Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Practical Considerations for Override Compensator Synthesis and Implementation ⋆

Phil March * **Matthew Turner** **

* *Department of Engineering, University of Leicester, University Road, Leicester. LE1 7RH, UK (e-mail: pm57@ le.ac.uk).*
** *Department of Engineering, University of Leicester, University Road, Leicester. LE1 7RH, UK (e-mail: mct6@ le.ac.uk).*

Abstract: This paper discusses some of the practical considerations involved in the synthesis and implementation of override compensators for systems which have outputs constrained to lie below certain thresholds. The paper assesses three different override architectures: the first, a generic override control scheme from the literature; the second, a sub-class of the first which offers easier tuning and implementation at the expense of flexibility; and the third, a new modification of the generic scheme which is more flexible than the second, but of similar complexity. The various schemes are demonstrated and compared using a simulation case-study of a permanent-magnet-synchronous-motor speed control system.

1. INTRODUCTION

Override compensators belong to the class of control strategies which are not used for primary control but are "retro-fitted" to existing controllers in order to enhance their performance or capabilities in some way. Perhaps the best known example of these retro-fitted control elements are the so-called anti-windup compensators which are added to a control system to deliver better performance and stability properties when actuator saturation is thought to be a problem. In a similar way, override compensators are retro-fitted to controllers which may be good in enforcing performance requirements on a certain set of "primary" controlled outputs, but in the process of doing so, allow a set of "secondary" outputs to stray outside some pre-specified set of limits. The function of the override compensator is to ensure that these secondary output limits are violated as little as possible, while ensuring the goals of the existing baseline controller are affected as little as possible. A good introduction to override compensation is provided in Glattfelder et al. [2004].

There are a number of different override compensation methods available in the literature ranging from simple ad-hoc methods born out of early industrial applications, for example in Glattfelder et al. [2003], to more sophisticated and powerful modern methods born out of optimal control theory. Many of the simple ad-hoc methods are intuitive and can essentially be designed by hand, although their application can be limited to relatively simple problems. In some of these early procedures, nonlinear stability is also not considered in the design phase and, rather, is ascertained purely *a postiori*. In contrast some of the modern methods, for instance in Turner et al. [2002a], encompass a much larger set of override compensator configurations and nonlinear stability is catered for directly in the design algorithms and systematic synthesis routines have been devised to simplify the design process.

⋆ This work was supported by the Engineering & Physical Sciences Research Council and TRW Automotive Ltd.

Despite this, these recently introduced modern methods, suffer from greater complexity than their ad hoc counterparts, which can make tuning more difficult and impede implementation. In particular, the method discussed in Turner et al. [2002a] advocates feeding override signals directly into the controller state equation, restricting its application to modern controllers in which the state-space realisation is known. Furthermore this method contains little guidance on which override signals play the more important role in constraining the secondary outputs from exceeding their limits, which may complicate tuning.

The aim of this paper therefore is to address some of the practical issues associated with the design, tuning and implementation of override compensators. In particular a new modification of the architecture proposed in Turner et al. [2002a] is developed, which is thought to have advantageous properties with respect to implementation. With the aid of a permanent-magnet-synchronous-motor (PMSM) simulation case study, it is shown how the choice of override architecture can considerably influence the design and performance of the compensator.

2. GENERIC OVERRIDE CONTROL PROBLEM

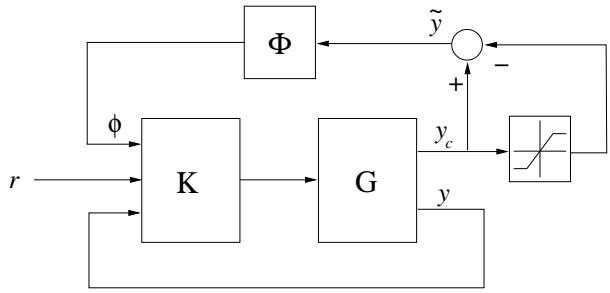

Fig. 1. Override compensation generic framework

A generic override control system is depicted in Figure 1. In this diagram, $G(s)$ represents the plant to be controlled with $y \in \mathbb{R}^{n_y}$ representing the set of measured outputs used by the baseline controller, $K(s)$, and $y_c \in \mathbb{R}^q$ representing the

set of outputs which we wish to limit. The plant's state-space realisation is given by

$$G(s) \sim \left\{ \begin{array}{rcl} \dot{x}_p &=& A_p x_p + B_p u + B_{pd} d \\ y &=& C_p x_p + D_p u + D_{pd} d \\ y_c &=& C_{pc} x_p + D_{pc} u + D_{pdc} d \end{array} \right. \quad (1)$$

