Design of robust fuzzy-logic control systems by multi-objective evolutionary methods with hardware in the loop

P. Stewart[a,*], D.A. Stone[a], P.J. Fleming[b]

[a] Electrical Machines and Drives Group, Department of Electronic and Electrical Engineering, University of Sheffield, Mappin St., Sheffield S1 3JD, UK
[b] Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin St., Sheffield, UK

Abstract

Evolutionary development of a fuzzy-logic controller is described and is evaluated in the context of hardware in the loop. It had been found previously that a robust speed controller could be designed for a DC motor motion control platform via off-line fuzzy logic controller design. However to achieve the desired performance, the controller required manual tuning on-line. This paper investigates the automatic design of a fuzzy logic controller directly onto hardware. An optimiser which modifies the fuzzy membership functions, rule base and defuzzification algorithms is considered. A multi-objective evolutionary algorithm is applied to the task of controller development, while an objective function ranks the system response to find the Pareto-optimal set of controllers. Disturbances are introduced during each evaluation at run-time in order to produce robust performance. The performance of the controller is compared experimentally with the fuzzy logic controller which has been designed off-line, and a standard PID controller which has been tuned online. The on-line optimised fuzzy controller is shown to be robust, possessing excellent steady-state and dynamic characteristics, demonstrating the performance possibilities of this type of approach to controller design.

Keywords: Fuzzy systems; Evolutionary/Genetic algorithms; Methodologies; Models and algorithms; Hardware-in-the-loop

1. Introduction

This paper investigates the potential of multi-objective control design with hardware in the loop. Tuning of PI parameters on-line has been achieved (Schroder et al., 2001) with multi-objective genetic algorithms applied to a sealed pump running on magnetic bearings. However, parameter and controller structure tuning on-line presents a further level of potential for control system design. A DC motor dynamometer rig and a micro-controller are used as a platform to develop and assess the control algorithms. In particular, an on-line tuned type (PI) and an off-line designed type (fuzzy logic) are considered for performance comparison. An automatic method for fuzzy logic control design is presented, utilising a multiobjective evolutionary algorithm for the optimisation process. Process uncertainty in the form of parameter variations has been investigated (Hughes, 2001) in which a model which includes parameter uncertainty and measurement noise is utilised with the aim of producing a controller which is also robust to disturbances. Automatic controller design considered here allows the evolutionary design to proceed in the presence of real-life measurement noise and parameter variation via the hardware in the loop. To further develop the theme of robust design, external disturbances are injected during each on-line chromosome assessment with the aim of increasing the controller robustness. Moreover, a complete plant cycle is performed during the evaluation phase for each chromosome.

Fuzzy logic control, comprising a fuzzification interface, rule base and defuzzification algorithm (Mamdani, 1974; Zadeh, 1973), has been applied to a wide variety of motion control applications (Betin et al., 2001; Guillemin, 1994). A vital region of interest concerns the implementation of the fuzzy controller. Several different approaches have been postulated to extract the

*Corresponding author. Tel.: +44-114-222-5841; fax: +44-114-222-5196.

E-mail address: p.stewart@shef.ac.uk (P. Stewart).

knowledge base from experts or training examples to construct the input–output membership functions and the fuzzy rulebase. These methods can be based on neural networks (Hong and Lee, 1996; Ishibuchi and Tanaka, 1993) or the application of fuzzy clustering techniques to construct a fuzzy controller from training data sets (Grauel and Mackenberg, 1997). It has been observed that the major drawback of most fuzzy controllers and expert systems is the need to predefine membership functions and fuzzy rules. In Hong and Lee (1996) is proposed based on fuzzy clustering techniques and decision tables to derive membership functions and fuzzy rules from numerical data. A natural evolution of the technique was to integrate Genetic Algorithms (GAs) into the Fuzzy logic design process (Chang and Wu, 1995; Homaifar and McCormick, 1995; Thrift, 1996). The robustness of the GA allows it to cover a multidimensional search space while ensuring an optimal or near-optimal solution, thus simultaneous design of membership functions and fuzzy control rules can be achieved (Wu and Liu, 2000). The development of these techniques to design optimal robust fuzzy logic controllers for example gas turbine engines (Jamshidi et al., 2003) and aerospace autopilots (Blumel et al., 2001) has arisen to satisfy the need which exists when expert heuristic knowledge does not exist to translate into controller design.

The performance of a particular control design is fundamentally tied to the accuracy of the model upon which it is based. This is especially true for iterative control design and optimisation procedures. The substitution of hardware in the loop for the software model opens up new possibilities for design based on real world performance indicies. In this paper the implementation of GA fuzzy design will be evaluated via an on-line experimental DC motor connected to a DC shunt load motor set to introduce dynamic disturbances. The performance of the resulting motion controller is compared with that of a manually tuned fuzzy controller. The results presented here demonstrate a convenient and practical method to produce a robust controller design on a prototype plant.

1.1. Multi-objective optimisation by evolutionary algorithm

Evolutionary algorithms are global parallel search and optimisation methods based around Darwinian principles, working on a population of potential solutions to a problem (in this case the on-line design of an optimal fuzzy logic controller via hardware in the loop). Every individual in the population represents a particular solution to the problem, often expressed in binary code. The population is evolved over a series of generations to produce better solutions to the problem.

The general multiple objective optimisation problem is described as (Jamshidi et al., 2003):

$$\min\{f_1(\mathbf{x}) = z_1, \ldots, f_j(\mathbf{x}) = z_j\}, \tag{1}$$

where

$$\mathbf{x} \in D. \tag{2}$$

The solution of $\mathbf{x} = [x_1, \ldots, x_i]$ is a vector of *decision variables*, and D is the set of feasible solutions. If each decision variable takes discrete values from a finite set, then the problem is combinatorial.

