Optimizing Nuclear Reactor Operation Using Soft Computing Techniques

J.O. Entzinger[1] and D. Ruan[2]

[1] University of Twente
Department of Mechanical Automation
PO-Box 217
NL-7500 AE Enschede, The Netherlands
Jorg@Entzinger.nl

[2] Belgian Nuclear Research Centre (SCK•CEN)
Department of Reactor Physics & Myrrha
Boeretang 200
B-2400 Mol, Belgium
DRuan@SCKCEN.be

Summary. The strict safety regulations for nuclear reactor control make it difficult to implement new control techniques such as fuzzy logic control (FLC). FLC however, can provide very desirable advantages over classical control, like robustness, adaptation and the capability to include human experience into the controller. Simple fuzzy logic controllers have been implemented for a few nuclear research reactors, among which the Massachusetts Institute of Technology (MIT) research reactor [1] in 1988 and the first Belgian reactor (BR1) [2] in 1998, though only on a temporal basis.

The work presented here is a continuation of earlier research on adaptive fuzzy logic controllers for nuclear reactors at the SCK•CEN [2, 3, 4] and [5] (pp 65–82). A series of simulated experiments has been carried out using adaptive FLC, genetic algorithms (GAs) and neural networks (NNs) to find out which strategies are most promising for further research and future application in nuclear reactor control.

Hopefully this contribution will lead to more research on advanced FLC in this domain and finally to an optimised and intrinsically safe control strategy.

Key words: Nuclear Reactors, BR1, Nuclear Plant Operation, Fuzzy Logic Control, Genetic Algorithms, Neural Networks

1 Introduction

In a broad range of applications fuzzy logic (FL) has established itself as a powerful alternative for classical (PID) control. However, when it comes to nuclear reactor operation, strict safety regulations prevent engineers from quickly developing and implementing new control methods. This makes the use of artificial intelligent control in this field still a tremendous challenge. Although research is going on and some successful implementations of FL were reported [1, 2], most implementations use very basic controllers with a static rule base. A good overview of FL applications in nuclear reactor operation can be found in [6] and in [7] for more recent developments.

1.1 Nuclear Reactor Control

The problem faced is how to control the power output of a nuclear reactor in an optimally safe and efficient way. The description below is based on the behaviour of the Belgian Reactor 1 (BR1, see Fig. 1), which is a graphite-moderated and air-cooled system fuelled with natural uranium.

Nuclear reactors have three key elements:

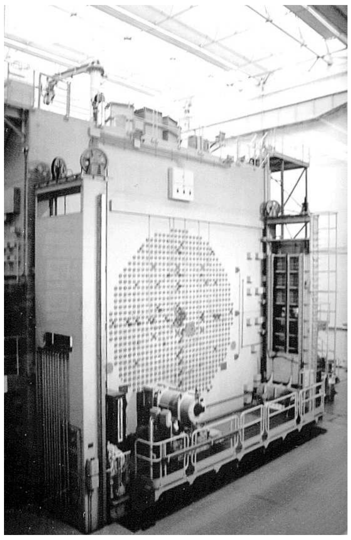
- radioactive fuel
- a moderator
- control rods

When the fuel is bombarded with free, low-energy neutrons, the fuel atoms split into two major fission fragments and release high-energy neutrons as well as energy in the form of heat. The free, high-energy neutrons are not likely to trigger other atoms to split. Therefore a moderator such as carbon (graphite) or hydrogen (in the form of water) is needed, so the high-energy neutrons lose energy by colliding into and bouncing off the moderator atoms. Now the free neutrons are slowed down, they are more likely to trigger other fuel atoms to split, causing a chain reaction as shown in Fig. 1(b).

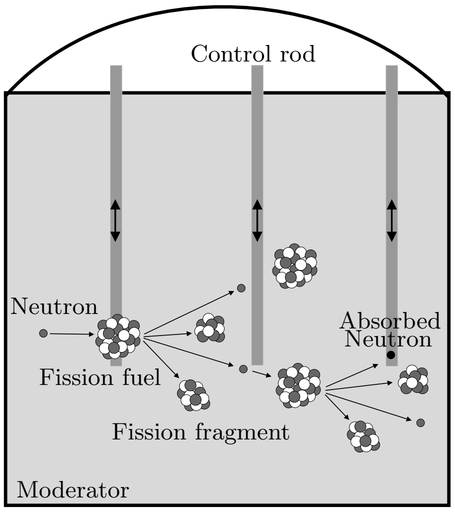
To control the chain reaction, control rods are inserted into or withdrawn from the reactor core. These rods are made of a material that absorbs neutrons, so when inserted further, more free neutrons are taken out of the chain reaction and the process is tempered. Of course for a steady-state process it is important to have one effective free neutron left from each fission. To raise the power output the control rods are temporarily withdrawn a little to keep slightly more free neutrons, increasing the number of fissions and thus the power output. When the power is near the desired level, they are inserted again to have a new steady-state, keeping one free neutron from each fission. There are three types of control rods:

- shim rods (also C-rods, for coarse control)
- regulating rods (also A-rods, for fine adjustments)
- safety rods (for very fast shutdown)

For a normal controller only the first two types are interesting, because the safety rods will only be controlled manually or by a supervisory controller. Normally only the regulating rods are active to level out small changes in the power output. Such changes could for instance be caused by a change in reactivity due to a raise of the reactor core temperature. When the regulation



(a) The BR1 reactor.



(b) Working principle of a nuclear fission reactor.

Fig. 1. The nuclear reactor.

rods are almost fully inserted or withdrawn, the shim rods are moved slightly, allowing the regulation rods to move back to a central position. Further the shim rods are only used at startup, shutdown and setpoint changes.

The BR1 nuclear reactor is currently controlled by a conventional simple on/off controller which only controls the regulation rods. The shim rods have to be moved manually, so the controller can only maintain a steady-state power level and is not used for setpoint changes. Of course also these setpoint changes should be carried out by a controller in future implementations. An early study of fuzzy logic control for the BR1 reactor ([2, 3]) resulted in a 1-year operation of a controller which actually did combine the shim rod and regulation rod control, however it was only used in steady-state situations.

