A DEVELOPMENT SPACE FOR MODEL BASED PREDICTIVE CONTROL

M. Huzmezan [*,1] **J.M. Maciejowski** [**]

[*] *Cambridge University Engineering Department, Trumpington St., Cambridge, CB2 1PZ, U.K.*
[**] *Cambridge University Engineering Department, Trumpington St., Cambridge, CB2 1PZ, U.K.*

Abstract. The paper aims to provide the reader with a brief introduction to the *MBPC* technique, an answer to the question "Why a *MBPC* development space?" and to present a particular proposal for such a Development Space. Then a description of the graphic user interface is given followed by a presentation of the development space at work with a MIMO aircraft model.

Keywords. Computer-Aided Design (CAD), Predictive Control (*MBPC*)

1. INTRODUCTION

Model Based Predictive Control has achieved practical success in various applications. In this paper we describe a *MBPC* Development Space suitable for designers involved in this type of control.

We call this CAD application a "Development Space" because we would like to emphasise the difference between this and a standard MATLAB toolbox. Having a user interface which relies on SIMULINK features, in an open environment for developers, this application goes beyond a normal toolbox in terms of customisation but, not as far as an application dedicated to a particular *MBPC* control problem. The tuning interface is constructed in such a way as to allow the designer to adjust various design parameters while the simulation is running.

The paper aims to provide the reader, through its sections, with a brief introduction to the *MBPC* technique, an answer to the question "Why a *MBPC* development space?" and then to give a flavour of the graphic user interface followed by a demonstration of the development space at work with a MIMO aircraft model.

[1] e-mail: {mh2,jmm}@eng.cam.ac.uk, tel:+44-1223-339222

2. THE *MBPC* TECHNIQUE

The defining feature of MPBC is the repeated optimisation of a performance objective over a finite horizon extending from a future time (N_1) up to a prediction horizon (N_2) (Clarke, 1994*b*). Figure 1 characterises the way prediction is used within the *MBPC* control strategy. Given a set-point, a reference is produced and used within the optimisation of the cost function (1). Manipulating the control variable, over the control horizon, the algorithm drives the predicted output, over the prediction horizon, towards the reference.

The future control movement is determined by minimising the following cost function:

$$J(k) = \sum_{l=N_1}^{N_2} \|(\tilde{y}(k+l) - r(k+l)\|^2_{Q(l)} + \sum_{l=0}^{N_u} \|\Delta u(k+l)\|^2_{R(l)} \tag{1}$$

subject to constraints on:

- the inputs levels: $u_l(j) \leq u(j) \leq u_u(j)$ where $t \leq j \leq t + N_u - 1$
- the input rates: $\Delta u_l(j) \leq \Delta u(j) \leq \Delta u_u(j)$ where $t \leq j \leq t + N_u - 1$
- the output (in general other linear combinations of states): $y_l(j) \leq \tilde{y}(j) \leq y_u(j)$ where $t + N_1 \leq j \leq t + N_2$

$r(k)$: is the a priori known vector reference signal for outputs (at time k), $Q(l)$ and $R(l)$ are weights (independent of time k) and $\|\alpha\|^2_Q = \alpha^T Q \alpha$. It is assumed that $\Delta u(j) = 0$ for $j \geq t + N_u$. The optimisation is usually carried out using a quadratic program solver.

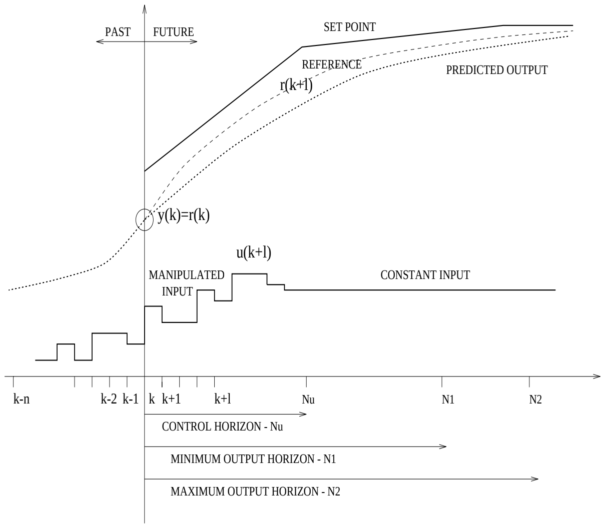

Fig. 1. The Prediction Strategy

The general structure of model based predictive control (*MBPC*) schemes is the following:

Optimiser contains the constrained cost function and it works iteratively in real time. The main task of the optimiser is to compute the present and future manipulated variable moves which minimises (1).