The image of solution $\mathbf{x}$ in the objective space is a *point*

$$\mathbf{z}^x = \lfloor z_1^x, \ldots, z_j^x \rfloor = \mathbf{f}(\mathbf{x}), \tag{3}$$

such that

$$z_j^x = f_j(\mathbf{x}), \quad j = 1, \ldots, J. \tag{4}$$

Point $\mathbf{z}^1$ *dominates* $\mathbf{z}^2$, $\mathbf{z}^1 \succ \mathbf{z}^2$, if $\forall j\ z_j^1 \leqslant z_j^2$ and $z_j^1 < z_j^2$ for at least one j. Solution $\mathbf{x}^1$ dominates $\mathbf{x}^2$ if the image of $\mathbf{x}^1$ dominates the image of $\mathbf{x}^2$.

A solution $\mathbf{x} \in D$ is *efficient* (*Pareto-optimal*) if there is no $\mathbf{x}' \in D$ that dominates $\mathbf{x}$. The point which is an image of an efficient solution is *nondominated.* The set of all efficient solutions is called the efficient set. The image of the efficient set in the objective space is called the *nondominated set* or *Pareto front.*

An *approximation to the nondominated set* is a set A of points (and corresponding solutions) such that $\neg\exists \mathbf{z}^1$, $\mathbf{z}^2 \in A$ such that $\mathbf{z}^1 \succ \mathbf{z}^2$, that is set A is composed of mutually nondominated points.

The point $\mathbf{z}^*$ composed of the best attainable objective function values is called the *ideal point.*

At every generational step, each individual of the population is run on the hardware, and its performance evaluated and ranked via a cost function. Individual performance is indicated by a fitness value, an expression of the solution's suitability in the solution of the problem. The relative degree of the fitness value determines the level of propagation of the individual's genes to the next generation. In the multi-objective evolutionary algorithm (MOGA) in use here, the rank of a certain individual corresponds to the number of individuals in the current population by which it is dominated. All nondominated individuals are assigned rank 1, while dominated ones are penalized according to the population density of the corresponding region of the trade-off surface. Fitness assignment is performed as follows (Blumel et al., 2001).

- Sort population according to rank.
- Assign fitness to individuals by interpolating from the best (rank 1) to the worst (rank $n \leqslant M$), the *Pareto ranking assignment process* (Jaszkiewicz et al., 2001), according to a (usually) linear function.
- Average the fitness of individuals with the same rank, so that all of them will be sampled at the same rate.

This keeps the global population fitness constant while maintaining appropriate selective pressure.

Evolution is subsequently performed by a set of genetic operators which stochastically manipulate the genetic code. Most genetic algorithms include operators which select individuals for mating, and produce a new generation of individuals. *Crossover* and *Mutation* are two well-used operators. The crossover operator exchanges genetic material between parental chromosomes to produce offspring with new genetic code. The mutation operator makes small random changes to a chromosome.

Trade-offs occur between competing objectives with the consequence that it is very rare to find a single solution to a particular problem. In reality a family of *non-dominated* solutions will exist. These *Pareto-optimal* (Fonseca and Fleming, 1995, 1998) solutions are those for which no other solution can be found which improves on a particular objective without a detrimental effect on one or more competing objectives. The designer then has the opportunity to select an appropriate compromise solution from the trade-off family based on a subjective engineering knowledge of the required performance. Individuals which represent candidate solutions to the optimisation problem (in this case fuzzy controller parameters such as membership functions, rule bases, etc.) are encoded as either binary or real number strings, producing an initial population of chromosomes by randomly generating these strings.

The population of individuals is evaluated using an objective function which characterises the individual's performance in the problem domain. The experimental system is run iteratively with each individual's set of controller parameters. The objective function determines how well each individual performs based on experimental data (in this case the current and velocity tracking performance and power consumption), and is used as the basis for selection via the assignment of a fitness value. The motivation in this case for combining GAs with fuzzy logic for control is to investigate a number of factors. Firstly, the design potential which can be gained by removing the need for knowledge solicitation to enable the fuzzy logic design. Secondly, to reduce the design time. Thirdly to examine a method for introducing robustness features into the fuzzy design. Finally, to investigate and define a method for multiobjective controller design where an accurate system model is either unavailable, or runs extremely slowly, a limiting factor in the process of iterative evolutionary design.

1.2. Hardware overview

The application consists of a brushed DC permanent magnet field motor fed by a four quadrant DC chopper drive operating at 5 kHz. Fig. 1 shows a schematic of the

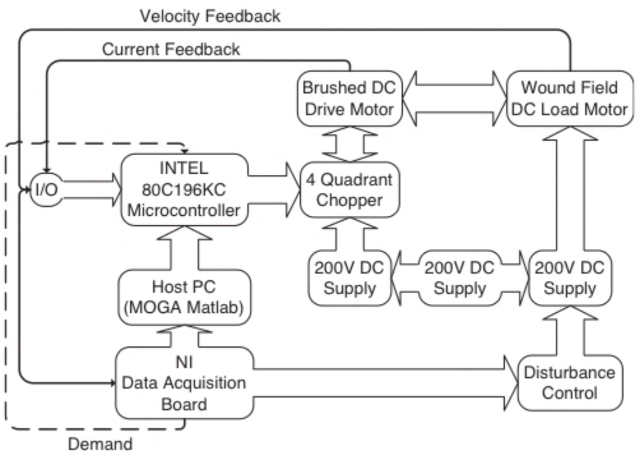


Fig. 1. On-line optimisation hardware setup.

on-line control system and hardware set-up. The objective is to perform robust closed loop speed control on this motor. The drive motor is connected via a flexible coupling to a field wound DC load motor which itself is fed directly by a 200 V DC supply. The disturbance torque from this load motor is independently controllable, based on the applied armature voltage. Current control is embedded in the INTEL 80C196KC microcontroller as is the fuzzy logic velocity controller. The microcontroller also hosts the velocity and current feedback signals from the motor set and chopper drive, respectively. The multi-objective optimisation programme runs under *Matlab* (Chipperfield et al., 1994), and resides on a PC. Candidate controllers are downloaded from this host to the microcontroller via the serial link and on-line debug facility allowing direct access to programme memory. Assessment of the candidate controllers is performed on the PC according to a pre-programmed performance cost function. A National Instruments data acquisition board performs signal acquisition to bring feedback signals into the PC, to facilitate performance evaluation via the objective function.

