Performance Evaluation of Multiplexed Model Predictive Control for a Large Airliner in Nominal and Contingency Scenarios

E. N. Hartley, J. M. Maciejowski and K-V. Ling

***Abstract*— Model predictive control allows systematic handling of physical and operational constraints through the use of constrained optimisation. It has also been shown to successfully exploit plant redundancy to maintain a level of control in scenarios when faults are present. Unfortunately, the computational complexity of each individual iteration of the algorithm to solve the optimisation problem scales cubically with the number of plant inputs, so the computational demands are high for large MIMO plants. Multiplexed MPC only calculates changes in a subset of the plant inputs at each sampling instant, thus reducing the complexity of the optimisation. This paper demonstrates the application of multiplexed model predictive control to a large transport airliner in a nominal and a contingency scenario. The performance is compared to that obtained with a conventional synchronous model predictive controller, designed using an equivalent cost function.**

I. Introduction

Multiplexed model predictive control (MMPC) was proposed in [1], [2] as a method for designing a constrained controller with reduced computational complexity in comparison to a conventional synchronous model predictive controller (SMPC) [3], [4], [5]. Instead of optimising over all inputs at each time step, the controller optimises over channels, each containing a subset of the inputs, in a periodic cycle. The most recently calculated open-loop input trajectory is assumed for each channel not manipulated at the current time step.

By reducing the number of decision variables, the optimisation problem is simplified, and therefore, the control update rate can be made higher. As highlighted in [1], [2], [6], [7], doing "something sensible" sooner may very well be preferable to doing the "optimal" thing later, and redundancy between actuators and coupling between states in the plant dynamics can be exploited.

Whilst it is well known that the structure of the MPC optimisation problem [8], [9], [10] can be exploited so that the computational complexity of each iteration of the QP (quadratic programming) solver is linear in terms of the prediction horizon, in the absence of exploitation of any problem-specific structure, the complexity remains cubic in terms of the number of control inputs. Even under the (conservative) assumptions that the number of iterations needed to solve each constrained QP remains constant, and that the sampling time will be divided by the number of channels (so all channels may be "serviced" by the MMPC in the same time allowed for a single evaluation of SMPC), the cubic speed-up of each individual quadratic program (QP) is traded for only a linear reduction in the time available for computation.

In [11], [12], using a simulator [13] derived from data obtained from the ill-fated El-Al Flight 1862 [14], [15], [12], it was demonstrated that, under the assumption of availability of the linearised model of the damaged aircraft, and a "fly-by-wire" approach (in that each control surface was treated as individually manipulable), a predictive controller would be capable of stably carrying out commanded manœuvres. However, computational requirements were not considered.

Neglecting the lateral x and y positions and the effects of flaps and landing gear, which are very nonlinear, 14 states and 27 control inputs are considered. Because input rate constraints are specified [12], some notion of the control surface position must also be included in the state vector, increasing the effective number of states to 41. It is therefore unsurprising that, even by present standards, solving the relevant QP for a modest prediction horizon (e.g. $N = 4$), and a sampling period of 0.11 s is not trivial.

This paper demonstrates that the conclusions of [11] hold with application of MMPC, and moreover, that in doing so, the computational load is substantially reduced with minimal loss of closed-loop performance.

II. Model trim and linearisation

The simulation model used here is that of a modified Boeing 747-200, with each control surface individually movable, as used by GARTEUR AG-16 [12], [16]. In the nominal scenario, it is assumed that all control surfaces are fully functioning. The contingency scenario corresponds to the static failure configuration of [16] in which severe damage has occurred to the right-hand wing, including separation of two engines. The mass and inertias of the plant are therefore rather different to those in the nominal scenario, and a substantial number of the control surfaces are inoperative or have reduced effect.

The trim-points used for this experiment are for straight-and-level flight at a true airspeed of 133.8 ms^{-1}, and a height of 600 m, obtained using the trim algorithm from [16] for both scenarios. Linearisation is performed by averaging the results from Simulink's `linmodv5` command with positive and negative deflections about the trim point. For the rest of this paper, unless otherwise stated, for a given time t, plant states $x(t)$ and inputs, $u(t)$ are expressed in terms of deviations from the trim point.

This work was supported by EPSRC grant number EP/G030308/1.

E. N. Hartley and J. M. Maciejowski are with the Department of Engineering, Cambridge University, United Kingdom {enh20,jmm}@eng.cam.ac.uk

K-V. Ling is with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore ekvling@ntu.edu.sg

III. Control objectives and assumptions

The objective of the controller in this scenario is threefold. Firstly, the controller must maintain nominal stability of the aircraft at the desired operating point. Secondly, the controller must be able to reject unmeasured disturbances caused by wind gusts and turbulence. Thirdly, the controller must track changes in operating point.