In this realisation, $x_p \in \mathbb{R}^{n_p}$ denotes the plant state, $d \in \mathbb{R}^{n_d}$, the disturbance acting on the plant and $u \in \mathbb{R}^m$, the control signal.

The linear baseline controller is assumed to have been designed for some primary performance objectives and is also assumed to have been designed such that the standard linear closed loop in Figure 1 (when $\Phi \equiv 0$) is internally stable. The controller's state-space realisation is given by

$$K(s) \sim \left\{ \begin{array}{rcl} \dot{x}_c &=& A_c x_c + B_{cr} r + B_c y + \phi_1 \\ u &=& C_c x_c + D_{cr} r + D_c y + \phi_2 \end{array} \right. \quad (2)$$

where $x \in \mathbb{R}^{n_c}$ is the controller state and $r \in \mathbb{R}^{n_r}$ is the reference. The signal $\phi = [\phi_1' \ \phi_2']'$ is produced by the override controller, $\Phi(s)$. As in Glattfelder et al. [2003] and Turner et al. [2002a], it is assumed that this signal remains zero unless the constrained output vector y_c violates one of its limits. Violation of output constraints is detected by comparing y_c to its saturated version, $\mathrm{sat}(y_c)$, as shown in Figure 1; if there is a difference the resulting signal $\tilde{y} = \mathrm{Dz}(y_c) := y_c - \mathrm{sat}(y_c)$ will be non-zero. When $\tilde{y} \neq 0$, this then elicits a reaction from $\Phi(s)$, hopefully to drive the signal $\tilde{y}$ to zero again swiftly. Note that the presence of the saturation (or deadzone) makes the problem nonlinear and care must be taken in making statements about stability and performance.

Remark 1. Note that to activate $\Phi(s)$, a limit must have been exceeded i.e. at least one element of $\tilde{y}$ must be non-zero. This implies that the override scheme is mainly useful for "soft" output limits in which limits can be exceeded slightly for small periods of time. As $\Phi(s)$ is linear it is not possible to ensure y_c remains below its threshold for all time, in general. However, this difficulty can be circumvented by choosing an artificial threshold to be a little lower than the true threshold, allowing the override controller to be activated before the real threshold is reached. Alternatively, for high performance applications, one might select a nonlinear override control scheme as advocated in Herrmann et al. [2007].

The design philosophy behind override control, then, is firstly to design a baseline controller for which the unconstrained system satisfies the primary performance objectives, and then augment the system with an override compensator to assist the system in respecting certain output constraints. This can be summarised as

Goal 1. To design $\Phi(s)$ such that

(1) The system in Figure 1 is globally asymptotically stable.
(2) If $\tilde{y}(t) = 0 \quad \forall t \geq 0$, then $\phi(t) = 0 \quad \forall t \geq 0$.
(3) $\|\tilde{y}\|_2 < \gamma \|[r' \ d']'\|_2$

Roughly speaking any $\Phi(s)$ which achieves the objectives of Goal 1 will ensure the overall closed loop system is stable, affected by the override controller only when constrained output limits are violated, and that the $\mathcal{L}_2$ gain between the constrained outputs and their saturated counterparts will be small.

Typical steps of an override compensator design process are as follows:

(1) Design a controller which satisfies the primary performance objectives
(2) Select additional plant outputs to constrain, and build saturation models
(3) Choose an appropriate override architecture
(4) Synthesise an initial compensator design
(5) Tune the design with the aid of simulation testing
(6) Implement the design in hardware for practical testing

Note that in the steps above, design and implementation are essentially treated as entirely separate processes. In addition, initial testing is often carried out in continuous time, leaving the practicalities of discretisation to be considered only at the stage of implementation. The ability to treat design and implementation separately is desirable but this property is not provided by all compensator architectures as will be revealed in the next section.