2. On-line PID control design

Various methods exist to tune the gains of a PID controller off-line to attain the prescribed transient response and steady-state error criteria. These methods generally involve some form of iterative approach to achieve performance criteria such as *rise—time, overshoot* and *settling—time* (Kuo and Hanselman, 1994). In keeping with the development of the fuzzy controller designed later in this paper, the PID control scheme was designed and tuned on-line. This on-line method has been shown to be extremely effective in a variety of applications including active magnetic bearings

(Schroder et al., 2001). In the application under consideration here, the PID controllers to be optimised comprise two cascaded (Kuo and Hanselman, 1994) control loops (Fig. 2). An outer loop performs tracking of a velocity demand, supplying a current demand based upon velocity error to the inner control loop which tracks the current demand based on current error. The output of the PID current controller is applied as a voltage to the DC drive motor via the PWM channel of the microcontroller and four quadrant DC chopper drive. The dynamic performance of the system is limited by current and voltage restraints specified by the motor manufacturer, namely

- Supply to chopper drive: 150 V DC
- Current limit: 1.5 A

The parameters to be tuned here are the proportional K_p, integral K_i and derivative gains K_d for both the current and velocity control loops, to achieve tracking performance, and also load disturbance rejection for the velocity loop.

$$C = E_t\left[K_p\left(1 + \frac{1}{K_i s} + K_d s\right)\right], \tag{5}$$

where E_t is the tracking error and C is the commanded control action. The optimisation engine chosen for this application is the multi objective genetic algorithm (MOGA) which runs on the PC platform (Chipperfield et al., 1994), and is interfaced to Simulink. Genetic algorithms can tolerate experimental noise, and are as such ideal for this application, since a population of potential solutions evolve by means of a selection and transformation process which is stochastic by nature. The on-line control optimisation was based on the structure of a standard manually tuned controller. A software interface allows the controller parameters to be altered on the microcontroller in real time from within the MOGA environment and also allows measured data to be read back into MATLAB for performance assessment. The optimisation procedure is constructed with the PID parameters for the two control loops as *decision variables* and direct measurements of the controllers performance to form the basis for assessment

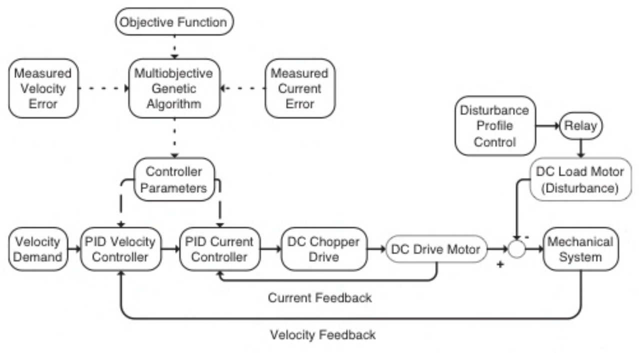


Fig. 2. On-line PID current and velocity controller optimisation setup.

as optimisation objectives. The current and velocity controllers were tuned concurrently, and performance was assessed in response to a step velocity demand of 200 rad s^{-1}. The objective function comprised three elements, and was of the form

- minimise $\int |e_i|\,dt$,
- minimise $\int |e_v|\,dt$,
- minimise $\int vi\,dt$,

where e_i is the current tracking error, e_v is the velocity tracking error, and $\int vi\,dt$ is the power consumption of the system. In this way, it is required that a search be performed for a pair of controllers which deliver the most accurate current and velocity tracking performance, while utilising the minimum possible energy. An extra feature which was built into the optimisation procedure was the injection of a disturbance via the load motor of approximately 0.3 N m which is approximately 30% of torque at the rated current of 1.5 A. In this way, it was expected that a controller set would be designed robust to external load disturbances. The optimisation algorithm was set according to the following parameters:

- Number of individuals per generation: 40
- Number of random immigrants per generation: 6
- Number of generations: 100
- Number of decision variables: 6 (PID parameters for two controllers)
- Number of objectives: 3 (current tracking, velocity tracking and power consumption)
- Number of immigrants per generation: 6 (random individuals to ensure complete search)
- Decision variable range: 0–2000
- All objectives set to minimise at zero.

2.1. *Results of on-line PID design*

The selection of the PID controller was extremely easy in this case. Minimisation of current tracking error also results in the minimisation of velocity tracking error. The power integral minimisation objective is to a greater extent a trade-off with the tracking objectives. Relaxation of the power criteria results in improved tracking performance up to a point where no improvement is achieved. From this point onwards, no improvement is made in tracking, even with the expenditure of larger amounts of energy. Evidently, the larger gains associated with controllers beyond this point waste energy in a more aggressive control action without any improvement in performance. Consequently, the controller gains which are associated with this boundary are chosen as the optimal set.

- Current controller gains: $P = 293.5$, $I = 50$, $D = 0$.
- Velocity controller gains: $P = 43.8$, $I = 13.8$, $D = 0$.

The solution was converged on the 15th generation. This response (Fig. 3) defines the baseline performance by which subsequent controllers will be compared, giving a rise-time from 0 to 200 rad s^{-1} in approximately 3.5 s in the case without external disturbances. The corresponding response (Fig. 4) defines the performance by which subsequent controllers will be evaluated under conditions of external disturbance. The switching pattern of the relay controlling the voltage to the load motor is shown in graph (c). The values shown have been scaled to the value of applied disturbance torque. This compares with a maximum torque of 1 N m from the drive motor at a rated current of 1.5 A. Under these conditions, a rise-time from 0 to 200 rad s^{-1} in approximately 4.5 s can be expected.