The linearisations of both the nominal and contingency models for a straight-and-level flight trajectory both contain unstable modes, so it is evident that the tracking error due to unmeasured disturbances will grow with catastrophic effects, unless actively corrected. A conventional output-feedback controller might operate in continuous time (e.g. [17], [16]), and thus the design will be limited primarily by the bandwidth of the control actuators, and the quality of the data from instrumentation. On the other hand, MPC operates in discrete-time, and due to the relatively heavy computational requirements introduces a delay between output measurement and control realisation. Therefore, the rejection of unmodelled disturbances (wind, turbulence and plant-model mismatch due to nonlinearities) is also limited by the sampling rate, which in turn is limited by the time in which the control action can be calculated.

The complexity of a conventional MPC controller may be reduced in many ways. For example, groups of similar actuators can be "ganged" together to move in parallel as would be the case in a classical design. Alternatively, some constraints can be ignored, instead favouring a careful choice of cost weightings to ensure that these are not likely to be encountered, possibly at the expense of nominal performance. But ultimately, by removing these degrees of freedom, and reducing the granularity of the controller specification, the flexibility afforded by MPC is gradually eroded.

The reason for examining multiplexed MPC (MMPC) for this application is clear. There are many control inputs, and performance can be improved by faster sampling. MMPC allows the sampling rate to be increased, whilst retaining the ability to optimise the trajectory of each control surface individually, accounting for rate constraints, saturations and (although not considered in this study) flight-envelope requirements.

IV. Observer, discretisation and target calculator

The decision variable in the optimisation is parameterised in terms of $\Delta u(k) = u(kT_s) - u((k-1)T_s)$, allowing actuator rate constraints to be specified. A penalty on $\Delta u(k)$ instead of $u(kT_s)$ and a disturbance estimate [18] is often used to obtain steady-state offset free control. The solution to the discrete-time algebraic Riccati equation (DARE), corresponding to infinite-horizon unconstrained LQR cost-to-go, is often used as the terminal state weighting matrix for synchronous MPC [19]. This choice of terminal cost is appropriate for the current study, because it means that a change in the constrained control horizon does not also require a complete re-tuning of the controller. An equivalent terminal weighting matrix, obtained by solving the discrete-time periodic Riccati equation (DPRE) is used for MMPC [1]. In this scenario there is redundancy between some of the inputs, and therefore, a solution to the DARE or DPRE does not exist unless $u(kT_s)$ also has a non-zero weighting in the cost function. The inclusion of this cost means that a disturbance observer and a target calculator are required for offset-free tracking in the presence of persistent disturbances.

Unusually, height, heading angle and true airspeed are considered directly here as the tracked outputs, to permit a single controller design. The salient results presented here would apply equally to a more conventional multi-layer design with roll, pitch and yaw as the controlled variables of an inner loop.

A. Observer

A discrete-time Kalman filter-based observer is specified in continuous-time and implemented in discrete time using the MATLAB `kalmd` command. For all scenarios, the observer runs with sampling period $T_{\text{obs}} = 0.005$ s, chosen to be faster than the sampling period T_s of any of the predictive controller designs presented subsequently. It is assumed that the 14 plant states can be measured, but an additional 10 integrating disturbance states are specified, acting as state disturbances on roll, pitch and yaw and their first derivatives, and the true airspeed, angle of attack, sideslip angle and height. Thus, the Kalman filter estimates the state $\begin{bmatrix} x(t)^T & \xi(t)^T \end{bmatrix}^T$ of the augmented model

$$G_{\text{aug}}(s) = \left[\begin{array}{ccc} A & B_\xi & B \\ 0_{10\times 14} & 0_{10\times 10} & 0_{10\times 27} \\ \hline C & 0_{14\times 10} & 0_{14\times 27} \end{array}\right] \tag{1}$$

where $0_{n\times m}$ denotes a matrix of zeros of size n by m, $x(t)$ is the plant state and $\xi(t)$ is the disturbance state. A further open-loop propagation is made at the sampling rate of the MPC controller, T_s by one time step, based on the previous control input. This allows a full time step at the MPC controller's sampling rate in which to perform calculation in real-time.

B. Discretisation

For a given controller sampling time T_s, using the usual zero-order-hold discretisation (e.g. [20]), let:

$$\Phi = e^{AT_s}, \Gamma = \int_0^{T_s} e^{A(T_s-\tau)} B \,\mathrm{d}\tau, \text{ and } \Gamma_\xi = \int_0^{T_s} e^{A(T_s-\tau)} B_\xi \,\mathrm{d}\tau.$$

C. Target calculator

The reference signal is assumed to be piecewise constant,

$$y_r(t) = y_r(k), t \in [kT_s, (k+1)T_s), \quad k \in \mathbb{Z}^+ \tag{2}$$

where $y_r(k) = \begin{bmatrix} V_{\text{TAS},r}(k) & \psi_r(k) & h_r(k) \end{bmatrix}^T$ consists of a column vector comprising the target true airspeed, yaw angle and altitude. The target calculator can then be specified in the usual manner (e.g. [21], [22], [23]).