1.2 Control of a Demonstration Model

Because it is difficult and time consuming to make a well resembling mathematical model of a nuclear reactor and moreover, the set of nonlinear differential equations would be hard to solve, newly developed controller types are tested on a demo model (derived from [4] and Chap. 4 of [5]). This model consists of a water tank which empties continuously through a small hole in the bottom and is filled simultaneously by two water flows as depicted in Fig. 2. The filling flows are controlled by valves, so the water level in the tank can be regulated.

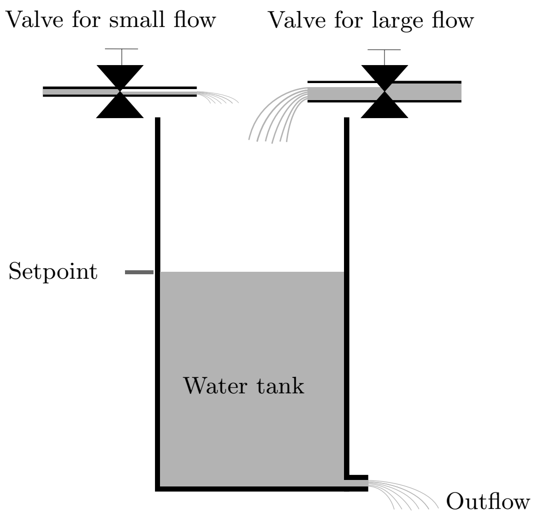


Fig. 2. Water tank demo model.

In this model the water level in the tank resembles the power output of the nuclear reactor. The speed of opening or closing the valves reflects the speed of inserting or withdrawing the control rods. One of the filling flows (VL) is made significantly larger than the other (VS) to copy the difference between shim rods and control rods. The control inputs are the difference between the current water level and the setpoint (D) and the change rate of the water level (DD).

Though this water tank demo model is much simpler than the nuclear reactor, it is still a non-linear system which is difficult to control. Therefore the demo is a good and representative testbed for the controllers investigated.

1.3 Directions of Exploration

For nuclear power plants it is important to quickly respond to a change in power demand. At the same time overshoot should be minimal for both safety and economic reasons. With these problems we find the classical control problem: the controller should be both fast and accurate.

The performance of the examined controllers will be evaluated based on their response to time to time setpoint changes, which is a more distinct and more interesting criterion for future applications than small disturbances in a steady-state.

Several options are investigated to improve a fuzzy logic controller (FLC) for speed, stability and accuracy. In this investigation soft computing techniques are used, mainly because of their robustness, their learning capabilities and their high level of problem independence. Options investigated are:

- Adaptive rule generation (Sect. 2)
- Input scaling (Sect. 2.4)
- Increasing the number of membership functions (Sect. 2.4)
- Optimization of membership functions by Genetic Algorithms (Sect. 3)
- Use of Neural Networks for plant simulation (Sect. 4.1)
- Fuzzy Neural Network controllers (Sect. 4.4)

A setup with an adaptive fuzzy rule base was already present from earlier research ([4] and Chap. 4 of [5]). Most tests are performed both with a static rule base and with an adaptive version.

Only a few possibilities are investigated, but some other options like different rule base adaptation mechanisms, membership function adaptation, plant identification techniques and advanced neuro-fuzzy systems might be interesting as well. Also non-fuzzy control and combinations of fuzzy logic with classical control have not been investigated, but are very well possible. It must be stressed that this is only an explorative investigation and no directly implementable result should be expected.

2 Rule Base Adaptation

A distinction will be made between the 'static' (reference) controller which has a fixed rule base and an 'adaptive' controller which has a rule base that is updated at each evaluation. The adaptive rule base introduces great flexibility, not only because it makes the controller (almost) problem independent, but also for instance because the controller can keep track of moving equilibria. However, in practice it is almost impossible to get an adaptive controller to work on-line in a nuclear reactor, due to safety regulations.

2.1 A Static Controller

All fuzzy variables can take five different values: Positive Big, Positive Small, Zero, Negative Small and Negative Big or PB, PS, ZE, NS and NB for short. For the valves VL and VS positive means opening and negative means closing, for the water level error D positive means too high and negative means too low and for the rate DD positive is raising and negative is dropping.

The membership functions (MFs) are shown in Fig. 3 and the static rule base is given in Table 2.1. The rule base table should be read as following: the value PS in the upper right corner of the rule base for VL means: '**If** $D = PB$ **and** $DD = NB$ **then** $VL = PS$'.

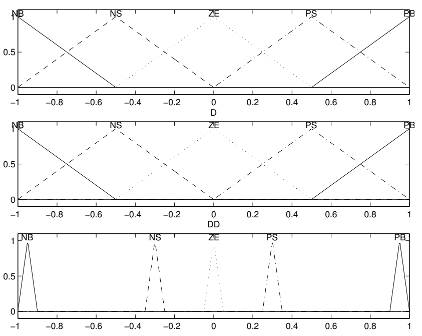
Table 1. Static rule base for VL and VS.

VL rule base

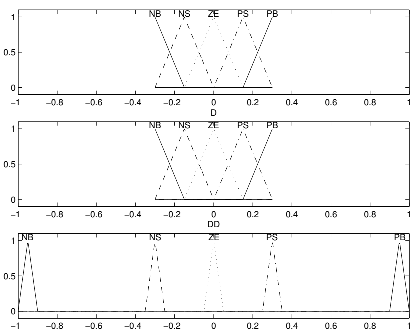
DD \ D	NB	NS	ZE	PS	PB
NB	PB	PB	PB	PS	PS
NS	PB	PB	PS	PS	PS
ZE	PB	PS	ZE	NS	NS
PS	PS	ZE	NB	NB	NB
PB	ZE	NB	NB	NB	NB

VS rule base

DD \ D	NB	NS	ZE	PS	PB
NB	ZE	ZE	ZE	ZE	ZE
NS	ZE	ZE	NS	PS	NS
ZE	ZE	PS	ZE	PS	PS
PS	ZE	NS	ZE	NB	NB
PB	ZE	NB	ZE	NB	NB



(a) Membership functions for VL



(b) Membership functions for VS

Fig. 3. Input and output membership functions.