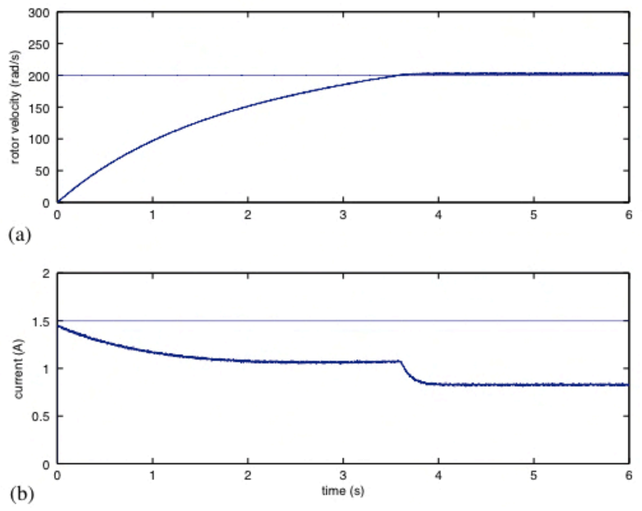


Fig. 3. Motor step demand tracking response without external torque disturbance: (a) velocity response, (b) current response.

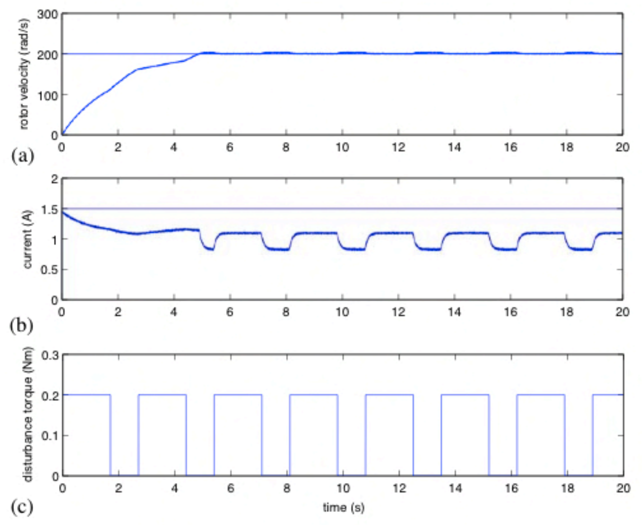


Fig. 4. Motor step demand tracking response with external torque disturbance: (a) velocity response, (b) current response, (c) estimated disturbance torque.

3. Off-line fuzzy-logic controller design

A fuzzy-logic velocity control scheme had been developed for this system previously in order to investigate the implementation issues involved with this type of control structure. Although claims are made concerning the reduction of development time (Driankov et al., 1993), in fact the development time to produce the fuzzy controller off-line was significantly greater than the time required to manually produce and tune a robust PID tracking controller. This can to a certain extent be explained by unfamiliarity with the technique of fuzzy design, a factor which is exacerbated by the complexity of the design procedure. The designer must choose input and output membership functions, a meaningful rule base, and an effective defuzzification strategy. In essence this requires the implementation of a controller with many degrees of freedom in the design, and consequently a complex implementation to achieve robust design.

An iterative design approach was utilised, to investigate the effects of the various degrees of design freedom in order to design the best controller. The most effective control structure was found to be input membership functions for error ($v(k)$) and change of error ($\Delta v(k)$) at time k, where

$$\Delta v(k) = v(k) - v(k-1). \qquad (6)$$

The form of the membership function is shown in Fig. 5, the input functions are linked to the controller output by a rule base of the form;

- IF error is Positive Big THEN output is Positive Big,
- IF error is Positive Small THEN output is Positive Small,
- IF error is Zero THEN output is Zero,
- IF error is Negative Small THEN output is Negative Small, and

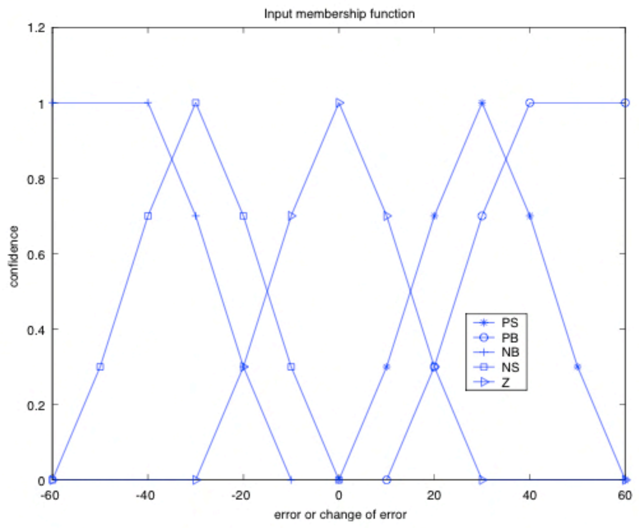


Fig. 5. Input membership functions for v and Δv.

- IF error is Negative Big THEN output is Negative Big.

This rule base is repeated for change of error, and was implemented experimentally, the structure being shown in Fig. 6. The error and change of error controllers were constructed as follows. The fuzzy inference rule base is implemented using the intersection operator. A matrix of input and output sets included in each rule is constructed. Assuming for example, two classical sets A and B in a universe U, with membership functions μ_A and μ_B, the minimum operator *intersection* can be defined as (Driankov et al., 1993)

$$\mu_{A\cap B}(x) = \min(\mu_A(x),\ \mu_B(x)). \tag{7}$$

The overall transfer surface for the controller was achieved by combining the matrix representation of all the individual rules into one overall matrix and applying the maximum operator *union*. This operation exemplifies the Cartesian cross product operator defined on n classical sets $A_1,\ldots,A_n$ as

$$\begin{aligned} X_{i=1}^{n} &= A_1 \times \cdots \times A_n \\ &= ((x_1,\ldots,x_n) | x_1 \in A_1,\ldots,x_n \in A_n). \end{aligned} \tag{8}$$

The resulting transfer characteristic for velocity error is shown in Fig. 7. A corresponding surface consequently exists for change of velocity error.