Algorithm 1 (Target calculator):

$$\min_{x_\infty(k), u_\infty(k)} \begin{bmatrix} x_\infty(k)^T & u_\infty(k)^T \end{bmatrix} Q_d \begin{bmatrix} x_\infty(k) \\ u_\infty(k) \end{bmatrix} \tag{3a}$$

subject to

$$\begin{bmatrix} \Phi - I & \Gamma \\ H_r & 0 \end{bmatrix} \begin{bmatrix} x_\infty(k) \\ u_\infty(k) \end{bmatrix} = \begin{bmatrix} -\Gamma_\xi \hat{\xi}(k) \\ y_r(k) \end{bmatrix} \tag{3b}$$

and

$$u_{\min} \le u_\infty \le u_{\max}. \tag{3c}$$

The matrix H_r is given by

$$H_r = \begin{bmatrix} - & \hat{e}_{V_{\mathrm{TAS}}}^T & - \\ - & \hat{e}_\psi^T & - \\ - & \hat{e}_h^T & - \end{bmatrix} \tag{3d}$$

where $\hat{e}_\Omega$ is the ith column of the 14×14 identity matrix, where i is the numerical index of the state $\Omega \in \{V_{\mathrm{TAS}}, \psi, h\}$. Matrix $Q_d > 0$ is the weighting matrix used in the MPC controller design presented in the following section. Within the scope of this study, it is assumed that there is always a feasible solution satisfying (3b) and (3c) however these constraints can be "softened" to allow a best-effort solution when no feasible solution exists.

V. Baseline synchronous MPC (SMPC) design

A synchronous MPC design is implemented as a baseline for comparison. In order to compare the performance of MPC controllers operating at different sampling rates, the cost function is specified in continuous time. This is slightly complicated by the desire to include Δu in the discrete-time cost function, as this is discontinuous. Define

$$\tilde{x}(t) = \begin{bmatrix} x(t) \\ u(t - T_s) \end{bmatrix} \tag{4}$$

and

$$\tilde{e}(t) = \begin{bmatrix} x(t) \\ u(t - T_s) \end{bmatrix} - \begin{bmatrix} x_\infty(t) \\ u_\infty(t) \end{bmatrix}. \tag{5}$$

It should be noted that $u(t)$ is piecewise constant. The cost function can be specified in two parts. In continuous time, for $Q > 0$ the cost of being at state $x(t)$ and applying constant input $u(k)$ for period T_s

$$\ell_c(k) = \int_{kT_s}^{(k+1)T_s} \tilde{e}(\tau)^T Q \tilde{e}(\tau)\, \mathrm{d}\tau \tag{6}$$

subject to

$$\dot{\tilde{x}}(\tau) = \overbrace{\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}}^{\tilde{A}} \tilde{x}(\tau) + \begin{bmatrix} B \\ \delta(\tau - (k+1)T_s) I \end{bmatrix} \Delta u(k) \tag{7}$$

where $\delta(\cdot)$ denotes the Dirac delta function. The discrete-time equivalent cost of the continuous part of the cost function can be found by calculating a matrix exponential [24] to provide discrete-time weighting matrices for the stage cost function, assuming zero-order hold on the inputs:

$$\ell_d(k) = \begin{bmatrix} \tilde{e}(kT_s) \\ \Delta u(k) \end{bmatrix}^T \begin{bmatrix} Q_d & N_d \\ N_d^T & R_d \end{bmatrix} \begin{bmatrix} \tilde{e}(kT_s) \\ \Delta u(k) \end{bmatrix}. \tag{8}$$

This stage cost does not sufficiently capture the desire to penalise moves in the control input between time steps. The purpose of the penalty on Δu in the discrete-time cost function is to limit the rate of change of the control surface. To capture this, let $R_\Delta = R_d + R_{\Delta 0}/T_s^2$, where the matrix $R_{\Delta 0} > 0$ is a design parameter. The synchronous MPC problem is shown in Algorithm 2, using the conditional notation $\tilde{x}((i+k)T_s|iT_s)$ to denote a prediction of the augmented state $\tilde{x}$ at time $(i+k)T_s$ given the measurements at time iT_s. The "Q" norm $\|x\|_Q^2$ is used as a short-hand notation for $x^T Q x$ with a compatibly sized matrix Q and column vector x.

