A solution to the Reduced Structure Control problem

Amin Nobakhti

Abstract— **The Reduced Structure Controller design problem was previously defined in [1]. Thereof, a two-stage process for the design of the forementioned controllers were proposed. In this paper, a single-stage, Genetic Algorithm approach to solving the problem is proposed.**

I. Introduction

Installation, wiring, maintenance and training costs in addition to factors such as the requirement for an analog realization of the controller, means that in industry, there are considerable pressures to keep the structure of multivariable controllers simple, not just by having low order cross-couplings, but as few of them as possible. However, whilst there are numerous techniques to address model order reduction of multivariable controllers, other than the insufficient centralsied/decentelaised classification, there is little to address the actual structure of the controller, as in, for example the number, and configuration of its cross-couplings. The few that there are, have the reverse problem of only focusing on the cross-couplings. The reduced structure concept is a crude, nonetheless potentially rewarding, attempt at addressing this void. It serves two purposes; first, in addition to the centralised and decentralised, it proposes three additional class of controllers, and secondly offers a single platform from which *both* the issue of the order of the controller and its number of sub-controllers may be addressed simultaneously. In this paper, a solution is proposed for the problem of designing one of these three classes; namely the Basic-RS class. The general RDS problem is defined as such,

Given a plant and some level of desired performance criteria, what is the simplest controller structure which will deliver that?

The definition is very generic, since it does not require a prior determination of what the performance criteria is. For example, it could be some time-domain feature, such as ISE, overshoot, settling time, or frequency based ones, such as bandwidth, peak values, or indeed some economic or maintenance index. This freedom to choose the object of design is particularly attractive because whereas in industry there is a unified need to simplify the controller structures, each industry, and for that matter, each installation will have its own particular design objective. At this point the RDS terminology is recalled. Note that these represent a more generalized set, than those presented in [1].

If $C_t(s)$ is the controller Template, and $C_x(s)$ is a sub-controller whose structural complexity is upper bounded by $C_t(s)$, then

$$C_x(s) \text{ is Basic-RS if } \quad \begin{cases} C_{x_{ij}}(s) = 0, \quad or \\ C_{x_{ij}}(s) = Ct_{ij}(s) \end{cases}$$

$$C_x(s) \text{ is RS if } \quad \begin{cases} C_{x_{ij}}(s) = 0, \quad or \\ C_{x_{ij}}(s) = \text{constant}, \ or \\ C_{x_{ij}}(s) = C_{t_{ij}}(s) \end{cases}$$

$$C_x(s) \text{ is RDS if } \quad \begin{array}{l} \text{The dynamics of} C_{x_{ij}}(s) \\ \text{is upper bounded by } C_{t_{ij}}(s) \end{array}$$

To close this section an example will be given to illustrate these various controller classifications. Consider the case where the template controller is chosen to be,

$$C_t(s) = \begin{pmatrix} \frac{\alpha_0+\alpha_1 s}{\beta_0+\beta_1 s+\beta_2 s^2} & \beta_3 \\ \frac{\alpha_2}{\beta_4+\beta_5 s^2} & \frac{\alpha_3+\alpha_4 s+\alpha_5 s^2+\alpha_6 s^3}{\beta_6+\beta_7 s+\beta_8 s^2+\beta_9 s^3} \end{pmatrix} \qquad (1)$$

An example of a Basic-RS controller based on this template would then be,

$$C_{B-RS}(s) = \begin{pmatrix} 0 & \beta_3 \\ \frac{\alpha_2}{\beta_4+\beta_5 s^2} & 0 \end{pmatrix}. \qquad (2)$$

An example of a RS controller,

$$C_{RS}(s) = \begin{pmatrix} \frac{\alpha_0}{\beta_0} & 0 \\ \frac{\alpha_2}{\beta_4+\beta_5 s^2} & \frac{\alpha_3}{\beta_6} \end{pmatrix}, \qquad (3)$$

and finally an example of a RDS controller would be,

$$C_{RDS}(s) = \begin{pmatrix} \frac{\alpha_0+\alpha_1 s}{\beta_0+\beta_1 s} & 0 \\ \frac{\alpha_2}{\beta_5 s^2} & \frac{\alpha_3+\alpha_4 s}{\beta_6+\beta_7 s+\beta_8 s^2+\beta_9 s^3} \end{pmatrix}. \qquad (4)$$

II. The Genetic formulation of the RDS controller design problem

The appeal of suing Genetic Algorithms to solve this problem is the possibility of having a single-stage design problem. Namely; in [1] it was proposed to design the *centralised* controller via some other technique, then evoke the proposed tools to end up with the RDS controller. However with Genetic Algorithms being optimisation algorithms, provided the RDS problem can be expressed as an optimisation, then a single stage design problem is feasible.