3. OVERRIDE CONTROL ARCHITECTURES

3.1 Architecture 1

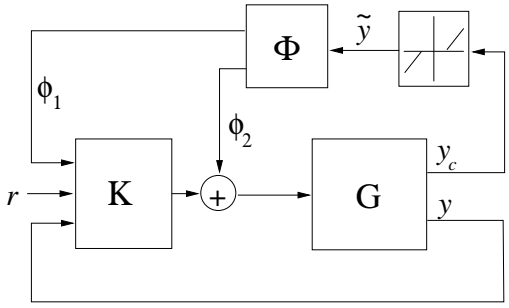

Fig. 2. Override Architecture 1

The first override design architecture under consideration is that depicted in Figure 1 which is the most general linear override control architecture. It can be equivalently represented as shown in Figure 2. Note that the override compensator is given the ability to manipulate the state equation of the controller and the controller output directly, as in (2), giving it maximal freedom in ensuring output limits are violated as little as possible.

This architecture is discussed extensively in Turner et al. [2002a] and algorithms are given there which enable a compensator $\Phi(s)$ to be synthesised such that the following $\mathcal{L}_2$ gain bound holds

$$\left\| \begin{bmatrix} W_1^{1/2} \tilde{y} \\ W_2^{1/2} \phi \end{bmatrix} \right\|_2 < \gamma \left\| \begin{bmatrix} r \\ d \end{bmatrix} \right\|_2 \quad (3)$$

In the above inequality, $W_1 > 0$ and $W_2 > 0$ are positive definite weighting matrices chosen by the designer to trade-off the importance of various signals in the optimisation process: W_1 is used to shift weight between particular constrained outputs and W_2 is used to limit override control activity in given channels. Note that in this case, $\phi \in \mathbb{R}^{n_c+m}$, making $W_1 \in \mathbb{R}_+^{q \times q}$ and $W_2 \in \mathbb{R}_+^{(n_c+m)\times(n_c+m)}$. Typically both W_1 and W_2 are chosen as diagonal matrices, requiring $q + n_c + m$ design parameters to be chosen.

3.2 Architecture 2

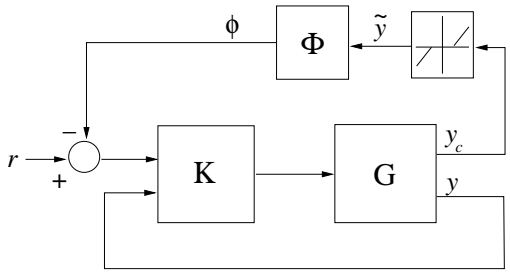

Fig. 3. Override Architecture 2

Figure 3 depicts the second override architecture in which the override controller has restricted freedom in how it can influence the controller, $K(s)$. It is now restricted to being allowed to "back-off" the reference. Although this architecture sacrifices flexibility when compared to Architecture 1, it is often favoured in practice due to its simplicity in both implementation and in tuning. Architectures 1 and 2 are related by noting that, effectively $\phi_2 \equiv 0$ in Architecture 2, making the two ϕ's related by

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}_{arch1} = \begin{bmatrix} B_{cr} & 0 \\ D_{cr} & 0 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}_{arch2} \tag{4}$$

in (2). In this case $\phi \in \mathbb{R}^{n_r}$ and hence the weighting matrix $W_1 \in \mathbb{R}_+^{n_r \times n_r}$ will often be of a lower dimension than that in Architecture 1. Note that the advantage of this architecture compared to Architecture 1 is that *no knowledge* of the state-space realisation of $K(s)$ is required. Providing the transfer function of $K(s)$ is known, $\Phi(s)$ can be designed using any realisation because the override signal is not injected into the state equation. This architecture may also be preferred because the override control contribution is clearly visible from how much the reference is "backed off".

