Designing a robust controller for a high redundancy actuator using genetic algorithms

Optimal Passive Fault Tolerant Control of a High Redundancy Actuator

Thomas Steffen, Konstantinos Michail, Roger Dixon, Argyrios Zolotas, Roger Goodall

Control Systems Group,Department of Electronic and Electrical Engineering,Loughborough University, Loughborough, LE11 3TU, UK, e-mail:{t.steffen,k.michail,r.dixon,a.c.zolotas,r.m.goodall}@lboro.ac.uk

Abstract: The High Redundancy Actuator (HRA) project deals with the construction of an actuator using many redundant actuation elements. If one element fails, this changes the behaviour slightly, but the system still remains operation. A key challenge in this project is to design a passive fault tolerant controller that maintains the required performance in the presence of faults. This paper shows how to achieve this with structurally simple controllers, by optimising the parameters using a genetic algorithm.

Keywords: high redundancy actuator, fault-tolerant control, passive fault tolerance, fault accommodation, robust control, genetic algorithm, controller optimisation.

1. HIGH REDUNDANCY ACTUATION

High Redundancy Actuation (HRA) is a new approach to fault tolerant actuation, where an actuator comprises of a large number of actuation elements (see Fig. 1). Faults in the individual elements can be accommodated without resulting in a failure of the complete actuator system.

The concept of the High Redundancy Actuation (HRA) is inspired by human musculature. A muscle is composed of many individual muscle cells, each of which provides only a minute contribution to the force and the travel of the muscle. The aim of this project is to use the same principle of co-operation with existing actuation technology to provide intrinsic fault tolerance.

An important feature of the HRA is that the actuator elements are connected both in parallel and in series. This makes it possible to deal with the two main fault modes: lock-up and loose elements. In case of a lock-up fault, the available travel is slightly reduced, but the elements in series can still move. If an element fails loose, this reduces the maximum force, but the elements in parallel are still active.

Controlling an HRA is a challenging problem, because of the complexity of the system. The goal is to find a controller that is no more complex than a typical PD or PID type controller as used for a conventional actuator. The controller needs to be robust enough to deal with the change of system behaviour caused by faults. Previous work is done by Chen and Jiang (2005) and by Dan and G.Yang (2006) based upon using active fault tolerance control schemes and passive fault tolerant controller design is considered by Zhao and Jiang (1998). The method proposed here is via passive fault tolerant control where, the system becomes insensitive to faults by design of a robust controller (Patton, 1997). Because no structured approach is available to solve this problem, this paper investigates a number of promising control structures.

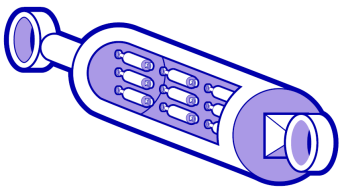


Fig. 1. Configuration of a High Redundancy Actuator

For the optimisation of the controllers parameters, a heuristics approach is used. For a good overview see Dreo et al. (2006). Such optimisation methods are able to search randomly in a predefined search space, aiming to find the optimal controller parameters. Evolutionary algorithms are one of the heuristic approaches that exists and implemented in control engineering (Fleming and Purshouse, 2002). The Non-dominated Sorting Genetic Algorithms II (NSGA-II) introduced by Deb et al. (2002) is used here. A recent implementation is considered by Michail et al. (2008) where it is part of the sensor optimisation framework for a MAGLEV suspension.

Section 2 introduces the idea and the model of the HRA. Section 3 presents the control structures that have been used. Section 4 presents the control objectives and functional boundaries as well as the genetic algorithm used to solve the control problem. Simulation results are analysed in Section 5 and the paper closes with the conclusion and outlook in Section 6.

2. SIMPLIFIED MODEL OF THE HRA

The HRA considered here uses direct electromagnetic actuation (voice coil principle). Other technologies are also possible, but they may lead to a slightly more complex model.