2.2 An Adaptive Controller

The adaptive controller is just like the static controller, only the conclusion part of all rules is initialised to Negative Big (which is intrinsically safe) and its rule base is adapted while controlling the system according to two simple rules:

- If the water level is too low and still dropping, open the valve faster (or close it slower)
- If the water level is too high, close the valve faster (or open it slower)

or in more technical terms:

- **If** $D < 0$ **and** $DD \leq 0$ **then** raise the conclusion part of the most triggered rule with 1
- **If** $D > 0$ **then** lower the conclusion part of the most triggered rule with 1

Raising here means that the next value out of $\{NB, NS, ZE, PS, PB\}$ will be taken, i.e., NB becomes NS, NS becomes ZE etcetera. The conclusion part for $D = ZE$ & $DD = ZE$ is always set to ZE to make sure the system will stabilize.

These rules are chosen in such a way that overshoot is minimized, thus restraining the speed of the controller. Other rules and additional rules have been tried but did not provide a generally better solution: speed could only be gained at the expense of more overshoot.

2.3 Comparison of Simulation Results

The adaptive controller initializes itself very quickly and its response is often much faster than that of the static controller (see Fig. 4), which means that requests for more power can be met faster and that there is less waste of energy when the power need decreases. Disadvantages are of course the overshoot (although it is only slight) and the (short) fluctuation when meeting a new setpoint value.

The adapted rule base is presented in Table 2.3, where the changed rules are printed bold and italic. It may seem that not many rules have adapted, but this is quite logical because the four rules in the lower left corner and the centre rule will never change when using the proposed adaptation rules and the rules applying to a too high water level (i.e., the two most right columns) are supposed to make the water level decrease as soon as possible, which means they should have NB as a conclusion part. This means only the eight upper left rules (centre rule excluded) should change, which is exactly what happens. The upper rows do not change because of an improper scaling of the DD input variable.

Table 2. Rule base for VL and VS after adaptation.

VL rule base

DD \ D	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NB	NB
NS	NB	NB	NB	NB	NB
ZE	***PB***	***PB***	ZE	NB	NB
PS	NB	NB	NB	NB	NB
PB	NB	NB	NB	NB	NB

VS rule base

DD \ D	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NB	NB
NS	NB	***PB***	***PB***	NB	NB
ZE	***NS***	***PB***	ZE	NB	NB
PS	NB	NB	NB	NB	NB
PB	NB	NB	NB	NB	NB

2.4 Enhancements

Some enhancements can be made to the adaptive controller to improve its performance. The first thing would be a scaling of the inputs. The level in

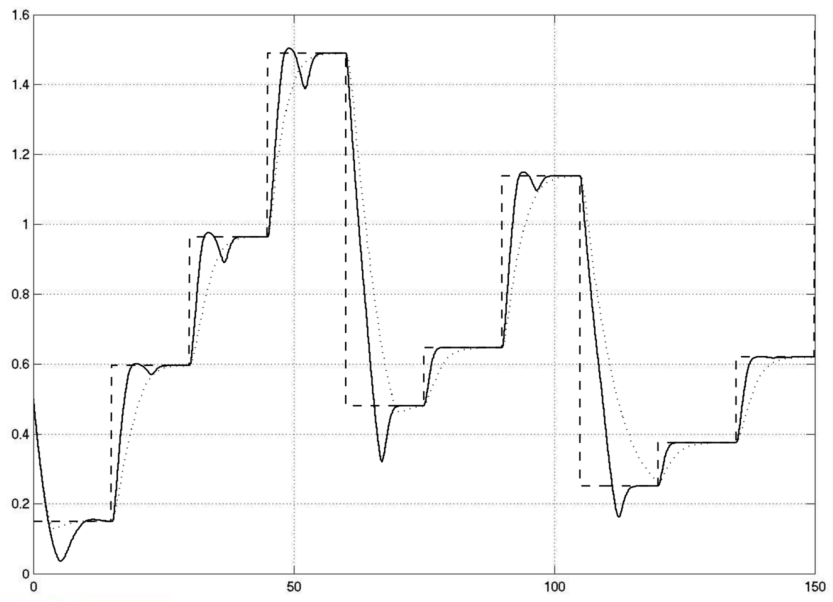


Fig. 4. Comparison of the static (*dotted*) and adaptive (*solid*) controller responses with respect to the setpoint value (*dashed*).

the water tank can change with a rate roughly between $-0.3\,m/s$ and $0.9\,m/s$ due to the design of the tank. This means a scaling of the DD input could be to multiply the value of DD with a factor 3 if the rate is negative and leave it as is when it is positive. Although the level can vary between 0 and 2 (so D can vary between -2 and 2) the input D will not be scaled because errors larger than 1 rarely occur and even if they do, they are dealt with just as well with the current scaling.

A second enhancement solves the problem that the adaptation algorithm sometimes gets trapped during initialisation. This is due to a limit approach of the equilibrium between in- and outflow such that $DD > 0$ so neither of both adaptation rules will be triggered, even not when a setpoint change occurs. Changing the first adaptation rule from '**If** $D < 0$ **and** $DD \leq 0$ **then** ...' to '**If** $D < 0$ **and** $DD \leq 10^{-2}$ **then** ...' solves this problem.

The performance of the adaptive controller with these enhancements is much better, as can be seen in Fig. 5. The only problem with the scaling is that it makes the controller much more problem specific. A better way would be to scale inputs dynamically, or to have an enormous input range and then dynamically set the positions of membership functions within a meaningful range.

A test with seven instead of five membership functions did not show a significantly better response.