Algorithm 2 (Synchronous MPC): Minimise:

$$\begin{aligned} J_d(i) = {} & \|\tilde{e}((i+N)T_s|iT_s)\|_{P_d}^2 \\ & + \sum_{k=0}^{N-1} \Big(\|\tilde{e}((i+k)T_s|iT_s)\|_{Q_d}^2 \\ & + \|\Delta u(k|i)\|_{R_\Delta}^2 + 2\tilde{e}((i+k)T_s|iT_s)^T N_d \Delta u(k|i) \Big) \end{aligned}$$

subject to

$$\begin{aligned} \tilde{x}(iT_s|iT_s) &= \tilde{x}(iT_s) && \text{(from observer)} \\ \hat{\xi}(i|i) &= \hat{\xi}(i) && \text{(from observer)} \\ \hat{\xi}(i+k+1|i) &= \hat{\xi}(i+k|i) \end{aligned}$$

$$\begin{aligned} \tilde{x}((i+k+1)T_s|iT_s) = {} & \begin{bmatrix} \Phi & \Gamma \\ 0 & I \end{bmatrix} \tilde{x}((i+k)T_s|iT_s) \\ & + \begin{bmatrix} \Gamma_\xi \\ 0 \end{bmatrix} \hat{\xi}((i+k)|i) + \begin{bmatrix} \Gamma \\ I \end{bmatrix} \Delta u(i+k|i) \end{aligned}$$

and

$$\begin{aligned} \Delta u_{\min} \le \Delta u((i+k)|i) \le \Delta u_{\max} \\ u_{\min} \le u((i+k)T_s|iT_s) \le u_{\max}. \end{aligned}$$

The matrix P_d is the solution to the discrete-time algebraic Riccati equation (DARE) with cross-terms, associated with the above stage costs and prediction models (e.g. [25]).

VI. Multiplexed MPC (MMPC) implementation

The same target calculator and observer designs as used for the synchronous MPC (SMPC) case are used for the multiplexed MPC implementation. However, a different optimisation problem is posed [1], [2], [6], [7]. The inputs are grouped into channels. At each time step, the optimisation may only manipulate the inputs in the active channel, and the channels become active in a repeating sequence (see e.g. [7] for a graphical explanation). For this application, the 27 inputs are grouped into 7 channels, 6 of which have 4 inputs and 1 of which has 3 inputs (Table I). The channels are defined so that symmetric pairs of similar control surfaces are grouped together. Letting M be the number of channels, and defining $\sigma(i) = \mod(i-1, M) + 1$, the optimisation problem is summarised in Algorithm 3.

Algorithm 3 (Multiplexed MPC with cross-terms): Minimise

$$\begin{aligned} J_d(i) = {} & \|\tilde{e}((i+M(N-1)+1)T_s|iT_s)\|_{P_{\sigma(i+M(N-1)+1)}}^2 \\ & + \sum_{k=0}^{M(N-1)} \Big(\|\tilde{e}((i+k)T_s|iT_s)\|_{Q_d}^2 + \|\Delta u((i+k)|i)\|_{R_{\Delta,\sigma(i+k)}}^2 \\ & + \tilde{e}((i+k)T_s|iT_s)^T N_{d,\sigma(i+k)} \Delta u((i+k)|i) \Big) \end{aligned}$$

TABLE I

MMPC CHANNEL ALLOCATION

Channel	Inputs[a]			
1	Stabiliser	Null	Up. Rudder	Lo. Rudder
2	Engine 1	Engine 2	Engine 3	Engine 4
3	R. I/B Ail.	L. I/B Ail.	R. O/B Ail.	L. I/B Ail.
4	Sp. 1	Sp. 12	Sp. 2	Sp. 11
5	Sp. 3	Sp. 10	Sp. 4	Sp. 9
6	Sp. 5	Sp. 8	Sp. 6	Sp. 7
7	R. I/B Elev.	L. I/B Elev.	R. O/B Elev.	L. O/B Elev.

[a]Key: R. Right, L. Left, Up. Upper, Lo. Lower, I/B inboard, O/B outboard, Ail. aileron, Elev. elevator, Sp. spoiler panel

subject to

$$\begin{aligned}
\tilde{x}(iT_s|iT_s) &= \tilde{x}(iT_s) && \text{(from observer)}\\
\hat{\xi}(i|i) &= \hat{\xi}(i) && \text{(from observer)}\\
\hat{\xi}(i+k+1|i) &= \hat{\xi}(i+k|i)\\
\tilde{x}((i+k+1)T_s|iT_s) &= \begin{bmatrix}\Phi & \Gamma\\ 0 & I\end{bmatrix}\tilde{x}((i+k)T_s|iT_s)\\
&\quad + \begin{bmatrix}\Phi_\xi\\ 0\end{bmatrix}\hat{\xi}((i+k)|i) + \overline{\Gamma}_{\sigma(i+k)}\Delta u(i+k|i)
\end{aligned}$$

