OPTIMAL CONTROL APPLICATIONS AND METHODS
*Optim. Control Appl. Meth.* 2014; **00**:1–22
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/oca

# FPGA-Based Predictive Control System for Spacecraft Rendezvous in Elliptical Orbits

Edward N. Hartley* and Jan M. Maciejowski

*University of Cambridge, Department of Engineering, Trumpington Street, Cambridge. CB2 1PZ. UK.*

SUMMARY

A field programmable gate array (FPGA)-based model predictive controller (MPC) for two phases of spacecraft rendezvous is presented. Linear time varying prediction models are used to accommodate elliptical orbits, and a variable prediction horizon is used to facilitate finite time completion of the longer-range manœuvres, whilst a fixed and receding prediction horizon is used for fine-grained tracking at close range. The resulting constrained optimisation problems are solved using a primal dual interior point algorithm. The majority of the computational demand is in solving a system of simultaneous linear equations at each iteration of this algorithm. To accelerate these operations, a custom circuit is implemented, using a combination of Mathworks HDL Coder and Xilinx System Generator for DSP, and used as a peripheral to a MicroBlaze soft core processor on the FPGA, on which the remainder of the system is implemented. Certain logic that can be hard-coded for fixed sized problems is implemented to be configurable online, in order to accommodate the varying problem sizes associated with the variable prediction horizon. The system is demonstrated in closed loop by linking the FPGA with a simulation of the spacecraft dynamics running in Simulink on a PC, using Ethernet. Timing comparisons indicate that the custom implementation is substantially faster than pure embedded software-based interior point methods running on the same MicroBlaze, and could be competitive with a pure custom hardware implementation. Copyright © 2014 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: predictive control; MPC; spacecraft rendezvous; embedded systems; FPGA; time-varying systems; aerospace; receding horizon; variable horizon



This work was supported by the Engineering and Physical Sciences Research Council Grant Number [EP/G030308/1] as well as industrial support from Xilinx, Mathworks, and the European Space Agency.

*Correspondence to: E. N. Hartley, University of Cambridge, Department of Engineering, Trumpington Street, Cambridge. CB2 1PZ. UK. Email: enh20@eng.cam.ac.uk

Contract/grant sponsor: EPSRC; contract/grant number: EP/G030308/1



*Prepared using* ocaauth.cls *[Version: 2010/03/27 v2.00]*

## 1. INTRODUCTION

Model predictive control (MPC) offers the ability to explicitly handle physical and operational constraints whilst optimising a given performance metric through repeated solution of a receding horizon constrained optimal control problem (e.g. [1–3]). As an implicit control law, it is also particularly suited to scenarios where reconfiguration is necessary, since the prediction model, cost function and constraints can be updated online to reflect changes in plant parameters or objectives. This property has evoked recent enthusiasm for the application of MPC to spacecraft manœuvreing functions such as station and formation keeping [4–8], attitude control [9–14], rendezvous [15–26], collision avoidance [16–18, 27, 28] and interplanetary transfer [29, 30]. Moreover, [31] documents a successful in-flight test of MPC for tracking trajectories.

Since MPC relies upon the solution of a constrained optimisation problem at each time step, the computational demand is heavier than for a classical controller. For larger problem formulations, a contemporary guidance, navigation and control computer would require a dedicated co-processor to support the MPC [20]. Whilst the related optimisation problems are usually well within the capabilities of a contemporary desktop computer, as well as modern high-performance RISC microprocessors that have become ubiquitous for ground-based embedded applications, the computer clock rates in space environments are between one and two orders of magnitude slower due to a need to be robust to solar radiation, low power consumption requirements. Lengthy clearance procedures also mean that hardware suitable for space applications inevitably lags that for general purpose usage by a few generations. Rather than using a high-performance general purpose microprocessor for this purpose, an alternative is to employ a custom peripheral circuit, designed to carry out all or part of the predictive control function, trading high clock frequencies (which can be problematic in space applications due to radiation exposure) for parallelism. Custom hardware may also offer opportunities to affect power consumption.

Field Programmable Gate Arrays (FPGAs) are a form of *programmable hardware*, with which a system designer can implement a complex custom digital circuit without recourse to manufacturing a bespoke silicon chip. These devices are well-suited for prototyping purposes as well as for low volume but highly specialised applications (such as spacecraft avionics). FPGAs have already been demonstrated as a viable platform for the most typical form of MPC, which employs a quadratic cost function, linear constraints and a linear time-invariant (LTI) prediction model [14, 26, 32–42].

This paper instead presents an FPGA-based implementation of a linear time-varying (LTV) MPC-based controller for spacecraft rendezvous in an elliptical orbit, based on a simplified version of the inner two mission phases presented in [20]. The first phase consists of impulsive manœuvres between a sequence of pre-determined holding points. It was shown in [20] that an MPC-based approach could be beneficial in reducing propellant consumption when compared with a conventional approach during this phase. The second phase tracks a straight line trajectory from the final holding point up to a point a few metres away from the target from which final interception of the target must proceed passively. During this phase, an MPC-based approach with a non-conventional cost function [43] can yield fuel-saving benefits over a linear feedback law.

