Proceedings of the Conference on Modelling and Simulation 1994
eds. Guasch and Huber

AUTOMATED SYNTHESIS OF A FUZZY CONTROLLER FOR CARGO SHIP STEERING BY MEANS OF QUALITATIVE SIMULATION

Francisco Mugica–Alvarez *
Dpt. ESAII
Universidad Politécnica de Cataluña
Diagonal 647, 2da. planta
Barcelona 08028, Spain
Mujica@esaii.upc.es

François E. Cellier
Electr. & Comp. Engr. Dept.
University of Arizona
Tucson, AZ 85721
U.S.A.
Cellier@ECE.Arizona.Edu

ABSTRACT

In this paper, Fuzzy Inductive Reasoning, *FIR*, will be applied to the systematic design of an autopilot of a cargo ship to demonstrate that this methodology works well when applied to highly non–linear plants. The application at hand shows as well the use of several fuzzy inductive reasoners coupled in parallel to resolve a single MIMO controller into several MISO controllers. The functionality of the approach is demonstrated by simulating the quantitative plant together with its qualitative controllers in a mixed quantitative/qualitative simulation enviroment. An important issue in the context of the systematic design presented here is the use of a new technique for obtaining the inverse model dynamics of the plant. In order to evaluate the performance of the Fuzzy Inductive Reasoning Based Controller, *FIRBC*, open–loop and closed–loop design stages are tested by means of simulation, and the results are compared with a Fuzzy Model Reference Learning Controller, *FMRLC*, recently proposed for use in the same application.

INTRODUCTION

Control of complex processes, such as highly non–linear, time–varying, or variable–structure systems, is still a topic of great interest to the engineering community. Classical controllers, such as the PID controller, don't work adequately when used to operate highly non–linear plants, plants subjected to strong environmental perturbations (disturbances), or plants with vastly different operating points. In all these cases, it is necessary to constantly re–calibrate the controller parameters for optimal control system performance.

Adaptive controllers [3] have been introduced as a means to automate the process of controller parameter re–adjustment. However, the first generation of adaptive controllers was based on an assumption of a linear plant to be controlled. More recently, a second generation of adaptive controllers incorporating Artificial Intelligence [9], Fuzzy System [13], and/or Neural Network [15] technologies was introduced to circumvent the previously mentioned limitations of first generation adaptive controllers.

The fuzzy controller has potential advantages when the specifications of control require robustness, adaptability, and flexibility to either environmental perturbations or effects of unmodeled plant dynamics. Fuzzy controllers are basically logic controllers using multivalued logic. The fuzzy controller discretizes the non–linear high–dimensional continuous operating space of the controller into discrete classes, then optimizes the behavior of the controller in terms of these discrete classes as a finite state machine, and finally uses the fuzzy membership information to smoothly interpolate beetween neighboring discrete points in the continuous operating space.

The application of fuzzy controller technology has, however, some drawbacks. One of them is that, until now, fuzzy controllers were always designed heuristically and in an ad hoc manner, based on expert knowledge of a human operator. Whereas this property of fuzzy control is seen as an *advantage* by some researchers (no complicated theory is needed to come up with a fuzzy controller for a highly non–linear plant; since the fuzzy controller is not based upon a plant model, it may be more robust to effects of unmodeled plant dynamics), it makes the tuning of a fuzzy controller with multiple inputs and multiple outputs a very tedious and awkward undertaking.

This article deals with the *systematic* design of fuzzy controllers, preserving the benign properties of the fuzzy controller technology, while simplifying the design process and drastically reducing the time needed to tune the fuzzy controller for optimal control performance. The proposed methodology makes use of qualitative modeling and simulation based upon Fuzzy Inductive Reasoning (FIR) [5]. In [7], the principal feasibility of the proposed controller design technique has been demonstrated by means of a very simple, synthetic, and linear plant to be controlled, and the methodology was outlined in detail.