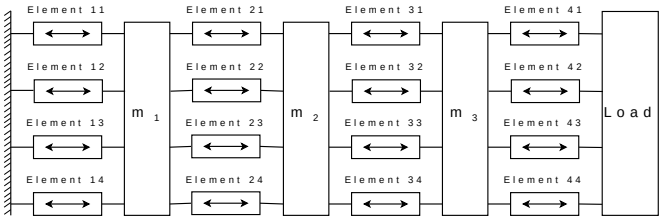


Fig. 2. 4×4 HRA Configuration

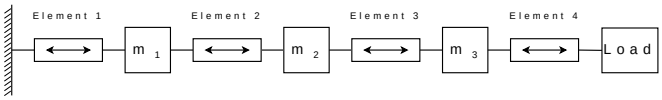


Fig. 3. Four Actuation Elements in Series

Single Element

A single actuation element behaves like a spring-damper combination

$$m\ddot{x} = ku - d\dot{x} - rx \tag{1}$$

where x is the position, m is the moving mass, k is the input coefficient, d is the damping factor (accounting for mechanical and electrical damping), r is the elasticity of the spring, u is the electrical input, and x is the position of the mass. For the state space model, the system state

$$\mathbf{x} = (\dot{x}\ x)^T$$

is used. The parameters for the model are given as

$$\begin{array}{rlrl} m &= 1 \text{ kg} & k &= 10 \text{ N/V} \\ d &= 10 \text{ Ns/m} & r &= 1 \text{ N/m} \end{array}$$

Further details on the modelling of the actuator can be found in Davies et al. (2008b).

HRA Model

Parallel elements can be modelled as a single actuator, because the elements are linked together, representing one moving mass. However, elements in series consist of individually moving masses. So a 4×4 HRA (see Fig. 2) can be modelled using four moving masses (see Fig. 3). As shown in Steffen et al. (2008), it is possible to align the dynamics of all masses by cancelling the additional poles with input-decoupling zeros. Then the connecting masses have no influence on the input-output behaviour, and only the behaviour of the load mass remains. So equation (1) still applies, if the parameters are adjusted accordingly:

$$m\ddot{x} = ku - \frac{1}{n}(d\dot{x} - rx)$$

where n denotes the number of elements.

For the example application, $n = 4$ and intermediate masses of $m_1 = 0.2\,\text{kg}$ are used, leading to the SISO transfer function

$$G_0(s) = \frac{1}{(s + 2.4)(s + 0.104)} \tag{2}$$

for the nominal system (fault-free situation).

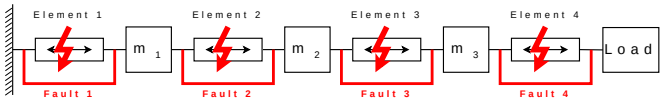


Fig. 4. Lock-up faults

Behaviour with Faults

This paper only deals with lock-up faults, because they have a more significant influence on the dynamics than other fault modes. A lock-up of an element means that the two masses it links move at the same speed, so they can be modelled as one mass (Fig. 4). For example, a lock-up of the second element leads to

$$G_{F2}(s) = 10\frac{s + 134.5}{s + 134.1}\frac{s + 41.6}{s + 41.1}\frac{1}{(s + 3.3)(s + 0.103)} \quad , \tag{3}$$

and a double fault in elements 2 and 3 leads to the transfer function

$$G_{F23}(s) = 10\frac{1}{(s + 5)(s + 0.102)} \quad . \tag{4}$$

The main change in behaviour is the movement of the faster pole from 2.4 to 3.3 or 5. Compared to this, all other changes are of minor significance. While the simulation of the results will use the full model, it is reasonable to use a simplified model for the controller design.

Simplified Model with faults

In order to speed up the controller design problem, only the following two characteristic transfer functions will be used to represent fault cases. The first one is an approximation of all cases with a single lock-up actuator fault:

$$G_{Fa}(s) = 10\frac{1}{(s + 3.3)(s + 0.103)} \quad . \tag{5}$$

The deviation from the actual transfer functions is small, and should be handled easily by any reasonably robust controller. The second function represents a lock-up of two elements:

$$G_{Fb}(s) = 10\frac{1}{(s + 5)(s + 0.102)} \quad . \tag{6}$$

3. CONTROL STRUCTURES

One of the main themes of this paper is to compare different control structures in how well they deal with faults in the HRA. There are three different control strategies compared as listed in Table 1.

