RECONFIGURABLE CONTROL USING CONSTRAINED OPTIMIZATION

J.M.Maciejowski
Cambridge University Engineering Dept.
Cambridge CB2 1PZ, England
Tel: +44 1223 332732. Fax: +44 1223 332662
Email: jmm@eng.cam.ac.uk

3 March 1997

Semi-Plenary paper for ECC97.

Abstract

A conceptual scheme for reconfiguring control systems in the event of major failures is advocated. This addresses a 'big problem' which is understandable by the man in the street, delivers enormous benefits if it can be made to work, and requires interesting and challenging research from control and systems theorists and practitioners. The scheme relies on the convergence of several technologies which are currently emerging: Constrained predictive control, High-fidelity modelling of complex systems, Fault detection and identification, and Model approximation and simplification. Much work is needed, both theoretical and algorithmic, to get this scheme to work, but we believe that there is enough evidence, especially from existing industrial practice, for the scheme to be considered achievable. After outlining the problem and proposed solution, the paper briefly reviews constrained predictive control, object-oriented modelling, which is an essential ingredient for practical implementation, and the prospects for automatic model simplification. The paper emphasises some emerging trends in industrial practice, as regards modelling and control of complex systems. Examples from process control and flight control are used to illustrate some of the ideas.

Keywords: Control reconfiguration, Fault tolerance, Predictive control, Modelling, Model approximation, Fault detection.

1 Introduction

We are interested in reconfiguring control systems when a major failure occurs. For example the partial or complete loss of a control surface on an aircraft, or failure of an important compressor in a process plant. In the event of such a failure, at least three inter-related questions arise:

1. Is it possible to continue to control the plant closely enough to the original specification that continuation of the original mission, or of the usual product, is possible?
2. Is it possible to control the plant, but with a much reduced specification, so that modifying the original mission, or production of a lower-quality product, is necessary?
3. Is it possible to abandon the mission, or to shut down the plant, without incurring disaster ('Get me home' mode)?

We make an initial assumption that it is not possible to anticipate all possible failure modes, so that an approach involving switching to precomputed control strategies is not possible. Most proposals for reconfigurable control systems take the 'precomputed' approach [27, 29, 31], although [33] detects a trend towards the kind of approach advocated in this paper and [16] makes the same assumption as ours. Of course it is sensible to precompute strategies for a set of failure modes which are relatively likely to occur, so our interest is in what to do if the failure mode is outside this anticipated set. The reasons for doing so are, firstly, that the set of anticipated failure modes, such as those revealed by a FMECA assessment, can be very large, and it is impractical or too expensive to precompute strategies for all the eventualities. Secondly, unanticipated failures also occur. Thirdly, a subjective impression is that single-mode failures are generally handled successfully, for example in aircraft operations; disasters generally occur when an initial failure is associated with other collateral damage, so that two or more of the anticipated failure modes occur simultaneously. In this case the number of possibilities grows combinatorially with the number of single-mode failures, and of course it becomes quite impossible to anticipate more than a small proportion of them.

The problem we have defined is outside the usual paradigms of adaptive or robust control, in that the sets of available inputs and outputs can change, and that the control specifications may change. The dynamics of the plant can change drastically and discontinuously, which is outside the usual robust

control paradigm but within that of some adaptive control research. Of course it is also true that the problem is well beyond the range of any control theory we currently have. We are proposing it as a 'Grand Challenge' problem to the control community. It has the following characteristics, which are desirable for such a challenge:

- The problem is generic and of great importance.
- The problem is immediately understood by the general public, and efforts to solve it are generally supported.
- Virtually every branch of control research can potentially contribute to solving it.

It is not enough to propose a problem; there should be at least some prospect of being able to solve it, before it becomes interesting. Our belief is that technologies are currently emerging, and entering industrial practice, which make it plausible to envision a general approach to the solution of the problem we have posed. There are 4 technologies which we have in mind:

1. Constrained Model-Based Predictive Control (MBPC), increasingly widely used in the petrochemical sector, and being introduced into other process industries.
2. 'High-fidelity' first-principles nonlinear simulation models, sufficiently detailed to contain representations of individual components.
3. Effective approximation/identification algorithms for multivariable systems.
4. Fault detection and identification (FDI) capability.

In addition to these technologies, it seems that research in Hybrid Systems is reaching a stage at which really useful results for our problem can be anticipated, so that analysis of some of the apparently most intractable aspects of the problem might be achieved. (We have in mind particularly the approach being pioneered by Morse [30], in which the set-up is quite close to that involved in the reconfigurable control problem.)

Basically the idea is that the following sequence of steps occurs:

1. When a failure occurs, the FDI system pinpoints the nature of the failure.
2. The 'high-fidelity' model is updated to reflect the failure.
3. Approximation/identification algorithms are run on data generated from the updated high-fidelity model, and/or from the failed system, to get a simple linearised model — probably augmented by linear inequalities — which captures the new external behaviour.
4. Constrained MBPC is run on the resulting model; it is given enough degrees of freedom (a big enough set of manipulated variables) to allow it to find a feasible control strategy.
5. The resulting system is analysed to determine whether the objectives need to be modified. (Some iteration of this step and the previous two is needed in general. This is the step in which hybrid systems theory, or something like it, is needed.)

In this paper we will briefly review 3 of the 4 technologies listed above (omitting FDI), and explain their relevance to the overall scheme. We will then present a brief example, which will illustrate some aspects discussed in the paper.

2 Constrained MBPC

2.1 Review of MBPC

Constrained Model Based Predictive Control (MBPC), also known by several other names, such as Receding Horizon Control (RHC) [28], Generalised Predictive Control (GPC), [5] Dynamic Matrix Control (DMC), [7] etc, is now the most widely used advanced control technique in the process industries, and it is the control methodology at the heart of this proposal. It is distinguished from other control methodologies by the following three key ideas:

- An explicit 'internal model' is used to obtain predictions of system behaviour over some future time interval, assuming some trajectory of control variables.
- The control variable trajectory is chosen by optimizing some aspect of system behaviour over this interval.