In the present paper, the methodology will be applied to the systematic design of an autopilot of a cargo ship to demonstrate that the approach also works for non–linear plants as they are encountered in real–world applications. The cargo ship application was chosen since there already exists an ample number of previous publications dealing with this system [1, 2, 3, 11, 12, 13]. This makes it possible to compare the results obtained by the newly proposed systematic fuzzy controller design technique with those previously obtained and most recently summarized in [13].

*Supported by Consejo Nacional de Ciencia y Tecnología – México

SHIP STEERING MODEL

Direct Model

The ship dynamics can be described using a model with two input variables, the rudder angle, δ, and the engine thrust, f, and two output variables, the heading of the ship, ψ, and its velocity, u. However, for the purpose of the autopilot design, this model is unnecessarily complicated. Most autopilots are designed to keep the ship on a predetermined course while minimizing fuel consumption. To this end, it suffices to work with a simplified model that uses the rudder angle, δ, and the ship velocity, u, as inputs, and the heading of the ship, ψ, as its single output. Such a model was proposed in [4]. It can be encoded as:

```
model class CargoShip
  terminal delta, u, psi
  local psid, psi2d, psi3d, deltad
  local H, K, tau1, tau2, tau3
  local tau1inv, tau2inv, tau12inv, lu
  parameter K0 = -3.86, tau10 = 5.66
  parameter tau20 = 0.38, tau30 = 0.89
  parameter l = 161.0, a = 1.0, b = 1.0
    tau1inv = 1.0/tau1
    tau2inv = 1.0/tau2
    tau12inv = tau1inv * tau2inv
    lu = l/u
    K = K0/lu
    tau1 = tau10 * lu
    tau2 = tau20 * lu
    tau3 = tau30 * lu
    psid = der(psi)
    psi2d = der(psid)
    psi3d = der(psi2d)
    deltad = der(delta)
    H = (a * psid * psid + b) * psid
    psi3d + (tau1inv + tau2inv) * psi2d + tau12inv * H- >
      = K * tau12inv * (tau3 * deltad + delta)
end
```

using a Dymola [5, 10] description. The ship dynamics are captured by a single third-order non-linear differential equation in the ship heading, ψ. The rudder dynamics, $\dot{\delta}$, are also taken into account. τ_1, τ_2, and τ_3 are three time constants that depend on the ship velocity, u, and the ship length, l. K is a gain value that also depends on the same quantities. Finally, H describes a non-linear damping term. The notation should be fairly self-explanatory.

Reference Model

The ship is supposed to behave similar to the following second-order linear reference model:

```
model class Reference
  terminal psir, psim
  local psimd, psim2d
    psimd = der(psim)
    psim2d = der(psimd)
    psim2d + 0.1 * psimd + 0.0025 * psim = 0.0025 * psir
end
```

where ψ_m represents the reference heading of the ship, and ψ_r denotes the set value of the heading. Both the ship model and the reference model are exactly as used in [13] (Eqs. (7) and (9)).

DESIGN PHYLOSOPHY

If the *inverse plant dynamics* are perfectly known, the control problem becomes trivial [7]. A naïve approach to trivial model reference control of a plant using the inverse dynamics is outlined in Fig.1.

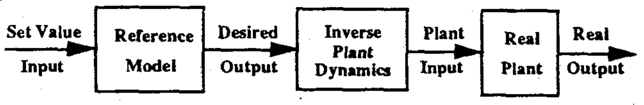

Figure 1: Naïve Controller

If the plant dynamics are exactly known, the two rightmost boxes cancel out, and the overall system behaves exactly like its reference model. The real output will be exactly equal to the desired output. Of course, the naïve controller doesn't work. The reader may consider a linear plant for simplicity. If the plant is strictly proper (more poles than zeros), the inverse plant is non-proper, i.e., exhibits differential behavior. If the plant is non-minimum phase (zeros in the right half plane), the inverse plant is unstable. In that case, an unstable pole-zero pair is being cancelled between the inverse dynamics model and the direct plant model, a nono for every control engineer. Finally, since this is open-loop control, the controller cannot correct for unmodeled plant dynamics or disturbances.

