Predictive Control of a Boeing 747 Aircraft using an FPGA ⋆

Edward N. Hartley * **Juan L. Jerez** ** **Andrea Suardi** **
Jan M. Maciejowski * **Eric C. Kerrigan** ***
George A. Constantinides **

* *Engineering Department, Cambridge University, CB2 1PZ, United Kingdom (e-mail: {enh20,jmm}@eng.cam.ac.uk).*
** *Dept. of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ, United Kingdom (e-mail: {jlj05,a.suardi,gac1}@imperial.ac.uk)*
*** *Dept. of Electrical and Electronic Engineering and Department of Aeronautics, Imperial College London, SW7 2AZ, United Kingdom (e-mail: e.kerrigan@imperial.ac.uk)*

Abstract: New embedded predictive control applications call for more efficient ways of solving quadratic programs (QPs) in order to meet demanding real-time, power and cost requirements. A single precision QP-on-a-chip controller is proposed, implemented in an field-programmable gate array (FPGA) with an iterative linear solver at its core. A novel offline scaling procedure is introduced to aid the convergence of the reduced precision solver. The feasibility of the proposed approach is demonstrated with a real-time hardware-in-the-loop (HIL) experimental setup where an ML605 FPGA board controls a nonlinear model of a Boeing 747 aircraft running on a desktop PC through an Ethernet link. Simulations show that the quality of the closed-loop control and accuracy of individual solutions is competitive with a conventional double precision controller solving linear systems using a Riccati recursion.

1. INTRODUCTION

For large predictive control problems, the computation and storage of the explicit solution (Bemporad et al., 2002) is not a feasible option and the QP has to be solved online at every sampling instant. For a wide range of control applications that could benefit from employing predictive control, this approach is still either too computationally demanding to meet real-time requirements with conventional computing platforms, or the cost and power requirements of the necessary hardware are unfavourable.

FPGAs are reconfigurable chips that can be customised for a specific application and have the potential to address all of these problems. For this class of devices, transistor scaling is still improving the performance and reducing the cost in every new technology generation, meaning that it has become possible to implement complex floating-point algorithms exploiting wide parallelisation. For predictive control applications, FPGAs are easily embedded as a system component, and offer cycle-accurate timing guarantees. The low clock frequencies compared to high-performance microprocessors translate to lower power consumption, whereas the performance can be regained through parallelisation and customisability.

Resource usage depends on computation precision. More operations can be performed in parallel by using fewer bits to represent each number, hence numerical issues are of crucial importance for an efficient FPGA implementation. In this paper a novel procedure for scaling the QP matrices allows implementation of an interior-point solver in single precision floating-point, where the solution of systems of linear equations, which is the computational bottleneck of the algorithm, is achieved with a parallel implementation of an un-preconditioned iterative linear solver. The resulting solution accuracy is competitive with a double precision implementation of a factorisation-based method, even at the later iterations of the interior-point method, when the equations to be solved become ill-conditioned.

There have been several prior FPGA implementations of predictive controllers. In Ling et al. (2008), a high-level synthesis tool that converts a C-like description into a hardware description was used to show that it was feasible to achieve speeds comparable to MATLAB implementations for an aircraft example with four states. Vouzis et al. (2009) proposed a mixed software/hardware implementation where the core matrix computations are carried out in parallel custom hardware, whereas the remaining operations are implemented in a general purpose microprocessor. The performance was evaluated on two state systems. Recently, Wills et al. (2012) proposed a (very) reduced precision interior-point implementation that solved a condensed QP with impressive computation times and demonstrated its feasibility on a experimental setup with a real two state plant.

For this implementation, the control of a nonlinear model of a Boeing 747-200 is considered. The resulting QP, posed

⋆ This work was supported by the EPSRC (Grants EP/G031576/1 and EP/I012036/1) and the EU FP7 Project EMBOCON, as well as industrial support from Xilinx, the Mathworks, and the European Space Agency.

in a non-condensed form, considering 12 states and 17 constrained inputs, is substantially larger than any prior FPGA demonstration. In addition, controlling this plant is significantly more numerically challenging than in any related prior work. A faster-than-real-time setup where the QP solver is implemented in a Xilinx Virtex 6 chip on an ML605 Evaluation board, controlling the nonlinear model of the plant running in Simulink on a desktop computer, communicating using UDP/IP over a 100 MBit Ethernet link is used to demonstrate that the real-time requirements for the application can be met with current FPGA technology.

The remainder of the paper is organised as follows: Section 2 presents the linearised model of the Boeing 747 aircraft and defines the optimal control problem; Section 3 gives the key characteristics of the interior-point algorithm and the hardware architecture; the scaling procedure is described in Section 4; Section 5 describes the details of the experimental setup; and results and conclusions are presented in Section 6.