- Only an initial segment of the optimized control trajectory is implemented; the whole cycle of prediction and optimization is repeated, typically over an interval of the same length. The necessary computations are performed on-line.

It naturally handles the control of multivariable plant, and takes account of information on constraints arising from equipment limitations, safety requirements, etc. In its usual form it does this by combining linear dynamic models with linear inequalities, which seems to be a very powerful combination, since the linear model keeps the dynamics simple, while the inequalities can be used to represent important nonlinearities, as well as constraints. The usual formulation of MBPC also has a quadratic cost functional; when combined with a linear model and linear inequalities this leads to a Quadratic Programming optimization problem. Since this problem must be solved on-line, the fact that the problem is convex is most important, and the additional structure available in a QP problem is important for predicting properties such as solution time.

To be specific, the cost to be minimised typically has the form

$$J(k) \quad = \quad \sum_{i=N_1}^{N_2} ||M\hat{x}(k+i|k) - r(k+i)||^2_{Q(i)} \quad + \quad \sum_{i=1}^{N_u} ||\Delta u(k+i)||^2_{R(i)} \quad (1)$$

and the minimisation is performed subject to constraints such as

$$|\Delta u_j(k+i)| \quad \leq \quad V_j \quad (2)$$

$$|u_j(k+i)| \quad \leq \quad U_j \quad (3)$$

$$|(M\hat{x})_j(k+i|k)| \quad \leq \quad X_j \quad (4)$$

where $u(k)$ is the (control) input vector at time k, $\Delta u(k) = u(k) - u(k-1)$, and $Mx(k)$ is the vector of variables which are to be controlled; $x(k)$ is the state of the plant. $\hat{x}(k+i|k)$ is a prediction of $x(k+i)$ made at time k, M is some matrix (for example, $M = C$ in the usual linear state-space model if only outputs are to appear in $J(k)$), and $r(k)$ is some reference (desired) trajectory for $Mx(k)$. The integers N_1, N_2 and N_u, as well as the weighting matrices $Q(i)$ and $R(i)$, are in principle chosen to represent some real performance objectives (such as profit maximisation in a process application [32]), but in practice they are often tuning parameters for the controller. It is assumed that the control signals are constant after the end of the optimization horizon, namely that $\Delta u(k+i) = 0$ for $i > N_u$. In the inequalities $u_j(k)$ denotes the j'th component of the vector $u(k)$, etc, and V_j, U_j, X_j are problem-dependent nonnegative values.

The predicted values $M\hat{x}(k+i)$ which appear in the cost function are usually obtained from a linear 'internal model', which is a predictor derived from a

linear approximation to the plant. The predictions usually assume a constant output disturbance over the prediction horizon, the level of this disturbance being estimated from the initial one-step prediction error. This disturbance model results in a kind of integral action being present in the controller: a persistent error between the output and set-point vectors is attributed by the controller to an increasing disturbance, and consequently an increasing control signal is generated, until the error is removed.

Relatively little is known about constrained MBPC theoretically, but it is now receiving a lot of attention. Nominal stability has been well investigated, even when constraints are active [6]. There are several parameters to be chosen when an MBPC scheme is implemented (principally prediction and control horizons, and weights for the cost functional) and there are now some reliable guidelines for choosing these so as to assure nominal stability. Essentially, the prediction horizon has to be made large enough, and theorems exist which specify how large it has to be. Alternatively, terminal constraints have to be imposed on the controlled variables. The use of infinite horizons is attractive, except that the imposition of constraints is then difficult, but some proposals have been made even in this direction [15].

It is still not really known how tolerant MBPC is to mis-modelling, or how to improve its robustness to modelling errors. One suggestion for obtaining robustness is to replace the quadratic optimization by a min-max formulation, but this gives a much harder optimization problem. Nevertheless Allwright has shown [6] that even this problem can be solved relatively efficiently. Progress has also been made on tuning the various parameters in the standard MBPC formulation so as to obtain robustness [18].

This usual MBPC formulation results in a piecewise-constant linear control law, with switching between laws occurring whenever the set of active constraints changes. Some research in hybrid systems emphasises the analysis of piecewise-constant linear systems. Putting this another way, it may be fruitful to consider a constrained MBPC controller as a hybrid system which switches between various linear control laws, depending on the active constraints.

It is important to point out that constrained MBPC is now an established control technique which is used routinely, particularly in the petrochemical sector, and on very large problems — tens of controlled variables, hundreds of manipulated variables, and thousands of constraints in some applications. On the other hand, it is typically used to implement a higher-level control layer on top of existing conventional controllers, and in an industry with very slow dynamics. This probably explains why it was brought into use even before any

kind of stability proof was developed, since it was always possible to disable it if there were any signs of developing problems. For the application to reconfigurable control which we are proposing this is probably unacceptable, and some theoretical advances to provide reassurance of correct functioning will be necessary.

2.2 MBPC as a tool for reconfiguration

Constrained MBPC has some inherent 'self-reconfiguration' capability [23], if there are redundant actuators and one of these actuators fails. In systems with redundant actuators, a 'daisy-chaining' arrangement in which one actuator is used in normal operation, but another one is brought into operation if the first one saturates or fails, is shown in figure 1. This figure shows the use of a model of the saturation characteristic of an actuator. Such a scheme is usually intended to deal with actuator saturation. But it is also effective in case the actuator used for normal operation (Actuator 1) fails, for example by getting stuck at a constant value, providing that integral action is present in the controller. The back-up actuator is not brought into operation immediately, but a persistent error in the controlled output leads to an increasing control signal from the integral action, until the daisy-chaining system 'thinks' that the normal actuator has become saturated, whereupon the back-up actuator (Actuator 2) is brought into play.

