

## **GEFASA V1.0 for Closed Loop Flatness Control**

### **1. Introduction:**

A component-based program development is characterized by a number of software entities equipped with connectors. Component-based programs can be software oriented or hardware oriented depending on the application. The current project is a component-based software, which consists of two components namely Mill model and the Fuzzy state machine. The components are connected as reconfigurable agents in the Flatness Server Architecture employed by the ABB Stressometer system.

An application consists of a set of components and a method to connect them such that certain expected system behavior is achieved. There are many different possible ways to connect components, such as by initial (static) configuration, (e.g., by means of an Architecture Description Language (ADL)) or by using a dynamic configuration approach such as emerging architectures, dynamic architectures or supervised reconfigurations.

A trend in component design is that components are more autonomous, more complex, have more responsibilities and the number of components in applications increases. This trend is obvious with agent architectures but even those component designs that are not agent-based have the same characteristics.

In designing industrial applications today, there is an emerging consensus that some kind of homogenous, high-level information representation is needed for agents, such that a predictable behavior can be achieved for all agents. A fuzzy automaton-based approach offers clear benefits for developing agents of reconfigurable architecture.

The focus of the current research project is to attain Flatness Control in a closed loop environment using a Mill model with 3 actuators and 32 measurement zones. First the Mill model is implemented in closed loop control

architecture and the main goal of this project is to test fuzzy state machine-based control. The Mill model is then implemented by reconfigurable agents in the Flatness Server Architecture employed by the Stressometer system. The whole project is developed and implemented using java.

## **2. Background:**

Among the problems that characterize industrial process control innovation, and which are not domain-related, the most difficult ones are as follows:

- (a) How can new knowledge be introduced into a system,
- (b) How can the system activate stored domain knowledge in an autonomous way,
- (c) How can the knowledge be validated (or otherwise detected as inappropriate) and
- (d) How can the system recover if the new, activated knowledge (or the currently active knowledge) is not suitable to handle the situation at hand.

The use of agent technology helps to answer question (a), in that paradigm an agent is defined as an architecture-neutral, mobile software entity that can act on behalf of a human and have decision making capabilities similar to a human [1]. The theory of software dynamical architectures describes the dynamics of the environment in which agents can act. The software architecture, by using an architecture broker, mediates the information flow among agents in order to achieve overall population-level goals, and also makes various resources (computational power, sensors, and actuators) available to the agent. The activation of the appropriate knowledge is accomplished via identification of operations performed by agents that are capable of finding the right model among a set of available models [2]. Thus the answer to problem (b) is defined as the appropriate interaction among the software architecture mechanisms and the agents. Agents with model-based target seeking algorithms utilizing fuzzy logic, interacting in a distributed large system are strong candidates to replace traditional communications channels among units of a distributed system at a higher system level.

A typical agent-based application consists of an environment in which the agents can communicate and a library of agents activated either by already active agents or by a specialized supervisory agent. Some of the typical properties of such systems are:

- Transparency for distributed, networked operations. For example, activation of the remote agents is done in the same way as for local agents, i.e., a re-distribution of agents is done seamlessly and without functional degradation. The interconnection of local agents is done in the same way as for remote agents, etc.
- Architecture integrity supervision. The system has built-in functionality that checks that all necessary agents are active and reachable. If this is not the case, the supervisor can restart missing components, or agents.
- Distributed basic functionality. This is available in the environment such as access to the file system, database, timers, etc. across the distributed system in a homogenous way.

One of the useful functions, which the research presented here is focusing on, is a state machine model that exists identically in each agent. Solutions of this type are common today and have been described earlier, e.g., in [3]. This is the supervisory state level and all agents have the same set of states. At any time instance each agent can have a different active state. The purpose of this state machine is that other agents, or a supervisor can query if an agent is in a proper active state and if it is not then will react appropriately. Each agent is responsible for the state transitions of its own state machine. External agents can only query the current state.

A fuzzy automaton [4] can implement new knowledge by means of the states of the goal path of an event-driven, sequential control algorithm while providing an effective approximation method to model continuous and discrete signals in a single theoretical framework. With respect to problem (c) knowledge validation is achieved by quantifying the degree of deviation from the nominal operating conditions due to unexpected events caused by either abrupt, or gradual changes in the system, or in the environment of the system. With respect to

problem (d) these properties can facilitate the development of computationally inexpensive fault detection and identification (FDI) algorithms, and automated recovery from faults (subject to further research). Regarding to FDI, the evaluation of the state transitions between states of a large, complex system will be done by focusing only on clusters of relevant states along the goal path. A reconfigurable virtual fuzzy automaton will be used to model those clusters of states [5].

### **3. Flatness Measurement and Stressometer System:**

Flatness is the measure of the relative elongation of the strip per unit length caused due to the uneven force distribution in the strip. The unit for measuring flatness is “i-unit” which means a relative elongation of 1 per 100000. The stressometer system is a cylindrical measurement roll, about 30cm in diameter and slightly longer than the strip width. A number of force transducers are installed below the steel sheet of the measurement roll. The transducers measure with great precision the force of the strip acting on the measurement roll. The force of a strip with a width of 1 meter acting on the measurement roll is measured in around 20 zones. The force on each zone is a measure of the internal stress in the strip; a flat strip must have the same stress in all zones. The stressometer system can measure with a precision of 0.5 I-units (means 5 microns of elongation in a 1m strip). More details about the ABB Stressometer system can be found in the appendix C.

### **4. The Software Agent GEFASA:**

The development of a software environment for the Intelligent Fuzzy Controllers Laboratory has been divided into three phases. The primary objective of Phase 1 is to create a program referred to as the Generic Encapsulated Fuzzy Automaton Software Agent (GEFASA) aimed for Object Oriented Control Systems using Java. This program is then loaded as a reconfigurable agent in the FSA (Flatness Server Architecture) broker running on the Stressometer System by ABB (Phase1 completed in August 2003). The Stressometer System includes several industrial computers and PLCs (Programmable Logic

Controllers) along with ABB's software development environment for complex control systems.

The notion of fuzzy automaton in this research is based upon the premises as follows: it can stay in more than one crisp (i.e., Boolean) state at once, to a certain degree in each. Those degrees are defined by a state membership function. That is, a fuzzy state is made of some crisp states. For each fuzzy state there is just one dominant crisp state, though, for which the state membership is a 1 (full membership). Each dominant state is associated with a linguistic model of the plant for inference. For each fuzzy state a composite linguistic model is devised by the composition of the linguistic models of all contributing crisp states. A set of associated state membership degrees are used for the composition. The transitions between fuzzy states are based upon the transitions defined between their dominant crisp states. There is an underlying Boolean finite state machine to implement the fuzzy automaton. The states of this Boolean automaton are the dominant (crisp) states of the fuzzy automaton. Fuzzy (i.e., continuous signal) inputs (and outputs) are mapped to sets of two-valued logic variables. In addition, other input signal types include two-valued ones and continuous signals with a threshold. Conditions for state transitions are defined by any combination of signals of these three types. Fuzzy outputs are obtained by computing the compositional rule of inference using the composite linguistic models. Defuzzified outputs are devised by using a suitable defuzzification algorithm. Two-valued (Boolean) outputs are devised in the usual manner [6].

The fuzzy automaton model implemented in the FSA broker along with an input stimuli generator can be found in detail in [7] and [8].