3.3 Architecture 3

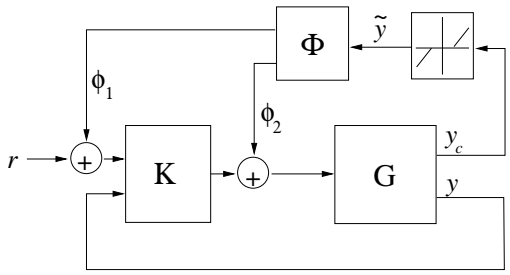

Fig. 4. Modified Override Architecture

We now propose a third compensation architecture in which the override compensator is given authority to modify the reference signal and the controller output (Fig. 4). This is a special case of of Architecture 1, viz. (5), and could perhaps be described as a middle ground between Architectures 1 & 2. It offers simpler tuning and implementation than Architecture 1, particularly for application to high order controllers, whilst providing increased flexibility in the design compared to Architecture 2. The architecture can be seen to be a special case of Architecture 1 by making the substitution

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}_{arch1} = \begin{bmatrix} B_{cr} & 0 \\ D_{cr} & I \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}_{arch3} \tag{5}$$

In this case, $\phi \in \mathbb{R}^{n_r+m}$ and thus less freedom is apparent when compared to the original architecture (assuming $n_r < n_c$), but obviously significantly more is available compared to Architecture 2. Note once again, that as the override signals are injected at the input and output of the controller, respectively, it is possible to use this architecture without injecting signals into the controller state equation, making it similarly attractive, in terms of implementation, to Architecture 2.

3.4 Static and dynamic variants

As in many control applications, it is always desirable to keep the number of states in a controller to a minimum if possible. In override control, the first LMI-algorithm provided in Turner et al. [2002a] allows the construction of a *static* override controller i.e. Φ is simply a static matrix. In practice however, this may not be desirable due to

i) Noise on the constrained outputs, $y_c(t)$, causing spurious activation of the override controller Φ.
ii) Inadequate frequency shaping. Stability may be provided, but performance could still be poor.

Therefore, it is often preferable, particularly in practice, to include filters in the override controller. Typically these take the form of 'output' filters, for filtering out noise present on the constrained plant outputs, and 'input' filters, to shape the frequency content of the override signal, ϕ. The dynamics of the output filter are absorbed into the linear plant model, G, and hence do not alter the structure of the design problem. The dynamics of the input filter form part of the compensator according to $\Phi(s) = \tilde{\Phi}(s)k_\phi$ where $\tilde{\Phi}(s)$, our input filter, is a transfer function matrix chosen by the designer and k_ϕ is a matrix gain to be synthesised. Although the structure of the problem is different for static and dynamic compensation, the same performance cost function is optimised, albeit in a different realisation.

Note that these filters can be included in every architecture discussed here in essentially the same manner. Frequently, both input and output filters can be chosen as diagonal transfer function matrices of appropriate dimensions. By ensuring the DC gain of the filter is unity, only the bandwidths and filter type need to be chosen. First order low pass filters are usually a good choice.

3.5 Tuning Parameter comparison

In choosing an architecture for override control, a number of factors are relevant. Of these, tuning parameters are particularly important as they influence the design flexibility and complexity. Table 1 shows a comparison of tuning parameters for the dynamic compensators, assuming input and output filters are chosen as transfer functions of the form

$$F(s) = \text{diag}\left(\frac{a_1}{s+a_1}, \frac{a_2}{s+a_2}, \dots\right) \tag{6}$$

and the weights W_1 and W_2 are chosen to be diagonal.

Note that Table 1 only gives typical values and, of course, the order and structure of the filters can be arbitrary.

OR Arch.	Size of W_1	Size of W_2	Compensator order	Tuning parameters
1	q	$n_c + m$	$n_c + m + q$	**2**$(n_c + m + q)$
2	q	n_r	$n_r + q$	**2**$(n_r + q)$
3	q	$n_r + m$	$n_r + m + q$	**2**$(n_r + m + q)$

Table 1. Compensator states and tuning parameters

3.6 Application to multivariable systems

All three architectures can be applied to multivariable control systems but the merits of each differ. Architecture 1 has maximum versatility since it is able to manipulate each control input independently. Thus, the compensator need only modify the control inputs necessary to bring the saturating channel out of saturation. This should assist in minimising the effect of the override compensation on nominal control performance.