Consider the case where $C_t(s) \in \mathbf{C}^{m\times m}$ is a centralised controller being designed for the plant $G(s) \in \mathbf{C}^{m\times m}$. Let for the sake of this example, the chosen performance index be the ISE and assume that the optimal $C_t(s)$ can achieve and ISE of α. That is to say, for a controller with similar structure to C_t, no ISE lower than α is possible. Now, let the problem be that of designing an RDS controller for this plant, where $C_t(s)$ is chosen to be the template, and β is the worst ISE we are prepared to tolerate. Note, that since $C_t(s)$

The author is with the Control Systems Centre, The University of Manchester, Manchester M60 1QD, UK. This work is supported by EPSRC under grant GR/S91741/01. E-mail: a.nobakhti@ieee.org

is the template, no other controller is expected to produce an ISE lower than α, hence for all possible RDS controllers, it holds true that $\beta > \alpha$, meaning $\beta = \alpha + \Delta$. It is seen that Δ refers to the performance degradation beyond the minimum attainable. We shall refer to Δ as the *performance degradation* level. Now its easy to see how the problem may be set up as a minimization. The RDS controller will be the solution to the following minimization problem,

Minimise the structure of $C_{RDS}(s)$, such that $\Gamma \leq \Gamma_m + \Delta$, where Δ is the maximum performance degradation level, Γ_m is the performance level of the template controller and Γ is that of $C_{RDS}(s)$

Therefore, the problem may now be coded as a genetic optimisation. However, note the striking difference between this, and a typical instance where a controller design problem is posed as an optimisation. Usually, its the controller performance which is the object of the optimisation, and its structure becomes the constraint of the optimisation [2], [3]. In here however, the problem is the reverse; the structure of the controller is the primary object of the minimisation, and the worst case performance is the constraint. In the next section, this problem is solved for the Basic-RS case.

III. Design of Basic-RS controllers

In this paper only the Basic-RS problem is solved. However, this solution provides a framework within which the genetic RDS problem may be solved.

A. Genetic coding of Basic-RS controllers

If the problem is to be solved in one optimisation, simultaneously two tasks need to be carried out. First is to generate various structures contained within the template, and second to optimise each one. However, controllers of difference structures will have different number of optimisation parameters, and since all these controllers must co-exist within the same population, this clearly poses a problem. Standard Genetic Algorithms only allow chromosome lengths of equal size and whilst it is possible to employ special Genetic Algorithms which allow chromosomes of different lengths. However, this class of Genetic Algorithms are very limited in scope and they have not evolved beyond experimental status [4]. It is nonetheless possible to put this problem into a standard Genetic Algorithm , by using so called hybrid chromosomes. Consider the generic template controller,

$$C_t(s) = \begin{pmatrix} c_{11}(s) & \cdots & c_{1m}(s) \\ \vdots & \ddots & \vdots \\ c_{m1}(s) & \cdots & c_{mm}(s) \end{pmatrix}. \tag{5}$$

where,

$$c_{ij}(s) = \frac{N_{ij}(s) = n_{ij1} + n_{ij2}s + \ldots + n_{ijk}s^{k-1}}{D_{ij}(s) = d_{ij1} + d_{ij2}s + \ldots + d_{ijk+l}s^{k+l-1}} \tag{6}$$

Let,

$$\begin{aligned} N_{ij} &= \{n_{ij1}, n_{ij2}, \ldots, n_{ijk}\} \\ D_{ij} &= \{d_{ij1}, d_{ij2}, \ldots, d_{ijk+l}\} \\ \Psi &= \{N_{11}, N_{12}, \ldots, N_{mm}, D_{11}, D_{12}, \ldots, D_{mm}\} \end{aligned}$$