Phase 2 (completed in April 2004) involves the development of a Mill model having 3 actuators and 32 measurement zones. The idea is to provide Flatness Control in a closed loop using a PID controller to provide for a reference to the fuzzy automaton-based control.

In Phase 3 the system will be connected to an ABB R&D Lab in Sweden through the Internet and will periodically receive input files that have been passed to the controllers of an actual cold rolling mill, as well as files describing

the performance of the plant. This system will process the input files for the controllers using the Mill model developed in the Phase 2 and the fuzzy automaton model. Its results will be compared with the output results produced by the controllers of the cold rolling mill to evaluate the performance of the method using the fuzzy automaton. Phase 3 will create an experimental system to conduct research on algorithms and architectures for supervisory control using fuzzy automata. Also the fuzzy automaton developed will be applied to have the system respond to the faults dictated by the fuzzy rules. This application is useful for multi-pass mills where the strip is reduced in several steps.

### 5. THE MILL MODEL:

The Mill model is shown in Fig. 1. Mill1 is a function that models a cold rolling mill with 3 actuators and 32 measurement zones. PID stands for an industry-standard Proportional-Integral-Derivative Controller that is modeled as a simple recursive function.

It has two parts:

- static computation that has an inverted model  $ex = A \backslash e$  (pseudo-inverse function),
- a dynamic section that includes a PID, or alike.

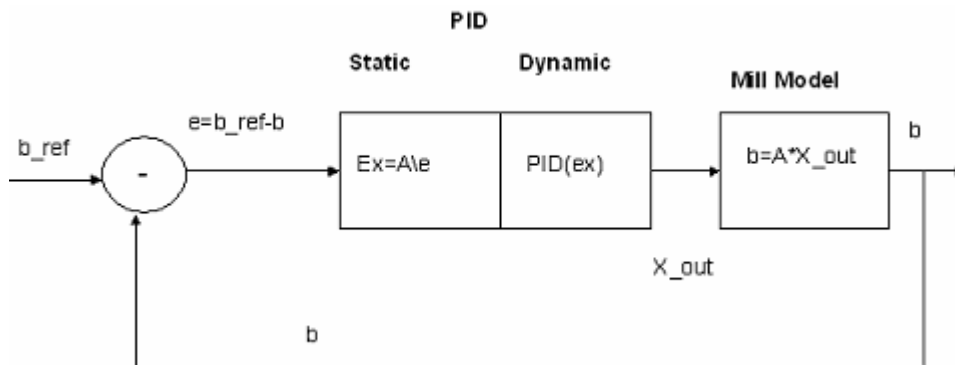


Fig1. Mill model for Flatness control in closed loop

The objective is to replace the computation for the pseudo-inverse function (i.e., replace  $ex = A \backslash e$  with a fuzzy approach) and leave the PID section alone. The flatness control program for the PID section is responsible for two tasks:

- Convert the error into an actuator setup, e.g., the least square of  $A \setminus b$ ,
- Implement the PID algorithm that makes a smooth conversion between different actuator setup points.

But instead of using the least square method the program follows the following rule for converting the error into an actuator setup: the actuator that takes out more of the error is given more power to do it. The input to the algorithm is  $e$ , referred to as the flatness error. It is a vector of the size  $32 \times 1$ . The output from the algorithm is  $e_x$  that is of size  $3 \times 1$ . It is the set point for the actuators.

The algorithm has the steps as follows:

**Step1:**

Compute the actuator parameters as if each actuator compensated alone for the error defined as  $x_1$ ,  $x_2$ , and  $x_3$ , respectively.

**Step2:**

Compute the residue  $r = e - Ax$ . The actuator that works the best will have the smallest residue. Factor  $A$  that determines the percent of the error compensated by each actuator must be determined in this step. Let  $A_1$ ,  $A_2$ , and  $A_3$  be columns in matrix  $A$  that describes each actuator,  $x_1$ ,  $x_2$ ,  $x_3$  be the parameters of each actuator, and the factors  $k_1$ ,  $k_2$ , and  $k_3$  are as follows:

- $k_1$  - part of the error compensated by Actuator 1,
- $k_2$  - part of the error compensated by Actuator 2,
- $k_3$  - part of the error compensated by Actuator 3, and  $K_1+K_2+K_3 = 1$ .

Then

- $A_1 * x_1 = k_1 * b$
- $A_2 * x_2 = k_2 * b$
- $A_3 * x_3 = k_3 * b$

**Step3:**

Compute factors  $k_1$ ,  $k_2$ ,  $k_3$  as follows:

- Compute the Euclidian norm of the residue  $r = A_1 * x_1 - b$  which provides the error left if Actuator1 acted alone. Do the same for Actuators 2 and 3.

- The 'best' actuator is the one with the lowest residue but the actuator with the lowest residue must have the highest factor. That is obtained by computing  $q1 = (r1+r2+r3)-r1$  and then  $k1 = q1/(q1+q2+ q3)$ .

#### **Step4:**

Compute the effective actuator parameter by making use of the following rule: the actuator that can compensate more of the error will work alone until some other actuator becomes more prevalent.

Hence, the program is state-based with three states, each of which corresponds to one specific actuator working alone.

The fuzzy logic section of the program consists of the following steps:

#### **Step1:**

Define crisp states. Set up conditions for triggering a state change. Define fuzzy If – Then rules and their relationship with each crisp state

#### **Step2:**

It has three parts:

- The initial part where the membership functions are computed and rules are given,
- The dynamic part where the fuzzy inference is done, and
- The interface to the program for actuator selection

#### **Step3:**

It is the implementation step:

- Compute the flatness error norm:  $\|e\| = \|b-b_{ref}\|$  and fuzzify it.
- Compute the inverse of the norm of the residue from each actuator (if it acted alone)  $\|r_i\|$   $i = 1, 2, 3$  and fuzzify them.
- Compute the parameter of each actuator  $x1, x2, x3$  and fuzzify them.

This step results in 7 fuzzy inputs  $\|e\|, \|1/r1\|, \|1/r2\|, \|1/r3\|, \|x1\|, \|x2\|, \|x3\|$ .

Some of these fuzzy inputs determine the states as shown in the Table below.

State	Norm(e) – in(1)	Norm(1/r1) – in(2)	Norm(1/r2) – in(3)	Norm(1/r3) – in(4)
1	High	High	Any	Any
2	High	Any	High	Any
3	High	Any	Any	High
4	Low	Any	Any	Any

The key parameters used in the fuzzification process are explained in detail in the INPUT1.m file attached in the appendix B

Explanation:

- If the error is high, then the flatness is adjusted using only one actuator, the one that has the highest capacity to correct the error. This corresponds to a control policy like using the shift actuators just to keep strip in the middle.
- If the error is low, all actuators come to play

The state transitions are as shown in Fig. 2.