With Architecture 2, the compensator makes use of the decoupling already present in the controller. If this decoupling is compatible with the override objective, this can be a very computationally efficient way to tackle a multivariable override problem. However, if the decoupling of the controller causes the compensator to influence more channels than required for bringing the system out of saturation, this approach could unnecessarily degrade performance. This problem is expected particularly in systems with more control inputs than reference inputs since two or more channels must be influenced by one reference signal.

Architecture 3 works in primarily the same manner as Architecture 2 in that the compensator makes use of the existing decoupling present in the nominal controller. However, the extra flexibility in being able to modify the control inputs directly adds a degree of freedom to modify the decoupling. This could assist performance for systems where the decoupling of the nominal controller is not compatible with override compensation.

3.7 Implementation Considerations

For practical application of an override control system, the state-space realisation used for implementing the controller and compensator can become an important consideration. This is particularly true when fixed point arithmetic is used since the speed and accuracy of computation can be adversely affected by a poorly optimised controller realisation. With fixed point arithmetic, scaling factors are applied to each constant, signal and state, dictating the allowable range of each signal and state, and also the resolution of the fixed point number representation. Thus, altering the controller realisation by the application of state similarity transformations can enable the designer to find the optimum balance between resolution and range.

Override compensator designs according to Architecture 1 are dependent upon the controller realisation. This means that if the controller realisation is changed at the stage of implementation, the override compensator will need to be re-designed. This may then require the state-space realisation of the compensator and the associated scalings to be optimised. Architectures 2 and 3 are independent of the controller realisation used. Thus, provided that an override compensator of architecture 2 or 3 provides adequate performance, their use can significantly reduce the complexity of implementation on a fixed point processor.

4. CASE STUDY: APPLICATION TO CURRENT LIMITATION IN A PMSM SPEED CONTROL SYSTEM

We consider the application of override compensation to limit the current in the Permanent Magnet Synchronous Motor (PMSM) speed control system of March et al. [2007] and March et al. [2008]. This control system adopts a Field-Oriented Control (FOC) strategy in which voltages and currents are described using two axis vector notation where the two axes, d and q, are orthogonal. For a background on PMSM control please consult Krishnan [2001] and Novotny et al. [2000].

For the purpose of override compensator design we consider the controller, K, to be a linear SISO state space model which generates the signal, iq_dmd, which is the q-axis current demand. This is used by a current controller as a reference for current regulation. The plant, G, is considered to be the nonlinear closed loop system consisting of the PMSM model, current controller and phase advance controller. The plant is linearised to give a linear model, $G(s)$, according to the state-space description of Section 2 with input iq_dmd and outputs $y = w_m$ and $y_c = [i_d \; i_q]'$. We consider an experimental second order controller, $K(s)$, which is essentially a low-pass filtered PI design. Plant and controller matrices are given in the appendix.

The signal that we wish to constrain is the magnitude of the current flowing in the motor. This translates to constraining the euclidean norm of the plant states i_d and i_q, which are d and q-axis components of the current vector, to the limit, I_{max}, that is we require

$$\left\| \begin{bmatrix} i_d \\ i_q \end{bmatrix} \right\| = \sqrt{i_d^2 + i_q^2} \leq I_{max} \tag{7}$$

Since this constraint is a nonlinear function of the states it is incompatible with the synthesis procedures discussed earlier which require an element-wise representation of the saturation function. However, by defining the time-varying current limit on i_q as

$$\bar{i_q}(i_d) = \begin{cases} \sqrt{I_{max}^2 - i_d^2} & |i_d| \leq I_{max} \\ 0 & |i_d| > I_{max} \end{cases} \tag{8}$$

Then instead of using a fixed limit saturation function in the override control scheme, it follows that use of the above time-varying limit in the saturation function can be used instead: $\mathrm{sat}_{\bar{i_q}(i_d)}(y_c)$. This scalar limit can then be used to construct the override compensated closed loop system of Figure 5.