The set Ψ contains the full set of optimisation parameters of the template controller. This set will construct one part of the hybrid chromosome. In the case of the Basic-RS controller, the second part will be a binary string, only used in the decoding process, which will indicate which sub-controllers are active in the Basic-RS controller. Consequently, each chromosome will be of the structure,

$$ind = \{\Upsilon \in \mathbf{R}^{1\times m^2}, \Psi \in \mathbf{R}^{1\times m^2(2k+l)}\} \tag{7}$$

where, the controller will be decoded as such,

$$C_{B-RS} = \begin{pmatrix} \upsilon_1 \frac{N_{11}(s)}{D_{11}(s)} & \cdots & \upsilon_m \frac{N_{1m}(s)}{D_{1m}(s)} \\ \vdots & \ddots & \vdots \\ \upsilon_{m(m-1)+1} \frac{N_{m1}(s)}{D_{m1}(s)} & \cdots & \upsilon_{m^2} \frac{N_{mm}(s)}{D_{mm}(s)} \end{pmatrix} \tag{8}$$

Thus although each Basic-RS chromosome will have the same length, they will decode into structures of different complexities. Furthermore its easy to see how this single-stage design will be achieved. The complexity string itself is part of the chromosome, meaning as with the actual optimisation parameters, its subject to the Genetic Algorithm operators such as mutation and crossover. Before proceeding to the design of the objective function, an example is provided to show how this hybrid chromosome will operate. Consider the case that $C_t(s)$ was a 3×3 PI controller. Then, the chromosome,

$$\begin{aligned} ind = \{&100011100 \vdots 7, 0, 1, 2, 0.1, 0, 1, -0.5, 100, \\ &0.01, 2, 11, 0, 13, -2, 0.1, -7, 17\} \end{aligned} \tag{9}$$

will deconede into the controller,

$$C(s) = \begin{pmatrix} 7 & 0 & 0 \\ 0 & \frac{100s+0.01}{s} & \frac{2s+11}{s} \\ \frac{13}{s} & 0 & 0 \end{pmatrix}. \tag{10}$$

B. Objective function evaluation

Inevitably, the cost function for this problem will involve some sort of comprise. Clearly, the controller with the highest performance, will *not* be 'best' controller. That, will be the controller with the smallest structure which satisfies the worst case performance condition. Thus, the penalty each controller receives must not only reflect its performance but also its structure. The relative weighting of this will be the compromise. i.e. hypothetically speaking, how much performance loss is acceptable per 'unit' of structural loss. In the Basic-RS case, the smallest unit will be a whole sub-controller. In the RDS case for example, this unit will be a single coefficient. One of the key advantages of using the Evolutionary Algorithms , is that the algorithms are problem-independent and need not know what they are actually optimising. All the Evolutionary Algorithm needs to know about two chromosomes, is that in the eventual environment or surface that the phenotypes will be reconstructed in, which one will correspond to a fitter (i.e. 'better') individual. This feature is very facilitating in the RS and the RDS optimisation problems, because all that has to be formulated is a systematic way of 'ranking' the individuals in the population in terms of their fitness. For the design of the Basic-RS controller, the following rank-based objective function is proposed.

```
  for ind=1,...,n.
    Step 1. Decode c using the complexity string, $\Upsilon_{ind}$ and
            the parameter string, $\Psi_{ind}$. Let $C_{ind}(s)$ be the
            resulting controller.
    Step 2. Close the loop with $C_{ind}(s)$ and calculate the
            error with this controller. Let this be $e_{ind}$.
    Step 3. If $e_{ind} > (1+\Delta)e_{min}$, let, $\theta_{ind} = 1$, else $\theta_{ind} =$
            $0$.
    Step 4. Let $\nu_{ind} = \sum\{\Upsilon_{ind}\}$.
    Step 5. Let $\Phi_{ind} = \{\theta,\ \nu_{ind},\ e_{ind}\}$
  end for loop
    Step 1. Stack all individuals that have $\theta = 1$ below those
            which have $\theta = 0$. Inside each group, sort the
            individuals based on $\nu$, such that the individuals
            on top have $\min(\nu)$ and those at the bottom have
            $\max(\nu)$. In each level of $\nu$, sort individuals in
            order of increasing $e$.
    Step 2. Assign each individual a cost value based on its
            place in the population. Namely if the $ind$ is the
            $m^{th}$ individual after the ranking, its fitness will be
            $\Gamma_{ind} = m$.
```

Remark 1: The error may be any calculatable expression related to the overall response of the system. Once it is chosen what the performance is, Δ represents the maximum degradation of that performance index which will be acceptable.