The utilisation of the centre of area defuzzification strategy (Driankov et al., 1993) results in a controller structure shown in (Fig. 8). The surface provides a nonlinear relationship between velocity error, change of velocity error, and the controller output.

3.1. *Results of off-line fuzzy logic controller design*

The performance of the off-line designed fuzzy logic velocity controller is presented in Fig. 9 for the non-disturbance case, and Fig. 10 for the case with external disturbance. In this case, a bi-directional velocity demand is supplied to the controller. In both the disturbed and undisturbed state, velocity tracking is comparable both in terms of rise time and steady-state accuracy to the PID controller. Although it is beyond the central remit of this paper, a substantial amount of time was spent selecting an appropriate defuzzification strategy and the selection of the input–output sets in order to achieve this tracking performance. Consequently, the investigation of an on-line fuzzy logic design becomes an attractive proposition which is described in the next section. The development for an

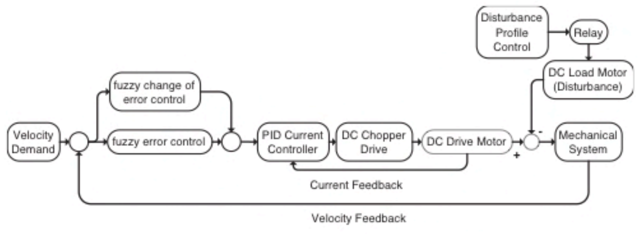


Fig. 6. Fuzzy controller implementation.

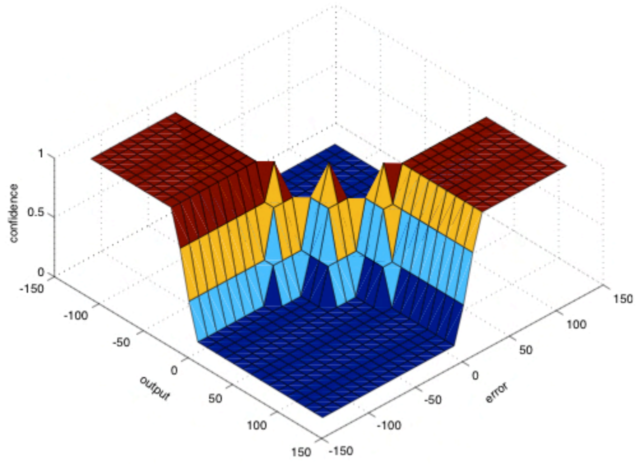


Fig. 7. Fuzzy transfer surface for velocity error.

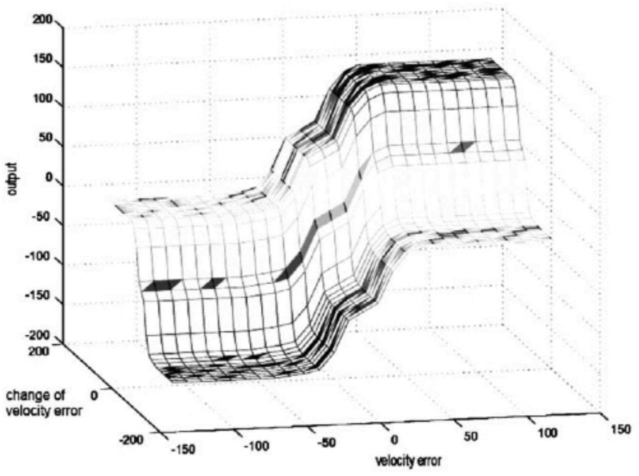


Fig. 8. Fuzzy controller output.

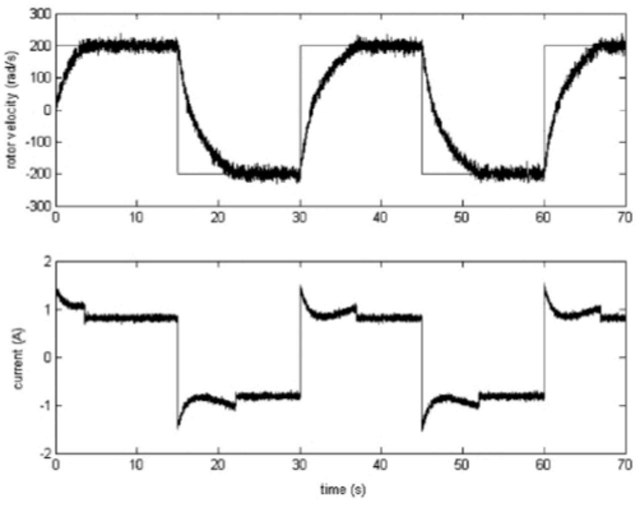


Fig. 9. Off-line designed fuzzy controller performance.

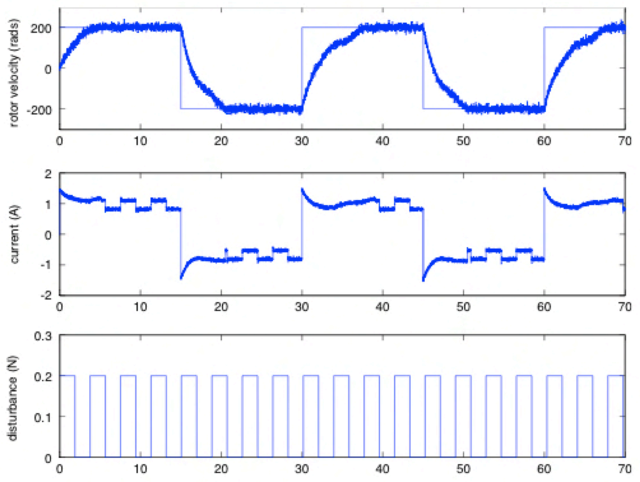


Fig. 10. Off-line designed fuzzy controller performance with external disturbance.

automatic design scheme with hardware in the loop will be considered and experimentally tested.