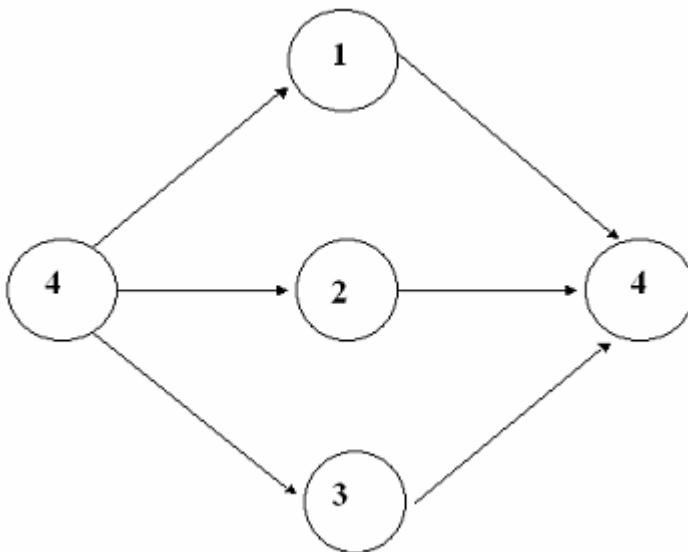


Fig2. State Transition Diagram

From S4 it can go to S1 or S2 or S3. From S1 or S2 or S3 it can go to S4. The fuzzy inputs used for the FSM are the actuator parameters  $x_1, x_2, x_3$ . So we have 3 fuzzy inputs. The fuzzy outputs are the values for the PID (or PI) parameters. The If-Then rules used for the fuzzy model building are as given below.

State1: high error, actuator1 is acts powerfully

+-----If Part-----+-----Then Part-----+

Rule	X1	X2	X3	P1	P2	P3	I1	I2	I3
1	L	L	L	L	L	L	H	H	H
2	M	L	L	M	L	L	M	H	H
3	L	M	L	M	M	L	H	M	H
4	L	L	M	M	L	M	H	H	M
5	H	H	H	H	M	M	L	L	L

State2: high error, actuator2 acts powerfully

+-----If Part-----+-----Then Part-----+

Rule	X1	X2	X3	P1	P2	P3	I1	I2	I3
1	L	L	L	L	L	L	H	H	H
2	M	L	L	M	M	L	M	H	H
3	L	M	L	L	M	L	H	M	H
4	L	L	M	L	M	M	H	H	M
5	H	H	H	M	H	M	L	L	L

State3: high error, actuator1 acts powerfully

+-----If Part-----+-----Then Part-----+

Rule	X1	X2	X3	P1	P2	P3	I1	I2	I3
1	L	L	L	L	L	L	H	H	H
2	M	L	L	M	L	M	M	H	H
3	L	M	L	L	M	M	H	M	H
4	L	L	M	L	L	M	H	H	M
5	H	H	H	H	H	M	L	L	L

State4: Low error, all actuators acts middle

+-----If Part-----+-----Then Part-----+

Rule	X1	X2	X3	P1	P2	P3	I1	I2	I3
1	L	L	L	L	L	L	H	H	H
2	M	L	L	M	L	L	M	H	H
3	L	M	L	L	M	L	H	M	H
4	L	L	M	L	L	M	H	H	M
5	H	H	H	H	H	H	L	L	L

The values for the above-mentioned fuzzy rules Low, Medium, High are as shown in Fig.3.

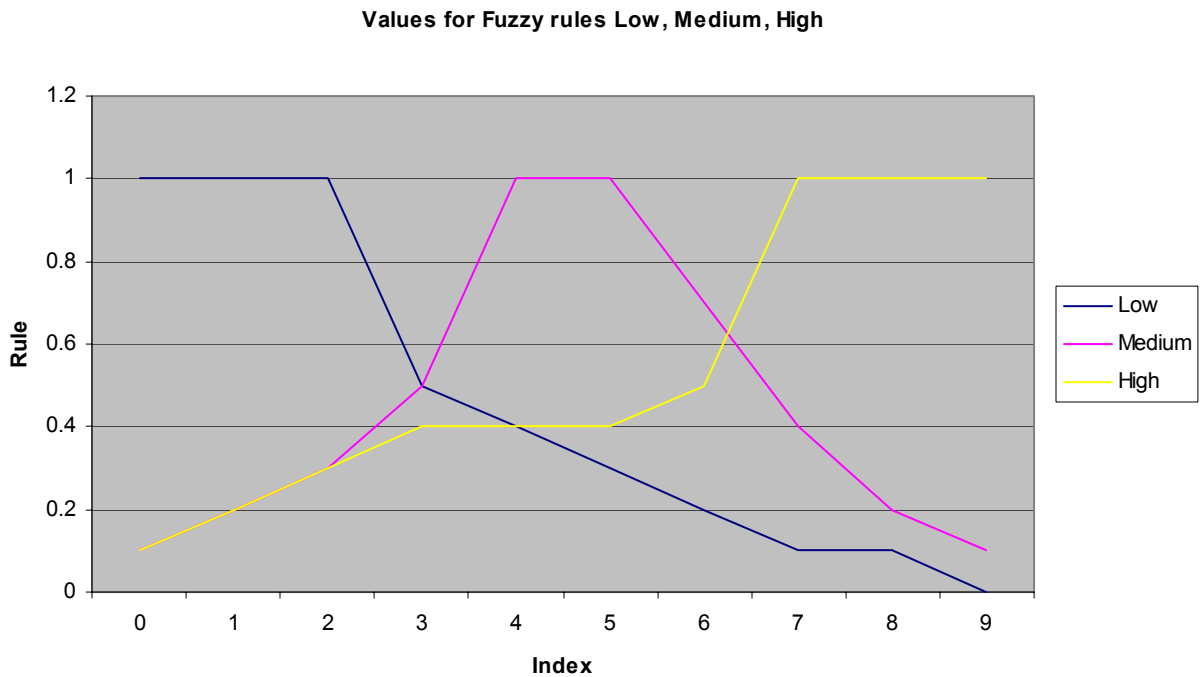


Fig3. Values for Fuzzy rules Low, Medium, High

## 6. THE MILL MODEL IN THE Flatness Server Architecture

The MillModel in the Flatness Server Architecture (FSA) consists of two fuzzy agents referred to as MillFSMSIM and MillFSM and two display panels known as MillFSMPanel and MillFSMSIMPanel. The whole model is implemented in a closed loop control in the Flatness Server Architecture as shown in Fig. 4.

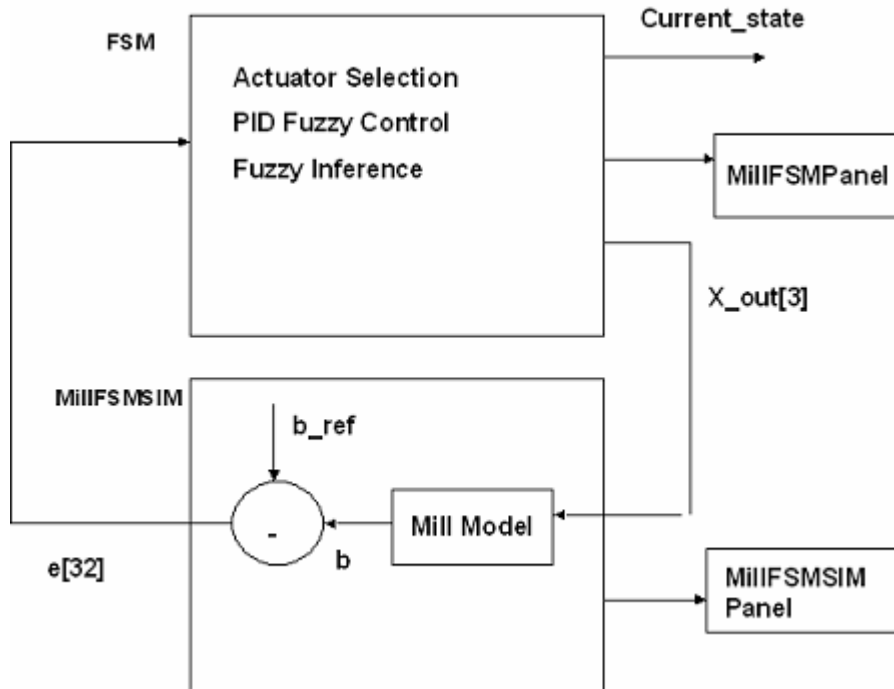


Fig4. Mill model in the FSA architecture

The module MillFSMMSIM computes the MillModel that models the mill with 3 actuators and 32 measurement zones. The input to this module is the vector  $x_{out}$  that is obtained as a defuzzified fuzzy output from the fuzzy inference process. Vector  $x_{out}$  is the command vector for some mechanical action. It is the input to the plant model that will again compute the error  $e$ , and the cycle will continue in this way equal to the number of cycles given by  $T_{MAX}$  (default is 100).