Remark 2: The set Φ_{ind} is called the *performance set* of *ind.* Individuals which have a $\theta = 1$ in their performance set, are those which fail to satisfy the maximum tolerance constraint.

Example 1: To see how this will work, consider the following five individuals as candidate Basic-RS, PI controllers for a plant $G(s) \in \mathbf{C}^{2\times 2}$.

$$\begin{aligned} ind_1 &= \{1,1,1,1,PI_{11},PI_{12},PI_{21},PI_{22}\} \\ ind_2 &= \{1,0,1,1,PI_{11},*,PI_{21},PI_{22}\} \\ ind_3 &= \{1,0,0,0,PI_{11},*,*,*\} \\ ind_4 &= \{1,0,0,1,PI_{11},*,*,PI_{22}\} \\ ind_5 &= \{1,0,0,1,PI_{11},*,*,PI_{22}\} \end{aligned}$$

Where the * symbol denotes a 'don't care' value, since it's corresponding decision variable is zero. Let the chosen performance index be the closed-loop ISE. Assume that the resulting ISE of these controllers are calculated and are as follows,

$$\{e_{C_1} = 1.1, e_{C_2} = 3, e_{C_3} = 10^7, e_{C_4} = 1.2, e_{C_5} = 2\} \quad (11)$$

Assume that e_{min} for the ISE is determined to be 1 and $\Delta = 1$, that is to say the minimum attainable ISE for $G(s)$ with 'any' controller is 1 and that up to a 100% degradation in performance may be acceptable. From this information the performance set of each controller may be constructed following the algorithm just presented as,

$$\begin{aligned} \Phi_1 &= \{0,\ 4,\ 1.1\} \\ \Phi_2 &= \{1,\ 3,\ 3\} \\ \Phi_3 &= \{1,\ 1,\ 10^7\} \\ \Phi_4 &= \{0,\ 2,\ 1.2\} \\ \Phi_5 &= \{0,\ 2,\ 2\} \end{aligned}$$

The controllers may then be ranked, as, prescribed earlier. First they are separated based on θ, then on order of ν_{ind}, and finally on order of e_{ind}. The subsequent rank based cost is also shown,

$$\begin{aligned} \Phi_4 = \{0,\ 2,\ 1.2\} &\Rightarrow \Gamma_4 = 1 \\ \Phi_5 = \{0,\ 2,\ 2\} &\Rightarrow \Gamma_5 = 2 \\ \Phi_1 = \{0,\ 4,\ 1.1\} &\Rightarrow \Gamma_1 = 3 \\ \Phi_3 = \{1,\ 1,\ 10^7\} &\Rightarrow \Gamma_3 = 4 \\ \Phi_2 = \{1,\ 3,\ 3\} &\Rightarrow \Gamma_2 = 5 \end{aligned}$$

Clearly the individual with the 'smallest' error, $C_1(s)$, did not end up as the 'fittest' or the best individual. The fittest was $C_4(s)$ which was the controller which achieved the lowest cost with the smallest structure.

The reader can easily see that under this scheme isomorphism does not exist. It is however, possible, to use the above algorithm to establish an objective function which obtains isomorphism between a given ind and a fixed, comparable, and absolute cost function value. The advantages of this are that it will allow the Evolutionary Algorithms to converge faster, because the ranking does not conceal genetic improvements. Consider in the given example, the controller $C_4(s)$ was replaced by a controller $C_{4a}(s)$ with a performance set $\Phi_{4a} = \{0, 2, 1.9\}$. This controller, would also end up having a cost of 1, although it clearly corresponds to a controller with inferior performance to $C_4(s)$. This means that as far as the Evolutionary Algorithm is concerned, the two controllers have the same fitness and it has no incentive to either choose the better one, or indeed further improvements. Thus to overcome these potential problems, an approach is proposed where there exists a one-to-one mapping. This is an extension of the ranking based approach proposed, with the difference being that each 'rank' of controller, as in, each level of structure, is assigned a cost higher than the total possible cost which a controller of the lower structure.

```
for ind=1,...,n.
    Step 1. Calculate the performance set $\Phi_{ind}$ as outlined earlier.
    Step 2. Let,
        $\Gamma C_{ind} = e_{ind}$
        $\Gamma R_{ind} = e_{min} + \Delta$
        $\Gamma F_{ind} = \frac{1}{1+\nu_{ind}} + \alpha e_{ind}$
        The cost of ind is then calculated as,
        $\Gamma_{ind} = \nu_{ind}(\Gamma C_{ind} + \Gamma R_{ind})(1-\theta) + \theta\eta\Gamma F_{ind}$
end for loop
```

Remark 3: The parameter η is a very large value such that there will be a large difference between controllers which satisfy the performance constraint and those which don't.