The present work is an extension of the preliminary work on implementing an FPGA-based control system presented in [24], which focuses solely on the impulsive manœuvre phase (although an alternative FPGA-based strategy for the final tracking phase, applicable only to circular orbits, is also considered in [26]). The primary contribution of this work is technological rather than theoretical, and is in addressing challenges in the hardware-acceleration of a non-standard MPC problem with a time-varying prediction model, and variable prediction horizon as well as mixed $\ell_1$ and quadratic objectives.

For the first (impulsive) phase, a variable horizon (VH) MPC is implemented by solving a sequence of convex linear programs (LPs) corresponding to each horizon length up to a defined maximum. The second (tracking) phase adopts a fixed-length receding horizon, and each MPC problem is a convex quadratic program (QP). The constrained optimisation problems that arise at each time step during both phases are solved using a primal-dual interior point (PDIP) quadratic progamming algorithm. Following a hybrid software-hardware paradigm [32–34], the algorithm is



DOI: 10.1002/oca

Table I. Scenario parameters

| Central body | Target | | | Chaser | |
|---|---|---|---|---|---|
| Mass | Semi-major axis | Eccentricity | Orbital period | Mass | Initial separation angle |
| $m_{\text{Mars}}$ | $a$ | $e$ | $T_{\text{orb}}$ | $m_{\text{chs}}$ | $\Delta\nu$ |
| $0.64185 \times 10^{24}$ kg | 4643 km | 0.20441 | $9.61 \times 10^{3}$ s | 1575 kg | 0.00303 rad |

co-ordinated by a Xilinx MicroBlaze soft-core processor, and a custom peripheral core (PCORE) circuit is implemented, using a combination of MathWorks HDL Coder and Xilinx System Generator for DSP, to assist in the construction and solution of the set of linear equations that arise at each iteration of the PDIP algorithm, using the minimum residual (MINRES) method. The MicroBlaze processor is comparable [44] in computational capabilities to the LEON2 processor often deployed in spacecraft. By posing LPs as QPs with a zero Hessian matrix, the same algorithm (and PCORE components) can be used with different numerical parameters for both phases.

In deference to convention, a synopsis of the residuum of the paper follows thus: Section 2 summarises key background material on prediction models and cost functions for predictive control purposes motivating the control design used in the present work, as well as prior examples of FPGA-based MPC controllers; Section 3 summarises the MPC formulation that has been applied; Section 4 summarises the algorithms employed; Section 5 describes the custom peripheral core component implemented on the FPGA to accelerate the solution of the QPs and the methods through which it was designed as well as analysing hardware resource and power usage; Section 6 presents the closed-loop simulation with the custom MPC controller implemented on a Xilinx ML605 Evaluation Board [45], using Ethernet to communicate with a nonlinear simulation of the spacecraft position dynamics, running in Simulink on a PC and comparisons with existing solution methods are presented. Finally, Section 7 concludes.

## 2. BACKGROUND AND DESIGN MOTIVATIONS

The control problem considered in this paper is the medium-short range phase of the rendezvous in the Mars Sample Return mission [20, 46], starting at the point where the controlled spacecraft (chaser) has been brought into approximately the same orbit as the spacecraft with which it must rendezvous (target), but with a significant separation in true anomaly (i.e. an in-track separation of a few tens of kilometres). Key parameters describing the scenario are presented in Table I. (Note that the scale of the semi-major axis means that the small value of initial separation in terms of true anomaly, $\Delta\nu$ corresponds to a sizeable in-track separation.)

For rendezvous in circular Keplerian orbits, the linearised Hill-Clohessy-Wiltshire (HCW) equations (e.g. [47–49]) can be used to predict the trajectory of the controlled spacecraft in a rotating reference frame whose origin is at the centre of mass of the target spacecraft. The latter is assumed to remain in a constant orbit. For elliptical orbits, the HCW equations are not sufficiently accurate, especially over long time periods (although [21] demonstrates good closed-loop performance with a probabilistically-constrained robust MPC for terminal phase rendezvous starting from a separation of 50 m using the LTI HCW equations and accounting for model mismatch due to target orbital eccentricities in the range $0.05 \leq e \leq 0.25$ as a bounded disturbance). More accurate state propagation matrices for relative translational dynamics in elliptical orbits are parameterised by the true anomaly of the target orbit [50–52]. For a passive target, the true anomaly varies only with time, so the models are linear time-varying (LTV). Unlike with a general nonlinear model the optimisation structure within the MPC controller is similar to that with LTI models, except that the prediction model varies deterministically over the prediction horizon, and *between controller evaluations*. The same computational techniques apply as with time invariant systems with the exception that some matrices must also be recomputed between time steps. In [4], an MPC controller is presented using an approximate discretisation of the continuous linearised parameter-varying equations of relative



DOI: 10.1002/oca

motion. The Yamanaka-Ankersen (YA) state transition matrix (STM) [51] which discretises the LTV dynamics more accurately has been used to form the prediction model for MPC in [6, 19, 20]. Gauss's Variational Equations (GVEs), which consider instead the relative position in terms of the differences in Keplerian orbital elements are also parameterised in terms of the true anomaly of the target and have been used in MPC by [5, 7, 19, 20].

