On-line multiobjective automatic control system generation by evolutionary algorithms

P Stewart[1], Member, IEEE, D.A. Stone[2], and P.J. Fleming[3].

[1,2]Electrical Machines and Drives Research Group, Department of Electronic and Electrical Engineering, University of Sheffield, UK., e-mail : p.stewart@sheffield.ac.uk

[3]Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK.,

***Abstract*— Evolutionary algorithms are applied to the *on-line* generation of servo-motor control systems. In this paper, the evolving population of controllers is evaluated at run-time via hardware in the loop, rather than on a simulated model. Disturbances are also introduced at run-time in order to produce robust performance. Multiobjective optimisation of both PI and Fuzzy Logic controllers is considered. Finally an on-line implementation of Genetic Programming is presented based around the Simulink standard blockset. The on-line designed controllers are shown to be robust to both system noise and external disturbances while still demonstrating excellent steady-state and dynamic characteristics.**

INTRODUCTION

This paper investigates the potential of robust, automatic, multiobjective control design with hardware in the loop. Tuning of PI parameters on-line [19] has been achieved via multiobjective genetic algorithms applied to a sealed pump running on magnetic bearings for a nuclear powered submarine. 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 microcontroller is used as a platform to develop and assess the control algorithms. An on-line tuned controller type (PI) and 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 [12] 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. Finally a method based on *Evolutionary Programming*, choosing both controller structure, elements and parameters is presented. Again the optimisation procedure is conducted via hardware in the loop. Fuzzy logic control, comprising a fuzzification interface, rule base and defuzzification algorithm [18], [22], has been applied to a wide variety of motion control applications [1], [9]. A vital region of interest concerns the implementation of the fuzzy controller. Several different approaches have been postulated to extract the knowledge base from experts or training examples to construct the input-output membership functions and the fuzzy rule-base. These methods can be based on neural networks [11], [14] or the application of fuzzy clustering techniques to construct a fuzzy controller from training data sets [8]. 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 [11], a method 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 [2], [10], [20]. 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 [21]. The development of these techniques to design optimal robust fuzzy logic controllers for example gas turbine engines [15] and aerospace autopilots [13] has arisen to satisfy the need which exists when expert heuristic knowledge doesn't 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 perfomance indicies. In this paper the implementation of GA fuzzy design is evaluated by 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. The final implementation considered in this paper is Genetic Programming (GP), which is an extension of the Genetic Algorithm specifically for handling complex computational structures [16]. The solution of a problem by means of GP is achieved by searching combinations of symbolic expressions. In this case, the symbolic expressions are blocks from the *Simulink* library, with the subsequent controllers evaluated in real-time on hardware in-the-loop. These symbolic networks are supported by parameter sets (e.g. gain values, PID coefficients etc.) optimised by evolutionary al-

0-7803-8304-4/04/$20.00 ©2004 IEEE

gorithm.

A. Multiobjective 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 robust servo controllers 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 [15]:

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

where

$$\mathbf{x} \in D, \quad (2)$$

The solution of $\mathbf{x} = [x_1, ..., x_i]$ 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*

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

such that

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

Point $\mathbf{z}^1$ *dominates* $\mathbf{z}^2$, $\mathbf{z}^1 \succ \mathbf{z}^2$, if $\forall j\ z_j^1 \leq z_j^2$ 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 tradeoff surface. Fitness assignment is performed as follows [6];

- Sort population according to rank.
- Assign fitness to individuals by interpolating from the best (rank 1) to the worst (rank $n \leq M$), the *Pareto ranking assignment process* [7], 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 *nondominated* solutions will exist. These *Pareto-optimal* [4], [5] 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 an 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.