Remark 4: The parameter α controls the levels of priority for individuals which violate the performance criteria. If this is small, an infeasible individual's structural order becomes the dominant part of the cost, but if this is large, its error will. Using the α parameter will help to significantly reduce the optimisation time in the following way: until an individual is found which satisfies this constraint, it will force the algorithm to increase the complexity of the controller, and subsequently reduce the error. This has the advantage that if the initial performance specifications are infeasible themselves, this will result in the most optimal full complexity controller. If on the other hand, all infeasible individuals were simply assigned a cost of infinity, this would face problems because not only it would not give the algorithm any clues as to which 'direction' is feasibility, but also, if feasibility does not exist, one then did not have the option of finding what is the best infeasible individual.

The function first assigns a cost to a controller based solely on its performance. This cost is then biased such that ranking will be achieved. This is done by adding to a controller of structure index ν_{ind}, the maximum possible cost which a controller of a structure $\nu_{ind} - 1$ may achieve. Note that if such a controller with the smaller structure generates a θ of one, then it will be automatically pushed to the bottom end of the stack because of the large parameter η. Thus the controller at the top of the stack will be the one which satisfied the performance criteria and also has the smallest structure.

The operation of this cost function may be illustrated through a simple example. To ease comparison, the same set of controllers as used in the previous example will be used. Let as before $\Delta = 1$ and let $\eta = 10^{10}$, $\alpha = 1$. The cost of each controller may then be calculated as follows,

$$\begin{aligned}
\Gamma_1 &= 4 \times (1.1 + 2.1)(1-0) + 0 \times 10^{10} \times 1/(1+4) + 1 \times 1.1 = 12.8 \\
\Gamma_2 &= 3 \times (3 + 2.1)(1-1) + 1 \times 10^{10} \times 1/(1+3) + 1 \times 3 = 3.25 \times 10^{10} \\
\Gamma_3 &= 1 \times (10^7 + 2.1)(1-1) + 1 \times 10^{10} \times 1/(1+1) + 1 \times 10^7 \approx 10^{18} \\
\Gamma_4 &= 2 \times (1.2 + 2.1)(1-0) + 0 \times 10^{10} \times 1/(1+2) + 1 \times 1.1 = 6.6 \\
\Gamma_5 &= 2 \times (2 + 2.1)(1-0) + 0 \times 10^{10} \times 1/(1+2) + 1 \times 2 = 8.2
\end{aligned}$$

If one lists the above in order of increasing cost, the following set will be obtained,

$$\begin{aligned}
\Gamma_4 &= 6.6 \\
\Gamma_5 &= 8.2 \\
\Gamma_1 &= 12.8 \\
\Gamma_2 &= 3.25 \times 10^{10} \\
\Gamma_3 &\approx 10^{18}
\end{aligned}$$

It is noted that the relative order of each feasible controller has not been changed, that is to say, under the ranking approach proposed previously, the fittest controller was $C_4(s)$, the second fittest one was $C_5(s)$ and so on. What has changed however, is that the cost assigned to each individual is unique and only associable to an individual with that chromosome. For example it was illustrated that under the previous scheme if $C_4(s)$ was replaced $C_{4a}(s)$ with a performance set $\Phi_{4a} = \{0, 2, 1.9\}$, this new controller would generate the same eventual cost as $C_4(s)$. It is easy however to confirm that under this new scheme the cost of $C_{4a}(s)$ will be $\Gamma_{4a} = 8$. Thus, whilst $C_{4a}(s)$ will still be the fittest individual (as was the case previously), this time it is readily seen that it is not as fit as $C_4(s)$ which had a cost of $\Gamma_4 = 6.6$.