2. BOEING 747 MODEL AND CONTROL PROBLEM

The FPGA-based controller is demonstrated through the control of a nonlinear model of a Boeing 747-200, modified to enable all control surfaces to be individually manipulated (Edwards et al., 2010; van der Linden et al., 2011).

The predictive controller is used to track state and input reference signals from an external target calculator, enabling roll, pitch and airspeed reference set points to be tracked with no steady state offset. These references are provided by a higher level controller—in this case an outer PID loop, which tracks a piecewise-linear altitude and yaw-angle reference trajectory.

A linear prediction model is obtained by linearising the nonlinear model around an equilibrium trim point for straight and level flight at an altitude of 600 m and an airspeed of 133 ms^{-1} (corresponding to a point soon after take-off), and discretising this at a sampling period $T_s = 0.2$ s. The actual state and input vectors for the nonlinear plant at time step k will be denoted $x(k)$ and $u(k)$ respectively. The state and input at the equilibrium trim point will be denoted $\overline{x}$, and $\overline{u}$.

Defining $\delta x(k) \triangleq (x(k) - \overline{x})$ and $\delta u(k) \triangleq (u(k) - \overline{u})$ as the deviations from the trim point, and letting $w(k)$ denote an unmeasured disturbance, the prediction model takes the form

$$\delta x(k+1) = A\delta x(k) + B\delta u(k) + B_d w(k). \tag{1}$$

Twelve states are considered in the prediction model: roll rate (rad/s), pitch rate (rad/s), yaw rate (rad/s), airspeed (m/s), angle of attack (rad), sideslip angle (rad), roll (rad), pitch (rad), and four engine states. A further two states, yaw angle (rad), and altitude (m) will be considered by the observer and outer-loop controller. Seventeen bound constrained inputs are controlled (see Table 1).

2.1 Control system architecture

The scenario considered is the tracking of a piecewise linear trajectory consisting of a period of straight and level flight, followed by a 90° change in heading followed by a 200 m descent, followed by a 10 ms^{-1} deceleration.

Table 1. Input constraints

	Input	Feasible region	Units
1,2	Right, left inboard aileron	$[-20, 20]$	deg
3,4	Right, left outboard aileron	$[-25, 15]$	deg
5,6	Right, left spoiler panel array	$[0, 45]$	deg
7,8	Right, left inboard elevator	$[-23, 17]$	deg
9,10	Right, left outboard elevator	$[-23, 17]$	deg
11	Stabiliser	$[-12, 3]$	deg
12,13	Upper, lower rudder	$[-25, 25]$	deg
14–17	Engines 1–4	$[0.94, 1.62]$	–

An observer is used to obtain filtered state esimates, $\delta\hat{x}(k)$ and obtain an estimate $\hat{w}(k)$ of a further 10 modelled integrating disturbance states. Time varying yaw, altitude and airspeed setpoints are provided to outer PID control loops (implemented on the PC), which provide roll, pitch and airspeed set points. A separate target calculator is used to obtain a target equilibrium pair $(\delta x_\infty, \delta u_\infty)$ for the required roll, pitch and airspeed setpoint. Letting $u_{\min}$ and $u_{\max}$ be the vectors corresponding to the lower and upper bounds of the manipulated inputs, $\delta u_{\min} \triangleq u_{\min} - \overline{u}$ and $\delta u_{\max} \triangleq u_{\max} - \overline{u}$, $Q \geq 0$ and $R > 0$, the target calculator solves

$$\min_{\delta u_\infty, \delta x_\infty} \delta x_\infty^T Q \delta x_\infty + \delta u_\infty^T R \delta u_\infty \tag{2a}$$

subject to

$$\begin{bmatrix} (A-I) & B \\ H_r & 0 \end{bmatrix} \begin{bmatrix} \delta x_\infty \\ \delta u_\infty \end{bmatrix} = \begin{bmatrix} -B_d\hat{w} \\ r \end{bmatrix} \tag{2b}$$

$$\delta u_{\min} \leq \delta u_\infty \leq \delta u_{\max}. \tag{2c}$$

For this study, it is assumed that an equilibrium pair compatible with these constraints exists. This is used as a setpoint for the predictive controller which is implemented on the FPGA. The linearisation point remains unchanged.