4. On-line fuzzy logic controller design

Evolutionary algorithms have been used to optimise various aspects of intelligent control systems. In particular, the algorithm can generate the fuzzy rule-base, and tune the parameters of the associated membership functions. The application of evolutionary algorithms to fuzzy optimisation is broadly split into two general areas; namely membership function tuning and rulebase design with tuning. GA has been applied (Tzes et al., 1998) to the off-line tuning of fuzzy membership functions, using a *fuzzy clustering* technique a fuzzy model was developed to describe the friction in a DC-motor system. In this case, the GA was seeded initially by the results obtained by fuzzy clustering. The results were greatly improved over those obtained by the non-tuned version. An *asynchronous* evolutionary algorithm has been used to generate membership functions to facilitate the rapid prototyping of fuzzy controllers (Kim et al., 1995). This approach utilized parallel processing, being implemented on a 512 processor CM-5 Connection Machine. The application in question was a simulated space-based oxygen production system. Evolutionary methods have also been used where the derivation of an obvious set of fuzzy rules is not immediately apparent. In this case, the designer may either pre-specify a number of rules, or allow the number of rules to become an extra degree of freedom in the design. In all cases, the computational intensiveness of the designed optimisation technique must be borne in mind, particularly in the case of on-line optimisation.

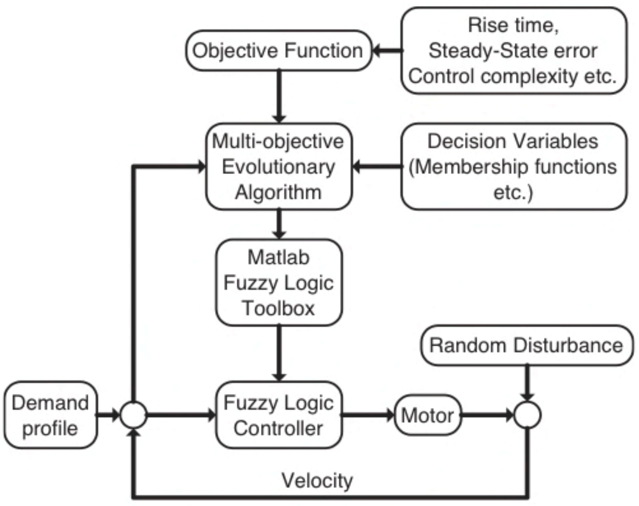
Due to the considerable computational and experimental considerations implicit in this method, certain constraints are included in the bounds of the decision variable vector in order to bring the automatic design time down to a reasonable level. A flowchart of the experimental set-up is shown in Fig. 11 and contains a number of elements.

The objective function contains the elements of performance and design to be minimised, principally

- rise-time in response to step changes in velocity demand,
- steady-state error in response to step changes in velocity demand,
- overall $\int vi\,dt$ power utilisation for a complete plant cycle,
- control complexity, i.e. the structure of the fuzzy logic controller is to be kept as simple as possible.

The decision variable vector contains the elements of controller design which are implemented in each individual during the evolutionary process. The decision variables include the number of inputs, number of membership functions for each input and output, number of rules in the rule base, and- or-ignore conjugates in each rule, and finally the defuzzification algorithm. The selected values in the decision variables vector are passed to the Matlab Fuzzy Logic Toolbox to be constructed into a controller file. In order to reduce the necessary execution time to converge to a satisfactory conclusion the decision variable vector is bounded as follows:

- number of inputs: 1–2
- number of membership functions for each input 3–5
- membership functions limited to triangular, with 2 base and one peak co-ordinate
- number of rules: 3–5
- conjugates: and, or, none
- defuzzification: centre of maximum.



Objective Function
Rise time, Steady-State error Control complexity etc.
Multi-objective Evolutionary Algorithm
Decision Variables (Membership functions etc.)
Matlab Fuzzy Logic Toolbox
Random Disturbance
Demand profile
Fuzzy Logic Controller
Motor
Velocity

Fig. 11. On-line fuzzy logic design setup.

In addition, a random +/−0.2 N m disturbance is injected during each experimental run to introduce an element of robustness into the design procedure. For each iteration of the design, the fuzzy controller was run on the motor rig and its performance ranked. It was found that the selected controller appeared early on in the procedure (generation 17 in a population of 10), in an initial run of 50 generations. The Pareto-optimal set of solutions included several configurations and combinations of membership functions, including one which was markedly similar to the solution defined by the off-line fuzzy design with on-line tuning. The solution chosen for presentation here, however, exhibits the required dynamic and steady-state performance but is coupled with a minimal set of membership functions (comprising an additional objective) and rules which presents computational advantages.

4.1. Results of on-line fuzzy logic controller design

The first results to present are those which show the dynamic and steady-state performance of the velocity controller. The undisturbed case is shown in Fig. 12, and the disturbed case in Fig. 13. In both cases, the velocity-tracking response of the system is comparable with earlier designs achieved by both PI and fuzzy logic control. One difference of particular interest is the current waveform in both cases which exhibits high-frequency components. This effect has been commented upon (Zhu et al., 2002) in the context of fuzzy logic control design, concluding that some off-line or on-line tuning is necessary to eliminate or effectively reduce the harmonics. In the case of the off-line fuzzy logic controller described earlier in this paper, the harmonics were reduced by on-line tuning. For future work in this case, the addition of frequency analysis to the objective function to minimise the unwanted harmonics would be a beneficial area of research. Hardware and computational constraints precluded the implementation of this analysis on-line at this time, but it is intended that the investigation of this phenomenon on an upgraded rig be