For the application in the present paper, the YA equations [51] provide a suitable prediction model of the dynamics of the relative position and velocity of the chaser with respect to the target in a local-vertical local-horizontal (LVLH) cylindrical reference frame (CRF) centred on the target (Figure 1(a)), with the $z_{\mathrm{crf}}$-axis pointing towards the central body, the $y_{\mathrm{crf}}$-axis normal to the orbital plane, and the $x_{\mathrm{crf}}$-axis completing a right-handed set. This is preferred to a Cartesian reference frame because it reduces linearisation errors at larger in-track separations [47].

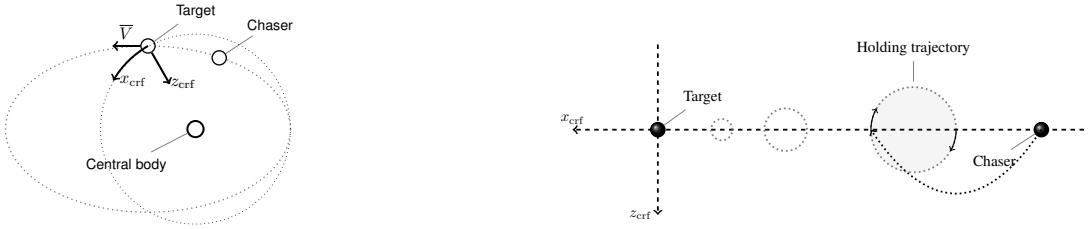
### 2.1. *Impulsive phase*

During the impulsive phase, the control objective is to guide the chaser towards the vicinity of the target via a sequence of pre-determined "holding points" at which it must remain pending clearance to continue (Figure 1(b)). Such trajectories are often informally described as "*hopping*", since in the ideal case (without any parametric or additive uncertainties) they comprise impulsive inputs at the start and finish separated by a long period of free drift. In the present scenario, the holding points are defined as being centred at $5000\ \mathrm{m}$, $2000\ \mathrm{m}$, $1000\ \mathrm{m}$, $500\ \mathrm{m}$ and $200\ \mathrm{m}$ in front of the target. The time spent at each holding point is unknown *a priori*, so at a given instant the goal is to enter (or remain in) the next holding point. Each transfer should take approximately half an orbit, propellant consumption should be minimised, and short-term accuracy is not too critical. The phase terminates once the final holding point has been reached. Input constraints can become active during the acceleration/deceleration impulses at longer ranges from the target.

MPC controllers commonly employ a quadratic cost function of predicted future states and inputs. However, for longer range manœuvres, minimisation of fuel consumption and completion of the manœuvre in finite time can be more important than accurate tracking of a reference trajectory. Accordingly, [15, 19, 20] employ a 1-norm cost function on inputs to reflect the direct correlation between impulsive accelerations and fuel consumption. Moreover, a variable prediction horizon (VH) helps facilitate finite-time manœuvre completion [15, 20, 30, 53, 54]. VH-LTV-MPC imposes a particularly high computational burden, since the prediction model must be updated at each time step, and the variable horizon optimisation is not convex. Nevertheless, a global optimum can be obtained by using mixed-integer programming [15, 54, 55] or enumeration of a sequence of convex problems [19, 20]. The latter strategy is chosen for the impulsive phase in the present paper.

### 2.2. *Tracking phase*

In the final tracking phase, the chaser spacecraft starts from the holding point which is centred approximately $200\ \mathrm{m}$ in front of the target in the CRF. The control objective is to maintain a line of approach parallel to the instantaneous velocity of the target (not in the $x$ direction in the CRF), offset by a distance of $0.113\ \mathrm{m}$ in the $z$ direction. The latter is the approximate distance the chaser




(a) Cylindrical reference frame

(b) Holding points in cylindrical reference frame

Figure 1. Cylindrical LVLH reference frame



DOI: 10.1002/oca

will travel open-loop in the $z$ direction over 30 s when its in-track velocity is $0.1\ \mathrm{ms}^{-1}$, since at ranges $\leq 3$ m from the target no control must be applied. The MPC regulates the position in the lateral directions in the moving reference frame, and regulates the velocity in the in-track direction.