2.2 MPC Regulation Problem

Define the notation $\delta x_i \triangleq x(k+i|k)$, $\delta u_i \triangleq u(k+i|k)$ and let $\theta = \begin{bmatrix} \delta x_0^T & \delta u_0^T & \dots & \delta x_N^T \end{bmatrix}^T$, and $\|x\|_M^2 \triangleq x^T M x$. Letting $Q \geq 0$ and $R > 0$ be weighting matrices, and $P > 0$ be the solution to the discrete time algebraic Riccati equation (Franklin et al., 1990), the optimal control problem (OCP), of horizon length N, solved at each time step is

$$\min_\theta \; \| (\delta x_N - \delta x_\infty) \|_P^2 + \sum_{k=0}^{N-1} \left(\| (\delta x_k - \delta x_\infty) \|_Q^2 + \| (\delta u_k - \delta u_\infty) \|_R^2 \right) \tag{3a}$$

subject to

$$\delta x_0 = \delta\hat{x}(k) \quad \text{(from observer)} \tag{3b}$$

$$\delta x_{k+1} = A\delta x_k + B\delta u_k + B_d\hat{w}(k),\; k \in 0, \dots, N-1 \tag{3c}$$

$$\delta u_{\min} \leq \delta u_k \leq \delta u_{\max} \qquad k \in 0, \dots, N-1. \tag{3d}$$

For simplicity, state constraints are neglected, and with only input constraints considered, no extra measure is needed to guarantee that a solution of the QP exists.

2.3 Finite-horizon optimal control problem as a quadratic program

Letting $\otimes$ be the Kronecker product, $\oplus$ be the matrix direct sum, $I_i \in \mathbb{R}^{i\times i}$ be an identity matrix, and $\mathbf{1}_i \in \mathbb{R}^{i\times 1}$ be a vector of ones, let

$$H \triangleq (I_N \otimes (Q \oplus R)) \oplus P \tag{4a}$$

$$G \triangleq \begin{bmatrix} I_N \otimes \begin{bmatrix} 0_{m\times n} & I_m \\ 0_{m\times n} & -I_m \end{bmatrix}, \; 0_{2Nm\times n} \end{bmatrix} \tag{4b}$$

$$F \triangleq \begin{bmatrix} -I_n & & & \\ A & B & -I_n & \\ & & \ddots & \ddots \\ & & & -I_n \end{bmatrix} \tag{4c}$$

$$h \triangleq \begin{bmatrix} -\mathbf{1}_N^T \otimes \begin{bmatrix} x_\infty^T Q & u_\infty^T R \end{bmatrix} & -x_\infty^T P \end{bmatrix}^T \tag{4d}$$

$$g \triangleq \mathbf{1}_N \otimes \begin{bmatrix} \delta u_{\max}^T & -\delta u_{\min}^T \end{bmatrix}^T \tag{4e}$$

$$f \triangleq \begin{bmatrix} -\hat{x}^T(k) & -(\mathbf{1}_N^T \otimes \hat{w}^T(k) B_d^T) \end{bmatrix}^T. \tag{4f}$$

The OCP (3a-d) is posed as a quadratic program:

$$\min_\theta \frac{1}{2}\theta^T H \theta + h^T \theta \quad \text{subject to} \quad G\theta \le g,\; F\theta = f. \tag{5}$$

3. HARDWARE IMPLEMENTATION

In this section the main characteristics of the hardware architecture are described. For more details, see Jerez et al. (2011). At each time step, the quadratic program (5a–c) is solved using Algorithm 1 (Wright, 1993) assuming general state and input constraints and dense cost and dynamics matrices.

Algorithm 1. primal-dual interior-point (PDIP) algorithm

1. **Initialization** $\theta_0 := 0.05$, $\nu_0 := 0.3$, $\lambda_0 := 1.5$, $s_0 := 1.5$, $\sigma := 0.35$, $I_{IP} = 18$.

for $k = 0$ to $I_{IP} - 1$

2. **Linearization** $\mathcal{A}_k = \hat{\mathcal{A}} + \begin{bmatrix} \Phi_k & 0 \\ 0 & 0 \end{bmatrix}$ $b_k := \begin{bmatrix} r_k^\theta \\ r_k^\nu \end{bmatrix}$ where

$$\hat{\mathcal{A}} \triangleq \begin{bmatrix} H & F^T \\ F & 0 \end{bmatrix}, \quad \Phi_k \triangleq G^T W^{-1} G, \quad W^{-1} \triangleq \Lambda_k S_k^{-1}$$

$$r_k^\theta \triangleq -\Phi_k\theta_k - h - F^T\nu_k - G^T(\lambda_k - \Lambda_k S_k^{-1} g + \sigma\mu_k s_k^{-1}),$$

$$r_k^\nu \triangleq -F\theta_k + f,$$

$$\mu_k \triangleq \frac{\lambda_k^T s_k}{Nl + 2p}.$$

3. **Solve** $\mathcal{A}_k z_k = b_k$ for $z_k =: \begin{bmatrix} \Delta\theta_k \\ \Delta\nu_k \end{bmatrix}$

4. $\Delta\lambda_k \triangleq \Lambda_k S_k^{-1}(G(\theta_k + \Delta\theta_k) - g) + \sigma\mu_k s_k^{-1}$