The MillFSMSIM module consists of the following inputs and outputs:

- input:  $x_{out}$  (3x1 matrix)
- output:  $e$  (32x1 matrix)
- $b_{ref}$  (32x1 matrix)

that is the set point for the “reference profile” for flatness. The mechanical process that is being controlled must achieve a result equal to this matrix. The current profile  $b$  is calculated and is compared with  $b_{ref}$  to determine the type of mechanical action required. The difference  $e=b-b_{ref}$  is an input to the control system implemented by the fuzzy automaton. Also after the control loop goes for

100 (default) iterations the MillFSMSIM agent writes calculated  $b$ , state changes,  $x_{out}$ , and the rest to be corrected to a matlab file for graphics purposes. The flowchart for the MillFSMSIM agent is shown in Fig. 5.

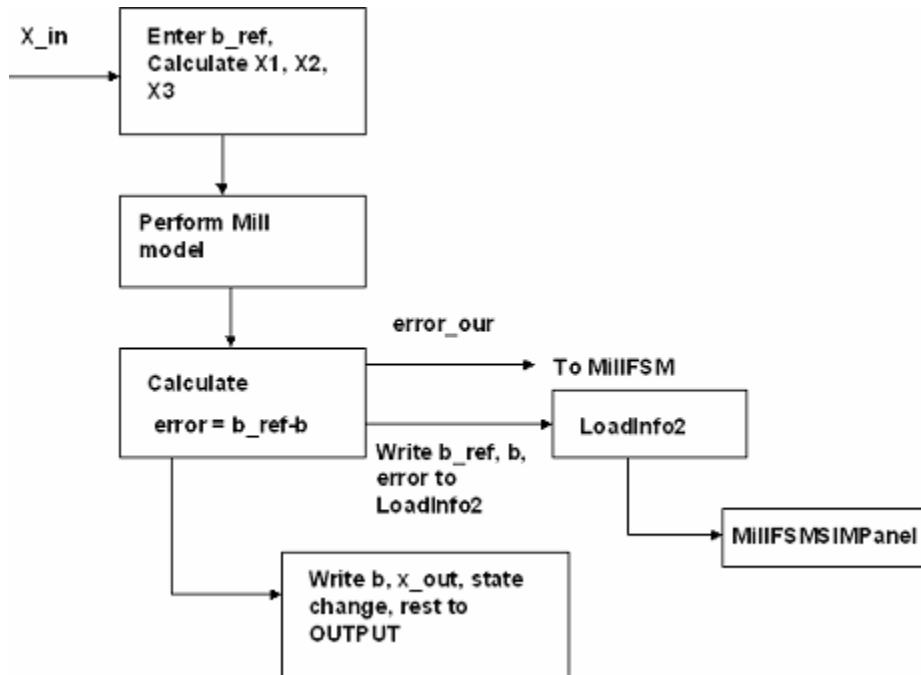


Fig5. MillFSMSIM module in the FSA Broker

The MillFSM module computes the composite linguistic model  $R^* = G * R_s$  once for a model and then performs the fuzzy model building and inference computations as well as evaluates the necessary state changes. The module has one input vector  $e$  ( $32 \times 1$ ) which is the error calculated in the MillFSMSIM module, one output vector  $x_{out}$  ( $3 \times 1$ ) which is given as input to the MillFSMSIM module, and one output value  $current\_state$  which depicts the necessary state changes. This module provides both fuzzy outputs as function of fuzzy inputs and Boolean outputs as function of Boolean inputs. The MillFSM module has the following inputs and outputs:

- input  $e$  ( $32 \times 1$  vector), the error value computed in the MillFSMSIM module by making use of the above mentioned mill model,
- output  $x_{out}$  ( $3 \times 1$  vector), obtained as a defuzzified value from the fuzzy inference process that acts as the command vector for some mechanical action.

- output *Current\_State*, describes the necessary state changes.

The flowchart for the MillFSM agent is shown in Fig. 6.

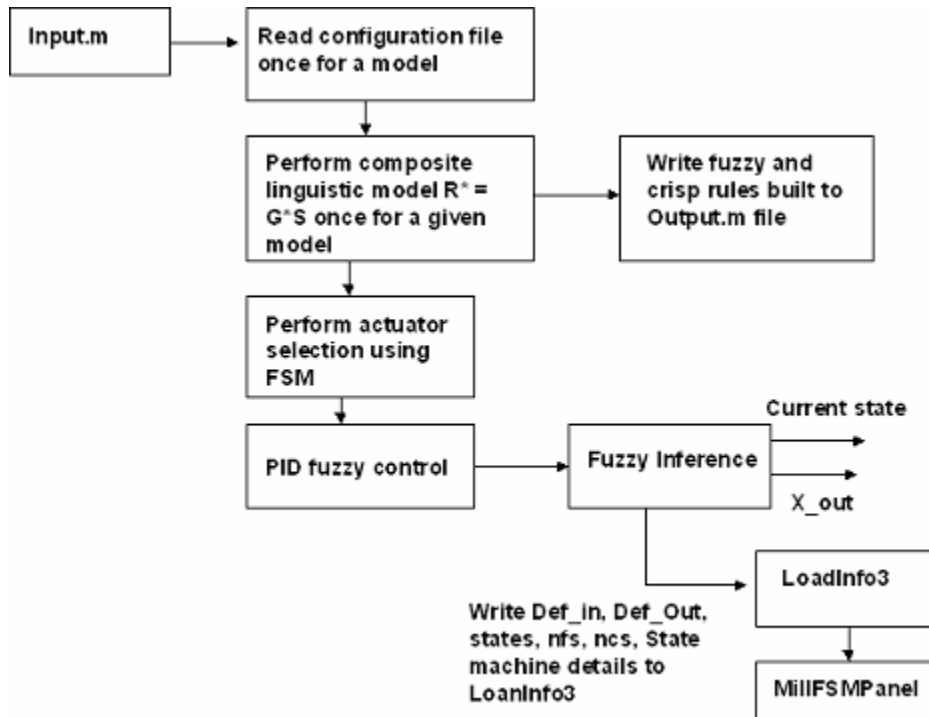


Fig6. MillFSM module in the FSA Broker

The MillFSMSIM agent and the MillFSM agent run in the broker in a closed loop configuration for 100 (default) iterations. The Broker also contains two panels that works in coordination with the MillFSMSIM and MillFSM agents. They are:

MillFSMSIMPanel

MillFSMPanel

The MillFSMSIMPanel displays the reference profile, calculated  $b$  and the calculated error vectors. All the vectors are of size  $32 \times 1$  and are displayed for every iteration the control loop goes through.