MPC with a pure $\ell_1$-norm cost function can be sensitive to noise and exhibit awkward behaviours such as dead-beat control at one extreme and idle control at the other [56]. Here, a quadratic cost function is more desirable. However, MPC with a quadratic cost function will tend to lead to long periods of relatively low and continuous thrust, whereas for a system propelled by gas thrusters, a degree of sparsity in the control actions is preferable. Such behaviour can be achieved by combining the quadratic cost with an additional $\ell_1$ term on the input, [43, 57, 58], informally dubbed as $\ell_{\mathrm{asso}}$-MPC in homage to the Least Absolute Shrinkage and Selection Operator (LASSO) used in regularised least-squares regression. MPC with this class of cost function has been demonstrated for the terminal phases of spacecraft rendezvous in *circular* orbits in [25, 26].

### 2.3. *Hardware implementation of predictive controllers*

Recently, interest has arisen in using FPGAs to exploit opportunities for parallelism in the numerical algorithms used to implement MPC, with a variety of approaches proposed. A hybrid software-hardware design is advocated by [32–34]. The first uses a primal barrier method and the custom accelerator is used to calculate the gradient vector and Hessian matrix of the augmented cost function at each iteration, whilst the latter two use an active set method, with a custom circuit expediting matrix/vector-vector multiplication. Full hardware implementations of MPC controllers are documented in [35] (active set), [36–41] (interior point), [26, 42] (first order) although [26, 40, 41] use a soft-core microprocessor to bridge between the MPC and the outside world. At the other end of the spectrum, [14] implements MPC using a custom soft-core processor with non-standard numerical representations to minimise the FPGA resource (and power) requirements. These implementations consider systems with LTI prediction models, and fixed prediction horizons. Deviation from these characteristics can considerably complicate the implementation, since it is no longer feasible to hard-code the prediction model or pre-processed QP matrices in ROM at design time, as in [39, 40]. Furthermore, certain finite state-machine logic that can be hard-coded for fixed sized problems will need to be configurable online, in order to accommodate different QP problem sizes. Nevertheless, these complications are necessary for the proposed application in this work and thus comprise one of its main technological contributions.

## 3. PREDICTIVE CONTROL FORMULATION

In predictive control, the control input applied to the plant at each sampling instant is computed by finding the solution to an optimisation problem, which minimises some function of a prediction of the future plant state and inputs values, whilst enforcing constraints on these trajectories.

### 3.1. *Generic formulation for variable horizon MPC*

Let $n_x$ and $n_u$ be the number of states and inputs respectively, and define $x_i \in \mathbb{R}^{n_x}$ and $u_i \in \mathbb{R}^{n_u}$ to denote the predicted state and control input $i$ time steps into the future, $x(k)$ denote the actual measured (or estimated) state at time $k \in \mathbb{Z}_+$, and $\theta(k) \triangleq [x_0^T, u_0^T, \ldots, x_N^T]^T$ be a stacked vector of predicted inputs and states computed at time $k$. Let $\mathbb{X}_i \subseteq \mathbb{R}^{n_x}$ and $\mathbb{U}_i \subseteq \mathbb{R}^{n_u}$ denote the set of feasible states and inputs respectively, $i$ time steps into the future, and $\mathbb{T}_N(k+N) \subseteq \mathbb{R}^{n_x}$ denote the (time varying) set of feasible states at the end of the prediction horizon. Let $A_i$, and $B_i$ denote the time-varying state update matrices $i$ steps into the future. Letting the scalar valued function $\ell_i(x_i, u_i) \geq 0$ for all $(x_i, u_i)$ be a stage cost function, $J_N(x_N) \geq 0$ for all $x_N$ be the terminal cost, then assuming a variable prediction horizon $N \leq N_{\max}$, (as in the first, impulsive, rendezvous phase considered) the optimisation problem posed at each time step is:

$$J^* = \min_{N,\theta} J_N(x_N) + \sum_{i=0}^{N-1} \ell_i(x_i, u_i) \tag{1a}$$



DOI: 10.1002/oca

$$\text{s.t.} \quad x_0 = x(k), \tag{1b}$$
$$x_{i+1} = A_i x_i + B_i u_i, \quad i \in \{0, \dots, N-1\}, \tag{1c}$$
$$x_i \in \mathbb{X}_i, \quad i \in \{1, \dots, N-1\}, \tag{1d}$$
$$u_i \in \mathbb{U}_i, \quad i \in \{0, \dots, N-1\}, \tag{1e}$$
$$x_N \in \mathbb{T}_N(k+N) \tag{1f}$$
$$N \leq N_{\max}. \tag{1g}$$

The control $u(k) = u_0$ is applied to the plant, the rest of $\theta(k)$ is discarded and the process is repeated at the next sampling instant.

Standard receding (fixed) horizon MPC is a special case, with the additional constraint $N = N_{\text{fix}}$, where $N_{\text{fix}} \leq N_{\max}$ is the chosen fixed horizon length, and the variable horizon problem can be solved by consideration of a sequence of $N_{\max}$ fixed horizon MPC problems (some of which may be infeasible). In the present context, this is preferable to using mixed integer programming, because solution of a sequence of convex problems leads to a conceptually simple algorithm, which is more expedient for an embedded implementation.