Table 1. Control strategies used

Control Structure	abbrev.	Fig.	Type	Degrees Of Freedom
Classical controller	PID	5	SISO	1
P-I Phase advance	PI-PA	6	SISO	2
State feedback	SF	7	MIMO	1

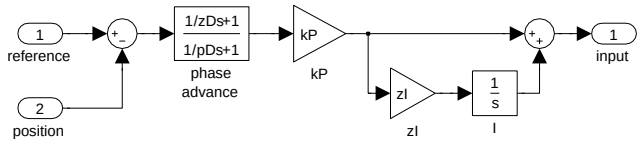


Fig. 5. Classical PID Controller

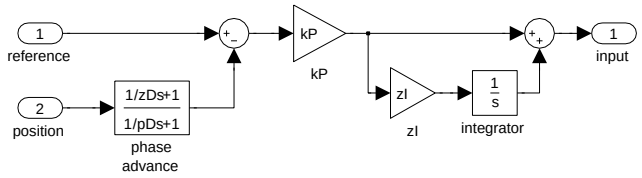


Fig. 6. PI Phase Advance Controller

Traditional PID

The first two structures are traditional PID controllers, implemented as a PI controller with a phase advance compensator. They both implement the controller behaviour

$$G_{\mathrm{PID}}(s) = k_P \frac{s + k_I}{s} \frac{\frac{s}{k_D} + 1}{\frac{s}{p_D} + 1} \ , \tag{7}$$

with four parameters: k_P is the proportional gain, k_D and k_I are the zeros of the phase advance compensator and the integral branch, and p_D is the realisation pole of the compensator. First, a typical implementation as shown in Fig. 5 is used here. To avoid wind up in the integrator, it is stopped while the input is at the limit of 24 V, but this is not shown in the diagram.

PI with Feedback Phase Advance

A common variation of the PID controller is also tested. The difference is that the phase advance compensator is only applied to the feedback signal, but not to the reference (see Fig. 6). This configuration is common in mechanical systems, and it is known to deal better with input limitations.

State Feedback

The third control structure includes the measurement of the speed of the actuator in addition to the position. While the speed could be found by calculation the derivative of the position, measuring it directly is more accurate. The speed measurement is added to the position error, which replaces the phase advance compensator (see Fig. 7). The resulting controller has two possible interpretations: it is a PID controller with a real derivative, or it is a state feedback controller with an error integral state. The control law is:

$$u = k_P \left(x_{\mathrm{ref}} - x - k_D \dot{x} + k_I \int_0^t x_{\mathrm{ref}} - x \, dt \right) \ . \tag{8}$$

This structure has two significant advantages. Because no phase advance compensator is used, this structure is much less sensitive to measurement noise. So higher gains can be used without excess noise levels in the control input. The other advantage is that it has only three parameter, which simplifies the optimisation process.

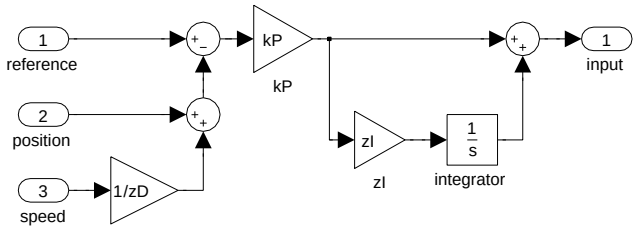


Fig. 7. State Feedback Controller

The speed can be measured directly, or it can be inferred from other measurements, such as the acceleration or the induced voltage in the actuator coils. In any case it will require an additional sensor, because a state observer would not work well in the different fault cases.

4. OPTIMISATION OF CONTROLLER PARAMETERS

Control Objectives

The control application considered here is position control, and the goal is to reach the target position soon and with very little overshoot. Consequently, two objectives are considered: the settling time and the overshoot. Both are defined in terms of a reference step response, where the system moves from $x(0) = 0$ to $x_{\mathrm{ref}} = 0.1m$.