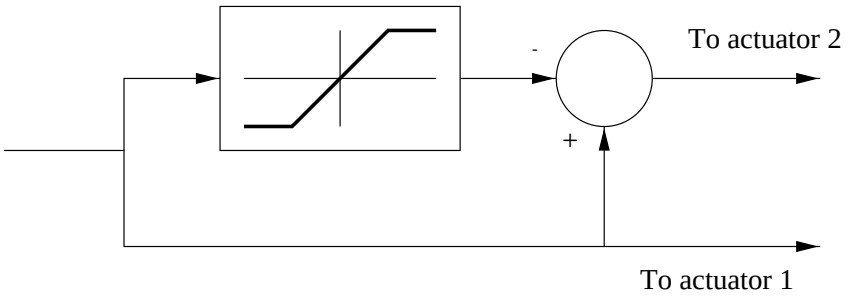

Figure 1: 'Daisy-chaining' of redundant actuators

Constrained MBPC exhibits essentially the same behaviour, providing that actuator saturation is modelled by suitable hard constraints, and that integral action is present (which it is in the standard problem formulation). Suppose that a plant has redundant actuators, and that one of these actuators becomes stuck at a position other than its correct equilibrium position. The controller

'thinks' that all the actuators are set to correct equilibrium positions but, because of the failure, the output does not approach the set-point. In the controller, this discrepancy is attributed to an output disturbance; this disturbance is assumed to persist at the same level into the future, and the actuator settings are therefore changed in order to compensate for the estimated disturbance. Now the setting of the failed actuator does not really change, so a discrepancy in the output vector remains. But, since a similar error persists in the face of an apparently 'larger' actuator signal, the controller attributes this error to a larger disturbance than it estimated previously. Consequently a greater change in the setting of the i'th actuator is demanded. This process is repeated until the controller 'thinks' that the failed actuator has reached its saturation level. It now moves the other actuators more vigorously to combat the perceived very large disturbance. The output approaches the set-point more closely, and as it does so, the *estimated* error is reduced, although not to zero. If the failure is compatible with the set-point specification, enough control action is eventually applied to return the plant to the correct set-point.

It is noteworthy that this behaviour occurs without the need to anticipate certain patterns of actuator failure, or to design schemes to handle them — it comes 'for free' with constrained MBPC. (We have assumed asymptotic stability of the MBPC scheme in the presence of the failure, in this scenario.) In this scenario the response of the MBPC controller occurs only after it has 'deduced' from feedback information that the gain of the usual actuator has been reduced. No Fault Detection information is assumed to be available. Clearly availability of external information about the problem would allow corrective action to be taken sooner, and thus more effectively.

Failures which can occur can be of three types:

1. Actuator failures — reduced range of actuator, possibly to zero, or 'hard-over' failures (where an actuator remains at one of its extreme positions).
2. 'Internal' failures — those in which some some part of the plant fails, with the consequence of changing significantly the plant dynamics and gains. (An actuator failure which changed only the dynamics of the actuator itself, but not its steady-state effectiveness, would be an 'internal' failure.)
3. Sensor failures — some measurements become unavailable, or incorrect, or unusually noisy.

Of course many failures will be combinations of these. For instance, losing some of the tail structure of an aircraft due to metal fatigue or battle damage

may cause a control surface (actuator) to disappear, significant changes in moments of inertia (hence an 'internal' failure), and perhaps loss of a dynamic pressure measurement (sensor). But to keep life tolerably simple, disregard such combinations for the moment. This is not entirely unrealistic. One of the most widely-publicised examples of control reconfiguration in recent years was the Sioux City incident [13], in which a rear engine and most hydraulic systems were lost, with the pilot flying the aircraft by controlling thrust to the two surviving engines. Although this was an example of an initial failure causing further collateral failures, only actuator failures were involved.

Actuator failures are probably the easiest to deal with, providing there is some degree of redundancy. As argued above, constrained MBPC accommodates such failures to some extent, even if there is no explicit FDI information to say that a failure has occurred. The situation is of course improved if such information is available, and it is easy to incorporate the information in the MBPC framework, primarily by modifying the explicit constraints on the corresponding actuator levels, or constraining the appropriate element of Δu to be zero if the actuator is jammed.

'Internal' failures are more difficult to handle, unless their effects are sufficiently small to be dealt with by the inherent robustness of the normal control system. Otherwise, FDI information is essential, in order to update the internal model used by MBPC. Conceptually this is straightforward and we shall argue below that we have the modelling technology available to do this in practice. The major difficulty is that of obtaining correct FDI information.

Sensor faults are potentially the most difficult to deal with, from the point of view of correcting for them within MBPC. If a sensor provides the only measurement of a controlled output variable, then it will not be possible to continue control without modifying the MBPC cost function. At the very least the control of that variable will have to be abandoned. It may be possible to substitute the control of another variable for the unavailable one, if required, or perhaps to replace the cost function by one specifying higher-level objectives. But this is moving into much deeper waters than in the cases considered earlier. Some higher-level supervisor is now required to adjust the MBPC formulation — and to decide whether it needs adjustment. It is not clear to what extent such a supervisor could be generic, in the sense that it could deal with unanticipated failures. Not all sensor failures need have such drastic consequences. If the measurement which is lost does not appear directly in the cost function, but is used to improve the quality of some estimated variable (as in a data fusion scheme), then it is the internal model used by MBPC which must be updated; in this case the position is similar to that after an 'internal' fault.

It is simplistic to believe that it is enough to change the internal model, or to change the constraints, in order to represent a fault, and that the MBPC controller will thereafter issue satisfactory control inputs. In some cases this will work. But in general the controller will need to be re-tuned to give satisfactory performance. 'Tuning' here means adjusting the horizons and weights which appear in the cost function, and possibly active management of constraints — it seems to be necessary to 'soften' certain constraints in order to retain feasibility, which is necessary to ensure stability in some cases. We believe that it is not too fanciful to expect general tuning strategies to be developed, which will be satisfactory for the majority of cases. (It is important to remember that in the context of recovery from failures, emphasis is on 'satisfactory' performance, which may be much worse than the performance one would design for in normal operation.)

3 High-fidelity complex models

'High-fidelity' dynamic models are increasingly being built of complex plant. They have been built and used in the aircraft and space industries for many years, but they are now being used also in the process industries. For example, every oil platform operating in the North Sea has an extremely complex first-principles dynamic model of its gas and/or oil processing operations. These models are typically not built for the purpose of designing controllers in the first instance. They are usually built for training and safety certification purposes; but since they exist, why not exploit them also for control purposes?