### *3.2. Predictive Control Problem as a QP*

Let

$$\ell_i(x_i, u_i) = \begin{bmatrix} x_i \\ u_i \end{bmatrix}^T H_i \begin{bmatrix} x_i \\ u_i \end{bmatrix} + h_i^T \begin{bmatrix} x_i \\ u_i \end{bmatrix} + c_i, \tag{2a}$$
$$J_N(x_N) = x_N^T H_N x_N, \tag{2b}$$

where $H_i \geq 0$, the terminal constraint is an equality constraint of the form $F_N x_N = f_N$ and that input and state constraints can be written jointly in the form:

$$G_i \begin{bmatrix} x_i \\ u_i \end{bmatrix} \leq g_i. \tag{2c}$$

Letting $\oplus$ denote the direct matrix sum, $\otimes$ denote the Kronecker product, $I_p$ denote the $p$-dimensional identity matrix, $\mathbf{0}_{q\times r}$ denote a $q \times r$ matrix of zeros and $\mathbf{1}_{q\times r}$ denote a $q \times r$ matrix of ones, and let $n_{xu} = n_x + n_u$, then let

$$H = \left( \bigoplus_{i=0}^{N-1} H_i \right) \oplus H_N, \quad h = \begin{bmatrix} \mathbf{1}_{N\times 1} \otimes h_i \\ h_N \end{bmatrix},$$

$$F = \begin{bmatrix} I_N \otimes \begin{bmatrix} -I_{n_x} & \mathbf{0}_{n_{xu}} \end{bmatrix} & \mathbf{0}_{Nn_x \times n_x} \\ \mathbf{0}_{n_x \times Nn_{xu}} & -I_{n_x} \\ \mathbf{0}_{n_T \times Nn_{xu}} & F_N \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n_x \times N(n_{xu})} & \mathbf{0}_{n_x} \\ \bigoplus_{i=0}^{N-1} \begin{bmatrix} A_i & B_i \end{bmatrix} & \mathbf{0}_{Nn_x \times n_x} \\ \mathbf{0}_{n_T \times Nn_{xu}} & \mathbf{0}_{n_T \times n_x} \end{bmatrix}$$

$$f = \begin{bmatrix} -x(k)^T & (\mathbf{0}_{Nn_x \times 1})^T & f_N^T \end{bmatrix}^T$$

$$G = \left[ \left( \bigoplus_{i=0}^{N} G_i \right), \mathbf{0}_{Nn_c \times n_x} \right], \quad g = \begin{bmatrix} \mathbf{1}_{N\times 1} \otimes g_i \end{bmatrix}.$$

The QP representation of the MPC controller, with $N$ fixed can be written as:

$$\min_{\theta} \quad \frac{1}{2}\theta^T H \theta + h^T \theta + \sum_i c_i \tag{4a}$$
$$\text{subject to} \quad F\theta = f, \quad G\theta \leq g \tag{4b}$$

where $\theta = \begin{bmatrix} x_0^T & u_0^T & x_1^T & u_1^T & \dots & u_{N-1}^T & x_N^T \end{bmatrix}^T$. This "uncondensed" way of posing the optimisation problem associated with the MPC scales favourably in terms of prediction horizon [59–61] and is advantageous for model reconfiguration, since once the time-varying state-space matrices are computed, the remaining effort in constructing the optimisation problem is data transfer rather than the multiplication of large, dense matrices.



DOI: 10.1002/oca

### 3.3. Impulsive phase: Application specific details

During the longer range impulsive phase, fuel economy is important. Therefore a long prediction horizon is desired to enable complete manœuvres to be predicted. Since position accuracy during this phase is of secondary importance, a sampling period of $T_s = 300$ s is used in order to avoid an excessively large number of prediction steps. To improve numerical conditioning of the matrices within the PDIP algorithm, the state and input vectors are scaled [40, 41] so that relative positions are in units of 20 m, velocities in units of $1/60$ m/s and input impulses in units of $1/330$ m/s. (As also observed in [40, 41], applying a scaling is crucial to achieving good performance, since the convergence and solution accuracy of the iterative MINRES method used at each PDIP iteration is dependent on the conditioning of the set of linear equalities to be solved.) Letting $A_{\mathrm{YA}}(t_1, t_2)$ denote the Yamanaka-Ankersen [51] state propagation matrix corresponding to the start time $t_1$ and end time $t_2$ and un-scaled state vector $[x_{\mathrm{crf}}, y_{\mathrm{crf}}, z_{\mathrm{crf}}, \dot{x}_{\mathrm{crf}}, \dot{y}_{\mathrm{crf}}, \dot{z}_{\mathrm{crf}}]^T$, and letting $T_Q$ be a diagonal scaling matrix, $T_Q = \mathrm{Diag}(0.05, 0.05, 0.05, 60, 60, 60)$, then the time varying state prediction matrix at time step $k$ in terms of the scaled state vector are:

$$A_i = T_Q A_{\mathrm{YA}}((k+i)T_s, (k+i+1)T_s)T_Q^{-1}. \tag{5}$$

The plant inputs are discretised impulsively, corresponding to impulsive changes in velocity ($\Delta V$) at the beginning of the sampling period and split into positive and negative components