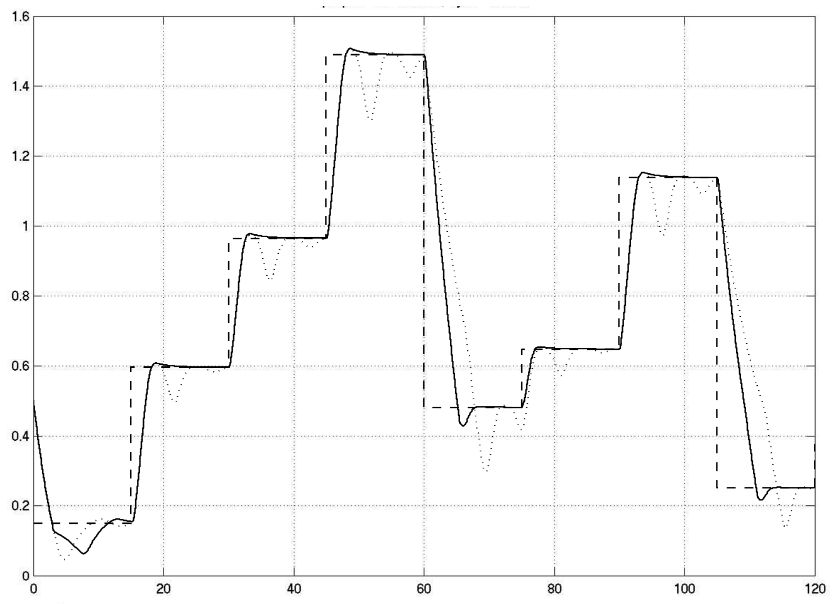


Fig. 5. Scaling the water level change rate (*solid*) removes the fluctuations in the output of the original adaptive controller (*dotted*). The *dashed* line represents the setpoint value.

2.5 Performance Comparison with Changing Flow Rates

To test the adaptive controller for robustness, simulations have been carried out with a changing maximum flow rate through the refilling pipes. The rates now increase or decrease with values which change over time and can run up to 25% of the original maximal flows. In the first test this value is chosen randomly every few seconds, in the second test the change is sinusoidal to resemble the slow and continuous changes in reactor physics.

Both the static and the adaptive controller still do their work properly when subjected to the changing flow rates. The adaptive controller still has a much faster response and is still able to maintain a relatively stable water level at the setpoint value.

To measure the influence this distortion has on the controller, the difference between the water level in the original system has been compared to the level in the system with changing flow rates. For a long run (1000 seconds) the mean difference in water level and the mean percentage difference are calculated. The percentage difference is interesting because it compensates for the fact that for high water levels the valves will be opened wider, so a sudden change in flow will have a larger effect.

Table 3 shows that the adaptive controller needs some time to adapt itself to every flow rate change: when the flow rate changes every 5 seconds the

Table 3. Robustness of the static and adaptive controller to random flow changes.

	Mean difference Random flow changing every			Mean percentage difference Random flow changing every		
	1s	2s	5s	1s	2s	5s
Static	0.0110	0.0102	0.0069	1.47	1.33	1.21
Adaptive	0.0210	0.0102	0.0054	3.44	1.43	0.81

Table 4. Robustness of the static and adaptive controller to sinusoidal flow changes.

	Mean difference Sine frequency			Mean percentage difference Sine frequency		
	0.1rad/s	0.05rad/s	0.01rad/s	0.1rad/s	0.05rad/s	0.01rad/s
Static	0.0068	0.0048	0.0029	1.16	0.87	0.58
Adaptive	0.0060	0.0028	0.0018	1.34	0.44	0.27

adaptive controller is robuster than the static one, but when the changes come faster after each other the static controller is robuster. To the more gradual flow changes of the sinusoidal distortion the adaptive controller is always robuster, as Table 4 shows.

3 Optimization of Fuzzy Logic Controllers

For a fuzzy logic controller there are many parameters that can be tuned, for instance:

- the rules
- the weights applied to each rule
- the number of membership functions
- the shape of the membership functions
- the positioning of the membership functions in the input and output space

The rules are already tuned by the adaptation algorithm, which appears to work very well. Adapting the weights of the rules could be interesting, but a lot of parameters would be needed. The shape and number of MFs usually have a negligible effect. The positioning of the membership functions within the input space seems an interesting topic because it could solve the scaling problem that we saw in Sect. 2.4 in a much more problem independent way.

3.1 Genetic FLC Optimization

Genetic algorithms (GAs) are a soft computing technique to perform optimisations in a way comparable to evolutionary development [8, 9]. They can handle very complex and coupled systems without the need of derivative information to define a search direction. Also GAs can very well handle search spaces with a lot of local optima, which make them very suitable for many applications [10, 11, 12, 13]. A drawback is that convergence is relatively slow and large numbers of optimisation variables will slow down the process even more. Considering all, the GAs can very well be applied to optimize the performance of an FLC for an arbitrary system by tuning the locations of the MFs.

Genetic Parametrization

To optimize the membership functions, a set of parameters must be found that satisfyingly defines their positions in the input space. Satisfyingly here means that we want as few parameters as possible and the parameters must still have as much 'physical relevance' as possible, to make the crossover operator a meaningful tool to reach convergence. Also the parametrization must prevent ambiguity (switching the positions of MFs) and practically impossible or unwanted variants.

The parametrization chosen uses the 'remain range', the space left over between the MFs that are already set and the input ranges, to determine the positioning of the tops of the MFs (see Fig. 6). A few constraints were added to the parametrization:

- The central MF always has its top at $x = 0$
- The x position of the top of one MF is one of the base points for the neighbouring MFs (so no gaps between MFs can arise)
- The MFs for VL and VS are the same, they are just scaled down for VS (like in the static controller case).

Some optimizations with independent MFs for VL and VS have been performed, the results however were not better and often even worse than optimizations with the same MFs for VL and VS. This could be due to the fact that the parameter vector is twice as long in this case, which makes it more difficult to for the GA to converge.

The GA will create different individuals (possible solutions) using this parametrization and apply crossover, mutation and selection to reach an optimum. Actually an individual is only a vector of eight values: the four top-positions for the D input and the four for the DD input.

Fitness Function

The fitness function is the heart of the genetic algorithm, because it defines how well each individual performs. The optimum obtained by the GA is highly dependent on the fitness function, therefore it has to be designed with care. It is important to make an inventory of all characteristics of good (or bad) controller performance, but also a representative control simulation has to be designed.