Unlike in typical optimisations, it is necessary to consider not only the nominal (fault free) case, but also the behaviour of the system under faults. The two objective functions are then defined as the average overshoot (M_p) and the average settling time (t_s) over these three cases:

$$M_p = \frac{1}{3} \left(M_{p,f0} + M_{p,f1} + M_{p,f2} \right) \tag{9}$$

$$t_s = \frac{1}{3} \left(t_{s,f0} + t_{s,f1} + t_{s,f2} \right) \tag{10}$$

The subscripts $\cdot_{f0}$, $\cdot_{f1}$ and $\cdot_{f2}$ denote the case with no fault and the cases with one fault and two faults respectively.

The only constraint assigned to the optimisation is that the overshoot is less than 2%

$$M_p < 2\% \quad . \tag{11}$$

Any excessive overshoot is added as a penalty to both criteria.

Multi-Objective Constraint Optimisation

The evolutionary algorithm selected is based on the non-dominated sorting of the individuals in the chromosome based on the ranking and the crowding distance of the individuals. A comparison between the different existing algorithms is found in Zitzler et al. (2000). The results show that the convergence of the NSGA algorithm (later developed into NSGA-II) is good even in the presence of local minima, and it is therefore suitable for the problem under consideration.

The parameters used are critical for the optimisation procedure. The parameters chosen here are shown on table 2. The crossover probability is generally selected to be

large in order to have a good mixing of genetic material. The mutation probability is defined as $1/n_v$, where n_v is the number of variables ($n_{v_{SF}} = 3$,$n_{v_{PI-PA}} = 4$,$n_{v_{PID}} = 4$). For the simulated binary crossover parameter (SBX) and the mutations parameter it was decided to use the value of 20 for each since they provide good distribution of solutions for the algorithm operations.

Table 2. NSGA-II Parameters

Parameter	setting
Crossover probability	0.9
Mutation probability	$1/n_v$
SBX parameter	20
Mutation parameter	20
Rigid bounds	1
Population	200
Generations	150

There is no systematic method to define those values as they depend on the nature of the problem. Instead, these values were selected after numerous trial runs. In addition, the search space is limited to reasonable parameters based on manual controller designs. To achieve the optimal solutions within limits the penalty function approach is used. For more details see Deb (2001).

Computational Time

The main problem with heuristic optimisation methods is that they require the objective function to be evaluated many times resulting in large computational effort (Hidalgo and Fernandez, 2005). The exact number depends on the specific problem, but it is in the order of 10000 times for the problem considered here. Each evaluation requires a full simulation run of the system model, and this is where most of the processing time is spent.

By keeping the model as simple as possible, each simulation run can be kept to well under one second on a standard PC. To further speed up the process, a trick is used to achieve vectorisation. Because MATLAB is an interpreted language, the overhead per operations can be high, and vectorisation (the processing of several values at a time) leads to significant performance gains by better amortisation of this overhead.

In Simulink, the same result can be achieved by using vector signals, where each element represents a signal for a different controller. With just one simulation, it is possible to generate the trajectories for a number of controllers at the same time. While this restricts the numbers of blocks and signals that can be used (see Fig. 8), it is quite feasible for the simple system discussed here. With a suitable solver and the correct amount of vectorisation (between 10 and 100 signals), it can lead to a performance gain of nearly a factor of ten. This brings the time for a full optimisation run down to less than a minute.

5. SIMULATION RESULTS

Pareto Front of Optimal Controllers

The results of the optimisation process are shown in Fig. 9. Because the objective contains two criteria, there are many