$\Delta s_k \triangleq -s_k - (G(\theta_k + \Delta\theta_k) - g)$

5. **Line Search** $\alpha_k \triangleq \max_{(0,1]} \alpha : \begin{bmatrix} \lambda_k + \alpha\Delta\lambda_k \\ s_k + \alpha\Delta s_k \end{bmatrix} > 0.$

6. $(\theta_{k+1}, \nu_{k+1}, \lambda_{k+1}, s_{k+1}) \triangleq$
$(\theta_k, \nu_k, \lambda_k, s_k) + \alpha_k(\Delta\theta_k, \Delta\nu_k, \Delta\lambda_k, \Delta s_k)$

end

Table 2. Resource usage of different circuit blocks. An FPGA consists of look-up tables (LUTs), flip-flops (FFs), embedded RAM blocks and multiplier blocks (DSP48s).

	MicroBlaze	MINRES solver	Sequential stage
LUTs	7125(4.7%)	61341(40.7%)	1960(1.3%)
FFs	6930(2.3%)	88217(29.3%)	3039(1.0%)
BRAMs	20(4.8%)	76(18.3%)	14(3.4%)
DSP48s	3(0.4%)	205(26.7%)	2(0.3%)

The hardware implementation is split into two distinct blocks. One block accelerates the computational bottleneck of the algorithm (solving the linear equations in step 3 in Algorithm 1) implementing a parallel minimum residual (MINRES) solver. MINRES is an iterative method, where the main operation is performing a matrix-vector multiplication (banded in this case) at each iteration. In the present implementation, this operation is accelerated by computing dot-products in parallel by performing all the multiplications concurrently and passing the results through an adder reduction tree. The other block implements the remaining operations in Algorithm 1. It consists of a sequential machine with custom instructions that reduce instruction storage requirements and is close to 100% efficient, i.e. it produces one result every clock cycle.

There are several reasons why an iterative method can be preferable in this context. Firstly, the main operation is very easy to parallelise and consists of multiply-accumulate operations which map efficiently onto hardware in terms of resources. Secondly, these methods allow one to trade accuracy for computation time by varying the number of iterations (see Section 6). Finally, it is possible to exploit the finer structure of the matrix to reduce memory requirements since there is no filling and most of the elements do not change and are repeated throughout the matrix. This last point is quite important because being able to store all problem data using on-chip memories is essential for accelerating an iterative application like MINRES. If one could not store the matrix on-chip, then the amount of I/O and computation would be of the same order as each iteration, hence the speed of the application will be determined by how fast one could load the data in and out of the chip regardless of the parallelism employed.

Table 2 shows the resource usage for the different blocks in the implemented controller. The linear solver accounts for more than 96% of the resources used by the solver, except for memory which is approximately 81%. Table 2 also indicates that the controller easily fits in a modest size modern FPGA with plenty of resources to spare.

Implementing the controller in hardware gives the possibility of providing cycle accurate computation time guarantees. For the current controller, computation time is

$$\frac{I_{IP} P Z (I_{MR} + 66)}{f_c} \text{ seconds}, \tag{6}$$

where $Z = N(2n+m)+2n = 229$ and $M = 2n+m = 41$ are the size and halfband of $\mathcal{A}_k$, I_{IP} and I_{MR} are the number of interior-point and MINRES iterations, respectively, and

$$P := \left\lceil \frac{2Z + M + 12\lceil \log_2(2M-1) \rceil + 230}{Z} \right\rceil = 4. \tag{7}$$

The clock frequency is denoted by f_c, which is 250MHz in this implementation.

4. OFFLINE PRECONDITIONING

For each PDIP iteration, the convergence of the MINRES algorithm used to solve the indefinite linear system $\mathcal{A}_k z_k = b_k$, and the worst-case accuracy of the final estimate of z_k are influenced by the eigenvalue distribution of $\mathcal{A}_k$. If no scaling is performed on the prediction model and cost matrices for this application, inaccuracy in the estimates of z_k leads the PDIP algorithm to not converge to a satisfactory solution. Merely increasing the number of MINRES iterations by an order of magnitude fails to improve the solution, yet radically increases the computational burden.

Conventionally, preconditioning would be applied online in order to accelerate convergence, and reduce the worst-case solution error of z_k. In order to minimise design complexity and hardware resource demands, in the present paper this is performed offline through a systematic scaling of the optimal control problem.

Matrix A_k is not constant. However, W_k^{-1} is diagonal, and since there are only upper and lower bound constraints on inputs, the varying component of $\mathcal{A}_k$, Φ_k also only has diagonal elements. Moreover, as $k \to I_{IP} - 1$, the diagonal elements of W_k^{-1} corresponding to inactive constraints tend towards zero. Therefore, despite those diagonal elements of W_k^{-1} corresponding to active constraints becoming large, as long as only a handful these exist at any instant, the perturbation to $\hat{A}$ is of low rank, and has a relatively minor effect on the convergence of MINRES. It follows that rescaling the control problem to improve the conditioning of $\hat{\mathcal{A}}$ should also improve the conditioning of each $\mathcal{A}_k$ and accuracy of the corresponding z_k.