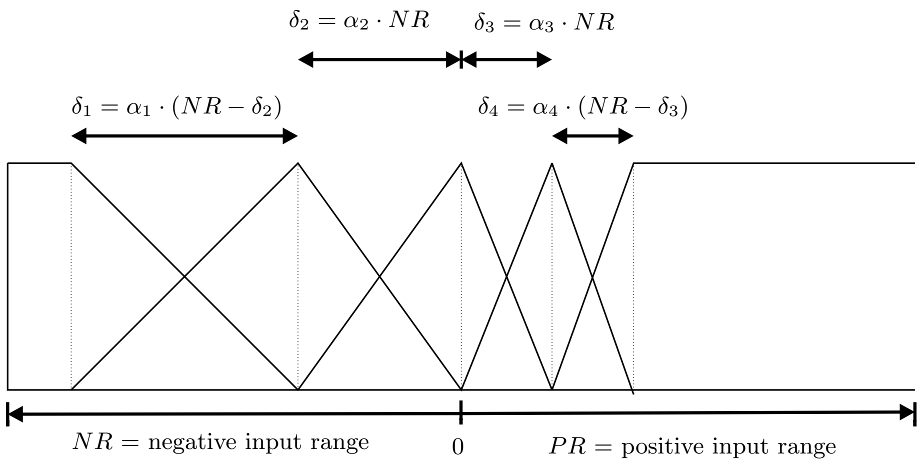

$\delta_2 = \alpha_2 \cdot NR$
$\delta_3 = \alpha_3 \cdot NR$
$\delta_1 = \alpha_1 \cdot (NR - \delta_2)$
$\delta_4 = \alpha_4 \cdot (NR - \delta_3)$
NR = negative input range
0
PR = positive input range

Fig. 6. Derivation of the membership functions from a parameter vector α.

Designing a representative control simulation is not trivial. The problems the controller will come across during actual operation should all be present in the simulation in the right proportions. At the same time the simulation should be as short as possible to save time. For an actual application some more time should be invested in the simulation design (i.e., the setpoint changes over time), also some checks on robustness might be added.

To see whether a certain simulation or fitness function is general enough to cover the whole operation range of the system, it might be a good idea to design multiple simulations and compare the resulting fitnesses for some different controllers. If both simulations are considered 'meaningful', a better fitness according to the one simulation should also have a better fitness according to the other.

At the moment four different fitness functions have been implemented. These differences can be found in the simulation time (30 or 250 seconds) and in the fitness criteria:

- **The simple fitness function** only uses the error area (difference between water level and setpoint integrated over time) as fitness value, where positive errors (i.e., overshoot) are multiplied by a factor larger than 1 because they are considered to be worse because of their possibly dangerous consequences in a nuclear reactor.
- **The extensive fitness function** also considers the mean settling time, mean overshoot, mean positive overshoot and mean negative overshoot, each with a different weight factor.

Optimisation Results

The improvement reached can be seen in Fig. 7 where the original (slow) response of the static controller and the response of the controller with optimized MFs are both plotted. In Fig. 8 the original and optimized membership functions are depicted. For this optimization the extensive fitness function was used with 250s simulation runs.

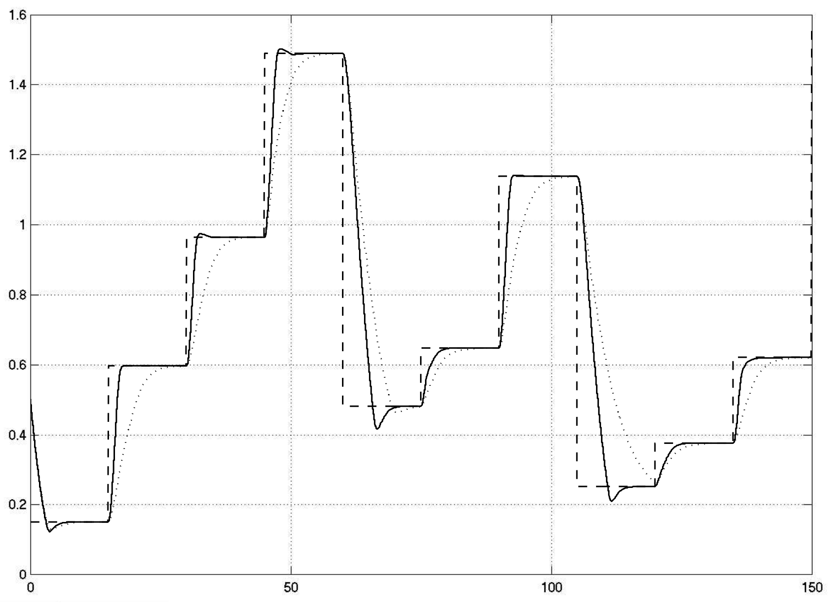
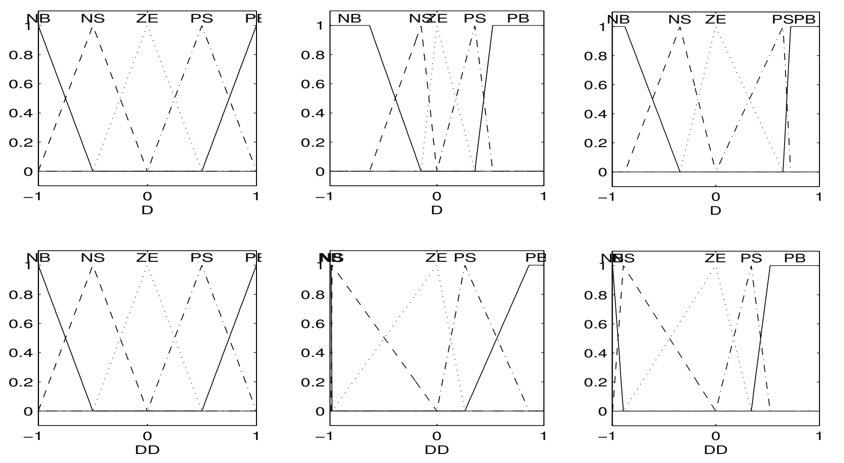


Fig. 7. Response of the original (*dotted*) and the optimized (*solid*) static controller with respect to the setpoint value (*dashed*).