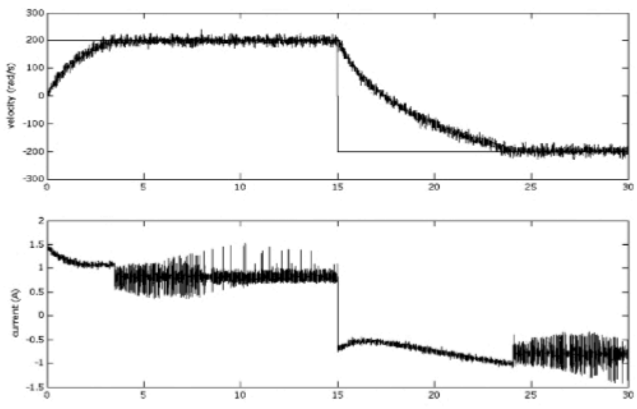


Fig. 12. On-line designed fuzzy logic velocity controller performance.

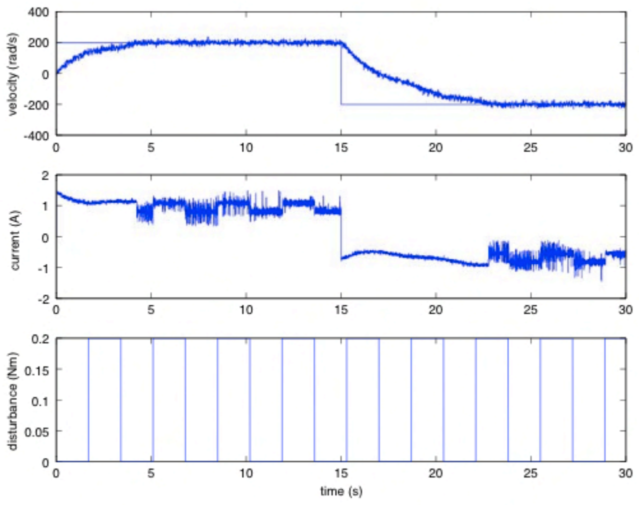


Fig. 13. On-line designed fuzzy logic velocity controller performance with disturbance.

performed at some future time. Although the performances of the various controllers are very similar, the structure of the on-line and off-line designed controllers are very different. Both have similar rule bases, but whereas the off-line design has inputs of both error and change-of-error, the automatically designed controller solely acts on error input. The membership functions which make up the input set are shown in Fig. 14, being the same number (5) as in the off-line designed case, but are far more closely clustered around the zero set. The membership functions which make up the output set are shown in Fig. 15 and are linked to the input set by the rule base;

- if velocity error is *negbig* THEN current demand is *negbig*,
- if velocity error is *negsmall* THEN current demand is *negsmall*,
- if velocity error is *zero* THEN current demand is *zero*,
- if velocity error is *posbig* THEN current demand is *posbig*,
- if velocity error is *possmall* THEN current demand is *possmall*.

The methods attached to the fuzzy logic controller were as follows.

- and:min
- or:max
- implication:min
- aggregation:max
- defuzzification:mom

5. Conclusions

The primary objective of this work, to assess the feasibility of automatically designing fuzzy logic con-

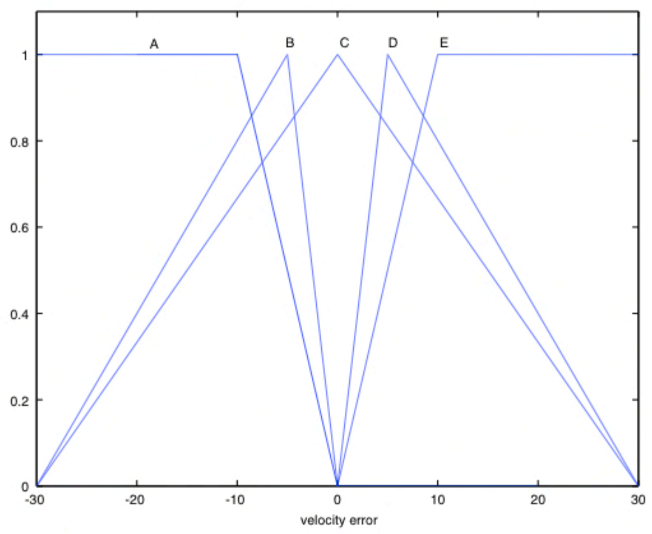


Fig. 14. On-line designed fuzzy logic velocity controller input membership functions. A, negbig; B, negsmall; C, zero; D, possmall; E, posbig.

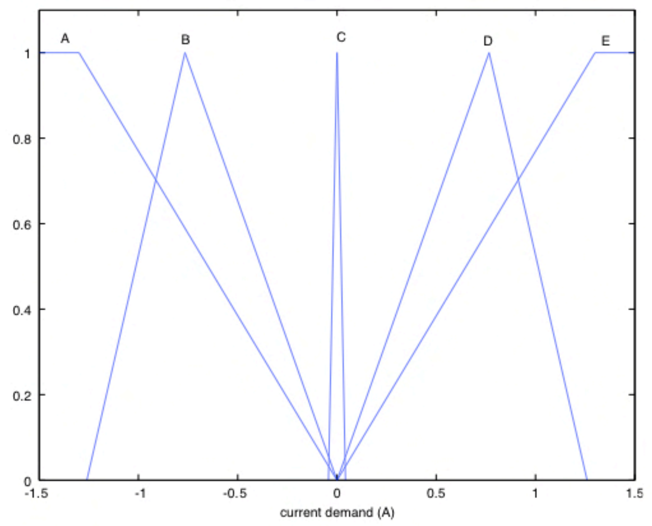


Fig. 15. On-line designed fuzzy logic velocity controller output membership functions. A, negbig; B, negsmall; C, zero; D, possmall; E, posbig.