So, if this approach doesn't work, why mention it? There is nothing fundamentally wrong with trying to approximate inverse plant dynamics. In fact, all controller designs are either directly or indirectly based on such approximations. The intrinsic control problem is: given a plant and a desired output trajectory, what is the best possible input trajectory that shall make the system output look as similar as possible to the desired output trajectory? In order to answer this question, one needs either directly or indirectly a knowledge of the inverse plant dynamics. Thus, what is wrong in the naïve controller design is not the fact that it is based on inverse plant dynamics, but only the way in which this knowledge was used.

The previous objections will now be countered one at a time. Most plants are strictly proper, i.e., the plant does not have a direct input/output coupling, a step applied to the input does not lead to a step of the output (in this formulation, the concept carries through also to non-linear plants). This is not a serious problem. All that needs to be done to solve it is to realize that both the reference model and the inverse plant dynamics are *models*, thus, these are sets of differential equations that don't need to be realized separately. It suffices to give the reference model sufficiently many poles to make the combined system consisting of the reference model cascaded with the inverse plant dynamics model at least proper. In the example at hand, the plant itself has three poles and one zero. Thus, the reference model needs at least two more poles than zeros to make the cascaded system proper. This is exactly what the reference model was used for in the first place (although many controller design techniques may not say so explicitly).

The cascaded model can be described as follows:

```
model CascadeSystem

  submodel (Reference) Ref
  submodel (CargoShip) Plant

  input psir, u
  output delta

    Ref.psir = psir
    Plant.psi = Ref.psim
    Plant.u = u
    delta = Plant.delta

end
```

The *CascadeSystem* model invokes two submodels, one of class *Reference* called *Ref*, the other of class *CargoShip* called *Plant*. The model references two external inputs (driving functions) and one output, and connects the subsystems into the system. ***Plant.psi* = *Ref.psim*** means that the variable ψ of the plant is to be set equal to the variable ψ_m of the reference model, etc.

The plant got connected into the cascade system "the wrong way around." Consequently, this is a higher-index model, since there exist constraints between outputs of integrators. However, Dymola is perfectly capable of automatically reducing the index of a higher-index model, thereby generating a model in state-space form [6].

The following Dymola script file:

```
enter model
  @ship.dym

differentiate

variable state ship.delta, ship.psi, ship.psid

partition

language acsl
outfile ship.csl
output model

stop
```

generates an ACSL [14] simulation program for the cascaded system. The *differentiate* command performs the index reduction. It turns out that the model was of index 4, thus the Pantelides algorithm [16] had to be applied thrice in a row to reduce the model to index 1. Although the sum of state variables of the two submodels is five, the overall system order is only three, thus three state variables need to be selected. The *partition* command then solves the computational causality assignment problem, determining which equation should be solved for what variable. Finally, an ACSL model is generated that can be simulated after plugging in appropriate driving functions.

Secondly, what about non-minimum phase behavior? Indeed, large ships have a tendency to exhibit non-minimum phase behavior. Such behavior in a human-controlled system is annoying at best. The pilot turns the steering wheel to the left, and the boat reacts by turning right. So, the pilot turns the steering wheel more left, and the boat turns more right [8]. The answer is again simple: a local controller needs to be built that moves the unstable zeros into the left-half complex plane. Subsequently, the original plant together with its local controller is treated as the new plant to be controlled.

The final objection is the most serious one, and this is where control engineers earn their income. The trick is to modify the design such that the same objectives are attained while changing the former open-loop control into a closed-loop control with additional properties relating to disturbance suppression and plant parameter sensitivity reduction.