For our purposes it is necessary to have models which are detailed enough to be able to represent failures. This means that the entities which are liable to failure, such as valves, sensors, compressors, control surfaces, etc, must be represented as entities in the model, so that it is possible either to remove the corresponding entity or to modify it appropriately. The models referred to in the previous paragraph typically meet this requirement, but they are generally embedded in proprietary software, without open external interfaces. Fortunately there is an emerging open methodology for building and maintaining (*ie* modifying) models of the kind we need, best represented by the object-oriented modelling languages *Omola* and *Dymola* and their associated tools [26, 8, 25]. (A long-standing methodology with similar properties is based on bond-graphs, but these do not appear to be suitable for modelling very complex systems, especially those involving multi-property streams such as fluid flows.)

These languages result from the realisation that modelling is quite distinct from simulation [2]. Simulation languages such as *Simulink* rely on a pre-analysis of the internal causality of a model by the model builder, which is why they typically use block diagram representations with fixed distinctions between the input and output variables of each block. (Note that this is not a point about graphical interfaces, but about the conceptual representation of simulation models. All simulation languages adhering to the so-called CSSL standard, such as *ACSL* or *ESL*, suffer from the same limitation, whether they are provided with a graphical interface or not.) In general it is impossible, in such languages, to represent a single real-world entity by a single entity in the model description. This is due to the fact that even a small change in the connectivity of real-world entities — such as removing a single entity — can lead to a radical change of causalities.

A simple example of this is provided by a resistor. In *Simulink* it is impossible to have a single block which will represent even the humble resistor in all circumstances. In fact infinitely many blocks are necessary. In addition to the obvious representations of R and $1/R$ in case the resistor is connected to a perfect current or perfect voltage source, every passive circuit in which the resistor is a component requires a two-port transfer matrix in which the 'R' appears more than once, and inextricably mixed up with other circuit components. Contrast this with an *Omola* representation, in which an individual resistor *is* associated with a single 'object':

```
R138 ISA Resistor WITH R=10 ohms;
```

and the generic model of a resistor states the defining relationships without implying causalities:

```
Resistor ISA TwoPort WITH
⋮
V = I * R;
⋮
END;
```

Note that the '`=`' here denotes equality rather than assignment, so that no computational sequence is implied. It is not necessary for `I` to be known first, and then to compute `V`. Indeed, they will in general both be computed 'simultaneously', after the '`V=I*R`' relation has been combined with all the relations defining the other components in the circuit and their connections.

Separate *Omola* statements describe how $R138$ is connected to other components. Furthermore, hierarchical model descriptions are supported, so that one can build reusable modules such as filters or phase-locked loops, parametrized by component values. In fact, the methodology of model-building in such a language is the same as that of defining classes in object-oriented programming languages, namely top-down stepwise refinement of behaviour definitions. So, for electric circuit models, *Omola* has classes such as `Two-port` which can then be given more specialised behaviours:

```
Bridged-T-Filter ISA Two-port
WITH
⋮
END;
```

Similarly, for chemical process models, basic classes such as `Tank` can be defined, and then used to define more specialised classes such as `Pressure-Vessel`.

It is worth re-emphasising that such reusable modules are possible *only* because relations are described, rather than computation flow. When a complete model is assembled, its hierarchical description is 'flattened', so that a large set of relations is obtained, which can then be analysed and the appropriate solution sequence determined. The key fact is that once a model description is complete, it is possible to obtain a simulation automatically, whereas translation in the reverse direction, from a simulation to a model, is not possible. Hence it is also true that, in the event of a failure, it is possible to update an *Omola*-type description and then obtain the resulting simulation model automatically, whereas this is not possible if one starts with a typical simulation language description.

Figure 2 shows a simplified version of an 'auto-cascade' refrigeration process used for liquefying natural gas [11]. A mixture of two refrigerants (propane and methane) is compressed by the compressor A and partially condensed in the condenser B. The remaining vapour (mostly methane) is cooled in the heat exchanger C and condensed in heat exchanger D. The condensate from B (mostly propane) is cooled in heat exchanger C and expanded to low pressure through valve E, causing its temperature to fall to about $-42\ ^\circ C$. The condensed methane is expanded to low pressure through valve F, its temperature falling to about $-161\ ^\circ C$ in consequence. After expansion, the resulting two-phase (vapour/liquid) methane passes through heat exchanger D again, where the liquid phase evaporates, obtaining the required enthalpy of evaporation from the natural gas and from the methane vapour coming from C,

thus cooling them. The methane vapour is then mixed with the two-phase propane coming from valve E and passes through heat exchanger C, where the liquid propane evaporates, cooling the incoming natural gas, the methane vapour flowing from B, and the propane liquid flowing from B. Finally the vapour mixture is returned to the compressor A.