Predictor employing the internal plant model, obtained as a state space liniarization around the operating point, provides the optimiser with predicted values of states and outputs.

Internal model represents the plant. In this paper we use a state-space model for the plant. This is a natural choice for flight control, since the states can be chosen to be physical variables, and this correspondence can be exploited to some extent.

Observer provides state estimates, which can be used in the predictor.

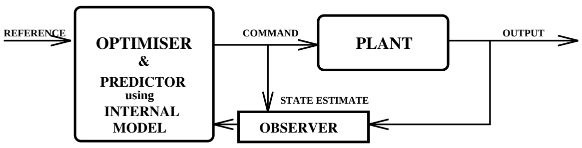

Fig. 2. The structure of *MBPC* schemes

3. AN *MBPC* DEVELOPMENT SPACE

The use of Model Based Predictive Control (*MBPC*) is increasing rapidly in the process industries, where it has proved to be extremely effective (Prett and Garcia, 1988; Clarke, 1994*a*), being a technique that deals naturally with constraints (Kouvaritakis *et al.*, 1992).

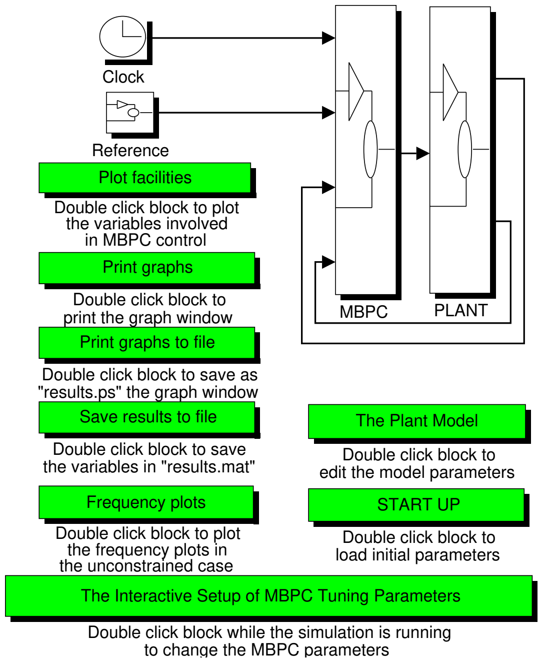

Fig. 3. The SIMULINK *MBPC* development space

Convenient general purpose *MBPC* design software for MIMO systems using state space internal models is not available. The existing software does not seem to allow a straightforward access from the MATLAB–SIMULINK environment (Morari and Ricker, 1995) or when MIMO plants are required to be controlled

(Scokaert *et al.*, 1995). In (Scokaert *et al.*, 1995) facilities such as an automatic procedure to define the optimisation horizons may fail in some particular cases. This limits the usefulness of the on-line identification feature which is provided by the toolbox.

Our Development Space is constructed using the MATLAB–SIMULINK software and its associated standard user interface. This environment can use state-space continuous or discrete *MBPC* internal models, and, the simulation plant model can be non-linear. It handles non square systems and disturbances. The Development Space assumes that either direct measurements of the plant states are available, or that they are estimated using a suitable observer, which the user must supply in the form of a Simulink block.

Explicit constraints on manipulated input changes, input variables, outputs and states are handled. Tools allow the user to handle unconstrained problems, characterised by an analytic solution, and constrained problems solved by means of hill climbing algorithms. identification. The use of nonlinear internal models for prediction and therefore, for the computation of the optimal control law is not currently possible.