$$B_i = T_Q A_{\mathrm{YA}}((k+i)T_s, (k+i+1)T_s) \begin{bmatrix} 0 & 0 \\ I_3 & -I_3 \end{bmatrix} T_R^{-1} \tag{6}$$

where $T_R = I_{6\times 6} \times 330^{-1}$. When the contribution of each is constrained to be non-negative, a 1-norm of the input vector can be computed by summation of its elements. Defining $\beta \in \mathbb{R}_+$ as a weighting parameter, and $u_{\max}$ as a vector of maximum input values, and $x_T(k, N)$ as a time varying terminal point that should be reached at time $k + N$:

$$\ell_i(x_i, u_i) = (1 + \beta\|u_i\|_1) \tag{7a}$$
$$J_N(x_N) = 0 \tag{7b}$$
$$\mathbb{X}_i = \mathbb{R}^{n_x}, \tag{7c}$$
$$\mathbb{U}_i = \{u : 0 \le u \le u_{\max}\}, \tag{7d}$$
$$\mathbb{T}_N = x_T(k, N). \tag{7e}$$

Therefore, for the MPC controller during the impulsive phase, $\forall i \in \{0, \ldots, N-1\}$: $H_i = \mathbf{0}_{n_{xu}\times n_{xu}}$, $h_i = \begin{bmatrix}\mathbf{0}_{1\times n_x} & \beta\mathbf{1}_{1\times n_u}\end{bmatrix}^T$, $F_N = I_{n_x}$, $f_N = x_T$, $G_i = \begin{bmatrix}\mathbf{0}_{n_u\times n_x} & -I_{n_u} \\ \mathbf{0}_{n_u\times n_x} & I_{n_u}\end{bmatrix}$, $g_i = \begin{bmatrix}\mathbf{0}_{n_u\times 1} \\ u_{\max}\end{bmatrix}$. The weighting parameter is chosen by empirical tuning as $\beta = 35/330$. The time varying terminal point is computed as

$$x_T(k, N) = T_Q \left(h_p \rho \begin{bmatrix} 1 & 0 & -es/\rho^2 & -k_o^2 es & 0 & k_o^2(\rho - \rho^2)\end{bmatrix}^T\right) \tag{8}$$

where $e$ is the target orbit eccentricity, $h_p$ is the nominal in-track distance to the holding point, $\rho = (1 + e\cos\nu(k+N))$, $\nu(k+N)$ is the true anomaly of the target predicted at time $(k+N)T_s$ by propagating the mean anomaly and solving Kepler's equation by Newton's method [48], $s = \rho\sin(\nu(k+N))$, $k_o^2 = h/p^2$, $h$ is the target's orbital momentum, and $p$ is the semi-latus rectum (e.g. [47–49]) of the target's orbit. The maximum thrust constraint is chosen as $u_{\max} = T_R\mathbf{1}_6$, i.e. a maximum $\Delta V$ of 1 $\mathrm{ms}^{-1}$ may be delivered at each time step.

### 3.4. Tracking phase: Application specific details

During the final tracking phase, control accuracy and mitigation of uncertainty is important. Therefore a much lower sampling period, $T_s = 1$ s, is used. Define vector $y_r = [0, 0, z_{\mathrm{ref}}, v_{\mathrm{ref}}, 0, 0]$. Where $z_{\mathrm{ref}}$ is the "radial" position setpoint in the rotating reference frame and $v_{\mathrm{ref}}$ is the in-track



DOI: 10.1002/oca

velocity setpoint. Letting $\gamma$ be the heading angle, defined by

$$\tan\gamma(k+i) = e\sin\nu(k+i)/(1+e\cos\nu(k+i)) \tag{9}$$

where $e$ is the orbital eccentricity, and $\nu(k+i)$ is the true anomaly of the target at time $k+i$, define $s_\gamma \triangleq \sin\gamma(k+i)$ and $c_\gamma \triangleq \cos\gamma(k+i)$, then let

$$C_i \triangleq \begin{bmatrix} c_\gamma & 0 & -s_\gamma & & & \\ 0 & -1 & 0 & & & \\ s_\gamma & 0 & c_\gamma & & & \\ & & & c_\gamma & 0 & -s_\gamma \\ & & & 0 & -1 & 0 \\ & & & s_\gamma & 0 & c_\gamma \end{bmatrix}. \tag{10}$$

The MPC is a fixed, receding horizon controller with a $\ell_{\text{asso}}$ cost function. Letting diagonal weighting matrices $\hat{Q} \geq 0$, and diagonal $R > 0$, $R_\lambda \geq 0$ then for this MPC:

$$\ell_i(x_i, u_i) = (C_i x_i - y_r)^T \hat{Q}(C_i x_i - y_r) + u_i^T R u_i + \|R_\lambda u_i\|_1 \tag{11a}$$
$$J_N(x_N) = x_N^T Q_N x_N, \quad N = N_{MPC} \tag{11b}$$
$$\mathbb{X}_i = \mathbb{R}^{n_x}, \tag{11c}$$
$$\mathbb{U}_i = \{u : 0 \leq u \leq u_{\max}\}, \tag{11d}$$
$$\mathbb{T}_N = \mathbb{R}^{n_x}. \tag{11e}$$