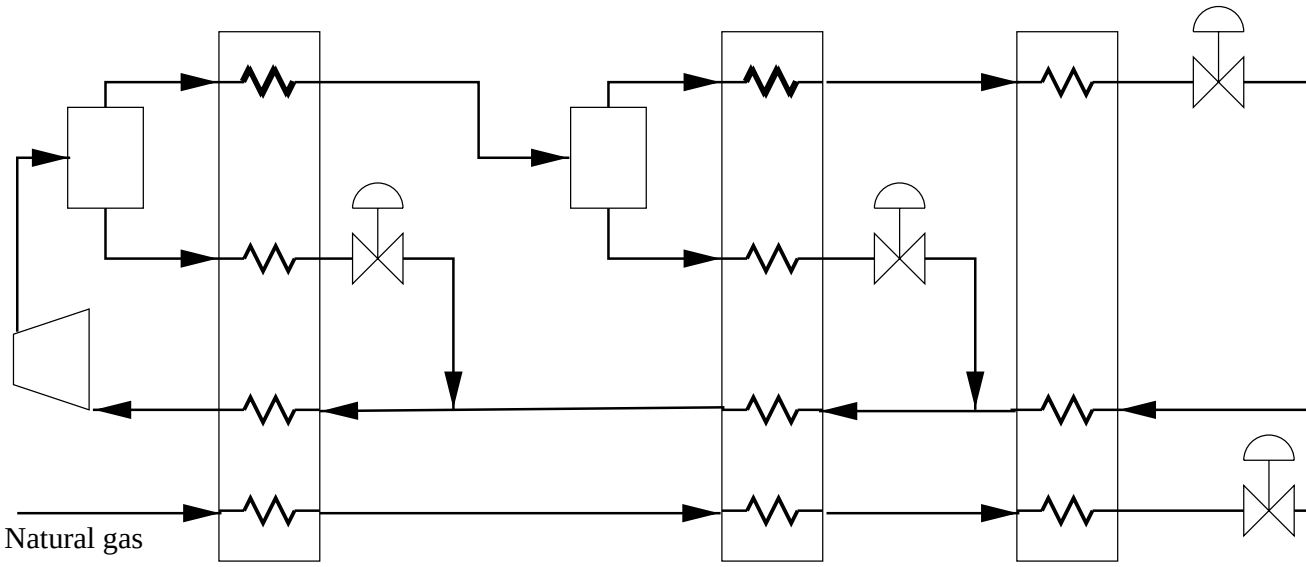

Figure 2: Natural gas liquefaction process

This simplified process is not,in fact, thermodynamically feasible, because the range of temperatures is too great to be attained by only two stages of heat exchange with practical refrigerants. In practice at least one more refrigerant — ethylene — and hence at least one more stage of heat exchange and expansion is needed. Furthermore, greater efficiency is obtained by using from 6 to 10 such stages, which is typical for an actual process. Each stage, comprising a heat exchanger and an expansion valve, is a good example of an entity that could be modelled as a reusable *Omola* module.

It is not feasible to react to every possible kind of failure by automatic update of a model, even if FDI information correctly identifies the failure. Suppose that one of the heat exchangers ruptures, for example. The change in its behaviour is quite drastic, as it ceases to be a heat exchanger and becomes a tank filling with liquid instead, with expansion occurring inside it as well. Representing this change would require understanding the physics governing the new behaviour, and formulating the appropriate relations — not something which

could be automated. On the other hand, suppose that the pipe connecting heat exchanger C to the expansion valve E ruptures. This phenomenon is much easier to represent, by connecting the pipe to an infinite reservoir at atmospheric pressure. It would be feasible to have a pre-defined model of a pipe rupture available for use in such cases, since this is a kind of failure which could be expected to occur. (Note that this does not imply anticipating specific failures, since the location of a rupture would not be assumed in advance — the module could be connected to whatever section of pipe was affected.) In practice a simpler solution would be available, since the ruptured section of pipe would almost certainly be isolated from the rest of the process by valves (not shown in the figure). This is easily represented as a model update, either by closing the isolation valves, if they exist as entities in the model, or by reducing the diameter of that pipe to zero, etc.

Correct management of the interface between the FDI system and the high-fidelity model, in order to allow such updates to be performed correctly, is not trivial, and requires some research. But it is probably one of the easier achievements required by our scheme.

An important consideration for our application is the speed of solution of complex high-fidelity models, which are necessarily nonlinear and therefore do not admit analytical solutions. It appears that some of the proprietary models mentioned earlier can be run very quickly on high-performance workstations without requiring special-purpose hardware [19]. This is achieved by modularisation of the models, with independent solution of each model and periodic reconciliation between modules. This approach currently depends on understanding of the model, however, and cannot be fully automated. On the other hand, advances in hardware performance will probably make such special techniques unnecessary, at least for low-bandwidth applications such as process control.

4 Approximation and identification

While complex high-fidelity models are needed for representing particular failure conditions in detail, they are not suitable for direct use as internal models in the constrained MBPC. It is a commonplace that control typically requires only simple models, which approximate the input-output behaviour of the plant to a 'reasonable' extent. Not only are complex models not required, but they are harmful because they impair the real-time performance of MBPC schemes.

For high solution speeds MBPC needs a linear model, in order to solve nothing more complicated than a QP problem. A simple model is desirable for the same reason.

The obvious way of obtaining a simple linear model is to linearise the complex nonlinear model about an operating condition and then simplify it by model reduction. However the initial linearisation is impractical by either symbolic differentiation, or by numerical differentiation as performed in typical simulation software. (Although symbolic differentiation is possible using computer algebra, even for very complex systems, high-fidelity models used in the process industries are very far from being in the $f(x, \dot{x}, u, t) = 0$ form, with 'nice' functions f — they often contain elements such as thermodynamic databases, for instance. Numerical differentiation is prone to major problems, for example missing dynamics which involve time delays.)

A more practical and effective way of obtaining a simple linear model is to perform some simulation experiments on the complex model, and to 'identify' a model for the data obtained from such experiments. This method of obtaining linearised models has been used for many years, for example for econometric models [20]. In addition to overcoming the problems mentioned above, this method can actually produce more accurate models than the theoretically linearised model, because the level of perturbation can be controlled, which allows some aspects of nonlinearities to be captured, whereas the theoretical approach is valid only for infinitesimal perturbations.

A very effective way of performing identification on data obtained from complex models is to use the so-called 'subspace methods' developed in recent years [17, 36, 35], and closely-related approximate realization algorithms developed rather earlier (which are reviewed in [22]). These are very effective for multivariable systems, generally giving very good reproduction of input-output behaviour with relatively simple models (as measured by the state dimension). One of their great advantages is that they can be run automatically, without user intervention and decision making. (The only essential decision needed is on the state dimension, and that can be automated. The author's recent experience includes a 2-input, 3-output, 10-section high-fidelity model of a pipeline carrying two-phase flow, being approximated very well by a linear model with fewer than 20 states, and the linearised model being obtained by a fourth-year undergraduate with almost no assistance — which is pretty close to being 'automatically'.) For use in MBPC schemes it is important that the simplified model should be stable (if the process being controlled is), and we now know how to guarantee stable models when using these methods [4, 21]. There are also variations of the subspace methods suitable for use when the plant is

operating under feedback.