Prior to scaling, $\text{cond}(\hat{\mathcal{A}}) = 2.3916 \times 10^9$. The objective of the exercise that follows is to obtain diagonal matrices $T_Q > 0$ and $T_R > 0$ to scale the linear state space prediction model and quadratic cost weighting matrices to improve the conditioning of $\hat{\mathcal{A}}$ using the following substitution: $A \leftarrow T_Q A T_Q^{-1}$, $B \leftarrow T_Q B T_R^{-1}$, $B_d \leftarrow T_Q B_d$, $Q \leftarrow T_Q^{-1} Q T_Q^{-1}$, $R \leftarrow T_R^{-1} R T_R^{-1}$, $\delta u_{\min} \leftarrow T_R \delta u_{\min}$, $\delta u_{\max} \leftarrow T_R \delta u_{\max}$. This substitution is equivalant to

$$\hat{\mathcal{A}} \leftarrow \hat{M} \hat{\mathcal{A}} \hat{M} \tag{8}$$

where

$$\hat{M} = \left(\left(I_N \otimes \left(T_Q^{-1} \oplus T_R^{-1} \right) \right) \oplus T_Q^{-1} \right) \oplus \left(I_{N+1} \otimes T_Q \right). \tag{9}$$

By constraining $T_Q = \text{diag}\{t_Q\}$ and $T_R = \text{diag}\{t_R\}$, the diagonal structure of Φ_k is retained. The transformation (8) is a function of both T_Q and its inverse, and both of these appear quadratically, so it is therefore likely that minimisation of any particular function of $\hat{M}\hat{\mathcal{A}}\hat{M}$ is not (in general) going to be a convex or particularly well conditioned.

In Betts (2010) some guidelines are provided for desirable scaling properties. In particular, it is desirable to normalise the rows and columns of $\hat{\mathcal{A}}$ so that they are all of similar magnitude. In Jerez et al. (2012) a diagonal preconditioner

$$\overline{M} \triangleq \left\{ \overline{M} : \ \overline{M}_{ij} = \begin{cases} \left(\sum_s |\mathcal{A}_{k,\{is\}}| \right)^{-1/2} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \right\} \tag{10}$$

is proposed for matrix $\mathcal{A}_k$ that guarantees $\lambda_i(\overline{M}\mathcal{A}_k\overline{M}) \in [-1, 1]$. This must be re-calculated at every iteration of the PDIP method and requires additional hardware for division and square root calculation to be instantiated in the FPGA. It can be observed that if the method of Jerez et al. (2012) were to be applied repetitively to a general square matrix of full rank, the 1-norm of each of the rows converges asymptotically to unity. Whilst iterative application of this preconditioner (i.e. repeatedly re-conditioning the same matrix) was not the intended application of Jerez et al. (2012), the offline method proposed subsequently exploits this observation.

The method proposed here for normalising $\hat{\mathcal{A}}$ follows naturally but with the further caveat that the structure of $\hat{M}$ is imposed to be of form (9). Consequently, it is not (in general) possible to scale $\hat{\mathcal{A}}$ such that all row norms are equal to an arbitrary value. Instead, the objective is to reduce the variation in row (and column) norms. Empirical testing suggests that normalising the 2-norm of the rows of $\hat{\mathcal{A}}$ (subject to (9)) gives the most accurate solutions from the PDIP method for the present application.

Noting that $\hat{\mathcal{A}}$ has a very particular structure, define the following vectors:

$$s_x \triangleq \left\{ s_x \in \mathbb{R}^n : \ s_{x,\{i\}} = \left(\sum_{j=1}^n Q_{ij}^2 + \sum_{j=1}^n A_{ji}^2 + 1 \right)^{1/2} \right\}$$

$$s_u \triangleq \left\{ s_u \in \mathbb{R}^m : \ s_{u,\{i\}} = \left(\sum_{j=1}^m R_{ij}^2 + \sum_{j=1}^n B_{ji}^2 + 1 \right)^{1/2} \right\}$$

$$s_N \triangleq \left\{ s_N \in \mathbb{R}^n : \ s_{N,\{i\}} = \left(\sum_{j=1}^n P_{ij}^2 + 1 \right)^{1/2} \right\}$$

$$s_\lambda \triangleq \left\{ s_\lambda \in \mathbb{R}^n : \ s_{\lambda,\{i\}} = \left(\sum_{j=1}^n A_{ij}^2 + \sum_{j=1}^m B_{ij}^2 + 1 \right)^{1/2} \right\}.$$

Define elementwise,

$$l_1 \triangleq \sqrt{s_u / \mu} \tag{11a}$$

$$l_2 \triangleq \{ l_2 \in \mathbb{R}^n > 0 : \ l_2^4 = ((N s_x + s_N)/(1 + N s_\lambda)) \} \tag{11b}$$

where

$$\mu = \frac{\left(N \sum s_x + \sum s_N + N \sum s_\lambda + n \right)}{(2(N+1)n)}. \tag{11c}$$

Algorithm 2. (Offline Preconditioning).