In this response we see the speed of the controller has improved significantly, at the cost of a little overshoot. The overshoot could be suppressed more by applying a higher weight to that in the fitness function. This is of course a decision that should be made according to the wishes of the controller designer and the specifications given.

When we compare the optimised MFs for the level (the upper plot in Fig. 8(b)) to the original ones (Fig. 8(a)) we see that the three central MFs are put closer together. This is not very surprising, because most control actions will be in this range and surely the accuracy in this range will be more important. When looking at the MFs for the rate (the lower ones) the most eye-catching thing is the disappearance of the NB membership function. This can very well be due to the fact that there is not much of a change in the rules between $DD = NS$ and $DD = NB$ (see Table 2.1). Also the fact that rates smaller than -0.3 are physically impossible might play a role here.



(a) Original MFs. (b) Optimised MFs for static controller. (c) Optimised MFs for adaptive controller.

Fig. 8. The original and optimised membership functions.

3.2 Genetic Adaptive FLC Optimization

Now we have seen the potential of membership function for a static FLC we will extend this principle for an adaptive controller. The same parametrization and the extensive fitness function as described in Sect. 3.1 are used. The main difference is that the optimization has now become an iterative process, where the controller first gets some time to adapt (using the adaptation mechanism from Sect. 2.2), after which the membership functions are optimized, followed by a new adaptation run and a new optimization run etcetera.

This process is likely to converge relatively fast, because the rule base and the MFs keep the same mutual relation. Also the demands on convergence can be at a lower priority, because if the GA has not fully converged in the first run, it can carry on in the second run. To make this principle work, the best found solution from the last optimization should be added to the initial population of the new GA run.

Figure 9 shows that the genetic optimization is able to remove the fluctuations in the original adaptive controller without doing much harm to the rest of the performance (there is even no overshoot any more). This means the optimization algorithm provides a very general way to improve controller performance, because engineering knowledge like applied for the input scaling in Sect. 2.4 is not needed any more. Using the GA the main disadvantages of the adaptive controller are solved.

The improved MFs (Fig. 8(c)) have the same characteristics as the optimised static controller MFs (Fig. 8(b)).

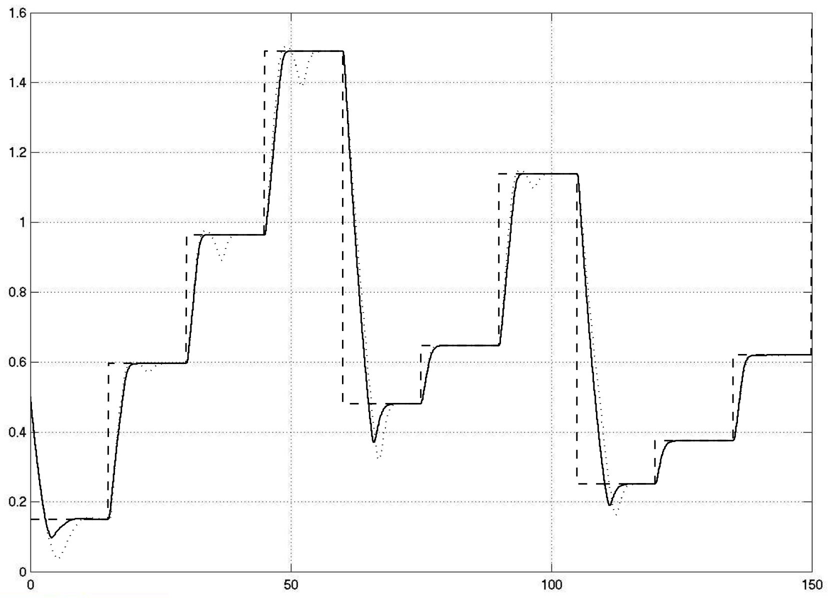


Fig. 9. Response of the original (*dotted*) and the optimized (*solid*) adaptive controller with respect to the setpoint value (*dashed*).

4 Application of Neural Networks

A problem with the setup as used in Sects. 2 and 3 is that in the nuclear reactor case safety regulations make it impossible to let a controller adapt itself on-line (i.e., when it is actually controlling the reactor). Therefore it is important to obtain a good model of the system so the controller can be adapted off-line which is fully safe. The adapted controller can now be checked for stability by engineers before it is actually implemented on-line.

It is difficult to obtain a good and fast model of a system (plant), especially when its dynamics are not fully known. This is often the case, because even if a mathematical formulation exists, most of the parameters in that formulation are uncertain or even change during operation. A neural network could be trained to mimic a plant's behaviour [14, 15, 16, 17] without any mathematical knowledge of the process and thus is a very general solution.

4.1 Tested Neural Networks

Several standard network types have been investigated:

1. Back Propagation (BP) networks
   - Feed-Forward back propagation (FF)
   - Cascade-Forward back propagation (CF)
   - Elman back propagation (ELM)
2. Radial Basis networks
   - Standard Radial Basis (RB)
   - Generalized Regression (GR)
3. Neuro-Fuzzy networks
   - Adaptive Neuro-Fuzzy Inference Systems (ANFIS)

All tests were performed using the standard networks provided by Matlab [18]. For the back propagation networks the values of weights and biasses are set using an iterative training scheme such as Levenberg-Marquardt. The radial basis networks use the distances of an input to all training sets to determine the influence of each neuron. An ANFIS is a true hybrid system where no clear distinction can be made any more between the fuzzy logic controller and the neural network.

The NNs were trained to predict the new D_{t+1} and rate DD_{t+1} based on the control inputs VL_t and VS_t and the last level D_t and rate DD_t. Different settings (such as number of neurons, number of hidden layers and spread factors for RB and GR NNs) and different data set sizes (2500, 5000 and 10000 data sets) have been used. Also an implementation with estimation of the level (D_{t+1}) only has been tested.