Once a subspace or similar method has been used to obtain an initial model from a complex model, parameter estimation on real plant data can be used to improve the model, or to 'track' gradual changes in the plant. Balanced parametrizations may be particularly suitable for this purpose, since they are well-conditioned numerically, and they ensure stability of the estimated model [3, 22]. Another interesting idea which fits well into the scenario envisaged here is that of *just-in-time* models, which are obtained (by approximation and/or estimation) as they are needed, at times and operating conditions which cannot be predicted [34].

A further question to be considered is which linearised model to use. Recall that the model required by MBPC is a long-range predictor. Furthermore, this predictor needs to perform well when operating in a feedback loop. The issues raised in [10] are relevant here.

5 Fault detection and identification

Fault detection and identification (FDI) is the key element of this whole proposal, and probably the most difficult one to make successful. We will, however, say less about it than about the other elements, due to lack of expertise in this area. There is intense activity on FDI and progress is being made, but it must be admitted that neither of these facts guarantees eventual success. The problem is inherently very difficult, particularly as the plant will be operating in closed loop, so that the controller may to some extent be compensating for the effect of a failure — the classic dilemma of 'dual control'.

But there are also some reasons for optimism:

1. The advent of self-validating components [9, 12] will make the FDI task easier, and trivial for some failures. The greatest scope for self-validation seems to be with sensors.
2. This proposal is particularly aimed at sudden and major failures, which are easier to detect and identify than gradual deterioration of components.
3. The availability of high-fidelity models (and corresponding measurements) in effect makes more information available to the FDI task, which allows the sensitivity/false alarm trade-off to be shifted to a better level.

While it may not be essential to provide guaranteed levels of performance of a reconfigurable control scheme in the event of a failure (since one may not be too choosy when trying to avoid disaster), it is certainly essential to ensure that a reconfigurable control system does not erroneously decide that a fault has occurred and then proceed to implement a relatively high-risk strategy. Therefore the question of false alarm occurrence is a key one for the present proposal.

6 Example

We present an example of reconfiguration performed by an MBPC flight controller, when the rudder of an aircraft jams, and it is required to change the aircraft heading (yaw angle). A linearised model of a civil airliner, derived from the 'RCAM' model used in the recent *GARTEUR* Flight Control Design Challenge [24], is used for this example. The internal model used by the MBPC controller is the same linear model. (So exact modelling is assumed initially.) There are 3 controlled variables: the yaw angle, the roll angle, and the sideslip, and 4 actuators: the rudder, the ailerons, the tailplane, and the engine thrust. A rudder jam (at the neutral position) is simulated by disconnecting the rudder demand signal (issued by the controller) from the rudder. A step demand is then made on the yaw angle. The results are shown in Figures 3 and 4. Each of the sub-figures in these figures shows 5 cases:

Case 1 Normal operation of the rudder. The rudder and aileron positions are constrained to be within ± 2 units on the graphs, which represents $\pm 20^\circ$ in each case.

Case 2 Jammed rudder. No FDI information supplied to the controller, so that the controller is not aware of the failure. The constraints on the rudder position demand are removed, in order to see the effects of actuator constraints (known to the controller) in this scenario. (The rudder demand has been reduced by a factor of 300 in the figure for this case.) Of all the cases, this gives the slowest yaw angle response.

Case 3 Jammed rudder. No FDI information supplied to the controller, but the usual $\pm 20^\circ$ constraints are restored on the rudder position. The yaw response is significantly faster in this case, because the controller stops relying on the rudder sooner, and makes more use of the ailerons, as can be seen from the larger roll angle.

Case 4 Jammed rudder. FDI information supplied — the constraint on the rudder demand has now been tightened to $\pm 0°$, so that the controller knows that it cannot move the rudder. In fact this makes so little difference that the plots for Cases 3 and 4 cannot be distinguished from each other. The reason for this is that in each case the controller moves the rudder demand to its constraint almost immediately, and then uses the ailerons and other actuators. Since the actual rudder position is the same in both cases, the two behaviours are virtually identical.

Case 5 Jammed rudder. FDI information supplied, as in Case 4. But now the weight on roll errors in the cost function has been reduced by a factor of 3, approximately. This leads to a much faster response of the yaw angle. It can be seen that a much larger roll angle develops during the first 10 seconds of the manoeuvre when this weight has been reduced, and the lift then has a larger component in the horizontal plane, which changes the aircraft's heading.

The manoeuvre simulated here is rather artificial. A more practical requirement than changing only the yaw angle would be to change the aircraft heading, without much concern for what combination of body angles was most appropriate to achieve it. Elsewhere we have argued [14] that a promising role for MBPC in flight control is at the flight management level, without considering reconfiguration. This example shows that such a higher level role is also appropriate for reconfiguration, since an MBPC-based flight manager would issue appropriate yaw, roll and sideslip set-points. In this case, however, the rudder jam would no longer be just an 'actuator fault', but what we earlier called an 'internal fault', which implies that a more complicated model update would be required after receiving FDI information.

7 Conclusion

The objective of our proposal is to find systematic ways of reconfiguring control systems in the event of major failure or damage. Benefits of doing this would be to enable safe operation or shut-down of industrial complexes following component failure, safe return to base of an aircraft following battle damage, etc.