with

$$\Delta u(i+k|i) = \Delta u(i+k|i-1) \text{ if } \sigma(i+k) \neq \sigma(i)$$

or, when $\sigma(i+k) = \sigma(i)$:

$$\begin{aligned}
\Delta u_{\min,\sigma(i+k)} &\leq \Delta u((i+k)|i) \leq \Delta u_{\max,\sigma(i+k)}\\
u_{\min,\sigma(i+k)} &\leq H_{\sigma(i+k)}u((i+k)T_s|iT_s) \leq u_{\max,\sigma(i+k)}.
\end{aligned}$$

The matrices $\overline{\Gamma}_{\sigma(i+k)}$ contain the N columns of $\begin{bmatrix}\Gamma^T & I\end{bmatrix}^T$ corresponding to the inputs in channel $\sigma(i+k)$, and the matrix $H_{\sigma(i+k)}$ contains the rows of the $n_u \times n_u$ identity matrix corresponding to the inputs in channel $\sigma(i+k)$, where n_u is the total number of plant inputs.

A. Discrete-time periodic Riccati equation (DPRE) with cross-terms

Since the discrete-equivalent cost of the continuous-time cost function is being used to allow direct comparison with the synchronous case, the solution to the discrete-time periodic Riccati equation *with cross terms* should be used as the terminal cost, or "cost-to-go":

$$\begin{aligned}
P_{\sigma(i)} &= Q_d + \tilde{\Phi}^T P_{\sigma(i+1)}\tilde{\Phi}\\
&\quad - \left(\tilde{\Phi}^T P_{\sigma(i+1)}\tilde{\Gamma}_{\sigma(i)} + N_{\sigma(i)}\right)\left(\tilde{\Gamma}^T_{\sigma(i)}P_{\sigma(i+1)}\tilde{\Gamma}_{\sigma(i)} + R_{\sigma(i)}\right)^{-1}\\
&\qquad \times \left(\tilde{\Gamma}^T_{\sigma(i)}P_{\sigma(i+1)}\tilde{\Phi} + N^T_{\sigma(i)}\right). \quad (9)
\end{aligned}$$

The solution can be found by performing a loop-shifting transformation [26] to obtain matrices

$$\overline{\Phi}_{\sigma(i)} = \tilde{\Phi} - \tilde{\Gamma}_{\sigma(i)}R^{-1}_{\sigma(i)}N^T_{\sigma(i)} \quad (10a)$$

$$\overline{\Gamma}_{\sigma(i)} = \tilde{\Gamma}_{\sigma(i)} \quad (10b)$$

$$\overline{Q}_{\sigma(i)} = Q_d - N_{\sigma(i)}R^{-1}_{\sigma(i)}N^T_{\sigma(i)} \quad (10c)$$

$$\overline{R}_{\sigma(i)} = R_{\sigma(i)} \quad (10d)$$

$$\overline{N}_{\sigma(i)} = 0 \quad (10e)$$

and then solving the standard DPRE for the periodic set of matrices $(\overline{\Phi}_{\sigma(i)}, \overline{\Gamma}_{\sigma(i)}, \overline{Q}_{\sigma(i)}, \overline{R}_{\sigma(i)})$ using the method proposed in [27].

B. QP formulation

Since the state vector is large in comparison to the prediction horizon, a dense QP formulation (e.g. [3]) is used. This is solved using an implementation of the dual active-set algorithm of [28] in Embedded MATLAB, so as to integrate easily with the plant model in Simulink.

VII. PERFORMANCE COMPARISONS

A. Trajectory Tracking (Qualitative Comparison)

Figure 1 shows the tracking of a trajectory that is piecewise continuous in its first derivatives, consisting of a 90° change in yaw angle, a 200 m descent and a 10 ms^{-1} reduction in true air speed. For this test, $N = 4$ for the MMPC and the SMPC (in real terms, this means that the MMPC covers $M(N-1)+1 = 22$ time steps). For the MMPC a sampling time of $T_s = 0.02$ s is chosen, and for the SMPC, $T_s = 0.11$ s is chosen so that the controllers have equal durations of prediction horizons.