Data: A, B, Q, R, P, ϵ

1. Let $t_Q = \mathbf{1}_n$, and $t_R = \mathbf{1}_m$.

Repeat:

2. Calculate l_1, l_2 as functions of current data, and define $L_1 \triangleq \text{diag}(l_1)$, $L_2 \triangleq \text{diag}(l_2)$.

3. Update:

$$t_Q \leftarrow L_2 t_Q, \ t_R \leftarrow L_1 t_R$$
$$A \leftarrow L_2 A L_2^{-1}, \ B \leftarrow L_2 B L_1^{-1}$$
$$Q \leftarrow L_2^{-1} Q L_2^{-1}, \ P \leftarrow L_2^{-1} P L_2^{-1}, \ R \leftarrow L_1^{-1} R L_1^{-1}.$$

Until: $(\|l_2 - \mathbf{1}\| < \epsilon) \cap (\|l_1 - \mathbf{1}\| < \epsilon)$

Output:

4. $T_Q = \text{diag}(t_Q)$, $T_R = \text{diag}(t_R)$.

Table 3. Effects of offline preconditioning

Scaling	$\text{cond}(\hat{\mathcal{A}})$	std $\|\hat{\mathcal{A}}_{\{i,:\}}\|_1$	std $\|\hat{\mathcal{A}}_{\{i,:\}}\|_2$	std $\|\lambda_i(\hat{\mathcal{A}})\|$
Original	2.39×10^9	2.92×10^3	2.42×10^3	2.42×10^3
Scaled	21.08	0.8087	0.1487	0.5538

Table 3 shows properties of $\hat{\mathcal{A}}$ that are likely to influence solution quality, before and after application of Algorithm 2 with $\epsilon = 10^{-7}$. These are the condition number of $\hat{\mathcal{A}}$, the standard deviation of the row 1– and 2–norms, and the standard deviation of the magnitude of the eigenvalues.

5. TEST BENCH

The proposed hardware-in-the-loop experimental setup architecture has two goals: providing a reliable real-time closed-loop simulation framework for controller design verification; and demonstrating a hardware coded controller that could be directly plugged into a real world plant.

Figure 1 shows the experimental setup. The QP solver, running on a Xilinx FPGA ML605 evaluation board, controls the nonlinear model of the B747 aircraft running on a desktop PC. Data transfer between controller and PC is performed through a 100 Mbit/s Ethernet link using UDP/IP. The communication latency of this interface is small compared to the computation time (see Table 4).

The desktop PC simulates the nonlinear plant model, the observer and target calculator faster-than-real-time using Simulink. At every sampling instant k, the observer estimates the next state $\delta\hat{x}(k+1|k)$ and disturbance $\hat{w}(k+1|k)$, and the target calculator provides the reference $(\delta x_\infty(k), \delta u_\infty(k))$ for the controller, represented as a sequence of single-precision floating point numbers in the payload of a UDP packet via an S-function. The FPGA returns the control action in another UDP packet, which is applied to the plant model at the *next sampling instant*.

On the controller side, the transferred data is captured by the Physical Layer Device (PHY) on an ML605 evaluation board, implementing the physical layer of Ethernet stack, and then transmitted to the FPGA chip. The data link layer, is implemented in hardware with a Media Access Control (Ethernet lite MAC) provided by the FPGA manufacturer. The transport and network layers of the UDP stack are also provided by the FPGA vendor and run on an embedded soft processor IP-core (Xilinx MicroBlaze). The decoded UDP packet is routed to a mixed software-hardware application layer.

The hardware application task consists of a custom hardware accelerator implementing the QP solver. The software application, executed on the same soft processor, bridges the communication between the Ethernet interface and the hardware accelerator. This architecture provides more system flexibility than a dedicated custom designed interface, with a small increase in FPGA resource usage (Table 2), power consumption and communication delay (Table 4), and allows easy portability to other standard interfaces, e.g. SpaceWire, CAN bus, etc., as well as an option for direct monitoring of controller behaviour.