FUZZY CONTROLLER SYNTHESIS

The overall design of a FIRBC can be decomposed into the following five stages:

- First, the cascade model consisting of the reference model and the inverse plant dynamics model must be obtained following the strategy explained in the previous section of this paper.
- Next, a qualitative model of the fuzzy controller(s) must be obtained based upon measurements of the reference input(s) of the control system, the output(s) of the reference model, and the output(s) of the cascade model.
- The design of the controller(s) must then be validated in open loop by comparing the plant input(s) suggested by the fuzzy controller(s) with the desired plant inputs obtained earlier in the design.
- Subsequently, the fuzzy controller can be integrated with the plant in a closed-loop configuration. Stability and performance of the control system are being tested.
- Finally, stabilization/tracking control loops can be added to the overall control structure at this point in time if this turns out to be necessary.

Stage I of the design has already been explained in the previous section of the paper. The symbolic formula manipulator inside Dymola has been proven capable of deriving a closed-form analytical model of the cascade system out of independent descriptions of the direct plant dynamics and the desired reference model.

However, before the next stage can be tackled, one remaining problem with the first stage needs to be addressed. In the direct plant dynamics model, the rudder deflection, δ, was used as an input. However, the model itself contains also the variable $\dot{\delta}$. This is awkward since it forces the simulation program to numerically differentiate a qualitatively computed input variable — an utterly dubious undertaking at best.

Several references have overcome this problem by neglecting the rudder dynamics, i.e., by setting $\dot{\delta} = 0$, claiming that the rudder dynamics are so much faster than the ship dynamics that they can be ignored. However, when trying this approach, it could be noticed that the rudder deflection often exceeds the physically allowable limits of 37°. The rudder deflection used to control the ship can be kept within the allowed range by placing a hard limiter between the controller and the plant. However, this introduces a cumulative error degrading the control performance beyond acceptable limits.

A second solution might be to replace the previously used input of the plant, δ, by its derivative, $\dot{\delta}$, and numerically integrate $\dot{\delta}$ into δ inside the plant model. However, this approach doesn't work very well either, since $\dot{\delta}$ is less strongly coupled to the plant output, ψ, than δ. This will be shown in due course. Thus, this approach weakens the performance of the fuzzy controller beyond recovery.

A third approach is to sever the relationship between δ and $\dot{\delta}$ inside the plant model, eliminating the equation *der*(*delta*) = *deltad* from the plant model. *delta* and *deltad* can now be treated as two separate input variables to the plant, and two separate (parallel) fuzzy controllers can be designed, one computing values of *delta*, and the other computing values of *deltad*. This approach worked beautifully.

Now, the attentive reader may wonder how the relationship between δ and $\dot{\delta}$ might be severed in a real ship. Where does the rudder need to be sawn apart in order to achieve *that* trick? Of course, this cannot be done. However, it is perfectly feasible to design a PID-type controller that takes as inputs the signals *delta* and *deltad* suggested by the two fuzzy controllers, and generates as output a single signal $\dot{\delta}$, such that $\dot{\delta}$ is as close as possible to *deltad*, yet driving δ to *delta*. This PID controller is harmless since it only corrects for errors that are of second-order small. Consequently, it does not need to be adjusted for different operating points.

Qualitative Fuzzy Controller Design

In the second design stage, the two qualitative controller models need to be identified. In order to do so, the following experiment is performed, as illustrated in Fig.2:

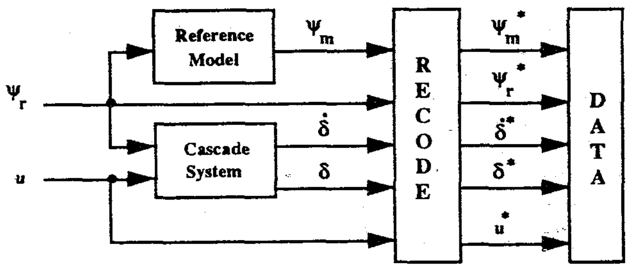

Figure 2: Qualitative model identification - Exp.1.

Binary random noise input is applied to the first system input, ψ_r, and a sinusoidal excitation is applied to the second system input, u. The reactions of the reference model, ψ_m, and of the cascaded model, δ and $\dot{\delta}$, are observed. All five quantities are recorded into a *raw data matrix*, where each column denotes one variable and each row denotes one data record (sampling point). The simulation was carried through 7000 seconds of simulated time. The first 6000 seconds were used for identifying the qualitative models. The remainder of the time history is used for validation purposes. The sampling time was chosen to be 1 second. The training data are shown in Fig.3.