Of course there are many problems with getting the scheme we have outlined to work. The main research problems include:

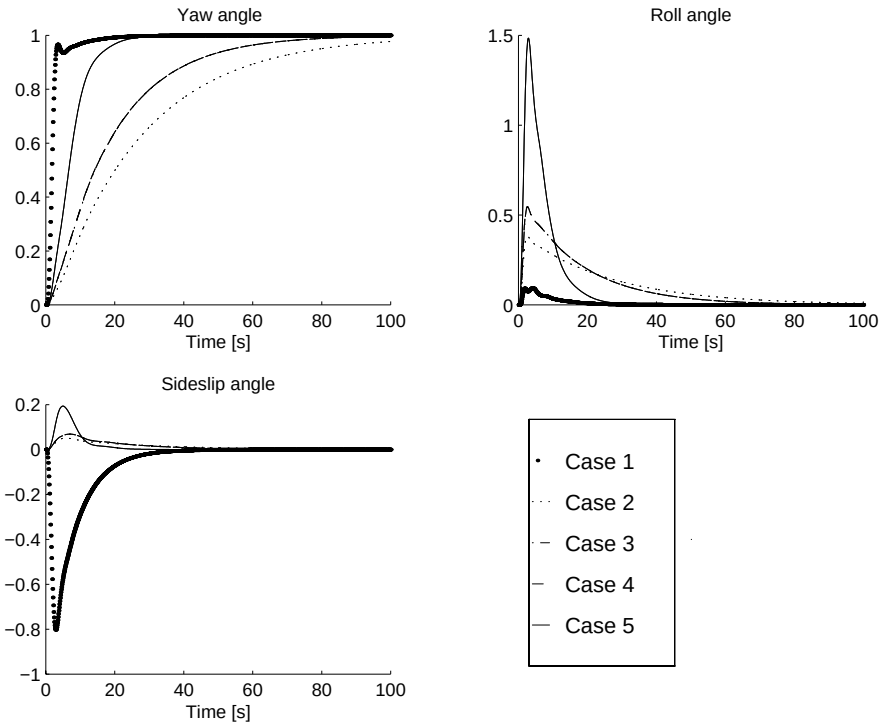

Figure 3: Yaw angle step demand with failed rudder: Controlled outputs

1. Getting reliable FDI.
2. Developing strategies for tuning the MBPC criterion on-line, and for constraint management, which will allow provably good reconfiguration schemes to be developed.
3. Ensuring the approximation/identification algorithms work in closed-loop and produce models suitable for control.
4. Getting the whole scheme to run quickly enough. (Not a problem for process applications, but definitely the bottleneck for aerospace.)
5. Automatic updating of the high-fidelity models.
6. Developing an appropriate conceptual 'system architecture' for integrating the 4 technologies successfully. (This probably has to be application-dependent.)

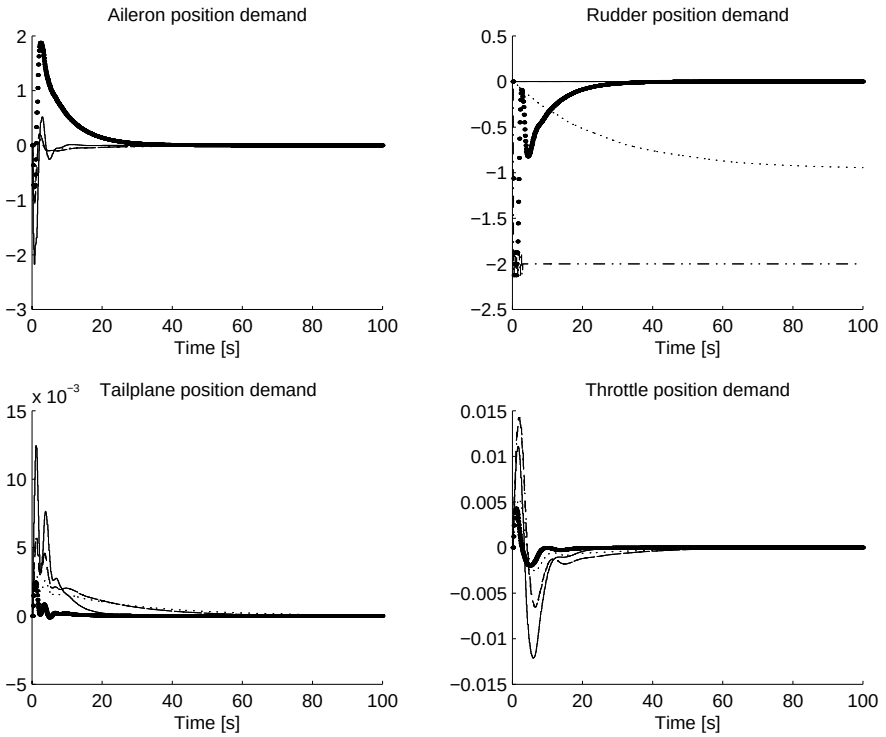

Figure 4: Yaw angle step demand with failed rudder: Actuator demands. (Key as for previous figure.)

We believe that there is a good chance of making the whole scheme work. The fact that MBPC controllers are used routinely in some industrial sectors, that complex models are built and run, that we have excellent methods for simplifying models, and that FDI is a major industrial and academic research area, together support this hypothesis. The approach may seem rather 'brute force', but it offers a unifying approach to reconfiguration problems, and offers much scope for good theoretical research which is needed in order to make it work.

One final thought. If we are prepared to do as much work on-line as suggested above, other approaches might also become feasible. Why not submit the linearised model of the failed system to a pole-placement algorithm, for instance, supplemented with some parameter tuning to get reasonable performance? Indeed, why not? In the control design community our common experience is

that design is difficult, even in an off-line environment. That may be partly due to our problem formulations — posing non-convex problems, typically. But there is also a 'cultural' component, in the sense that we usually attempt inherently difficult problems, trying to optimise performance, to satisfy many simultaneous objectives, and generally trying to improve on whatever has already been achieved. For the fault-tolerant / reconfigurable control problem, a different mind-set is required, as regards the control re-design sub-problem. 'Good enough' becomes much less demanding, in terms of performance, and in terms of performance guarantees. It may well be that, if one approaches the problem with that view, then several 'simple' approaches become feasible.

8 Acknowledgement

I would like to acknowledge many useful conversations with M.Huzmezan, who also performed the simulations shown in section 6.

References

[1] Babcock,P.S, Channelization — the 2-fault tolerant attitude-control function for the space station freedom, *IEEE Aerospace and Electronic Systems Magazine*, **11**, 9-16, (1996).

[2] Cellier,F.E, *Continuous System Modeling*, Springer-Verlag, 1991.

[3] Chou,C.T, and Maciejowski,J.M, System identification using balanced parametrizations, to appear in *IEEE Trans. Automatic Contr.*

[4] Chui,N.L.C, and Maciejowski,J.M, Realization of stable models with subspace methods, *Automatica*, **32**, 1587–1595, 1996.