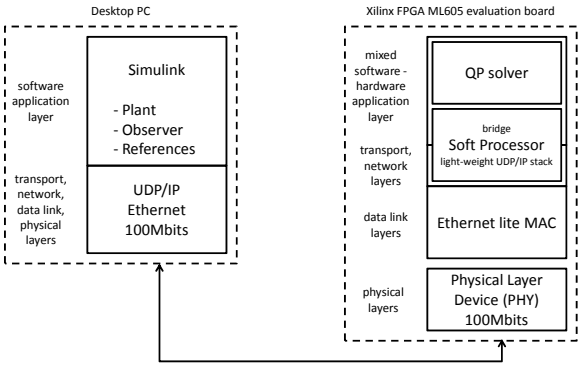


Fig. 1. Hardware-in-the-loop experimental setup. The computed control action by the QP solver is encapsulated into an UDP packet and sent through an Ethernet link to a desktop PC, which decodes the data packet, applies the control action to the plant and returns new state, disturbance and trajectory estimates.

6. RESULTS AND CONCLUSIONS

The MINRES-based PDIP MPC controller, implemented using single precision floating point arithmetic, and only off-line preconditioning of $\mathcal{A}_k$, and using a prediction horizon $N = 5$, has been used to successfully control the nonlinear model of a Boeing 747-200. Figure 2 shows the closed-loop state and control surface trajectories for the simulation of the manœuvre described in Section 2.1, with $I_{MR} = 66$.

To demonstrate the claim in Section 3 that solution accuracy can be traded for computation time, Table 4 presents timing and solution accuracy data for the MINRES-based controller over the course of a 600 s simulation, allowing 229, 115, 88 and 66 MINRES iterations for each PDIP iteration. The MINRES-based controller is compared with the algorithm of Rao et al. (1998), the dense active set method of Goldfarb and Idnani (1983) and an implementation of the un-preconditioned MINRES-based solver with the same pre-scaled prediction model, running on the PC. This is also compared with the algorithm of Rao et al. (1998) running directly on a 150 MHz MicroBlaze (the fastest allowed by the synthesis software). For the first two PC-based implementations, the algorithms are coded in MATLAB and converted to C-MEX functions with MATLAB Coder (R2011a), with responsiveness and memory integrity checks disabled, and no use of BLAS. The MINRES-based PC implementation is coded directly in C. Functions are compiled using `gcc` version 4.2.1 with -O3 optimisations.

Timing for PC-based algorithms is obtained using the `tic` and `toc` commands on a Macbook Pro with a 2.4GHz Intel i5, over a sequence of data obtained from a prior simulation. For FPGA-based controllers, time is measured from the point before the UDP packet is sent until the point after which the UDP packet reply is received. It can be seen that the FPGA-based implementation requires substantially fewer clock cycles to achieve competitive control accuracy, and that for the present application, the clock frequency could even be reduced (for example, to reduce power consumption or reduce heat dissipation requirements) whilst still achieving real-time control.

Table 4. Predictive controller performance comparisons over duration of simulation ($N = 5$)

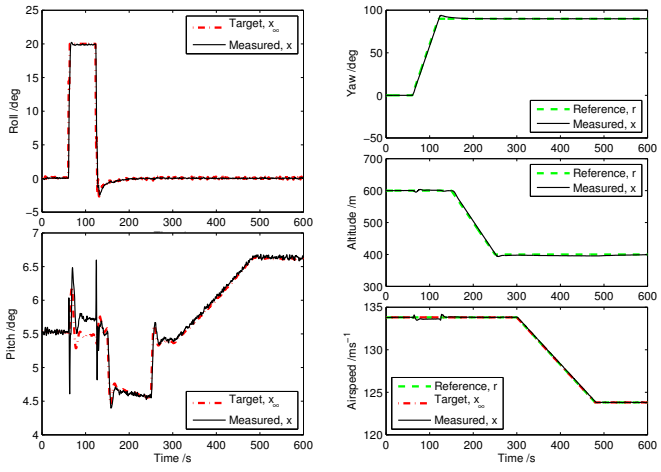
Implementation				Numerical accuracy		Cost	Max solution time (ms)			
Location	Algorithm	Precision	I_{MR}	$e_{\max}$	e_μ	summation	Total	QP	Comm	Clk cycles
FPGA	MINRES	float32	229	2.2×10^{-3}	3.3×10^{-5}	4.428×10^3	20.99	20.38	1.11	50×10^5
FPGA	MINRES	float32	115	1.0×10^{-2}	3.8×10^{-5}	4.420×10^3	13.52	12.41	1.11	31×10^5
FPGA	MINRES	float32	86	1.9×10^{-2}	4.5×10^{-5}	4.412×10^3	11.81	10.70	1.11	27×10^5
FPGA	MINRES	float32	66	3.7×10^{-2}	6.0×10^{-5}	4.401×10^3	9.89	8.78	1.11	22×10^5
PC	MINRES	float32	229	3.1×10^{-3}	4.5×10^{-4}	4.428×10^3	865	865	N/A	21×10^8
PC	RWR1998	float64	N/A	–	–	4.428×10^3	15.2	15.2	N/A	36×10^6
PC	GI1983	float64	N/A	1.3×10^{-4}	1.3×10^{-5}	4.428×10^3	6.4	6.4	N/A	15×10^6
MicroBlaze	RWR1998	float32	N/A	2.5×10^{-3}	3.8×10^{-4}	4.428×10^3	526.3	525.2	1.11	79×10^6