trollers on-line with hardware in the loop has been demonstrated. A hardware platform previously intended for fuzzy logic design, formed the hardware in the loop since it was well characterised. The design of a fuzzy logic controller by traditional off-line methods had required manual tuning on line to maximise performance, and in particular, to reduce current harmonics introduced by the control action. It has been shown experimentally that on-line fuzzy logic controller design is feasible, and also that excellent dynamic and steady-state performance can be achieved. The design was optimised without the solicitation of knowledge because of the stochastic nature of the evolutionary optimisation algorithm which searches the multidimensional space of membership functions and rules for combinations which can achieve the performance specified in the objective function. Controller design based around models and simulation is often limited by the veracity of the model under consideration. For example, electromagnetic actuators may be approximated by relatively simple expressions. However under certain circumstances, dynamic effects such as eddy currents, which are extremely difficult to model, need to be included in dynamic simulation. In this case, the differences between actual and simulated plant can make a significant difference to the controller performance. It appears that the on-line fuzzy controller design offers considerable advantages and is worthy of serious consideration, also the possibility of injecting random disturbances during the design phase resulting in a controller capable of rejecting at least bounded disturbances shows particular promise. This topic together with consideration of the effects of controller dynamics on the harmonic content of the current waveforms will form part of a further, investigation.

References

Betin, F., Pinchon, D., Capolino, G.A., 2001. Control of electrical drives subject to large variations of load: a fuzzy logic approach. Electromotion 8 (3), 155–168.

Blumel, A.L., Hughes, E.J., White, B.A., 2001. Multi-objective evolutionary design of fuzzy autopilot controller. In: Zitzler, E. (Ed.), Evolutionary Multi-criterion Optimisation, ISBN 3-540-41745-1. Springer, Berlin, pp. 669–680.

Chang, C.H., Wu, Y.C., 1995. The genetic algorithm-based tuning method for symmetric membership functions of fuzzy logic control systems. Proceedings of the International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies. pp. 421–428.

Chipperfield, A.J., Fleming, P.J., Pohlheim, H.P., 1994. A genetic algorithm toolbox for MATLAB. Proceedings of the International Conference on Systems Engineering, Coventry, UK, pp. 200–207.

Driankov, D., Hellendoorn, H., Reinfrank, M., 1993. An introduction to fuzzy control. Springer, Berlin ISBN 3-540-56362-8.

Fonseca, C.M., Fleming, P.J., 1995. An overview of evolutionary algorithms in multiobjective optimisation. Evolutionary Computation 3 (1), 1–16.

Fonseca, C.M., Fleming, P.J., 1998. Multiobjective optimisation and multiple constraint handling with evolutionary algorithms—Part 1: a unified formulation and Part 2: application example. IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans 28 (1), 26–37, 38–47.

Grauel, A., Mackenberg, H., 1997. Mathematical analysis of the Sugeno controller leading to general design rules. Fuzzy Sets and Systems 85 (2), 165–175.

Guillemin, P., 1994. Universal motor control with fuzzy logic. Fuzzy Sets and Systems 63 (3), 339–348.

Homaifar, A., McCormick, E., 1995. Simultaneous design of membership functions and rule sets for fuzzy controller using genetic algorithms. IEEE Transactions on Fuzzy Systems 3 (2), 129–139.

Hong, T.P., Lee, C.Y., 1996. Induction of fuzzy rules and membership functions from training examples. Fuzzy Sets and Systems 84 (1), 33–47.

Hughes, E.J., 2001. Evolutionary multi-objective ranking with uncertainty and noise. In: Zitzler, E. (Ed.), Evolutionary Multi-criterion Optimisation, ISBN 3-540-41745-1. Springer, Berlin, pp. 329–343.

Ishibuchi, H., Tanaka, H., 1993. Neural networks that learn from if-then fuzzy rules. IEEE Transactions on Fuzzy Systems 1, 85–97.

Jamshidi, M., dos Santos Coelho, L., Krohling, R.A., Fleming, P.J., 2003. Robust control systems with genetic algorithms, CRC Press Control Series, ISBN 0-8493-1251-5.

Jaszkiewicz, A., Hapke, M., Kominek, P., 2001. Performance of multiobjective evolutionary algorithms. In: Zitzler, E. (Ed.), Evolutionary Multi-criterion Optimisation, ISBN 3-540-41745-1. Springer, Berlin, pp. 241–255.

Kim, J., Moon, Y., Ziegler, B.P., 1995. Designing fuzzy net controllers using genetic algorithms. IEEE Control Systems Magazine 15 (3), 62–72.

Kuo, B.C., Hanselman, D.C., 1994. Matlab tools for control system analysis and design. Prentice-Hall International, NJ ISBN 0-13-099946-6.

Mamdani, E.H., 1974. Application of fuzzy algorithms for control of simple dynamic plant. Proceedings of the IEE 121 (12), 1585–1588.

Schroder, P., Green, B., Grum, N., Fleming, P.J., 2001. On-line evolution of robust control systems: an industrial active magnetic bearing application. IFAC Journal of Control Engineering Practice 9, 37–49.

Thrift, P., 1996. Fuzzy logic synthesis with genetic algorithms, Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 279–283.

Tzes, A., Peng, P.Y., Guthy, J., 1998. Genetic-based fuzzy clustering for DC-motor friction identification and identification. IEEE Transactions on Control Systems Technology 6 (4), 462–472.

Wu, C.J., Liu, G.Y., 2000. A genetic approach for simultaneous design of membership functions and fuzzy control rules. Journal of Intelligent and Robotic Systems 28, 195–211.

Zadeh, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man and Cybernetics 3, 28–44.

Zhu, Z.Q., Shen, Z.X., Howe, D., 2002. Comparative study of alternative fuzzy logic control strategies of Permanent Magnet Brushless AC drive. Proceedings of the 2002 International Conference on Control Applications, September 18–20, Glasgow, Scotland, UK, pp. 42–47.