[5] Clarke,D.W, Mohtadi,C, and Tuffs,P.S, Generalised predictive control — Parts 1 and 2, *Automatica*, **23**, 137–160, 1987.

[6] Clarke,D.W, (ed), *Advances in Model-Based Predictive Control*, Oxford University Press, 1994.

[7] Cutler,C.R, Cutler,C.R, and Ramaker,B.L, Dynamic matrix control — a computer control algorithm, *Proc. American Contr. Conf., San Francisco*, Paper WP5-B, 1980.

[8] Elmqvist,H, *A structured model language for large continuous systems*, Report LUTFD2/(TFRT-1015)/1-226, Lund Institute of Technology, 1978.

[9] Faulkner,A, Smart actuator systems: a practical solution? *Proc. Conf. Aerospace Hydraulics and Systems, London. September 1993* (London: Inst. Mechanical Engineers), 123-132, 1993.

[10] Gevers,M, Towards a joint design of identification and control?, in: Trentelman,H, and Willems,J.C, *Essays on Control: Perspectives in the Theory and its Applications*, (Boston: Birkhaüser), 1993.

[11] Gosney,W.B, *Principles of Refrigeration*, (Cambridge: Cambridge University Press), 1982.

[12] Henry,M.P, and Wood,G.C, The implications of digital communications on sensor validation, *Proc. IFAC symposium on On-line fault detection and supervision in the chemical process industries, Newark, Delaware, USA*, 1992.

[13] Hughes,D, and Dornheim,M.A, United DC-10 crashes in Sioux City, Iowa, *Aviation Week and Space Technology*, **131**, 96–97, 1989.

[14] Huzmezan,M, and Maciejowski,J.M, Flight management using predictive control, in: Magni,J-F, Bennani,S, and Terlouw,J, (eds), *Robust Flight Control — A Design Challenge*, Lecture Notes in Control and Information Sciences, vol. 224, (London: Springer-Verlag), 1997.

[15] Kothare,M.V, Balakrishnan,V, and Morari,M, Robust Constrained MPC using Linear Matrix Inequalities, *Automatica*, **32**, 1996.

[16] Kwong,W.A, Passino,K.M, Laukonen,E.G, and Yurkovich,S, Expert supervision of fuzzy learning systems for fault tolerant aircraft control, *Proc. IEEE*, **83**, 466–483, 1995.

[17] Larimore,W.E, System identification, reduced-order filtering and modeling via canonical variate analysis, *Proc. American Contr. Conf.*, San Francisco, 1983.

[18] Lee,J.H, and Yu,Z.H, Tuning of model predictive controllers for robust performance, *Computers in Chemical Engineering*, **18**, 15–37, 1994.

[19] Lin,J, and Griffiths,G.W, A shadow model approach to fault detection and isolation (FDI) in the process industry, *Paper presented at the InstMC Colloquium on Fault Diagnosis in the Process Industries: Model-based vs Knowledge-based, London*, January 1994.

[20] Maciejowski,J.M, and Vines,D.A, Decoupled control of a macroeconomic model using frequency-domain methods, *J. Econ.Dyn. and Contr.*, **7**, 55-77, (1984).

[21] Maciejowski,J.M, Guaranteed stability with subspace algorithms, *Syst. and Contr.Lett.*, **26**, 153–156, 1995.

[22] Maciejowski,J.M, Parameter estimation of multivariable systems using balanced realizations, in: Bittanti,S, and Picci,G, (eds), *Identification, Adaptation, Learning*, NATO ASI Series, (Berlin: Springer-Verlag), 1996.

[23] Maciejowski,J.M, The implicit daisy-chaining property of constrained predictive control, submitted to *Applied Mathematics and Computer Science* (Special issue on Process Control and Data Processing).

[24] Magni,J-F, Bennani,S, and Terlouw,J, (eds), *Robust Flight Control — A Design Challenge*, Lecture Notes in Control and Information Sciences, vol. 224, (London: Springer-Verlag), 1997.

[25] Marquardt,W, Trends in computer-aided process modeling, *Computers in Chemical Engineering*, **20**, 591–609, 1996.

[26] Mattsson,S.E, Andersson,M, and Åström,K.J, Object-oriented modeling and simulation, in: Linkens,D.A, (ed), *CAD for Control Systems*, Marcel Dekker, 1993.

[27] Maybeck,P.S, and Stevens,R.D, Reconfigurable flight control via multiple model adaptive control method, *IEEE Trans. Aerospace Electron. Syst.*, **27**, 470–479, 1991.

[28] Mayne,D.Q, and Polak,E, Optimization based design and control, *Proc. IFAC World Congress, Sydney, July*, 1993.

[29] Moerder,D.D, et al, Application of precomputed control laws in a reconfigurable aircraft flight control system, *J. Guidance, Navigation and Control*, **12**, 325–333, 1989.

[30] Morse,A.S, Supervisory control of families of linear set-point controllers — part 1: exact matching, *IEEE Trans. Auto. Contr.*, **41**, 1413–1431, 1996.

[31] Musgrave,J.L, Guo,T-H, Wong,E, and Duyar,A, Real-time accommodation of actuator faults on a reusable rocket engine, *IEEE Trans. Control System Technology*, **5**, 100–109, 1997.

[32] Prett,D.M, and Garcia,C.E, *Fundamental Process Control*, (Boston: Butterworths), 1988.

[33] Rauch,H.E, Autonomous control reconfiguration, *IEEE Control Systems Magazine*, **15**, 37–48, 1995.

[34] Stenman,A, Gustafsson,F, and Ljung,L, *Just in time models for dynamical systems*, Report LiTH-ISY-R-1882, Linköping University, Sweden, 1996.

[35] Van Overschee,P, and De Moor,B, *Subspace Identification for Linear Systems: Theory, Implementation, Applications*, Kluwer Academic Publishers, 1996.

[36] Verhaegen,M, and Dewilde,P.M, Subspace model identification, parts 1 and 2, *Int. Jnl. Control*, **56**, 1187–1241, 1992.