4. THE GRAPHIC USER INTERFACE (GUI)

The main functions of the development space (see Figure 3) are: *Interactive setup of the* MBPC *tuning parameters*, *Plant model setup*, *Start up*, *Frequency plot*, *Plot facilities* and *Print graph*.

The tuning parameters of the *MBPC* method are requested by the dialog box shown in Figure 4. It is important to note here that any change in the parameters while the simulation is running is passed immediately to the *MBPC* algorithm. This feature can be used to understand the influence of the various parameters and the way of tuning them.

The *Plant model setup* function enables us to setup the internal plant model for the *MBPC* control algorithm within a text editor window via a template file.

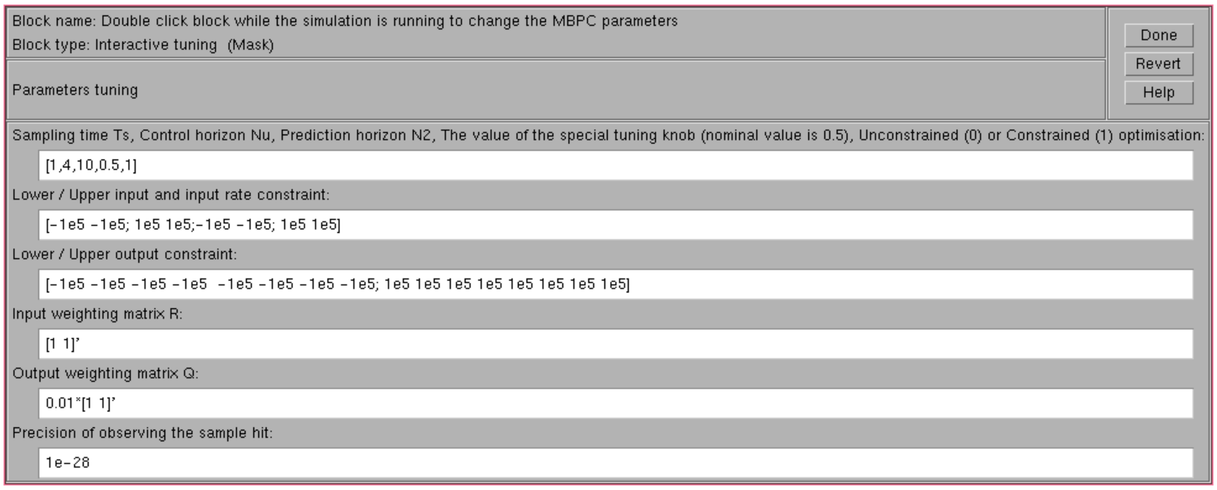

Fig. 4. The *MBPC* parameters dialog box

The environment shown in Figure 3 provides functions and has features that allow the user to design and test in simulation *MBPC* controllers for simple and complex MIMO plants. Such an approach supports once more the idea of a "development space" instead of a toolbox. As the only option, state space based internal models were used because of the reliability of the numerical algorithms involved and the possibility to interface them with other state-space based tools such as the subspace method – used in

The file to be edited has comments included that are intended to help the user in changing the parameters according to his particular problem.

Activating the *Start up* push button we will have all variables required by the *MBPC* method within the MATLAB workspace. SIMULINK uses specific variables from the workspace and passes them to the S-function as parameters. When a SIMULINK model is created, a new function, called *S-function*, becomes available in MATLAB. An *S-function* behaves like any other MATLAB function. A full description of the concept is given in (The MathWorks, 1995). Once all the variables are loaded and computed we can access the *Frequency plot* facilities that will provide us with information regarding the closed loop transfer function (complementary sensitivity) in the unconstrained case (see Figure 6). The frequency response of the controlled plant is saved as well in MVFR format for later use within the MFD Toolbox (Ford *et al.*, 1990).

When the simulation is finished, clicking on the *Plot facilities* push button brings forward a multiple graph (shown in Figure 7) showing measurements of the plant together with the variation of the states and the command provided by the *MBPC* controller.

5. DEVELOPMENT SPACE'S ARCHITECTURE

Functional blocks are: the reference block, the plant block and the *MBPC* block. The engine of the design environment is the *MBPC* controller, having the optimiser as a central point.