The MillFSMPanel displays the De-fuzzified Inputs, De-fuzzified Outputs, number of Fuzzy states, number of Crisp states, Current state, Previous state, times error taken by actuators either individually or as a whole and  $x_{out}$ . The display is modified for every iteration the control loop goes through. The whole Mill model implemented in the FSA architecture is as shown in Fig.7.

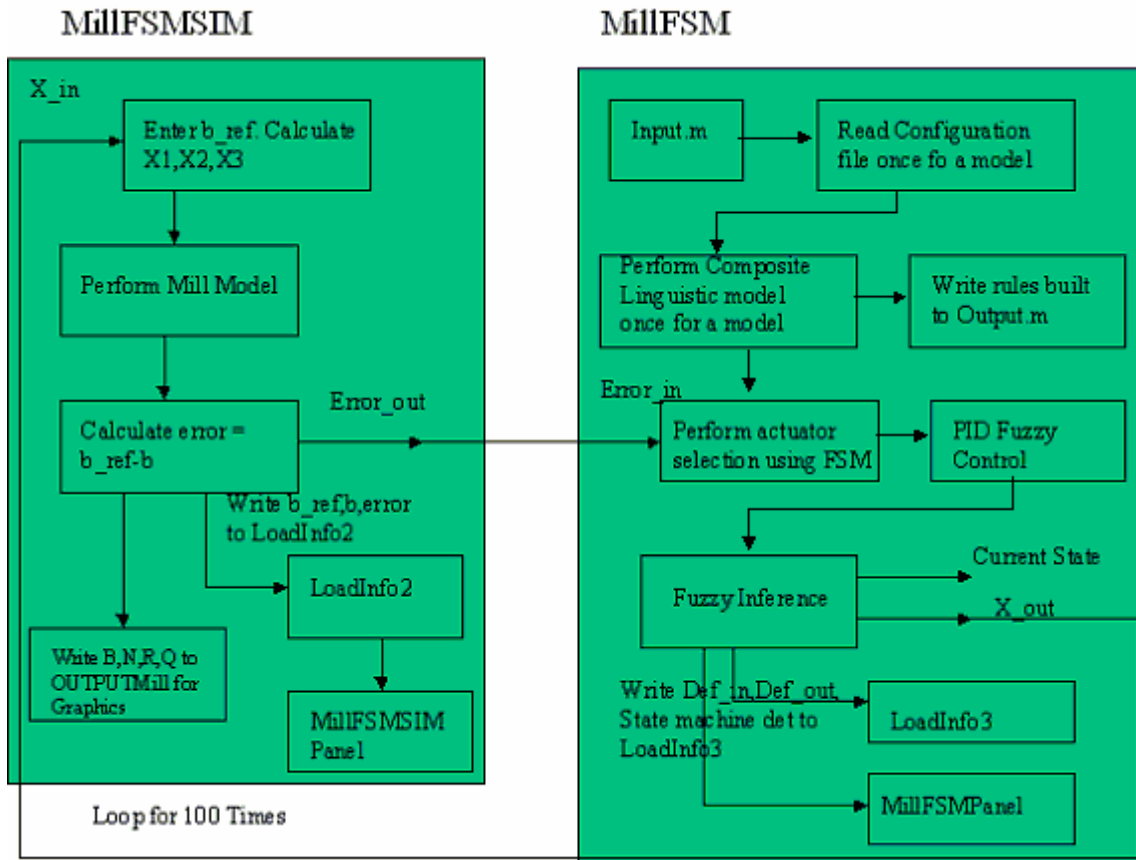


Fig7. Mill model in the FSA Broker

The Flatness Server Architecture broker consisting of the Mill model is as shown in Figures 8 – 11 below.