C. Example

An example is provided here to illustrate the design of the Basic-RS controller using the proposed approach. The problem being considered is Limbeer's plant, with the following transfer function matrix,

$$G(s) = \begin{pmatrix} \frac{s+4}{s^2+6s+1} & \frac{1}{5s+1} \\ \frac{s+1}{s^2+100s+10} & \frac{2}{2s+1} \end{pmatrix}. \tag{12}$$

For more details on this plant, see [5] The problem set here, is to find a Basic-RS PI controller for this plant. The desired closed-loop response functions are,

$$Sys_R(s) = \begin{pmatrix} \frac{1}{s+1} & 0 \\ 0 & \frac{1}{s+1} \end{pmatrix}. \tag{13}$$

The Basic-RS template is chosen as,

$$C_t(s) = \begin{pmatrix} \frac{\alpha_{11}s+\beta_{11}}{s} & \frac{\alpha_{12}s+\beta_{12}}{s} \\ \frac{\alpha_{21}s+\beta_{21}}{s} & \frac{\alpha_{22}s+\beta_{22}}{s} \end{pmatrix}. \tag{14}$$

Each individual in the Evolutionary Algorithm will have 12 parameters and is of the form,

$$ind = \{d_{11}, d_{12}, d_{21}, d_{22} \vdots \alpha_{11}, \beta_{11}, \alpha_{12}, \beta_{12}, \alpha_{21}, \beta_{21}, \alpha_{22}, \beta_{22}\}$$

where in this case,

$$\alpha_{ij} \in [-3, 3], \quad \beta_{ij} \in [-3, 3], \quad d_{ij} = \{0, 1\} \ \forall \ i, j = 1, 2$$

Each chromosome ind will be decoded into a controller in the following manner,

$$C_{ind}(s) = \begin{pmatrix} d_{11}\frac{\alpha_{11}s+\beta_{11}}{s} & d_{12}\frac{\alpha_{12}s+\beta_{12}}{s} \\ d_{21}\frac{\alpha_{21}s+\beta_{21}}{s} & d_{22}\frac{\alpha_{22}s+\beta_{22}}{s} \end{pmatrix} \quad (15)$$

The performance error for each controller is calculated as follows: let $R(t)$ be the array which contains the time response of $Sys_R(s)$ for time t. Let the time response of the closed loop system with the candidate controller be $Y(t)$. The errors for this candidate controller will then be calculated as,

$$e_{ind} = \sum_{t=0}^{T}\sum_{i,j=1}^{2} \mid r_{ij}(t) - y_{ij}(t) \mid . \quad (16)$$

First, the template is optimised for the time vector $t = \{0, 0.1, \ldots, 6.9, 7\}$ to find e_{min}. The following controller is found,

$$C_f(s) = \begin{pmatrix} \frac{1.217s+1.305}{s} & \frac{-0.1401s-0.4394}{s} \\ \frac{-0.1294s-0.04005}{s} & \frac{1.045s+0.52}{s} \end{pmatrix} \quad (17)$$

which obtains a cost of 1.6061. Thus, it was assumed for the remainder of this section that $e_{min} = 1.7$. Next, the reduced structure methodology presented before will be applied to this system. To illustrate the full operation of the algorithm, Δ will be gradually increased and the effects of this increased preparedness to tolerate worst performance is observed on the structure of the controller. The parameters of the cost function where chosen as follows,

$$\eta = 10^{10} \qquad \alpha = 0.1$$

1) Case 1: $\Delta = 0$: This is the case where there is zero tolerance and thus it is expected that the algorithm finds a full-complexity controller similar to the template controller (Eq. 17). In this case the following controller is obtained,

$$C_{\Delta=0}(s) = \begin{pmatrix} \frac{1.218s+1.309}{s} & \frac{-0.1336s-0.4446}{s} \\ \frac{-0.1415s-0.04277}{s} & \frac{1.046s+0.5203}{s} \end{pmatrix}, \quad (18)$$

which obtains a *total* cost of 13.2187. Of this, 1.6047 is IAE error and the remainder is structure cost of the controller. Note that this controller is in fact marginally better than the template controller optimised for this problem which was given in (Eq. 17), which obtained a cost of 1.6061. However, due to the stochastic nature of the Evolutionary Algorithm [6], [7]it is unrealistic to expect, although theoretically sound, that the $\Delta = 0$ controller and the optimised template controller will be identical. The response of the closed-loop system with this controller is shown in Figure 1.