Using *fuzzy recoding*, a technique explained in [5], each quantitative data value is then converted to an equivalent

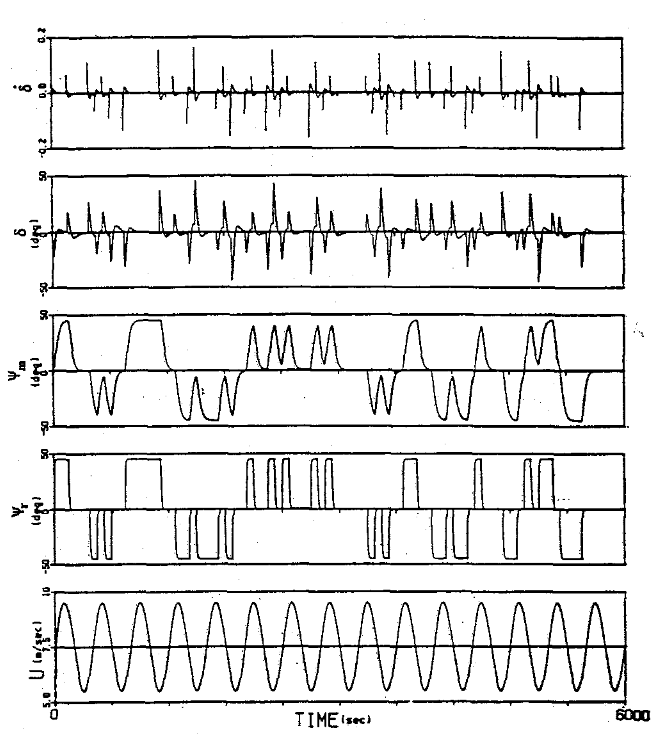

Figure 3: Training data set.

qualitative triple consisting of a class value, a fuzzy membership value, and a side value. As a result of this transformation, three data matrices are obtained, one containing the class values, the second containing the membership values, and the third containing the side values. These three matrices are then used in the process of fuzzy inductive reasoning.

In the next step, the two qualitative models are generated using optimal mask analysis as explained in [5, 7]. The topology of the two controller models is shown in Fig.4. An optimal mask is a temporal causal relation between the input variables and the output variable. The optimality function is based on the forecasting power of the mask, computed through use of the Shannon entropy measure.

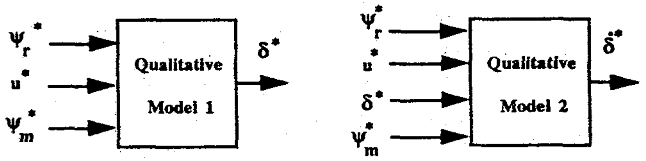

Figure 4: Controller topology - Exp.2.

Thereby, a qualitative triple of the first desired plant input, *delta*, is computed as a qualitative function of the two system inputs, ψ_r and u, the output of the reference model, ψ_m, and past values of all these quantities. A qualitative triple of the second desired plant input, *deltad*, is computed as a qualitative function of the two system inputs, ψ_r and u, and the variable δ. It turns out that the optimal mask for *deltad* does not make use of the output of the reference model, ψ_m, at all. This is the reason why the previously attempted solution of replacing δ by $\dot{\delta}$ as a control input to the plant failed.

Validation of the Qualitative Models

In the third design stage, the results of two qualitative simulations of the two models across 1000 seconds are compared with the last 1000 seconds of the previously performed quantitative simulation in open loop, in order to verify the quality of the forecast. This process is shown in Fig.5 where *REGEN.* is an abbreviation for *regeneration*, a technique converting qualitative triples back to quantitative variables. Thus, the *RECODE* module is the *fuzzifier*, whereas the *REGEN.* module is the *defuzzifier* in the system.