The control accuracy metrics presented are

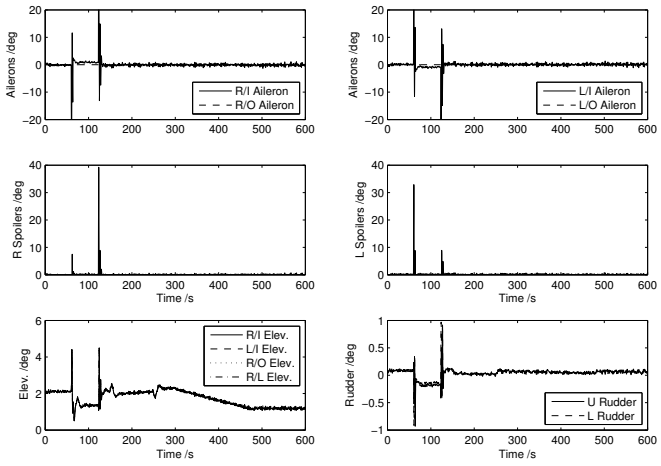
$$e_{\max} = \max_{i,k} \left| u^F_{\{i\}}(k) - u^*_{\{i\}}(k) \right| / \left(\delta u_{\max,\{i\}} - \delta u_{\min,\{i\}} \right)$$

$$e_\mu = \text{mean}_{i,k} \left| u^F_{\{i\}}(k) - u^*_{\{i\}}(k) \right| / \left(\delta u_{\max,\{i\}} - \delta u_{\min,\{i\}} \right)$$

where $\bullet_{\{i\}}$, $\bullet^F$ and $\bullet^*$ denote the ith element of a vector, solution from the FPGA, and a "true" (assumed) optimal solution found using double precision arithmetic conventional methods respectively. Even with a reduced number of MINRES iterations, and a fixed number of 18 PDIP iterations, the error between the obtained control action and the ideal solution is, on average, small. The sum of stage costs over the simulation is similar in all cases.



(a) Selected state trajectories



(b) Selected input trajectories

Fig. 2. Closed loop simulation trajectories ($I_{MR} = 66$)

REFERENCES

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Betts, J.T. (2010). *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*. SIAM, second edition.

Edwards, C., Lombaerts, T., and Smaili, H. (eds.) (2010). *Fault Tolerant Flight Control: A Benchmark Challenge*. Lecture Notes in Control and Information Sciences. Springer.

Franklin, G.F., Powell, J.D., and Workman, M.L. (1990). *Digital control of dynamic systems*. Addison-Wesley, Reading, Mass., 2nd ed edition.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Math. Prog.*, 27(1), 1–33.

Jerez, J.L., Constantinides, G.A., and Kerrigan, E.C. (2011). An FPGA implementation of a sparse quadratic programming solver for constrained predictive control. In *Proc. ACM Symp. Field Programmable Gate Arrays*. Monterey, CA, USA.

Jerez, J.L., Constantinides, G.A., and Kerrigan, E.C. (2012). Towards a fixed point QP solver for predictive control. In *Proc. IEEE Conf. on Decision and Control (Submitted)*.

Ling, K.V., Wu, B.F., and Maciejowski, J.M. (2008). Embedded model predictive control (MPC) using a FPGA. In *Proc. 17th IFAC World Congress*, 15250–15255. Seoul, Korea.

Rao, C.V., Wright, S.J., and Rawlings, J.B. (1998). Application of interior-point methods to model predictive control. *J. Optim. Theory Appl*, 99(3), 723–757.

van der Linden, C., Smaili, H., Marcos, A., Balas, G., Breeds, D., Runham, S., Edwards, C., Alwi, H., Lombaerts, T., Groeneweg, J., Verhoeven, R., and Breeman, J. (2011). GARTEUR RECOVER benchmark. URL `http://www.faulttolerantcontrol.nl`.

Vouzis, P.D., Bleris, L.G., Arnold, M.G., and Kothare, M.V. (2009). A system-on-a-chip implementation for embedded real-time model predictive control. *IEEE Trans. Control. Syst. Technol.*, 17(5), 1006–1017.

Wills, A.G., Knagge, G., and Ninness, B. (2012). Fast linear model predictive control via custom integrated circuit architecture. *IEEE Trans. Control. Syst. Technol.*, 20(1), 59–71.

Wright, S.J. (1993). Interior-point method for optimal control of discrete-time systems. *J. Optim. Theory Appl.*, 77, 161–187.