The blocks shown in Figure 5 are used to produce the signals required by the *MBPC* algorithm. The MATLAB workspace is provided with the signals involved in the control for analysis and plotting purposes. The optimiser is written as a SIMULINK S-function that has no states but runs a quadratic program. Only the first movement of the solution vector is used for control purposes.

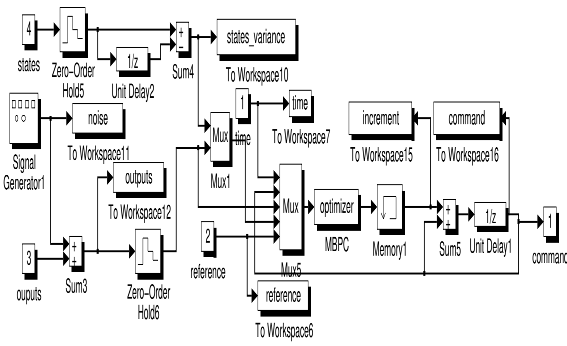

Fig. 5. The *MBPC* block

The plant block hosts the linear or nonlinear model of the plant. We note here that the main requirement of the Development Space is that the state measurements from the plant model are available as long as estimation of them is not used.

When a reduced model of the plant is employed via truncating the states, as the *MBPC* internal model, a state transformation is needed in order to provide the reduced model, employed for prediction, with the necessary initial measurement. Therefore, in the SIMULINK plant block we have included the necessary transformation matrix. Within the reference block the designer has to provide reference signals to the algorithm, reference signals which can be changed during the run. The reference trajectory is either generated during the simulation run, or it can be pre-computed and read from a file. The latter approach allows the *MBPC* to anticipate the future set-point movements.

6. THE DEVELOPMENT SPACE AT WORK

In order to gain more insight into the use this Development Space we are going to take a longitudinal aircraft model and pass it through all the stages required for a proper set-up and simulation using the *MBPC* environment.

The setting up involves two main steps:

(1) The definition of the plant model, used as internal model within *MBPC* algorithm. Therefore, we are using two models, one to represent the plant and one for the *MBPC* internal model.
(2) The definition of *MBPC* tuning parameters (control and prediction horizons, command and output weighting matrices).

$A =$	$\begin{bmatrix} -0.9825 & 0.0000 & -0.0007 & -0.0161 \\ 1.0000 & 0 & 0 & 0 \\ -2.1937 & -9.7758 & -0.0325 & 0.0743 \\ 77.3570 & -0.7675 & -0.2264 & -0.6684 \end{bmatrix}$
$B =$	$\begin{bmatrix} -2.4379 & 0.2912 & 0.2912 \\ 0 & 0 & 0 \\ 0.1837 & 9.8100 & 9.8100 \\ -6.4785 & 0 & 0 \end{bmatrix}$
$C =$	$\begin{bmatrix} 0 & -79.8667 & -0.0283 & 0.9996 \\ 0 & 0.0000 & 0.9996 & 0.0290 \end{bmatrix}$
$D =$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Table 1. The aircraft linear model

The first step can be accomplished once a linear continuous representation of the plant is produced using standard trim and liniarization procedures. Such a representation is used in order to define the discretized version of the model at the sampling time required. This discrete system is produced from the continuous time one by using specific commands from the MATLAB *μ-tools* or Control Toolbox. Model reduction can be applied in order to reduce the computational burden due to the large dimension of the internal model. This internal model will assume a simple way to model disturbances as being the difference between the real output and the model output and assumes that the same disturbance persists across the prediction horizon. The linear representation of the plant is given in Table ??

Constraints involved in the optimisation algorithm are saturation of actuators and actuator rate limits (see Table 2):

Limits	Constrained Variable Name	Unit
$-0.436 \leq \delta_E \leq 0.174$	elevator deflection saturation	rad
$-0.261 \leq \dot{\delta}_E \leq 0.261$	elevator deflection rate limits	rad/s
$0.009 \leq \delta_{Th} \leq 0.174$	engines throttle limits	-
$-0.028 \leq \dot{\delta}_{Th} \leq 0.028$	engine throttle slew rates	-

Table 2. Longitudinal channel actuator constraints

For the unconstrained case and the linear plant model we can plot the frequency response of the closed loop transfer function of the minimum and maximum singular values (see Figure 6).