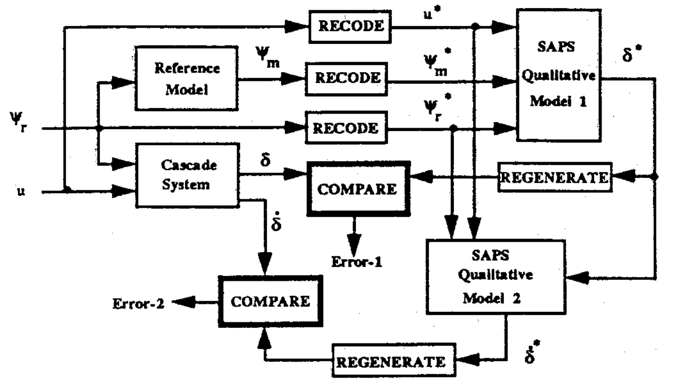

Figure 5: Validation process.

The good results of the validation stage are shown in Fig.6. Error signals were displayed since the original signals themselves are practically indistinguishable one from another.

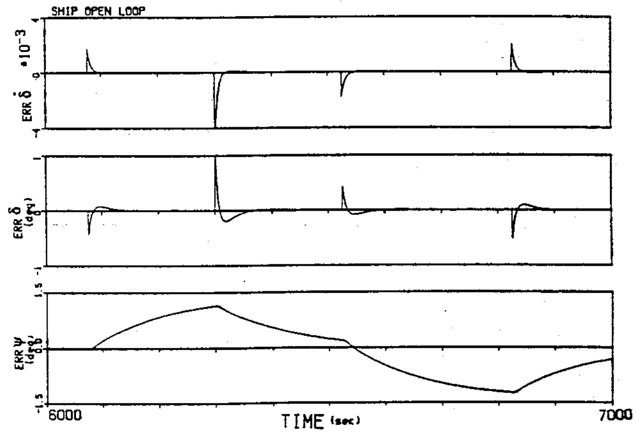

Figure 6: Forecast quality results.

Integration of Fuzzy Controllers

Once the errors in the previous experiment have been shown to be sufficiently small, the control loops can be closed as illustrated in Fig.7.

There is no direct error computation between the desired output, ψ_m, and the real output, ψ, as in most of the other control configurations. The trick is that the qualitative model was designed using ψ_m, but is now used with ψ_m replaced by ψ. This has the same overall effect. It can be noticed that the previously designed qualitative model has now become the fuzzy controller in this configuration.

However, no–one told the fuzzy controller that it is important to keep ψ close to ψ_m. In fact, the FIRBC doesn't even know what ψ_m is. Consequently, the ship position will ever–so–slowly drift away from its desired course. Thus, a tracking control loop is introduced, in which the difference between the desired ship position, ψ_m, and the real ship position, ψ, is fed back into the rudder deflection.

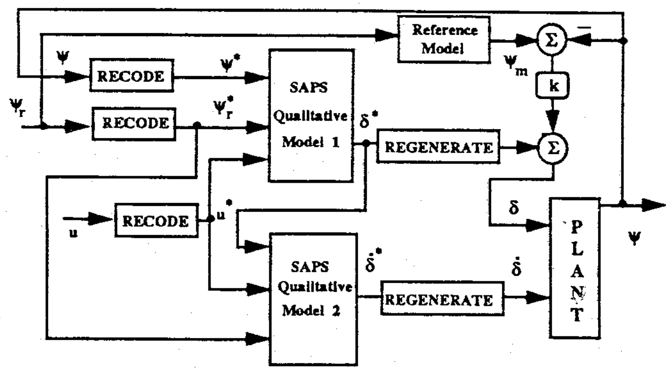

Figure 7: Fuzzy controller – FIRBC.