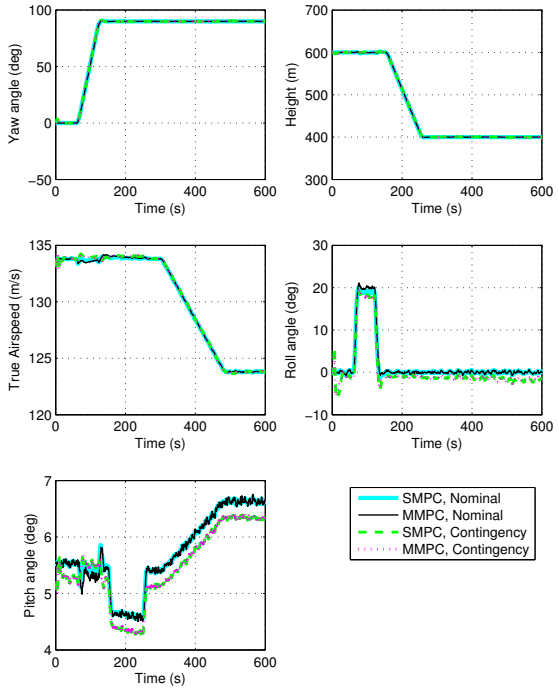

Fig. 1. Comparison of trajectories using SMPC and MMPC for nominal and contingency scenarios (actual state values)

For both the nominal and contingency scenarios, the MMPC is equally as capable as the SMPC of performing the manœuvres stably, despite presence of wind and turbulence. Whilst the behaviours of the pitch angle are different in the nominal and contingency cases, there is almost no difference between SMPC and MMPC in the two cases. The timing data obtained on a 2.8 GHz Mac Pro running MATLAB R2011a under Scientific Linux 6.0 in VirtualBox 4.0.12 using a single processor core is shown in Table II. The mean running time

for the QP solver is taken from the Simulink profiler tool. This time is then scaled by the ratio of the maximum number of QP solver iterations per problem to the mean number of iterations to obtain an estimate for the maximum time needed, and then this is presented as a percentage of the sampling time. It should be noted that although the SMPC implementation comfortably fits within the defined sampling period on the desktop workstation, an embedded application would not be afforded so powerful a processor.

TABLE II

TIMING DATA COMPARISON OF SMPC AND MMPC (PER QP)

MPC Type Scenario	SMPC Nominal	MMPC Nominal	SMPC Contingency	MMPC Contingency
T_s	0.11 s	0.02 s	0.11 s	0.02 s
Mean #iter/QP	16.02	2.71	18.48	3.26
Max #iter/QP	38	17	45	13
Mean T_{sol} (ms)	4.15	0.06	7.13	0.03
Max T_{sol} (ms)	9.84	0.36	17.36	0.13
(As % of T_s)	8.94%	1.81%	15.78%	0.66%

B. Straight and Level Flight Regulation (Quantitative Comparison)

To compare the rejection of disturbances quantitatively, the continuous-time integral of the tracking error, weighted by the continuous-time LQR cost weighting matrix Q used in equation (6) is considered for a $T_{\text{end}} = 60$ s period of straight-and-level flight in presence of wind and turbulence

$$\int_0^{T_{\text{end}}} x(t)^T Q x(t)\, \mathrm{d}t. \tag{11}$$

The results using this performance metric are presented in Table III and Table IV for the nominal scenario and the contingency scenario respectively (letting T_{hor} be the length of the prediction horizon in seconds).

In the nominal scenario, it is clear that for SMPC, reducing the sampling period is the best way to improve disturbance rejection, and that the effects of an increased prediction horizon are negligible. In comparison, for this example, MMPC operating at $T_s = 0.01$ s gives a performance metric equivalent to SMPC at $T_s = 0.02$ s. This is despite, for this configuration, seven sampling instants being required for all channels to be serviced. A similar equivalence can be seen between SMPC with $T_s = 0.08$ s and MMPC with $T_s = 0.04$ s. This suggests that MMPC is exploiting the plant redundancy.

For the contingency scenario, it appears that lengthening the prediction horizon is the most effective way of reducing the tracking error. However, as the prediction horizon becomes longer, the computation time becomes prohibitive (for $N = 29$ and $T_s = 0.01$ s the simulation did not finish within 8 hours, although it can be argued that using a sparse MPC formulation [8], [10] might be considered more appropriate for this length of prediction horizon).

C. Straight and Level Flight Regulation (Numerical Performance)

Tables V and VI show the maximum number of QP iterations of the active set QP solver over the 60 s. In addition

TABLE III

NOMINAL SCENARIO REGULATION PERFORMANCE OVER 60 S

(a) SMPC

N	T_{hor}	Sampling period T_s (s)						
		0.01	0.02	0.04	0.08	0.07	0.14	0.28
8	$8T_s$	0.350	0.370	0.411	0.496	0.475	0.621	0.881
15	$15T_s$	0.349	0.369	0.410	0.497	0.475	0.622	0.880
22	$22T_s$	0.348	0.368	0.410	0.496	0.475	0.622	0.880
29	$29T_s$	–	–	0.409	0.496	0.475	0.622	0.880