4.1 Performance comparisons

Figures 6, 7 and 8 show the override performance of the three compensator architectures applied to the linear control system

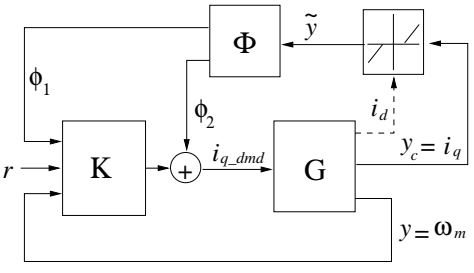

Fig. 5. Override case study block diagram

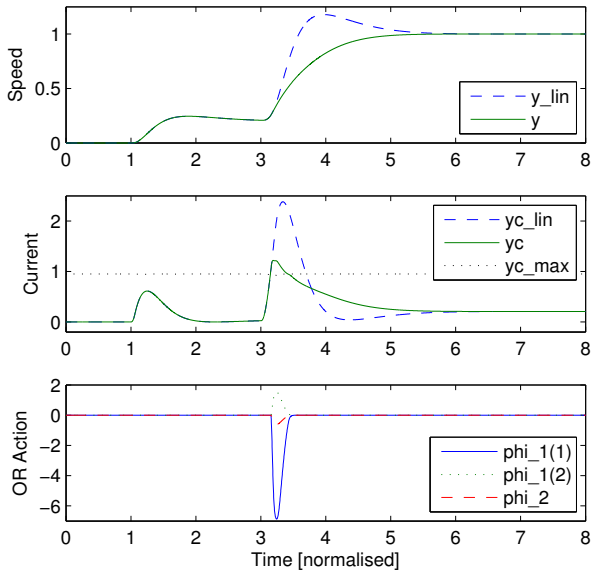

Fig. 6. Linear simulation with OR Architecture 1

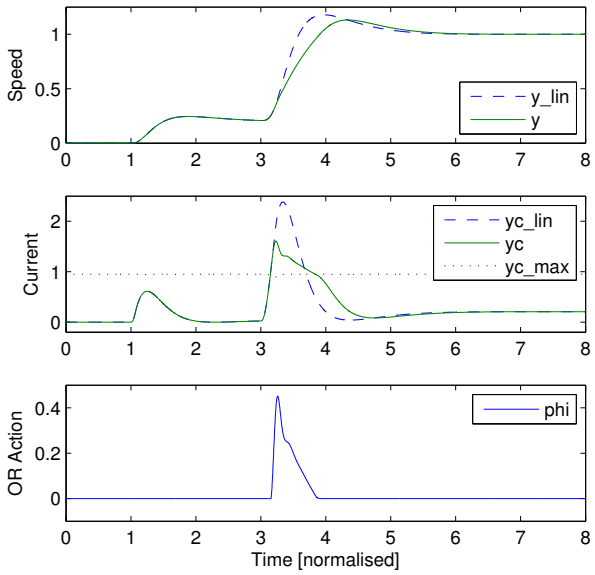

Fig. 7. Linear simulation with OR Architecture 2

model. For consistency in comparison, all input filters were chosen to be first order low-pass with a bandwidth of 1kHz, and the output filters were chosen to be first order low-pass with a bandwidth of 2.5kHz. The weighting matrix W_1 was chosen to be unity, and the elements of W_2 were chosen to be in the order of $1e{-}6$. However, to exploit the potential of the different architectures, some flexibility was required in W_2. Therefore, for Architecture 1, $W_2 = 1e{-}6$, but for Architecture 2, $W_2 = diag(1e-6, 1e-6, 1e-4)$, and for Architecture 3, $W_2 = diag(1e-6, 1e-6)$.