B. Hardware overview

The application consists of a brushed DC permanent magnet field motor fed by a four quadrant DC chopper drive operating at 5kHz. Figure 1 shows a schematic of the on-line control system and hardware setup. 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 200V 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 automatically designed

velocity controller. The microcontroller also hosts the velocity and current feedback signals from the motor set and chopper drive respectively. The multiobjective optimisation

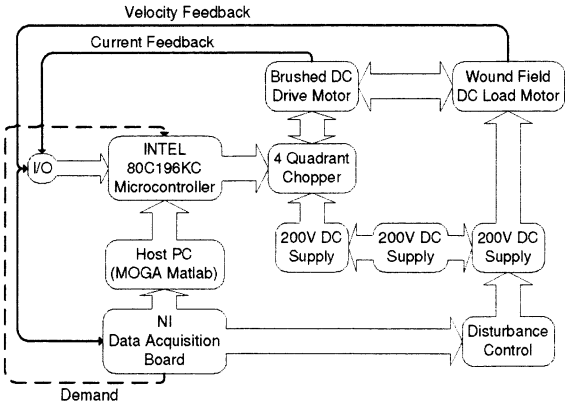

Fig. 1. Online optimisation hardware setup

programme runs under *Matlab* [18], 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.

I. 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 $settling-time$ [17]. 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 [19]. In the application under consideration here, the PID controllers to be optimised comprise two cascaded [17] control loops (figure 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

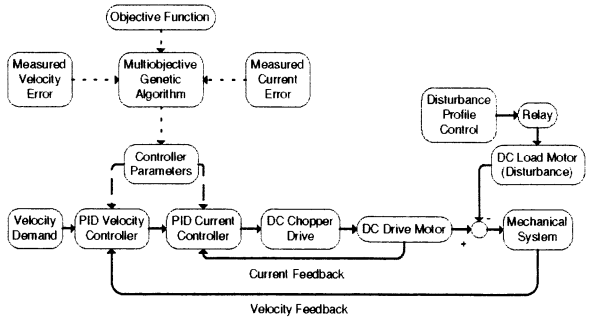

Fig. 2. Online PID current and velocity controller optimisation setup

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

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.

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

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

- minimise $\int |\epsilon_i| dt$
- minimise $\int |\epsilon_v| dt$
- minimise $\int vi.dt$

where ϵ_i is the current tracking error, ϵ_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 $0.3Nm$ which is approximately 30% of torque at the rated current of $1.5A$. 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.

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 tradeoff 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.

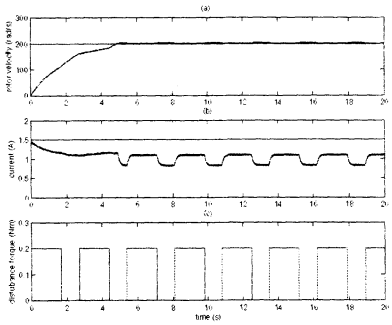

Fig. 3. Motor step demand tracking response with external torque disturbance. (a):velocity response, (b):current response, (c):estimated disturbance torque

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

The switching pattern of the relay controlling the voltage to the load motor is shown in graph (c) (figure 3. The values shown have been scaled to the value of applied disturbance torque. This compares with a maximum torque of $1Nm$ from the drive motor at a rated current of $1.5A$. Under these conditions, a rise-time from $0rads^{-1}$ to $200rads^{-1}$ in approximately $4.5s$ can be expected.

II. On-line fuzzy logic controller design

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 setup is shown in figure 4 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

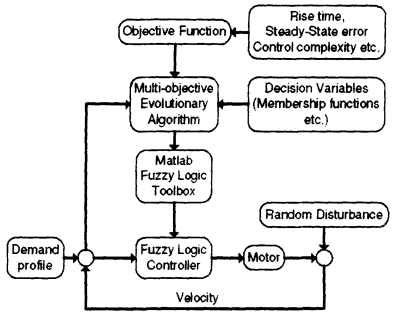

Fig. 4. On-line Fuzzy Logic design setup

- Control complexity, i.e. the structure of the fuzzy logic controller is to e 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

In addition, a random $+/-0.2Nm$ 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. 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 figure 5, and the disturbed case in figure 6 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 [23] in the context of fuzzy logic control design, concluding that some off-line or on-line tuning

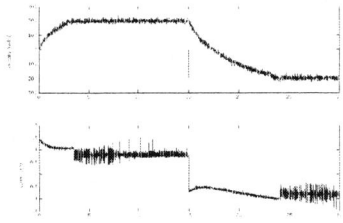

Fig. 5. On-line designed Fuzzy Logic velocity controller performance

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 performed at some future time. Although the performances