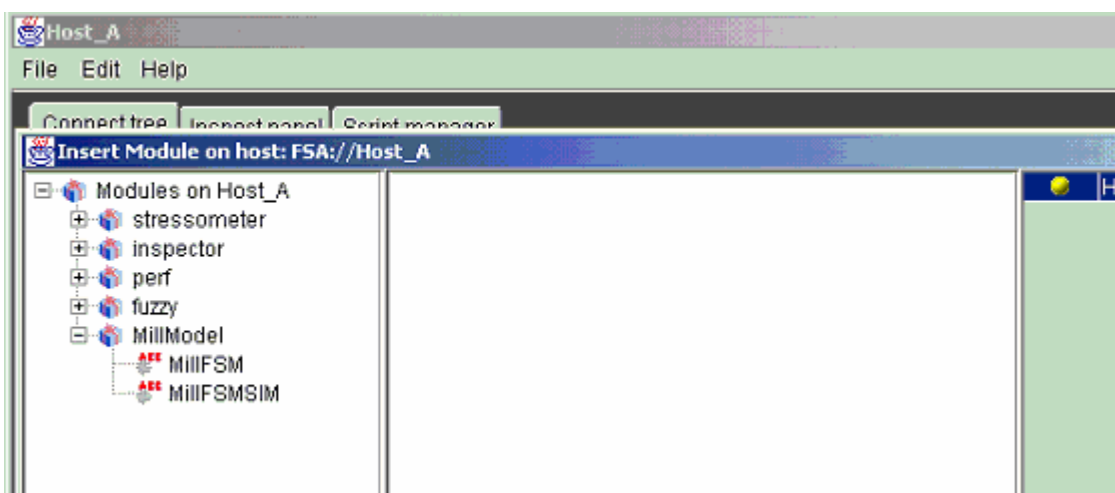


Fig8. Inserting Millmodel in the FSA architecture

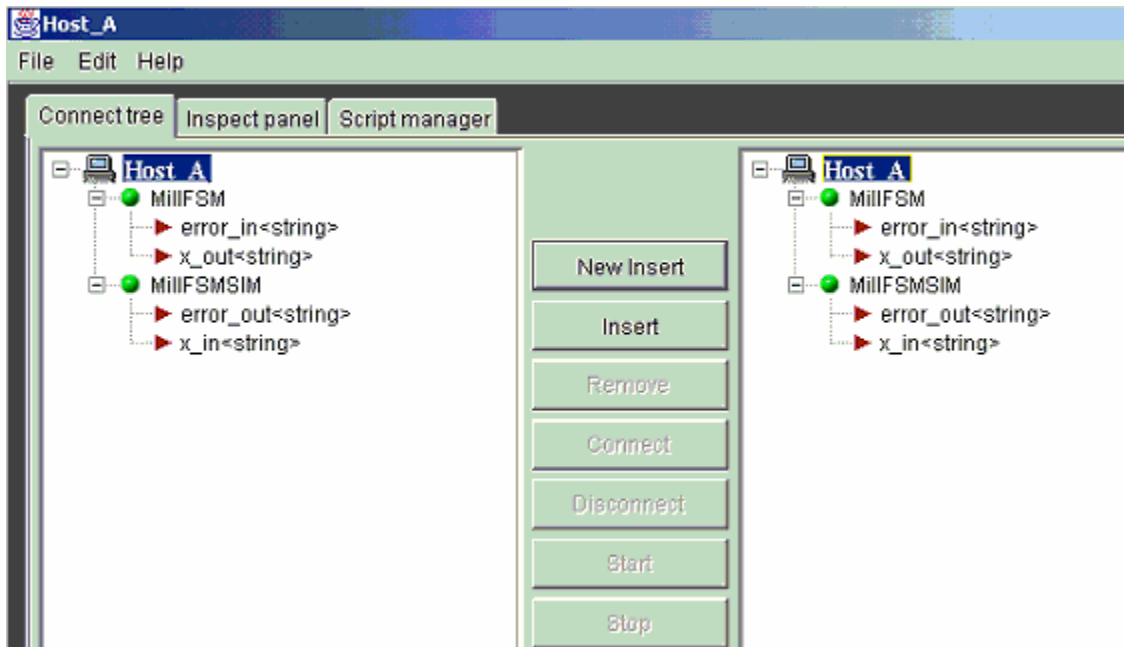


Fig9. MiIFSMSIM and MiIFSM agents in the FSA broker

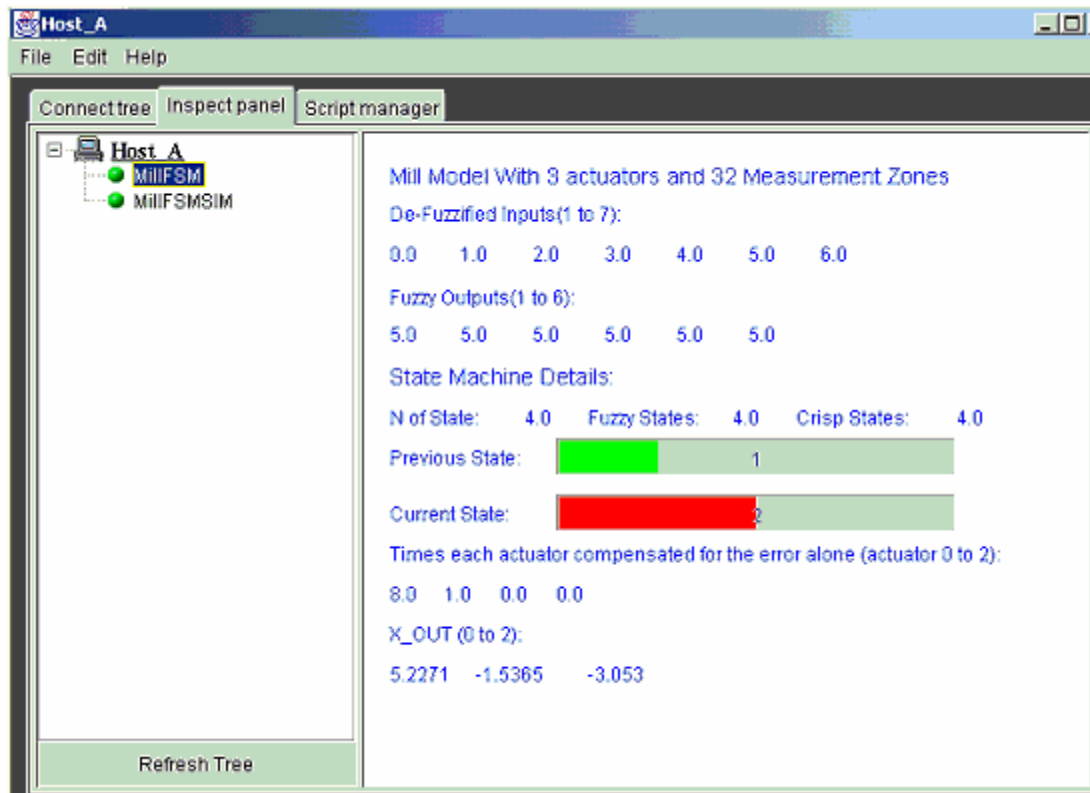


Fig10. MiIFSMSPanel in the FSA broker

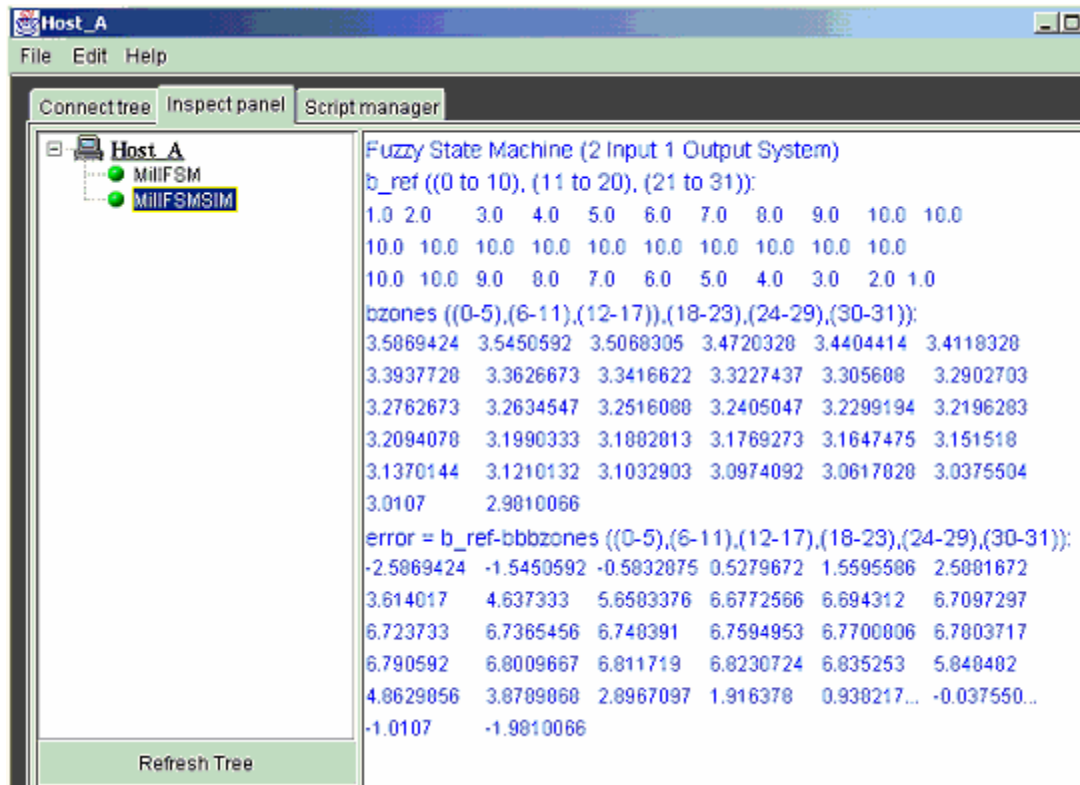


Fig11. MillFSMSIMPanel in the FSA broker

The simulation results for the closed loop flatness control are as shown in Figures 12 - 16 below.