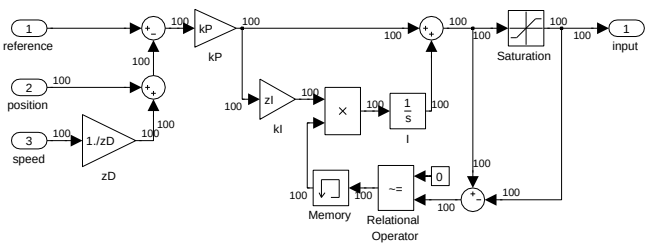


Fig. 8. Vectorised Controller Implementation

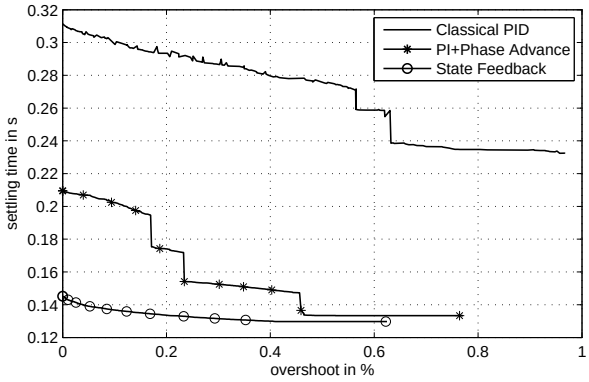


Fig. 9. Pareto Optimal Solutions

solutions that are each Pareto-optimal because no other solution can improve on both criteria. It follows logically that the Pareto curve has to be falling.

Still, the results clearly show that the three control structures lead to different results. The state feedback controller has the shortest settling time. The PI controller with phase advance in the feedback comes next. It is only very slightly slower for higher overshoot ($> 0.5\,\%$), but it is at a significant disadvantage for very low overshoot. The classical PID controller is significantly inferior, so much so that it cannot be recommended for this application.

Another interesting aspect of these optimal solutions is that there are jumps in the settling time. This indicates that the oscillation of the system violates the $1\,\%$ settling band, thus increasing the settling time significantly. This effect can be seen in more detail in the simulation results below. This effect would be unusual in the control of a single spring damper system, but it does happen because the controller has to deal with the plant under different fault cases. So these jumps are a sign that the controller cannot deal well with the resulting differences in behaviour.

Finally it is remarkable that the solution sets stop long before the penalty applies for the hard limit at $2\,\%$ overshoot. There are a number of explanations for this. The optimisation variable is the average over all three fault cases, while the penalty applies per individual case, so the average will be significantly below the threshold. Also the settling band of only $1\,\%$ penalises any higher overshoot. The simulations will show that the optimal controllers show nearly no oscillation in the system response.

Step Response by Fault Case

To test the ability of the designed controllers to deal with faults, a number of more detailed simulations are

Table 3. Settling Time Results by Design Overshoot M_p

M_p	Faults	PID	PI-PA	State-FB
0.0 %	f_0	172ms	208ms	133ms
	f_1	358ms	212ms	146ms
	f_2	408ms	212ms	160ms
0.1 %	f_0	168ms	197ms	123ms
	f_1	345ms	207ms	135ms
	f_2	392ms	207ms	151ms
0.5 %	f_0	161ms	122ms	146ms
	f_1	314ms	130ms	126ms
	f_2	363ms	147ms	144ms
0.6 %	f_0	217ms	122ms	148ms
	f_1	305ms	130ms	127ms
	f_2	368ms	147ms	143ms

f_0: fault-free, f_1: one fault, f_2: two faults

performed. Because of the high number of parameters, only a small cross-section of the results can be presented here. The results for an overshoot objective of 0.1 % and 0.5 % are chosen (see Table 5). While the amplification may seem rather high, the range of the position states (0.1 m) is small compared to the range of inputs (24 V), and the measurement resolution is high, so the controllers are quite reasonable. The detailed system model is used for all verifications, leading to the simulations in Figs. 10 and 11.

The detailed models are different from the models used for the controller design, so the response may differ slightly from the one used for the evaluation function. Even if the effect is small, it may make the difference between an oscillation being within the settling band or not. This explains why the results in the simulations may not match exactly with the prediction of the evaluation function. A tabular overview of the results is shown in Tables 3 and 4.

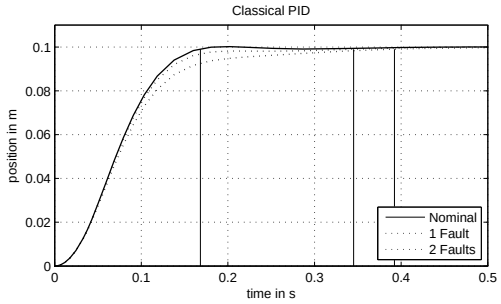
The simulations fully support the conclusions from the Pareto sets (Fig. 9). The classical PID controller (Fig. 10(a)) does not deal well with the behavioural changes of the system under different fault cases. This leads to a slow settling especially in the fault cases. A more aggressive tuning could improve this, but it would also result in a higher overshoot in the nominal case, which is forbidden by the constraints.