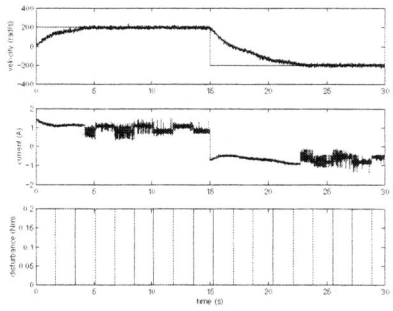

Fig. 6. On-line designed Fuzzy Logic velocity controller performance with disturbance

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

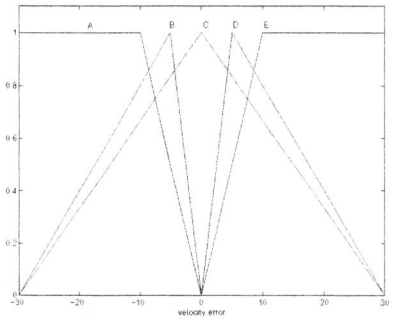

Fig. 7. On-line designed Fuzzy Logic velocity controller input membership functions. A:negbig, B:negsmall, C:zero, D:possmall, E:posbig.

membership functions which make up the input set are shown in figure 7, 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 figure 8 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*

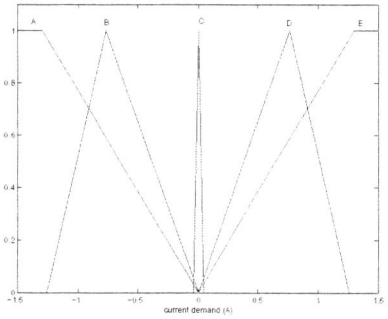

Fig. 8. On-line designed Fuzzy Logic velocity controller output membership functions. A:negbig, B:negsmall, C:zero, D:possmall, E:posbig.

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

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

III. GP CONTROLLER DESIGN

A symbolic approach was adopted to design the *RST* controller shown in block diagram form in figure 9.The *decision variables* which form the genetic material of the individuals in the population are the contents of the R, S and T controllers. Each genome contains information defining the symbolic contents and associated parameters of each of the controllers. The symbolic contents available are shown in

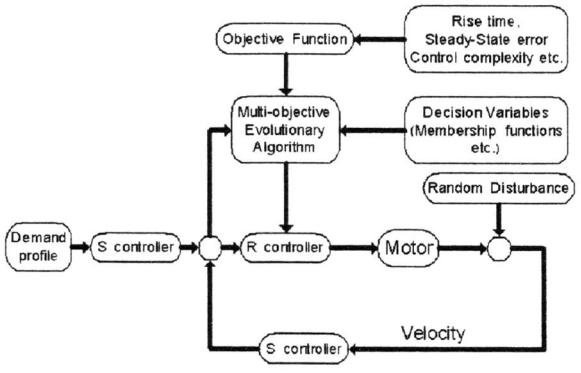

Fig. 9. Genetic Programming design setup

figure 10 An identical GA setup to the PI and Fuzzy Con-

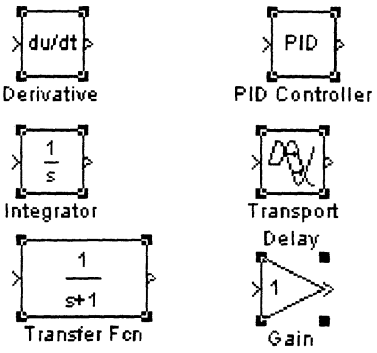

Fig. 10. Simulink symbolic library

troller optimisation was applied to the GP design. When greater computational power is available on-line, then it will be possible to include control structures other than continuous time controllers in the optimisation process. In this case however, the best control set in terms of the evaluation function turned out to be of a PI structure, with gains similar to those identified in the previous section on PI parameter optimisation. The tracking response of the controller is shown in

