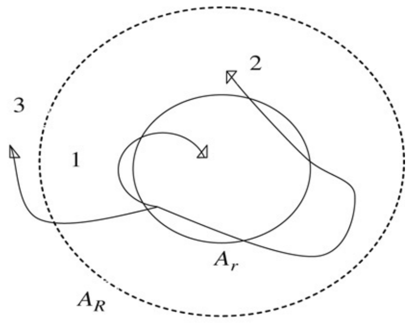


Fig. 21. Concept of stability.

Let us say five tasks have to be performed in a given period of time, and each task requires one person dedicated to it. Suppose there are six people capable of doing these tasks. As you have more than enough people, there is no problem in scheduling this work and getting it done. Of course, who gets assigned to which task depends on some criterion, such as total time for completion, on which some optimization can be done. But suppose these six people are not necessarily available during the particular period of time in question. Suddenly, the equation is seen in less than crisp terms. The availability of the people is fuzzy-valued. Here is an example of an assignment problem where fuzzy sets can be used. In this project we have shown a role for using fuzzy sets in control systems and the performance of the system, its ability to reject force and angular disturbances as well as its capability in tracking a human's driving inputs. This is the power of fuzzy logic.

Moreover, we propose for the next projects generalizing the fuzzy system based on the Java application, so that the corresponding user can change the rule base and the structure of the system for the purpose of learning the finer points of inverted pendulum fuzzy system design. Also we can develop a fuzzy laboratory on the InterNet with Java Applets to provide opportunities to learn fuzzy logic.

Appendix. Stability and instability

Essentially, stability means that the system trajectory can be kept arbitrarily close to the origin by starting sufficiently close to it. We can give a definition of stability and instability [16].

The equilibrium state $x = 0$ is said to be stable if, for any $R > 0$, there exists $r > 0$, such that if $\|x(0)\| < r$, then $\|x(t)\| < R$ for all $t \geq 0$. Otherwise, the equilibrium point is unstable.

More formally, the definition states that the origin is stable, if, given that we do not want the state trajectory $x(t)$ to get out of a ball of arbitrarily specified radius A_R, a value $r(R)$ can found such that starting the state from within the ball A_r at time 0 guarantees that state will stay within the ball A_R thereafter.

Conversely, an equilibrium point is unstable if there exists at least one ball A_R such that for every $r > 0$, no matter how small, it is always possible for the system trajectory to start somewhere within the ball A_R and eventually leave the ball A_R. As shown in Fig. 21 the implementation of the curves are asymptotically stable, marginally stable, and unstable, respectively.

All trajectories close to the origin moving further and further away to infinity are denoted "blow up". There are differences between instability and the notion of "blowing up". In linear systems, instability is equivalent to blowing up, because unstable poles always lead exponential growth of system states. However, for nonlinear systems, blowing up is only one form of instability. This subject has a very large scale for control systems.

References

[1] B.K. Çelik, Fuzzy Control Systems and Fuzzy Logic Based Inverted Pendulum Control System, Term Project, Computer Engineering Department, Kocaeli University, January 2004.
[2] L. Wang, A Course in Fuzzy Systems and Control, Prentice-Hall Inc., 1997.
[3] J.T. Ross, Fuzzy Logic with Engineering Applications, McGraw-Hill, Inc., 1995.
[4] G.J. Klir, B. Youan, Fuzzy Sets and Fuzzy Logic Theory and Applications, Prentice Hall Inc., 1995.
[5] Z. Yeh, K. Li, A systematic approach for designing multistage fuzzy control systems, Fuzzy Sets and Systems (2003).

[6] M. Margaliot, G. Langholz, Adaptive fuzzy controller design via fuzzy lyapunov synthesis, in: Proceedings of FUZZ-IEEE'98, 1998, pp. 354–359.

[7] T. Yamakawa, Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system, Fuzzy Sets and Systems 32 (1989) 161–180.

[8] S. Kawaji, T. Maeda, Fuzzy servo control system for an inverted pendulum, in: Proceedings of IFES'91, vol. 2, 1991, pp. 812–823.

[9] N. Yubazaki, J. Yi, M. Otani, K. Hirota, SIRMs dynamically connected fuzzy inference model and its applications, in: Proceedings of IFSA'97, vol. 3, 1997, pp. 410–415.

[10] J. Yi, N. Yubazaki, K. Hirota, Upswing and stabilization control of inverted pendulum system based on the SIRMs dynamically connected fuzzy inference model, Fuzzy Sets and Systems (2000).

[11] M. Margaliot, G. Langholz, A new approach to fuzzy modeling and control of discrete-time systems, IEEE Transactions on Fuzzy Systems 11 (2003).

[12] J. Yi, N. Yubazaki, Stabilization fuzzy control of inverted pendulum systems, Artificial Intelligence in Engineering 14 (2) (2000) 153–163.

[13] L.A. Zadeh, From circuit theory to systems theory, Proceedings of Institution of Radio Engineers 50 (1962) 856–865.

[14] L.A. Zadeh, Fuzzy algorithms, Information and Control 12 (2) (1968) 94–102.

[15] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Transactions on Systems, Man and Cybernetics 3 (1973).

[16] J.-J.E. Slotine, W. Li, Applied Nonlinear Control, Prentice-Hall, Inc., 1991.

[17] V.B. Rao, C++ Neural Networks and Fuzzy Logic, IDG Boks Worldwide, Inc., 1995.

[18] J. Jaworski, Java Developer's Guide, Sams.net Publishing, 1996.

[19] K.C. Hopson, S.E. Ingram, Developing Professional Java Applets, Sams.net Publishing, 1996.

[20] S.C. Chapra, R.P. Canale, Numerical Methods for Engineers, McGraw-Hill, Inc., 1989.