4.2 Results of the Back Propagation Networks

If we check the performance of the networks with a test-data set, the CF networks come out best, the FF networks come second and the ELM networks worst. For CF and FF networks it seems that large networks (more layers, more neurons) are better off with large data sets and small networks are better off with small sets. For ELM networks things only get worse when more neurons are added.

If we check the performance using the estimation error in an actual control simulation, things turn out very different. The ELM networks still perform miserable, but now the performance of the CF and FF networks are much more mixed and small data sets seem to be the way to go, preferably in combination with a small network. However, the best performance (by a CF network with 10 neurons, 2500 data sets, see Fig. 10) is still too poor to be really useful.

When only estimating the level things do not get much better. CF still seems the best choice, but now middle-class networks with small training sets perform best. Based on the simulation this is almost the same case. Training a network with level estimation costs about 2/3 of the time needed for training a network estimating both level and rate.

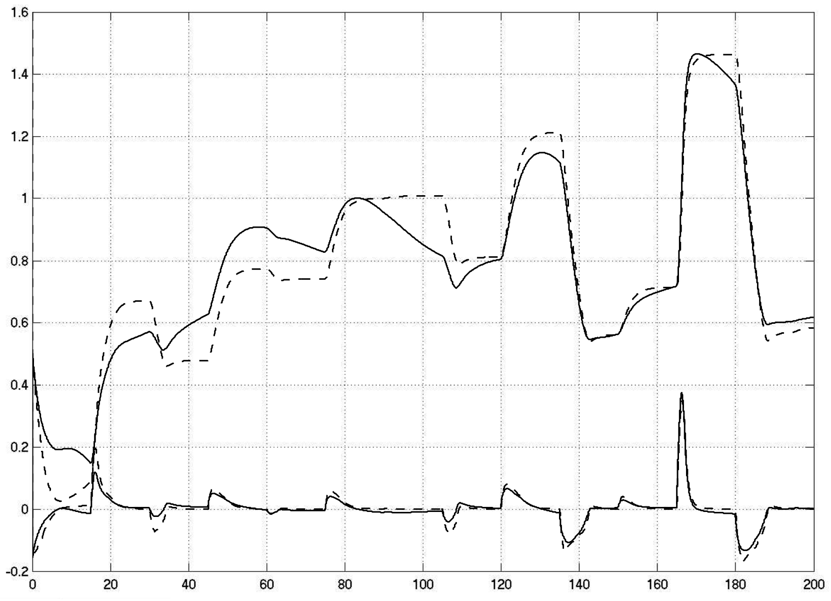


Fig. 10. Simulation using the best tested BP network providing a level and rate estimation (*solid*). The actual plant response (reference) is *dashed.*

4.3 Results of the Radial Basis Networks

It is very difficult to get normal RB networks to mimic the plant accurately. When using too many data sets the performance gets worse, probably due to conflicting data (almost the same inputs, but different outputs). It is therefore important to make a well-balanced dataset and choose the appropriate spread factor (first hidden layer bias value). When adaptive RB NNs are used (adding neurons one by one) results do not get any better and training takes a lot longer. Also training the level only comes with some problems: when the flow has stabilized, the NN does not see that the rate is 0 and estimates a slightly positive or negative value, thus changing the level when it should be steady.

The GR networks (Fig. 11, solid lines) seem to give better results than the BP-NNs. The shape is right and the lines are straight. However, at some points (5-15 s, 105-110 s, 135-165 s, 185-200 s) things are going wrong, as can be clearly seen in the rate estimate (the lower line). The problem between 5-15 s could be because the training data contain to few sets with these very low levels. However, if we would eliminate this first error we get the dotted line in Fig. 11, and we see that the estimate would still not be accurate enough to design a controller upon.

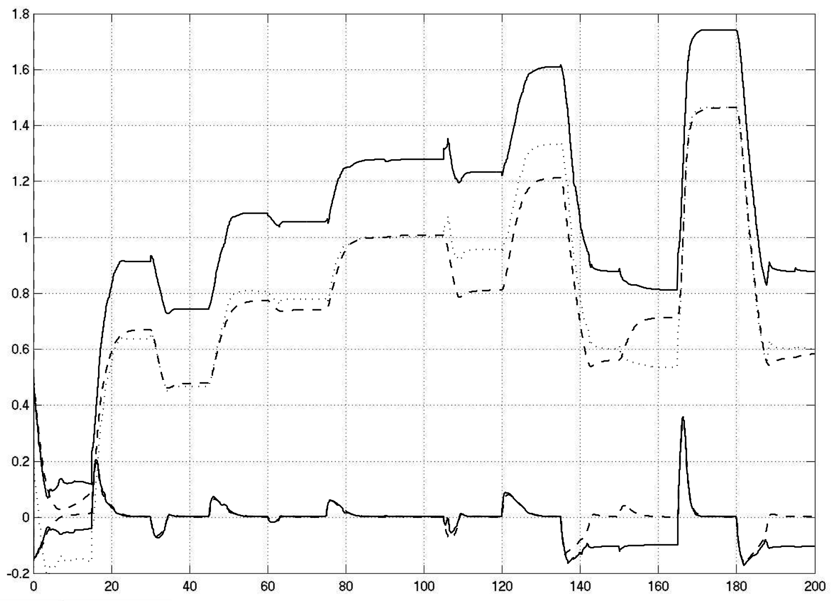


Fig. 11. Simulation using the best tested GR network providing a level and rate estimation (*solid*). The actual plant response (reference) is *dashed*, the estimation with compensation for the initial error is *dotted*.

4.4 Using the ANFIS as a Controller

Instead of mimicking a plant, NNs could also be used as a controller. Especially the ANFIS should be suited for such a task when trained with data that reflect 'good control'. This means an ANFIS might be used to convert the expert knowledge of a manual operator into a static controller. Another option would be to try inverse model control. This theory is based on the idea that if we know the transfer function of our plant, we can control it in an exact manner if we make a controller with the exact inverse transfer function. In classical control this is impossible because these inverse transfer functions do not exist in practice, but it should be possible by training an ANFIS.