Architecture 2 achieves a significant reduction in the violation of the output limit compared to the nominal controller, although the limit is still exceeded by approximately 60%. Three states are required, two for the output (noise) filter and one for feedback to the reference demand. Architecture 3 has the additional flexibility of modifying the control signal directly with the effect that a reduced $\mathcal{L}_2$ gain bound, γ, is achieved (Table 2). Feedback directly to the control signal requires one extra state but allows the compensator to respond faster, reducing the severity of output violation to 37%. Architecture 1 requires the most states since it has authority to modify all controller states and outputs but this allows a significantly lower $\mathcal{L}_2$ gain bound to be achieved, resulting in much tighter control of the constrained output. Output violation in this simulation is reduced to 22%.

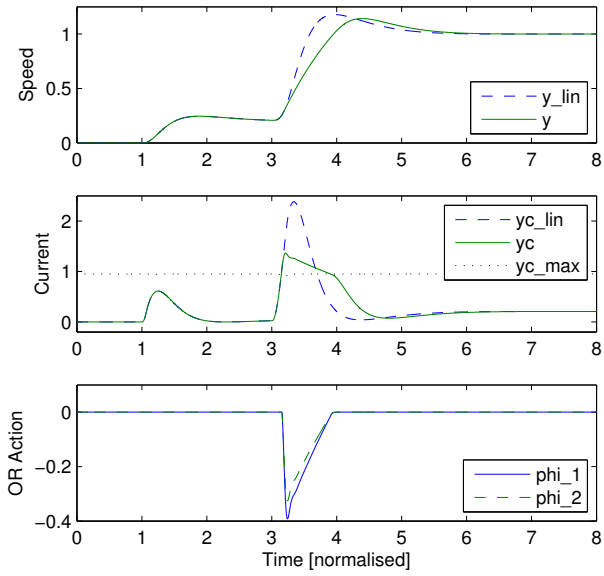

Fig. 8. Linear simulation with OR Architecture 3

OR Arch.	$\mathcal{L}_2$ gain, γ	Max. o/p violation (%)	States
no OR	n/a	138	0
1	13.72	22	5
2	27.35	60	3
3	26.78	37	4

Table 2. Performance measure comparison

Clearly, Architecture 1 provides the best performance but requires more states so the computational demands for implementation are increased. In addition, since the compensator is required to modify the controller states directly, coding is more complex. A further complication is that a practical controller implementation may require the controller realisation to be changed. With a first order controller it is simple to scale the override signal so the compensator will work as intended with the new state-space realisation. However, for higher order controllers this is non-trivial and a redesign of the override compensator would be required.

Architecture 2 combines the simplest tuning with the simplest implementation. Such designs often perform well and and do not require access to signals within the controller, making implementation simple. The number of states required for implementation does not rise if a higher order controller is used, and a change to the controller realisation has no effect on the function of the compensator.

Architecture 3 adds an extra level of flexibility in tuning over Architecture 2 which may improve performance. Implemen-

tation is quite simple, requiring only one additional state per control signal and access to these signals. Again, the function of the override compensator is independent of the controller realisation.

5. CONCLUSION

This paper has presented some of the practical issues regarding the implementation of override compensators designed using the formulae of Turner et al. [2002a]. In addition, guidelines have been given on the choice of architecture, and a modified architecture has been proposed which can provide an appealing compromise between performance and simplicity of design and implementation for systems with controllers of order greater than one.

The generic architecture, Architecture 1, has maximum flexibility over the structure of the compensator and so promises the lowest $\mathcal{L}_2$ gain performance level. However, the order of the compensator can be high and the tuning can be more complex. In addition, because the compensator directly alters the controller states, if a change in the controller realisation is desirable at the implementation stage, a redesign of the compensator is required.

Architecture 2 has the benefit of simplest synthesis and requires the least number of states for implementation. In addition, the function of the design is independent of the controller realisation used, and so the implementation stage in the process of design and verification can be tackled separately to those of synthesis and tuning. However, the $\mathcal{L}_2$ gain performance bound achieved can be limited due to the compensators inability to manipulate the controller states individually and its inability to directly alter the controller output.

Architecture 3 functions very similarly to Architecture 2 but with the additional flexibility of being able to directly alter the control signal. This can help to improve performance without significantly increasing the complexity of implementation or tuning. The function of this type of compensator is also independent of the controller realisation used.