(b) MMPC

N	T_{hor}	Sampling period T_s (s)				
		0.01	0.02	0.03	0.04	0.05
2	$8T_s$	0.368	0.405	0.444	0.481	0.525
3	$15T_s$	0.370	0.410	0.454	0.496	0.545
4	$22T_s$	0.369	0.409	0.453	0.493	0.541
5	$29T_s$	0.369	0.408	0.451	0.491	0.540
6	$36T_s$	0.368	0.408	0.450	0.490	0.539
7	$43T_s$	0.368	0.407	0.449	0.489	0.539

TABLE IV

CONTINGENCY SCENARIO REGULATION PERFORMANCE OVER 60 S

(a) SMPC

N	T_{hor}	Sampling period T_s (s)						
		0.01	0.02	0.04	0.08	0.07	0.14	0.28
8	$8T_s$	57.72	55.37	51.63	46.83	47.72	45.35	57.97
15	$15T_s$	53.72	48.56	41.38	36.02	36.57	40.02	57.42
22	$22T_s$	50.38	43.23	35.33	33.58	32.92	40.07	58.03
29	$29T_s$	–	38.95	31.93	33.67	32.62	39.99	57.70

(b) MMPC

N	T_{hor}	Sampling period T_s (s)				
		0.01	0.02	0.03	0.04	0.05
2	$8T_s$	56.32	52.50	49.26	46.83	45.05
3	$15T_s$	52.72	46.62	42.16	39.47	38.21
4	$22T_s$	49.45	41.84	37.29	35.26	34.21
5	$29T_s$	46.51	38.08	34.02	32.74	32.48
6	$36T_s$	43.87	35.21	31.79	31.49	31.63
7	$43T_s$	41.50	33.06	30.19	30.76	31.38

to each QP iteration being simpler for the MMPC, it can be seen that in this example, fewer iterations are required. This is primarily because this particular MPC problem is dominated by the input constraints, and therefore, each QP in the MMPC problem has significantly fewer constraints associated with it than the corresponding SMPC problem.

VIII. CONCLUSIONS

This paper has demonstrated the use of MMPC for a large airliner. The MMPC formulation is extended to allow cross-terms between inputs and states in the cost function, thus allowing the discrete equivalent integral cost functions of [24] to be used directly. The fault-tolerant properties of MPC demonstrated in [11] are retained by the multiplexed formulation, whilst computational complexity is significantly reduced. For this example, not only are the QP iterations simpler due to fewer decision variables at each sampling instant, the reduced number of constraints (only the constraints on input magnitudes and increments for the current channel must be posed) means that the maximum number of iterations seen over the course of a simulation is also greatly reduced. It is further noted, that due to the redundancy in the plant,

TABLE V

NOMINAL SCENARIO MAXIMUM NUMBER OF QP ITERATIONS PER STEP

(a) SMPC

N	T_{hor}	Sampling period T_s (s)						
		0.01	0.02	0.04	0.08	0.07	0.14	0.28
8	$8T_s$	69	72	72	72	72	67	71
15	$15T_s$	137	136	130	127	133	147	119
22	$22T_s$	205	195	193	213	216	214	162
29	$29T_s$	–	–	270	286	299	250	211

(b) MMPC

N	T_{hor}	Sampling period T_s (s)				
		0.01	0.02	0.03	0.04	0.05
2	$8T_s$	9	9	9	9	9
3	$15T_s$	13	13	13	13	13
4	$22T_s$	17	17	17	17	17
5	$29T_s$	21	21	21	21	21
6	$36T_s$	25	25	25	25	25
7	$43T_s$	29	29	29	29	29

TABLE VI

CONTINGENCY SCENARIO MAXIMUM NUMBER OF QP ITERATIONS PER STEP

(a) SMPC

N	T_{hor}	Sampling period T_s (s)						
		0.01	0.02	0.04	0.08	0.07	0.14	0.28
8	$8T_s$	103	99	96	93	96	87	83
15	$15T_s$	182	178	175	167	166	155	136
22	$22T_s$	264	255	253	237	240	218	207
29	$29T_s$	–	336	349	291	305	296	268

(b) MMPC

N	T_{hor}	Sampling period T_s (s)				
		0.01	0.02	0.03	0.04	0.05
2	$8T_s$	5	5	5	5	5
3	$15T_s$	8	9	9	8	9
4	$22T_s$	11	13	13	12	13
5	$29T_s$	15	17	16	16	15
6	$36T_s$	19	21	20	20	18
7	$43T_s$	23	25	24	23	22

the sampling period when using MMPC need not be faster by a factor of the number of channels M to obtain the same tracking performance as with SMPC for a given integral cost function.