Training an ANFIS with *inverse* plant data however, is almost the same as training an ANFIS to mimic the plant. Because none of the ANFIS plants trained made any sense in a control simulation, it is quite logic that ANFIS controllers did not work either.

5 Conclusion

When dealing with soft computing techniques it is difficult to make hard statements on performance, due to a lack of in-depth investigations at the current stage. Actually the vagueness which is the strength of these techniques is also their weakness when it comes to drawing conclusions. However some very meaningful conclusions can be drawn from the research done.

As shown by almost any test in this chapter fuzzy logic provides a powerful way to control a strongly non-linear system, even when the system is not fully defined or has changing properties. Although the strength of the applied soft computing methods would be that no detailed (mathematical) description of the plant would be needed, even when such a description is available FLC still has many benefits, such as its robustness and transparency.

A static controller (Sect. 2.1) is the simplest and safest solution in an industrial environment, however designing the rule base by hand costs a lot of time and engineering skills. Moreover, such controllers are very problem specific and are likely not to provide optimal control in real life applications.

Adaptive control can solve these problems because they need very limited knowledge of the system. Actually for the problem investigated only two simple 'common sense' rules are needed to fully control the strongly non-linear system, as shown in Sect. 2.2.

Tuning of the FLC is important and can be done by hand (Sect. 2.4) when some insight in the controller and the system to control is available. An automated tuning method using Genetic Algorithm optimization techniques was proposed and successfully tested in Sect. 3 for both the static and the adaptive controller. In both cases it appears to be very profitable to let the algorithm tune the controllers according to predefined desires such as minimum overshoot and short settling time. Using the proposed optimization, overshoot, fluctuations could be removed from the adaptive controller response and input scaling is no longed needed.

Due to safety regulations and practical problems controllers cannot be adapted and optimised on-line. Therefore an accurate model of the plant (i.e., the controlled system) is needed. To maintain the problem independence and superfluity of a mathematical description of the plant, Neural Networks were proposed in Sect. 4. From all tested networks, the Generalized Regression (GR) networks seem to be most suitable.

Although the simulation results of the GR NNs look quite reasonable, the exact values are by far not accurate enough to be applied in a real system. This may be due to the fact that neural networks generally provide an approximation instead of an exact answer combined with the fact that errors in the simulation accumulate. More balanced training data or other or adjusted types of networks might help to find a workable solution but it might just as well be impossible by principle to make an accurate plant simulation model using NNs. A first step in further research would be to find successful implementations in literature.

A promising solution has been found to make high performance FLCs for application in for instance nuclear reactor control. A subject for further research would be how to make the proposed methods suitable for practical application in an industrial environment.

References

1. J.A. Bernard. Use of a rule-based system for process control. *IEEE Control Systems Magazine*, 8(5):3–13, 1988.
2. D. Ruan. Initial experiments on fuzzy control for nuclear reactor operations at the belgian reactor 1. *Nuclear Technology*, 143:227–240, Aug 2003.
3. D. Ruan and A.J. van der Wal. Controlling the power output of a nuclear reactor with fuzzy logic. *Information Sciences*, 110:151–177, 1998.
4. D. Ruan. Implementation of adaptive fuzzy control for a real-time control demomodel. *Real-Time Systems*, 21:219–239, 2001.
5. D. Ruan, editor. *Fuzzy Systems and Soft Computing in Nuclear Engineering.* Studies in Fuzzyness and Soft Computing. Physica-Verlag, 2000. ISBN 3-7908-1251-X.
6. S. Heger, N.K. Alang-Rashid, and M. Jamshidi. Application of fuzzy logic in nuclear reactor control, part i: An assessment of state-of-the-art. *Nuclear Safety*, 36(1):109, 1996.
7. D. Ruan and P.F. Fantoni, editors. *Power Plant Surveillance and Diagnostics.* Studies in Fuzzyness and Soft Computing. Springer, 2002. ISBN 3-540-43247-7.
8. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, 1997. ISBN 3-540-60676-9.
9. J. Heitkötter and D. Beasley. The hitch-hiker's guide to evolutionary computation, 2000. URL `http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/`. FAQ for the comp.ai.genetic newsgroup.
10. R.L. Haupt and S.E. Haupt. *Practical Genetic Algorithms.* Wiley-Interscience, 1997. ISBN 0-471-18873-5.
11. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989. ISBN 0-471-18873-5.
12. D.C. Marinescu, H.J.Siegel, A.S. Wu, and H. Yu. A genetic approach to planning in heterogeneous computing environments. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, page 97. IEEE, 2003. URL `http://doi.ieeecomputersociety.org/10.1109/IPDPS.2003.1213199`.
13. J.O. Entzinger, R. Spallino, and W. Ruijter. Multilevel distributed structure optimization (paper #374). In I. Grant, editor, *CD Proceedings of the 24th International Congress of the Aeronautical Sciences (ICAS), Yokohama, Japan.* ICAS, Optimage Ltd., Edinburgh, UK, 2004. ISBN 0-9533991-6-8.
14. A. Andršik, A. Mszros, and S.F. de Azevedo. On-line tuning of a neural pid controller based on plant hybrid modeling. *Computers and Chemical Engineering*, 28:1499–1509, 2003.

15. J.F. Briceno, H. El-Mounayri, and S. Mukhopadhyay. Selecting an artificial neural network for efficient modeling and accurate simulation of the milling process. *International Journal of Machine Tools & Manufacture*, 42:663–674, 2002.
16. J.A. Roubos, S. Mollov, R. Babuška, and H.B. Verbruggen. Fuzzy model-based predictive control using takagi-sugeno models. *International Journal of Approximate Reasoning*, 22:3–30, 1999.
17. P. Gil, J. Henriques, A. Dourado, and H. Duarte-Ramos. Fuzzy model-based predictive control using takagi-sugeno models. In *Proceedings of ESIT'99 European Symposium on Intelligent Techniques, Crete, Greece*. ERUDIT, Jun 1999.
18. The MathWorks Inc. Matlab 6.5, the language of technical computing. Software with online help, 1984-2002. URL `http://www.mathworks.com`.