Again, to improve numerical conditioning of the matrices within the PDIP algorithm, the state and input vectors are scaled [40, 41] so that relative positions are in units of $1$ m, relative velocities are in units of $1/40$ m/s, and input impulses in units of $1/250$ m/s. Thus, the state and input scaling matrices are:

$$T_Q = \text{Diag}(1, 1, 1, 40, 40, 40) \tag{12a}$$
$$T_R = \text{Diag}(250, 250, 250, 250, 250, 250). \tag{12b}$$

Let $Q_i = T_Q^{-1} C_i^T \hat{Q} C_i T_Q^{-1}$, then for the tracking phase, the same Yamanaka-Ankersen prediction model is used (but discretised with $T_s = 1$ s and scaled with the different values of $T_Q$ and $T_R$), and the following changes are made to the QP function:

$$H_i = Q_i \oplus R \tag{13a}$$
$$h_i = \begin{bmatrix} -(T_Q^{-1} C_i^T \hat{Q} y_r)^T & R_\lambda \mathbf{1}_{1\times n_x} \end{bmatrix}^T \tag{13b}$$
$$A_{N-1} = \mathbf{0}. \tag{13c}$$

The (appropriately scaled) tuning weights are chosen as follows:

$$\hat{Q} = \text{diag}(0, 0.3, 0.3, 210, 0, 0) \tag{14a}$$
$$\hat{R} = 1575^2 \times \text{diag}(0.0002, 0.0008, 0.0008, 0.0002, 0.0008, 0.0008) \tag{14b}$$
$$R = T_R^{-1} \hat{R} T_R^{-1} \tag{14c}$$
$$R_\lambda = 0.5 \times 1575 \times T_R^{-1} \times \text{diag}(0.0448, 0.0082, 0.0082, 0.0448, 0.0082, 0.0082). \tag{14d}$$

The zero weighting in the "$x$" position reflects that the controller should regulate to a specific velocity along a path to the target without concern for the exact position on the path.

In order to simplify the implementation, in particular to avoid the need for additional logic (for enabling/disabling the terminal equality constraint) the prediction matrix $A_{N-1}$ is set to $\mathbf{0}$ in the "tracking" second MPC phase, thus decoupling the final stage of $\theta$.



DOI: 10.1002/oca

## 4. PREDICTIVE CONTROL ALGORITHMS

Each of the individual convex QPs is solved using a primal-dual interior point (PDIP) algorithm. (Note that an LP is a QP with a zero Hessian matrix, and unlike some algorithms for constrained QPs, the PDIP algorithm for QP does not require a strictly positive definite Hessian matrix.) This is described in Figure 2(a), to identify the parts of the algorithm accelerated using custom hardware components implemented in the FPGA fabric. The notation $n_c$ is used to denote the total number of inequality constraints, and in this context, subscripts $\cdot_k$ are used to denote the iteration index. As well as including a maximum number of iterations $I_{IP}$ to ensure that computation time is bounded (even if this leads to a suboptimal solution), a tolerance on the duality measure $\mu_k$ is also included to avoid numerical problems due to elements of $s_k$ becoming too small (or their reciprocal becoming too large). Unlike many interior point implementations, but in common with [39–41], Mehrotra's Predictor Corrector [62] is *not* used, since an iterative linear solver (Section 4.1) is used and thus there are no matrix factorisations to recycle.

Step 9 in Figure 2(a) is where the majority of the computational effort is required — solving the set of linear simultaneous equations $\mathcal{A}_k c_k = b_k$. The multiplications in steps 6 and 10 also take a considerable amount of time because the prediction matrices used to construct the equality constraints $F$ change for each problem. Therefore, steps 6–10 are implemented as a custom peripheral core (PCORE) in the FPGA fabric, whilst the remainder are implemented in software on the Xilinx MicroBlaze soft core processor. Building the remainder of the structure of the matrix $\mathcal{A}_k$ in step (7) is heavy in terms of RAM accesses, and therefore including this within the custom PCORE rather than building the large matrix in software and communicating it at a limited data rate is also beneficial.

Prediction of the future target true anomaly through integration of the mean anomaly rate and solution of Kepler's equation using Newton's method [48] and calculation of the prediction matrices using the YA equations [51] is performed in the MicroBlaze, and communicated to the custom PCORE. However, the PCORE is responsible for storing them and building a representation of $\mathcal{A}_k$.