The malevolent reader may now suspect that an oldfashioned P–controller was introduced through the back door that does all the work, whereas the fuzzy controller was only left in the system to make the paper pass by the reviewers. This is not true. It is like in driving school. It is really the student who does all the driving. The teacher only reaches over from time to time to give an ever–so–gentle nudge to the steering wheel in order to prevent the car from unfriendly encounters with nearby trees. It is really the fuzzy controller that does all the work. The feedback signal that comes through the tracking control loop is much smaller in magnitude than the signal that comes through the fuzzy controller. All it does is to provide an additional incentive to the plant for staying on track. The feedback gain, k, assumes a value of $k = 0.2$. This value was found through experimentation. Since the P–controller only corrects errors that are of second–order small, no adjustments to the operating point are needed.

In its feedback configuration, the fuzzy controller will be able to correct for the effects of environmental perturbations and unmodeled plant dynamics. However, if these effects become prominent, the control may deteriorate. For example, wind velocity and direction will have a substantial effect on ship performance, and consequently, an indirect correction may not be good enough. Yet, the solution is simple. All that needs to be done is to augment the plant model *CargoShip* such that it accounts for the effects of these two additional input variables. Dymola will then automatically include these variables also in the cascade model. They are then treated in exactly the same fashion as the ship velocity, u. The raw data model obtains two additional columns, and the qualitative model obtains two additional inputs, but everything else remains the same. Of course, since the qualitative model now has four different inputs, considerably more training data may be needed for the data base used by the fuzzy controller. However, all this can be done off–line, and is therefore relatively harmless.

In order to compare the performance of the FIRBC with earlier designs such as the FMRLC design advocated in

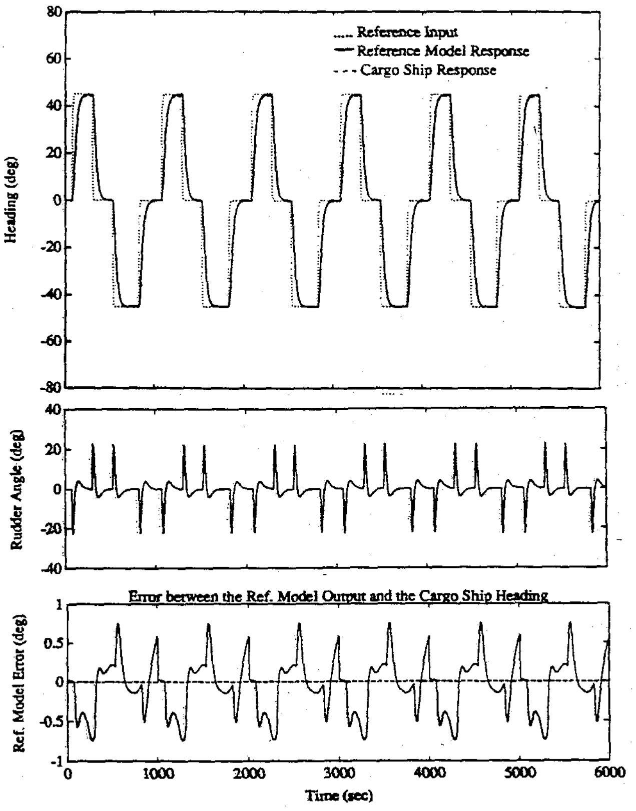

Figure 8: FIRBC comparative performance.

[13], the same experiment was reproduced that Layne and Passino used to test the performance of their design. The characteristics of this experiment are: constant ship velocity of 5.0 m/s, six cycles of perturbation of ψ_r: (1) course of 45° to the left during the first 250 seconds, (2) course of 0° during the nest 250 seconds, (3) course of 45° to the right during the following 250 seconds, (4) course of 0° during the last 250 seconds of the cycle. The results of this closed–loop experiment are shown in Fig.8. Both responses look practically identical, and both perform better than the Gradient and Lyapunov Model Reference Adaptative Controllers. Both techniques also avoid unrealistically large rudder deflections. In contrast to the FMRLC, the FIRBC has been identified for perturbations of both u and ψ_r, and does not need to be reidentified if the ship velocity changes.