REFERENCES

A.H. Glattfelder and W. Schaufelberger. A path form anti-windup to override control. *6th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2004)*, Vol. 3, pp. 1379-84.

A.H. Glattfelder and W. Schaufelberger. Control systems with input and output constraints. *Springer*, 2003.

G. Herrmann, M.C. Turner and I. Postlethwaite. A robust override scheme enforcing strict output constraints for a class of strictly proper systems. *Automatica*, to appear, 2007.

R. Krishnan. Electric motor drives, modelling, analysis & control. *Prentice Hall*, 2001.

A. Aceves Lopez and J. Aguilar Martin. Using multivariable nonlinear stability theory for override control systems. *Proceedings of the European Control Conference*, 1999.

P. March and M.C. Turner. Anti-windup compensator designs for permanent magnet synchronous motor speed regulation. *International Electric Machines and Drives Conference*, 2007.

P. March and M.C. Turner. Anti-windup compensator designs for permanent magnet synchronous motor speed regulation. *IEEE Transactions on Industrial Applications*, submitted for journal publication, 2008.

D.W. Novotny, and T.A. Lipo. Vector control and dynamics of AC drives. *Oxford Science Publications*, 2000.

M.C. Turner and I. Postlethwaite. Output violation compensation for systems with output constraints. *IEEE Transactions on Automatic Control*, Vol. 47(9), pp.1540-1546, 2002.

M.C. Turner and I. Postlethwaite. Output violation compensation for systems with output constraints. *University of Leicester Technical Report 02-02*, March 2002.

Appendix A. LINEAR PLANT AND CONTROLLER MATRICES

The linear plant model is parameterised by the following matrices.

$$A_p = \begin{bmatrix} -2.844 & 1.263 & -0.009809 & -0.158 & 0.05071 \\ 8.07 & -136.9 & 118.2 & 261.2 & -33.44 \\ 3.53 & -145 & -117 & -35.66 & 175.9 \\ 7.564 & -278.5 & -487 & -1312 & -329.8 \\ -3.429 & 116.1 & 213.1 & 1255 & -1362 \end{bmatrix}$$

$$B_{pd} = \begin{bmatrix} 1.543 \\ -2.204 \\ -0.9575 \\ -2.052 \\ 0.9302 \end{bmatrix} \quad B_p = \begin{bmatrix} -0.001138 \\ -0.008051 \\ 0.004471 \\ 0.03089 \\ -0.03901 \end{bmatrix}$$

$$C_p = [\, -1.542 \;\; 0.3585 \;\; 0.03594 \;\; 0.0286 \;\; -0.002264 \,]$$

$$C_{pc} = \begin{bmatrix} -0.0008787 & -0.3851 & -0.5068 & 1.028 & 0.8355 \\ 0.05575 & -2.14 & 0.8116 & 1.776 & -0.4106 \end{bmatrix}$$

$$D_{pd} = D_p = 0 \quad D_{pdc} = D_{pd} = [\, 0 \;\; 0 \,]'$$

The experimental second order controller state-space model is parameterised by the following matrices.

$$A_c = \begin{bmatrix} 0 & 0 \\ 0 & -200 \end{bmatrix}$$

$$B_{cr} = -B_c = \begin{bmatrix} 2.56 \\ -9.769 \end{bmatrix}$$

$$C_c = [\, 5.219 \;\; -9.769 \,]$$

$$D_{cr} = -D_c = 0$$

Appendix B. AMENDED STATE SPACE MATRICES FOR COMPENSATOR SYNTHESIS ACCORDING TO ARCHITECTURE 3

$$\bar{B} = \begin{bmatrix} 0 & B_p\tilde{\Delta} \\ B_{cr} & B_c\Delta D_p \end{bmatrix} \quad \bar{D}_y = [\, 0 \;\; \Delta D_p \,]$$

$$\bar{D} = \begin{bmatrix} 0 & D_{pc}\tilde{\Delta} \end{bmatrix} \quad L \in \mathcal{R}^{(n_r+m)\times q}$$