2) Case 1: $\Delta = 1$: This means the tolerance threshold on the ISE is 2.7. The algorithm found the following controller for this case,

$$C_{\Delta=1}(s) = \begin{pmatrix} \frac{1.199s+1.268}{s} & \frac{-0.1358-0.4391}{s} \\ 0 & \frac{1.048s+0.5209}{s} \end{pmatrix}, \quad (19)$$

with a cost of 10.3947, of which 2.2947 was purely the error cost and the rest was the complexity cost. Notice how the algorithm has found that for the specified tolerance level, an upper triangular controller is able to meet them. This does not mean that a full complexity controller will not meet the tolerance levels, but it means the *simplest* controller structure which is able to satisfy the specifications is an upper triangular one. It is easy to see that for $\Delta = 1$, a full PI controller, if it attains the minimum cost of 1.7, would generate a total error of $1.7 + 4 \times 2.7 = 12.5$ which is higher than the upper triangular controller.

The step response of the closed loop system with this controller is shown in Figure 1. Compared to the full PI controller it is seen that still there is almost exact matching on the diagonal responses, however there is slight increase in interaction in element (2,1), which is as expected since the controller has lost one degree of freedom by changing into an upper triangular structure.

3) Case 1: $\Delta = 2$: For this value of $\Delta = 2$ and for $\Delta = 3$, again, upper triangular controllers similar to the case $\Delta = 1$ were found.

4) Case 1: $\Delta = 4$: The controller found in this case was,

$$C_{\Delta=4}(s) = \begin{pmatrix} \frac{1.218s+1.331}{s} & 0 \\ 0 & \frac{1.02s+0.5265}{s} \end{pmatrix}, \quad (20)$$

which obtained a performance cost of 5.2666 and a total cost of 16.666. It is seen in this case that the controller structure has been reduced to just a decentralised diagonal controller. This indicates that a decentralised controller is not able to obtain a cost of less than 5.266 for this plant, and only once, up to or more that this amount of error, was declared tolerable (i.e. $\Delta + e_{min}$), did it become the solution to the optimisation. It is easily confirmable that for the case $\Delta = 4$, the -near optimal- full controller and the upper triangular controller would generate cost function values respectively of 24.5 and 19.36. The step response of the closed loop system with this controller is shown in Figure 1, where it can be seen that, as expected - the interactions in both channels have increased. However, the diagonal responses still are exactly the same as those of the reference responses also plotted.

To confirm the results, the following table was calculated manually by using the Evolutionary Algorithm to individually optimise the various controller structures. It is seen that the table indeed confirms the results.

Δ	Full PI	Upper Tri	Lower Tri	Diagonal	Best
0	1.666	$\theta = 1$	$\theta = 1$	$\theta = 1$	Full PI
1	12.4	10.36	$\theta = 1$	$\theta = 1$	Upp Tri
2	16.4	13.36	$\theta = 1$	$\theta = 1$	Upp Tri
3	20.4	16.36	18.76	$\theta = 1$	Upp Tri
4	24.4	19.36	21.76	16.6	Diag

It is important to note that the tolerance level *does not* set a priori error level which a controller will be found to satisfy. This is merely the worst case error one is prepared to tolerate. However, frequently the error will be less. For example for the case $\Delta = 1$ the smallest error found was 2.266 which is 0.44 below the threshold of 2.7, similarly for

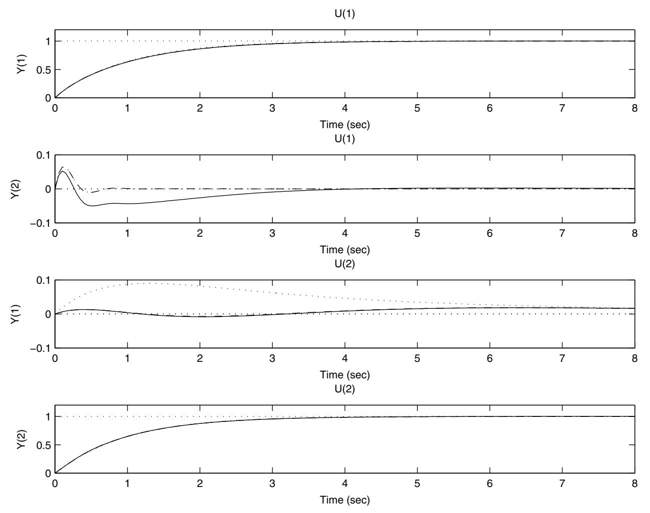

Fig. 1. Closed loop responses. $\Delta = 0$ solid, $\Delta = 1$ dashed, $\Delta = 4$ dotted.

the cases $\Delta = 3$ and $\Delta = 4$, the errors where respectively 1.44 and 0.434 below the threshold tolerance.