With respect to the amount of input energy spent on the control in order to obtain an accurate tracking of the reference course defined as $E = \delta^2$ [13], the FMRLC spends $E = 17.3368$ during the 6000 seconds of the experiment. The FIRBC spends $E = 13.111$ during the same period.

CONCLUSIONS

The proposed architecture is not truly a learning control scheme.It is an optimal control scheme that synthesizes an optimal set of fuzzy rules off–line from a set of training data. It is proposed that using the knowledge available in the form of a quantitative direct plant model is an advantage. Direct plant dynamics are fairly easy to come by, and as long as the process of generating from it the cascade model and from there the fuzzy controller can be fully automated, there is nothing wrong with this approach. Moreover, it would be fairly easy to build on top of the synthesized fuzzy control architecture presented in this paper a fuzzy membership adaptation scheme similar to the one advocated in [13] allowing the controller to adapt itself to slow variations in plant dynamics. As other fuzzy controller design techniques, also the newly proposed FIRBC methodology suffers from a high degree of heuristicism. No attempts have been made to prove stability or convergence. However, the results obtained look very promising indeed.

ACKNOWLEDGEMENTS

Research supported by Consejo Interministerial de Ciencia y Tegnologia (CICYT), project TIC93-0447. The authors are thankful to Dr. Rafael Huber of the Institut de Cibernètica for his support on this research.

References

[1] Amerogen, J. V. and A. Cate, "Model Reference Adaptative Autopilots for Ships," *Automatica*, **11**, pp. 441–449, 1975.

[2] Åström, K. J. and C. G. Kallström, "Identification of Ship Steering Dynamics," *Automatica*, **12(1)**, pp. 9–22, 1976.

[3] Åström, K. J. and B. Wittenmark, *Adaptative Control*, Addison-Wesley Publishing Company, 1989.

[4] Bech, M. and L. Smitt, *Analogue Simulation of Ship Maneuvers*, Technical Report, Hydro– og Aerodynamisk Laboratorium, Danske Tekniske Højskole, Lyngby, Denmark, 1969.

[5] Cellier, F. E., *Continuous System Modeling*, Springer–Verlag, New York, 1991.

[6] Cellier, F. E. and H. Elmqvist, "Automated Formula Manipulation Supports Object–Oriented Continuous–System Modeling," *IEEE Control Systems*, 13(2), pp. 28–38, 1993.

[7] Cellier, F. E. and F. Mugica (1994), "Inductive Resoning Supports the Design of Fuzzy Controllers," *Journal of Intelligent and Fuzzy Systems*, accepted for publication.

[8] Clark, D. W., "Self–Tuning Control of Nonminimum–Phase Systems," *Automatica*, **20**, pp. 501–517, 1984.

[9] de Silva, C., "An Analytical Framework for Knowledge–Based Tuning of Servo Controllers," *Engineering Aplications of Artificial Intelligence*, 4(3), pp. 177–189, 1991.

[10] Elmqvist, H., *A Structured Model Language for Large Continuous Systems*, Ph.D. dissertation, Report CODEN: LUTFD2/(TFRT–1015), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1978.

[11] Kallström, C. G., K. J. Åström, N. E. Thorell, J. Eriksson and L. Sten, "Adaptative Autopilots for Tankers," *Automatica*, **15**, pp. 241–252, 1979.

[12] Kallström, C. G. and K. J. Åström, "Experiences of System Identification Applied to Ship Steering," *Automatica*, **17**, pp. 187–198, 1981.

[13] Layne, J. and K. Passino, "Fuzzy Model Reference Learning Control for Cargo Ship Steering," *IEEE Control System*, **13(6)**, pp. 23–34, 1993.

[14] Mitchell & Gauthier Assoc., *ACSL: Advanced Continuous Simulation Language — User Guide and Reference Manual*, Concord, Mass, 1986.

[15] Narendra, K. S. and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, **1(1)**, pp. 4–27, 1990.

[16] Pantelides, C. C., "The Consistent Initialization of Differential–Algebraic Systems," *SIAM J. Sci. Stat. Comput.*, 9(2), pp. 213–231, 1988.