The PI controller with phase advance in the feedback path is significantly better (Fig. 10(b)). It shows a fast initial response, good settling behaviour, and consistency across the different fault cases. The state feedback controller (Fig. 10(c)) improves again. The main difference is that the system settles without any oscillation.

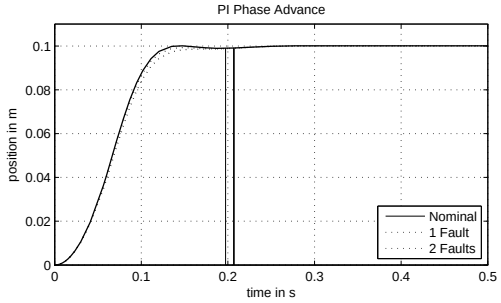
Similar result are found for the higher overshoot of 0.5 % in Fig. 11. The classical PID controller is still inferior to the two improved controllers, but the difference between PI with phase advance and state feedback is nearly invisible.

6. CONCLUSIONS AND FURTHER RESEARCH

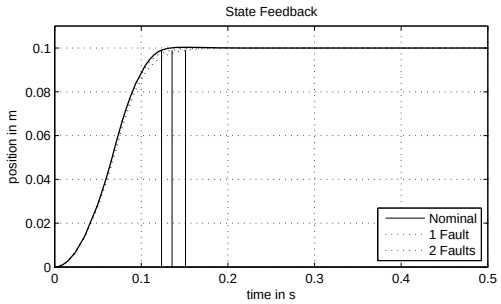
The simulations show that all the designed controllers can satisfy the requirements, despite their simple structure. The PID controller has some difficulties in dealing with



(a) *Classical PID with 0.1% overshoot*



(b) *PI Phase advance with 0.1% overshoot*



(c) *State feedback with 0.1% overshoot*

Fig. 10. Simulation results for Overshoot 0.1%.

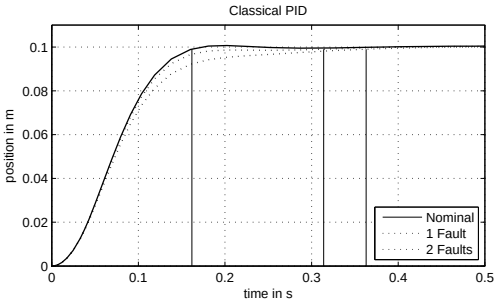
Table 4. Overshoot Variation Due To Faults.

M_p	Faults	PID	PI-PA	State-FB
0.0 %	f_0	0%	0.002%	0.01%
	f_1	0%	0%	0%
	f_2	0%	0%	0%
0.1 %	f_0	0.19%	0.10%	0.34%
	f_1	0.07%	0.10 %	0%
	f_2	0%	0.08%	0%
0.5 %	f_0	0.66%	0.42%	0.34%
	f_1	0.95%	0.25%	0.23%
	f_2	1.06%	0.28%	0.17%
0.6 %	f_0	1.27%	0.99%	1.04%
	f_1	0.29%	0.40%	0.39%
	f_2	0.21%	0.39%	0.37%

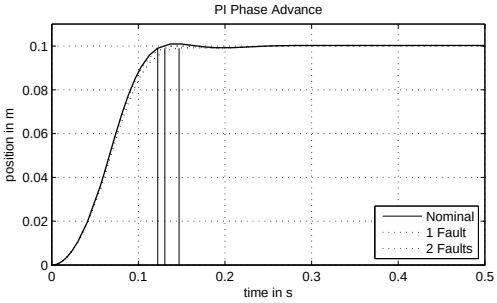
f_0: fault-free, f_1: one fault, f_2: two faults

faults, but this can be improved significantly by moving the phase advance block into the feedback path.