The next step involves the definition of the plant within the corresponding block as a nonlinear model. This assumes that the plant is known and that we are able to implement it via SIMULINK standard blocks or a *S-function*. Both linear and nonlinear models can be implemented at the designer's choice. In the case of the nonlinear implementation we have to make sure that the assumption of measurable states is not violated so that, we are able to provide the *MBPC* block with these signals.

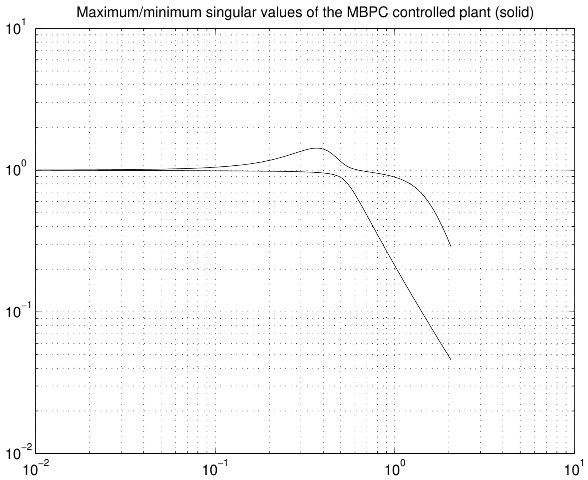

Fig. 6. The frequency plot facility output

Before starting a simulation it is as well to check the simulation parameters of the standard SIMULINK pull-down menu. The maximum step of the integration method has to be an adequate sub multiple of the sampling time at which the *MBPC* controller operates. In our case the *Min Step Size* $= 0.001 \times T_s$, *Max Step Size* $= 0.25 \times T_s$ and *Tolerance* $= 1 \times 10^{-3}$.

When running the simulation it is important to make sure that the precision required in detecting the sampling hit, the time spent in the on-line optimisation procedure and the offset time parameter present in most of the discrete time blocks of the *MBPC* controller are consistent. The sample time variable sets the sample time at which the discrete block is updated. It possible to specify an offset time that will give information about which blocks should be updated sooner or later then others (see (The MathWorks, 1995)). For our example $T_s = 0.1$ s and the offset is 1×10^{-8}. The amount of computation time for the quadratic programming solver block is varying from one internal model to another depending on the horizons used. Therefore, selecting the offset is a task that cannot be provided as an automatic feature of the Development Space due to the rules of thumb involved.

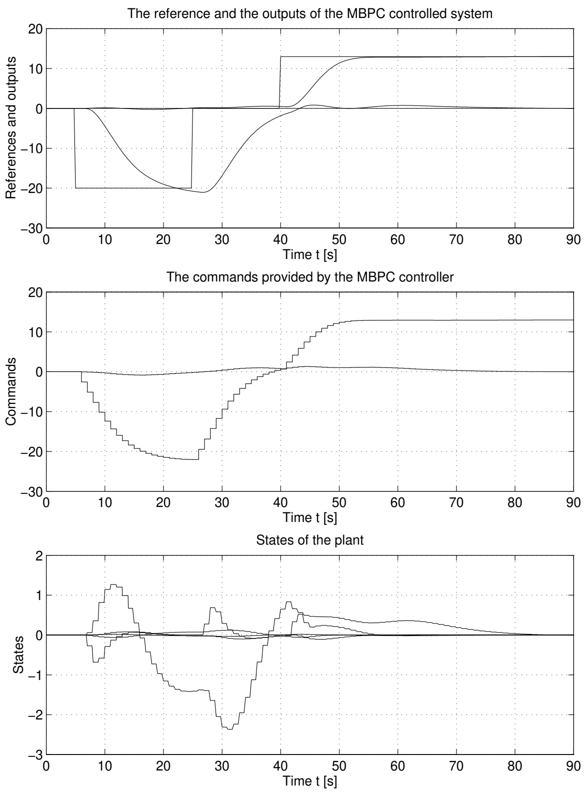

Fig. 7. The plotted simulation results

Once we have decided the initial tuning parameters we can proceed to time simulations. For the first stages of the tuning procedure it is recommended to have a short simulation time (six up to ten times the maximum time constant of the plant) and at the beginning to start in the unconstrained case and then to include the constraints. At this stage a trial and error process complemented by the designer's knowledge has to be performed in order to achieve the required performance for the controlled plant. For our example, the values within the dialog window for tuning parameters are chosen finally to be:

Sampling time, horizons, type of optimisation	[1,4,10,0.5,1]
Input and rate constraints	$\begin{bmatrix} -0.436 & +0.009 & +0.009 \\ +0.174 & +0.174 & +0.174 \\ -0.261 & -0.028 & -0.028 \\ +0.261 & +0.028 & +0.028 \end{bmatrix}$
States constraints	$\begin{bmatrix} -1e3 & -1e3 & -1e3 & -1e3 & -2 & -40 & 55 \\ 1e3 & 1e3 & 1e3 & 1e3 & 2 & 40 & 120 \end{bmatrix}$
Input weighting matrix	[3e5 6e5 6e5]'
Output weighting matrix	[1 1 1]'
Precision in observing the sample hit	1e-8

Table 3. Aircraft model *MBPC* tuning parameters

With these values the responses to step changes in vertical and air speed, shown in Figure 7, were obtained in the absence of disturbances. Any change of the design parameters will reflect instantaneously within the simulation being carried out.

7. CONCLUSIONS

In this paper the aim has been to provide the reader with information about developing and using a development space for *MBPC* control. The design environment was intended to be a useful tool for designers involved in this type of control. The architecture was defined in a flexible way to give opportunity to the designer to use this tool as a Development Space which can be customised to his needs. For example the use of predefined blocks and embedded algorithms will enable him to construct simulation models. Otherwise, a too dedicated user interface will limit the user to several predefined operations which will be too restrictive for an experienced user.

The features of the Development Space were defined after a careful look at the previous attempts such as: the Model Predictive Control Toolbox for use with Matlab (Morari and Ricker, 1995) and the Predictive Adaptive Control Environment (Scokaert *et al.*, 1995).

As the main conclusion we claim that we have developed a tool beyond the missing parts of the products reviewed above, maintaining the simplicity and flexibility required by such an environment.

8. ACKNOWLEDGEMENTS

We would like to acknowledge Prof. D.W.Clarke for providing us with the internal report (Scokaert *et al.*, 1995).

As well we would like to acknowledge Pembroke College Cambridge, The Lundgren Fund, ORS scholarship, Cambridge Overseas Trust and CT Taylor Fund by which Mihai Huzmezan is supported.

9. REFERENCES

Clarke, D. (1994*a*). *Advances in Model-Based Predictive Control.* Oxford University Press.

Clarke, D.W. (1994*b*). Advances in model-based predictive control. In: *Advances in Model-Based Predictive Control* (D. Clarke, Ed.). pp. 3–21. Oxford University Press.

Ford, M.P., J.M. Maciejowski and J.M. Boyle (1990). Multivariable Frequency Domain Toolbox. *The Mathworks Partner Series, GEC Engineering Reserch Centre, Cambridge Control Ltd.*

Kouvaritakis, B., J.A. Rossiter and A.O.T. Chang (1992). Stable Generalized Predictive Control: An Algorithm with Guaranteed Stability. *ProceedingsIEE — Part D* **139**(4), 349–362.

Morari, M. and N.L. Ricker (1995). Model Predictive Control Toolbox. *The Mathworks Partner Series.*

Prett, D.M. and C.E. Garcia (1988). *Fundamental Process Control.* first ed.. Butterworths Series in Chemical Engineering.

Scokaert, P.O.M., C.M. Chow and D.W. Clarke (1995). Simulink Predictive Adaptive Control Environment - Users Guide. *Report No. OUEL 2057/95.*

The MathWorks (1995). SIMULINK A program for Simulating Dynamic Systems. *The Mathworks Series.*