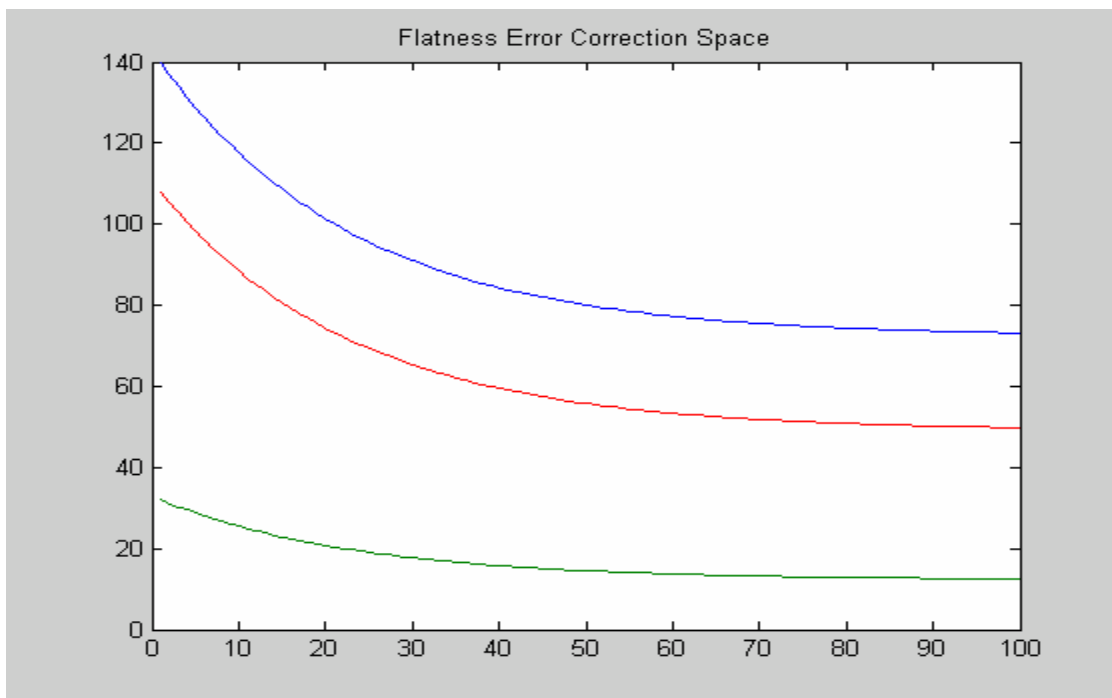


Fig12. Flatness error correction space

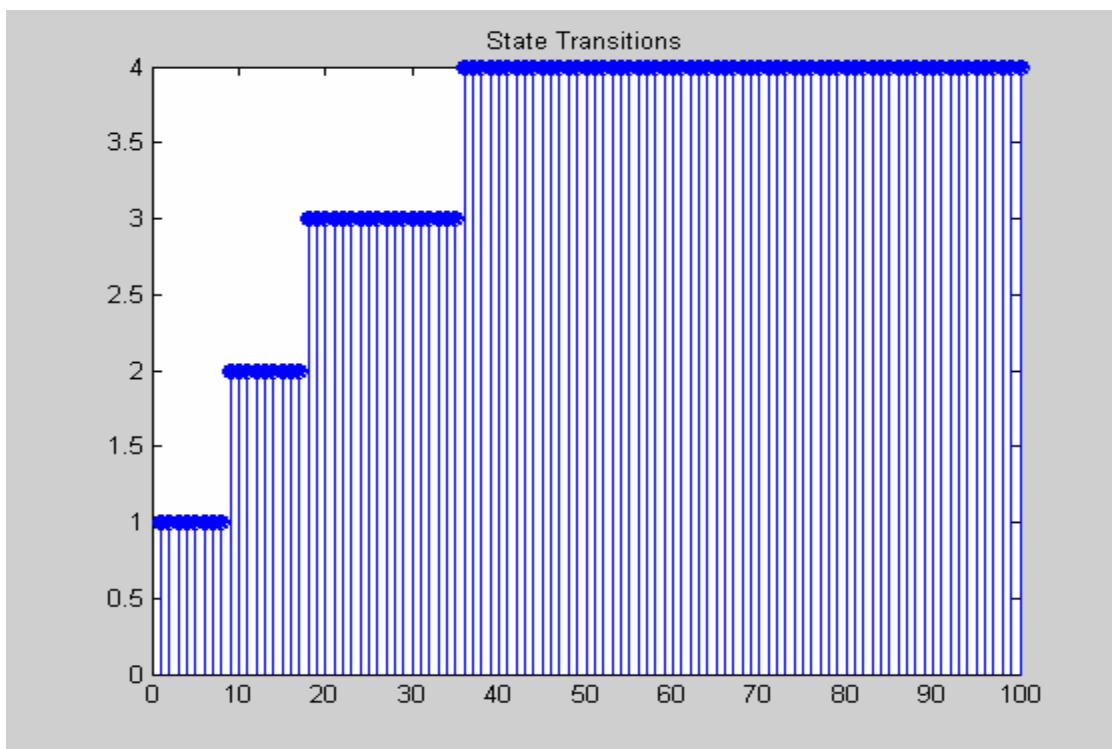


Fig13. State Transitions

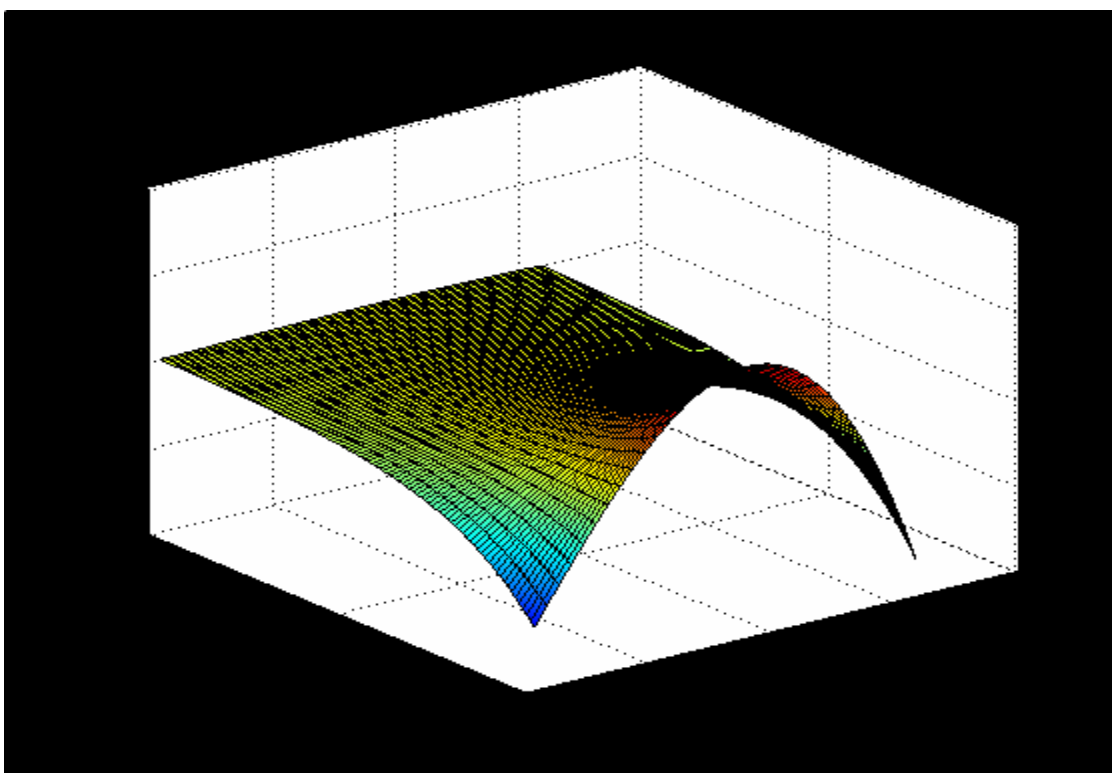


Fig14. Rest to be corrected, r

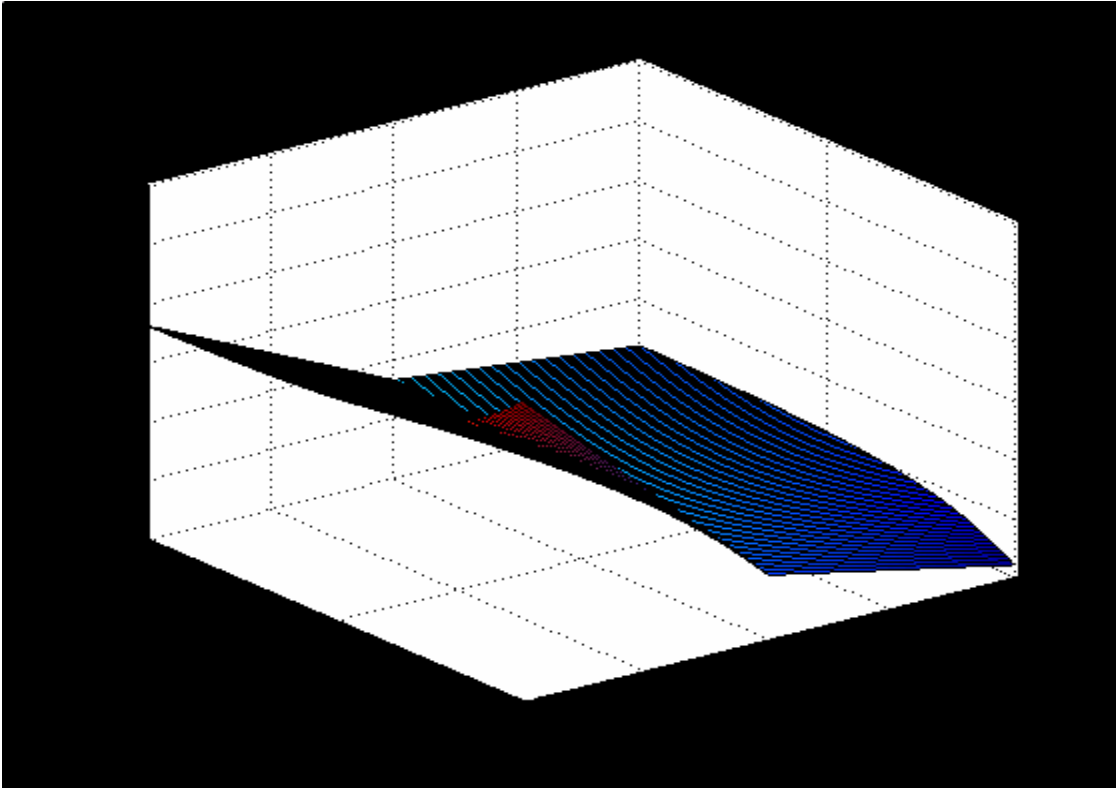


Fig15. Actuator Parameter, X

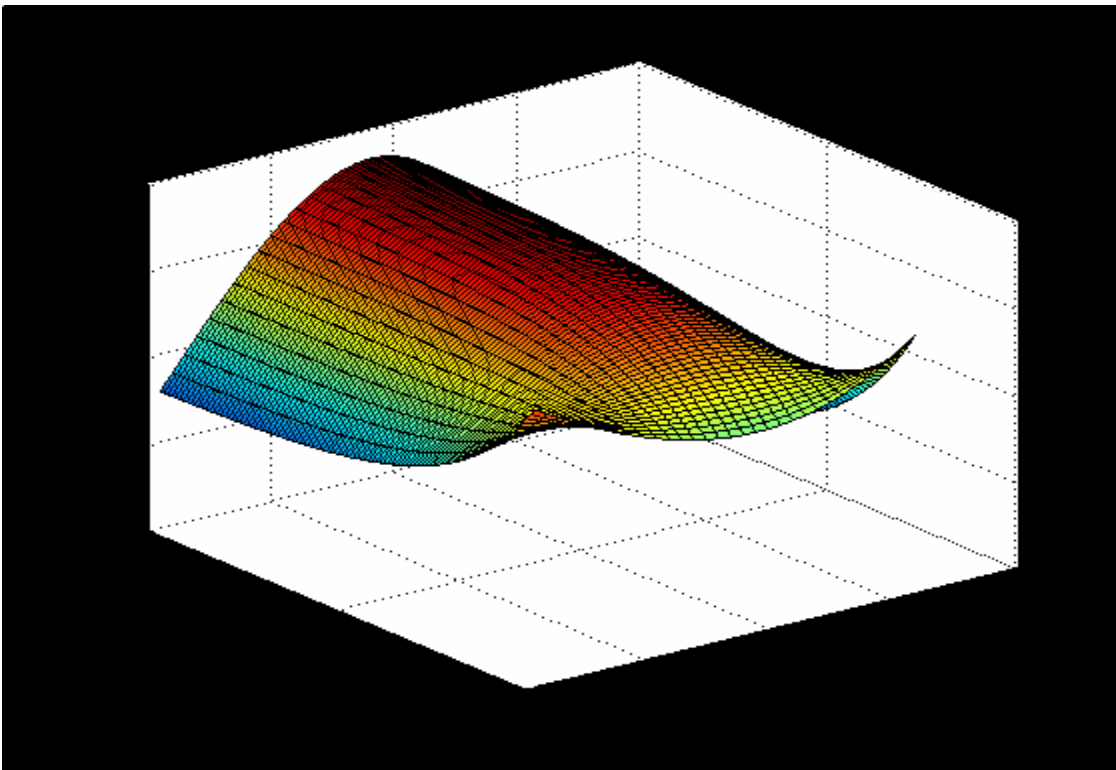


Fig16. Measured strip stress profile, b

## 7. REFERENCES:

- [1] M. Knapik, J. Johnson, Developing Intelligent Agents for Distributed Systems, Exploring Architecture, Technologies and Applications, McGraw-Hill, 1998.
- [2] MAS'99 Multi-Agent Systems in Production, Proc. of the First IFAC Workshop, Dec. 2-4, 1999, Vienna, Austria, Elsevier Science, 2000.
- [3] Selic, B., Gullekson, G., Ward, P.T., Real-Time Object-Oriented Modelling, John Wiley & Sons Ltd., NewYork, 1994.
- [4] J.L. Grantner, G. Fodor, D. Driankov, Hybrid Fuzzy-Boolean Automata for Ontological Controllers, Proceedings of the 1998 IEEE World Congress on Computational Intelligence, FUZZ-IEEE'98, Vol. I, pp. 400-404, Anchorage, Alaska, May 4-9, 1998.
- [5] J. L. Grantner, G. A. Fodor, D. Driankov, The Virtual Fuzzy State Machine Approach - A Domain-Independent Fault Detection and Recovery Method for Object-based Control Systems, 18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS'99, June 10-12, 1999, New York, NY, Proceedings, pp. 158-162.
- [6] J. L. Grantner, G. A. Fodor, Fuzzy Automaton for Intelligent Hybrid Control Systems, 2002 World Congress on Computational Intelligence, WCCI'02/FUZZ-IEEE'02, May 12-17, 2002, Hilton Hawaiian Village Hotel, Honolulu, HI, in the CD Proceedings, ISBN: 0-7803-7281-6
- [7] J.L.Grantner, R.Gottipati, G.A.Fodor, Intelligent Fuzzy Controller Laboratory, 2004 ASEE Annual Conference and Exposition, Saltlake City, Utah, USA, June 20-23 2004
- [8] J.L.Grantner, G.A.Fodor, R.Gottipati, Fuzzy Logic Enabled Agent for Supervisory Control, 2004 International Conference on Fuzzy Systems, Budapest, Hungary, July 25-29 2004