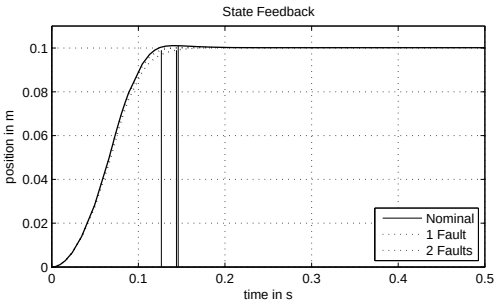
The key to a good robustness of the controller is a high gain. Together with a phase advance block, this could lead to excessive noise in the control input, if the resolution of the position sensor is not sufficient. The state feedback controller solves this problem, by taking the speed into account. While this does require an additional sensor,



(a) *Classical PID with 0.5% overshoot*



(b) *PI Phase advance with 0.5% overshoot*



(c) *State feedback with 0.5% overshoot*

Fig. 11. Simulation results for Overshoot 0.5%.

Table 5. Optimal Controller Parameters

M_p	Parameter	PID	PI-PA	State-FB
0.1 %	k_P	202	927	4000
	k_I	0.02	0.03	0.0
	k_D	8.6	19.2	32.4
	p_D	47.0	77.4	-
0.5 %	k_P	203	927	3856
	k_I	0.05	0.06	0.04
	k_D	8.6	19.5	32.9
	p_D	46.8	77.3	-

it allows both faster reaction and a higher gain without increased complexity in the controller.

All three controllers require only a few mathematical operations, so they can easily be implemented even in a very simple digital processor. More importantly, the controller order is independent of the configuration of the HRA. This means that even high orders such as 10×10 HRA configuration (as in Davies et al. 2008a) are feasible. In fact, the higher number of elements may even simplify the control, because the effect of individual faults is reduced.

ACKNOWLEDGEMENTS

This project is a cooperation of the Control Systems group at Loughborough University, the Systems Engineering and Innovation Centre (SEIC), and the actuator supplier SMAC Europe limited. The project is funded by the Engineering and Physical Sciences Research Council (EPSRC) of the UK under reference EP/D078350/1.

REFERENCES

W. Chen and J. Jiang. Fault-tolerant control against stuck actuator faults. *IEE Proceedings Control Theory and Applications*, 152 Issue 2:p138–146, 2005.

Y. Dan and G.Yang. Adaptive fault-tolerant tracking control against actuator faults with application to flight control. *IEEE Transactions on control systems technology*, 14 No.6:p1088–1096, 2006.

J. Davies, T. Steffen, R. Dixon, and R. M. Goodall. Multi-agent control of a 10x10 high redundancy actuator. In *Submitted to 23rd IAR Workshop on Advanced Control and Diagnosis*, 2008a. [submitted].

J. Davies, T. Steffen, R. Dixon, R. M. Goodall, A. C. Zolotas, and J. Pearson. Modelling of high redundancy actuation utilising multiple moving coil actuators. In *Proceedings of the IFAC World Congress 2008*, Jul 6-11 2008b.

Kalyanmoy Deb. *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley & sons Ltd, 2001.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

J. Dreo, P. Siarry, A. Petrowski, and E. Taillard. *Metaheuritics for Hard Optimization*. Springer-Verlg Berlin Heidelberg, New York, 2006. ISBN 3-540-23022-9.

P. J. Fleming and R. C. Purshouse. Evolutionary algorithms in control systems engineering: A survey. *Control Engineering Practice*, 10(11):1223–1241, 2002.

J. I. Hidalgo and F. Fernandez. Balancing the computation effort in genetic algorithms. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2:1645–1652 Vol. 2, 2005.

K. Michail, C. A. Zolotas, R. M. Goodall, and J. Pearson. Maglev suspensions - a sensor optimisation framework. *16th Mediterranean Conference on Control and Automation*, pages 1514–1519, 2008.

R.J. Patton. Fault-tolerant control systems: The 1997 situation. In *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, volume 3, pages 1033–1054, 1997.

T. Steffen, R. Dixon, R. M. Goodall, and A. Zolotas. Multi-variable control of a high redundancy actuator. In *Actuator 2008 – International Conference and Exhibition on New Actuators and Drive Systems – Conference Proceedings*, pages 473—476. HVG, 2008. ISBN 3-933339-10-3.

Q. Zhao and J. Jiang. Reliable state feedback control system design against actuator failures. *Automatica*, 34 (10):1267–1272, 1998.

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.