An interesting point should be observed at this stage. For no value of Δ, was a lower-triangular controller structure generated. This fact could have also been directly observed from the cost function. Let the cost function of the Upper triangular controller be,

$$\Gamma_{UT} = \nu_{UT}(\Gamma C_{UT} + \Gamma R_{UT})(1 - \theta) + \theta\eta\Gamma F_{UT} \qquad (21)$$

Assume that Δ is such that $e_{UT} < e_{min} + \Delta \Rightarrow \theta = 0$. The case which $\theta = 1$ means the controller is in violation so it doesn't not concern us at this point. Hence the cost of the upper triangular controller will be,

$$\Gamma_{UT} = \nu_{UT}(\Gamma C_{UT} + \Gamma R_{UT}) \qquad (22)$$

Since, for all Δ, $\Gamma R_{UT} = \Gamma R_{LT}$, the only way for the lower triangular controller to be selected is to have $\Gamma_{LT} < \Gamma_{UT}$, it implies that for some Δ, $\Gamma C_{UT} > \Gamma C_{LT}$, which is known to be false. Thus in this particular example the lower triangular controller will never be a solution for any value of Δ.

IV. Concluding Remarks

As part of the RDS design technique, three additional classes of controllers were proposed. In [1] some analytical tools were developed for the design of the so-called Basic-RS and RS controller classes. These tools were however limited. On one hand they reacquired the designer to 'interpret' the results, secondly, and more importantly, they could only be applied to an already existing control system, thereof, they constituted a 'two-stage' design process. The final limiting factor was in their extension to solve the RDS class, where considerable difficulties were experienced. Such problems lead to this work which is approaching the problem from a heuristic, rather than analytical perspective.

The advantages are many fold. First, it minimises the designer's involvement, as there is little to interpret. Second, as demonstrated here, it allows for the possibility of a single-stage design process, and last but not least provides a single framework which when extended could include all the three proposed classes of controllers. Another major advantage of this approach has been, that where as the initially developed tools were only designed for time domain performance indices, in this approach the performance index may be chosen as anything the designer chooses, because the Evolutionary Algorithms only work with pay-off values. In this work however, only the Basic-RS class were addressed, and the design approach was shown on an example. This example effectively demonstrated how the 'one-step' design procedure would work in practice. The designer provides the plant, some performance indicator and its corresponding optimal value, and a threshold above which no more performance degradations will be tolerated. The algorithm would then find the simplest controller which delivers a performance level within the threshold.

In retrospect, unlike the real life examples provided in [8] which showed amazing structure reductions for real life controllers, the example provided here is of little practical significance and its importance doesn't go beyond merely demonstrating the functionality of the algorithm. Although a 2×2 Basic-RS controller, has 12 distinct possible configurations, it is known only 4 of these are practically acceptable. However the real potential of this approach is when one thinks of higher dimension controllers which have thousands of possible configurations, not to mention the RDS case, where each configuration will itself have hundreds of different orders. Thus, in this respect, the worth of this work is more in showing the feasibility of the concept, than any immediate practical gains.

References

[1] A. Nobakhti and N. Munro, "Minimal strcuture control: A proposition," *IFAC Confrence on Control Systems Design (CSD'03)-Slovak Republic*, 2003.
[2] A. Nobakhti, N. Munro, and B. Porter, "Evolutionary achievement of diagonal dominance in linear multivariable plants," *Electronic Letters*, vol. 39, pp. 165–166, 2003.
[3] B. Porter, N. Munro, and A. Nobakhti, "Evolutionary dominance-based design of linear multivariable controllers," *Proc. European Control Conference, Cambridge, UK*, 2003.
[4] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlang, 1994.
[5] D. Limebeer, "The application of generalized diagonal dominance to linear systems satbility theory," *International Journal of Control*, vol. 36, no. 2, 1982.
[6] T. Back, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.
[7] D. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine learning*. Addidon-Wesley, 1989.
[8] A. Nobakhti and N. Munro, "Minimal strcuture control: Some examples," *IFAC Confrence on Control Systems Design (CSD'03)-Slovak Republic*, 2003.