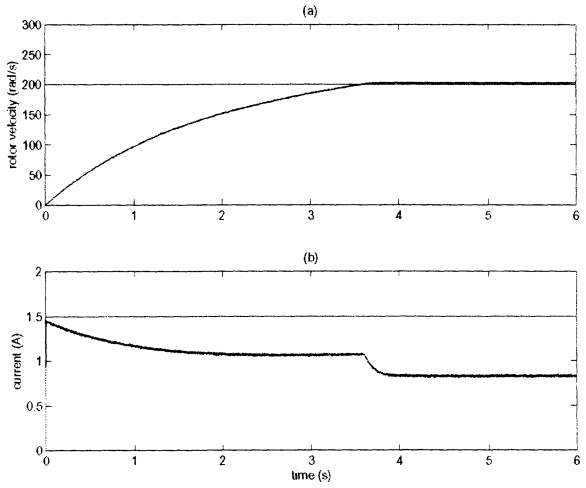

Fig. 11. Motor step demand tracking response without external torque disturbance. (a):velocity response, (b):current response

figure 11. Evolutionary algorithms have been applied to the design and optimisation of DC motor servo controllers. It has been shown that the search algorithm is effective in designing both fixed (PI) and variable (fuzzy) structure controllers. Applying a symbolic approach to Genetic Programming, a Genetic algorithm was applied to searching for controllers for an RST feedback controller. A PI controller with similar gains to that already identified was chosen as the optimal controller. It appears that a GA/GP approach could facilitate a *generic* approach to controller design, with *automatic* controller *structure* identification. Future work will allow the competition to occur not only between standard continuous time controllers, but also populations or *ecologies* of biologically inspired heuristics (eg neural etc.), to allow a search of non-standard control structures.

REFERENCES

[1] Betin F., Pinchon D. and Capolino G.A., Control of electrical drives subject to large variations of load: a fuzzy logic approach., *Electromotion*, 8(3), July-September 2001, 155-168.

[2] Chang C.H. and Wu Y.C., 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.*, 1995, 421-428.

[3] Chipperfield A.J., Fleming P.J. and Pohlheim H.P., A genetic algorithm toolbox for MATLAB, *Proceedings of the International Conference on Systems Engineering*, 1994, pp.200-207, Coventry U.K.

[4] Fonseca C.M. and Fleming P.J., An overview of evolutionary algorithms in multiobjective optimisation, *Evolutionary Computation*, 3(1), 1995, 1-16.

[5] Fonseca C.M. and Fleming P.J., 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), 1998, 26-37 and 38-47.

[6] Fonseca C.M. and Fleming P.J., Genetic algorithms for multiobjective optimisation: formulation, discussion and generalisation. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp.416-423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[7] Goldberg D.E., Genetic algorithms in search, optimisation and machine learning. *Addison Wesley Publishing company*, Reading, Massachusetts,1989.

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

[9] Guillemin P., Universal motor control with fuzzy logic, *Fuzzy Sets and Systems*, 63(3), May 1994, 339-348.

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

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

[12] Hughes E.J., Evolutionary multi-objective ranking with uncertainty and noise, *Evolutionary Multi-criterion Optimsation* Zitzler E. (Ed.), pp.329-343, ISBN 3-540-41745-1, Springer-Verlag, 2001.

[13] Blumel A.L., Hughes E.J. and White B.A., Multi-objective evolutionary design of fuzzy autopilot controller, *Evolutionary Multi-criterion Optimsation* Zitzler E. (Ed.), pp.669-680, ISBN 3-540-41745-1, Springer-Verlag, 2001.

[14] Ishibuchi H. and Tanaka H., Neural networks that learn from if-then fuzzy rules, *IEEE Transactions on Fuzzy Systems*, vol.1, 1993, 85-97.

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

[16] Koza J.R., Genetic Programing: On the programing of computers by means of natural selection, *MIT Press* Cambridge, 1992.

[17] Kuo B.C. and Hanselman D.C., Matlab tools for control system analysis and design, *Prentice-Hall International, New Jersey*, 1994, ISBN 0-13-099946-6.

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

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

[20] Thrift P., Fuzzy logic synthesis with genetic algorithms, *Proceedings of the 4th International Conference on Genetic Algorithms*, 1996, 279-283.

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

[22] Zadeh L.A., Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man and Cybernetics*, 3(1973), 28-44.

[23] Zhu Z.Q., Shen Z.X. and Howe D., Comparative study of alternative fuzzy logic control strategies of Permanent Magnet Brushless AC drive, *Proceedings of the 2002 International conference on Control Applications*, pp.42-47, September 18-20, 2002. Glasgow, Scotland. U.K.