A possibility for future work would be a multiplexed approach to the target calculation problem, to match the multiplexed control problem, and to compare the performance of MMPC with SMPC with sparse QP formulations.

REFERENCES

[1] K. V. Ling, J. M. Maciejowski, and B. F. Wu, "Multiplexed model predictive control," in *Proc. 16th IFAC World Congress*, Prague, Jul. 3–8 2005.

[2] A. G. Richards, K. V. Ling, and J. M. Maciejowski, "Robust multiplexed model predictive control," in *Proceedings of the European Control Conference*, Kos, Greece, July 2–5 2007.

[3] J. M. Maciejowski, *Predictive Control with Constraints*. Pearson Education, 2002.

[4] E. F. Camacho and C. Bordons, *Model predictive control*. London: Springer-Verlag, 2004.

[5] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Publishing, 2009.

[6] K. V. Ling, J. M. Maciejowski, and B. F. Wu, "Computing the cost of multiplexed MPC," in *Proc. 10th Int. Conf. on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, Dec. 2008, pp. 582–587.

[7] K. V. Ling, J. M. Maciejowski, A. Richards, and B. F. Wu, "Multiplexed model predictive control," University of Cambridge," Technical report CUED/F-INFENG/TR.657 (Accepted for Automatica), 2010.

[8] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *J. Optim. Theory and Appl.*, vol. 99, no. 3, pp. 723–757, December 1998.

[9] R. Bartlett and L. Biegler, "QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming," *Optimization and Engineering*, vol. 7, no. 1, pp. 5–32, 2006.

[10] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," in *IFAC World Congress*, Seoul, Korea, July 2008, pp. 6974–6997.

[11] J. M. Maciejowski and C. N. Jones, "MPC fault-tolerant flight control case study: Flight 1862," in *IFAC Safeprocess Conf.*, June 2003.

[12] C. Edwards, T. Lombaerts, and H. Smaili, Eds., *Fault Tolerant Flight Control: A Benchmark Challenge*, ser. Lecture Notes in Control and Information Sciences. Springer, 2010.

[13] C. A. A. M. van der Linden, *DASMAT — Delft University Aircraft Simulation Model and Analysis Tool*, TU Delft, 1996.

[14] M. H. Smaili, "Flight data reconstruction and simulation of El Al Flight 1862," Master's thesis, Delft University of Technology, 1997.

[15] M. H. Smaili and J. A. Mulder, "Flight data reconstruction and simulation of the 1992 Amsterdam Bijlmermeer airplane accident," in *AIAA Modeling and Simulation Technologies Conference and Exhibit, AIAA-2000-4586*, Denver, CO, August 2000.

[16] C. van der Linden, H. Smaili, A. Marcos, G. Balas, D. Breeds, S. Runham, C. Edwards, H. Alwi, T. Lombaerts, J. Groeneweg, R. Verhoeven, and J. Breeman, "GARTEUR RECOVER benchmark," 2011. [Online]. Available: http://www.faulttolerantcontrol.nl

[17] R. van Keulen, "Real-time simulation and analysis of the automatic flight control system of the Boeing 747-200," Master's thesis, TU Delft, 1991.

[18] J. M. Maciejowski, "The implicit daisy-chaining property of constrained predictive control." *Appl. Math. and Comp. Sci.*, vol. 8, no. 4, pp. 101–117, 1998.

[19] D. Chmielewski and V. Manousiouthakis, "On constrained infinite-time linear quadratic optimal control," *Systems & Control Letters*, vol. 29, no. 3, pp. 121–129, November 1996.

[20] K. J. Aström and B. Wittenmark, *Computer-controlled systems*, 2nd ed. Prentice Hall, 1990.

[21] K. R. Muske and T. A. Badgwell, "Disturbance modeling for offset-free linear model predictive control," *Journal of Process Control*, vol. 12, no. 5, pp. 617–632, 2002.

[22] G. Pannocchia and E. C. Kerrigan, "Offset-free receding horizon control of constrained linear systems," *AIChE Journal*, vol. 51, no. 12, pp. 3134–3146, 2005.

[23] F. Borrelli and M. Morari, "Offset free model predictive control," in *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, December 12–14 2007, pp. 4663–4668.

[24] C. F. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Trans. Aut. Control*, vol. 23, no. 3, pp. 395–404, 1978.

[25] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital control of dynamic systems*, 2nd ed. Reading, Mass.: Addison-Wesley, 1990.

[26] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.

[27] J. Sreedhar and P. Van Dooren, "Solution of the periodic discrete-time Riccati and Lyapunov equations," in *Proc. 2nd IEEE Conf. on Decision and Control*, San Antonio, Texas, December 1993, pp. 361–362.

[28] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Math. Prog.*, vol. 27, no. 1, pp. 1–33, 1983.