When a variable prediction horizon is used, some of the individual fixed horizon optimisation problems can be be infeasible for shorter horizons $N$, and this must be identified. Three heuristics (implemented in software) are used to identify infeasibility and/or a poor solution, based on methods used in [63]. The data within interior point algorithms can cover a wide dynamic range (and it is difficult to obtain *a priori* bounds on the magnitudes of the primal and dual decision variables and the step lengths), so in most parts of the implementation, floating-point arithmetic is used. Single precision (32-bit) is used in preference to the more pervasive double precision (64-bit) type since empirical evidence indicates that this is gives sufficient accuracy, and the latter would be emulated (slowly) in software on the MicroBlaze whereas the former can be handled natively. Also, the AXI communication bus used to transfer data from the MicroBlaze to the custom PCORE only allows 32-bit word lengths. Finally, single precision arithmetic requires fewer hardware resources than double precision in the PCORE. The exception to this design decision is within the Lanczos Process, where fixed point data types are used (together with problem-dependent pre-conditioning to bound the values [64–66]) to minimise resource usage.

### 4.1. *Iterative solution of linear system using MINRES*

As in [39, 40], the system of linear equations is solved using the minimum residual (MINRES) algorithm. This iterative method can be considered preferable to factorisation-based methods, since it is dominated by matrix-vector multiplications, which are very easily parallelised, and the number of iterations (i.e. the solution time) can be traded for solution accuracy. To facilitate description of how the hardware implementation is partitioned in the present design, the MINRES algorithm is summarised in Figure 2(b). The majority of the computation in the MINRES algorithm comprises the Lanczos process (steps 5–11). This method is implemented using fixed-point arithmetic, enabled by the preconditioner proposed by [64–66], which ensures that key variables are in the range $[-1, 1]$. Using the notation $Z_{k,mn}$ to denote the $n$th element of the $m$th row of matrix $Z$ at iteration $k$, by



DOI: 10.1002/oca

1: Initialisation: $\theta_0$, $\lambda_0$, $s_0$, $z_0$, $\sigma$, $I_{\text{IP}}$
2: **while** $k \leq I_{IP} - 1$ and $\mu_k \geq 10^{-6}$ **do**
▷ (Linearisation phase)
3: $\mu_k = z_k^T s_k / N n_c$
4: $w_k = z_k ./ s_k$ (elementwise)
5: $m_k = \begin{bmatrix} \theta_k \\ \lambda_k \\ \sigma\mu_k \mathbf{1}_{n_c} ./ s_k - \text{diag}(w_k) g_k + z_k \end{bmatrix}$

*(Steps 6–10 implemented in custom circuit)*

6: $\Phi_k = G^T \text{diag}(w_k) G$
7: $\mathcal{A}_k = \begin{bmatrix} H + \Phi_k & F^T \\ F & 0 \end{bmatrix}$
8: $b_k = \begin{bmatrix} -h \\ f \end{bmatrix} - \begin{bmatrix} H + \Phi_k & F^T & G^T \\ F & 0 & 0 \end{bmatrix} m_k$
▷ (Solution phase)
9: $\mathcal{A}_k c_k = b_k$, where $c_k = \begin{bmatrix} \Delta\theta_k \\ \Delta\lambda_k \end{bmatrix}$
10: Calculate $G(\theta_k + \Delta\theta_k) - g$

11: $\Delta z_k = \text{diag}(w_k)(G(\theta_k + \Delta\theta_k) - g) + \sigma\mu_k s_k^{-1}$
12: $\Delta s_k = -G(\theta_k + \Delta\theta_k) - s_k$
▷ (Line Search)
13: $\alpha_k = \max_{[0,1]} \alpha : \begin{bmatrix} \lambda_k + \alpha\Delta\lambda_k \\ s_k + \alpha\Delta s_k \end{bmatrix} > 0$
▷ (Update solution)
14: $(\theta, \lambda, z, s)_{k+1} = (\theta, \lambda, z, s)_k + \alpha\Delta(\theta, \lambda, z, s)_k$
15: $k \leftarrow k + 1$
16: **end while**

(a) PDIP

1: $\mathbf{v}_0$, $\mathbf{v}_1$, $\beta_1$, $\beta_2$, $\eta$
2: Calculate $\beta_2 = \sqrt{\mathbf{v}_1^T \mathbf{v}_1}$ and $\beta_2^{-1} = 1/\beta_2$
3: Set $\beta_1 \leftarrow \beta_2$
4: **for** $k = 0$ to $k_{\max}$ **do**
▷ Lanczos Process *(fixed point)*
5: Calculate $\beta_1^{-1} \mathcal{A}\mathbf{v}_1$
6: Calculate $\alpha = \beta_1^{-1} \mathbf{v}_1^T (\beta_1^{-1} \mathcal{A}\mathbf{v}_1)$
7: Calculate $(\beta_1^{-1} \mathcal{A}\mathbf{v}_1 - \beta_1 \mathbf{v}_0)$
8: Calculate $\mathbf{v}_2 = (\mathcal{A}\mathbf{v}_1 - \beta_1 \mathbf{v}_0) - \alpha\beta_1^{-1} \mathbf{v}_1$
9: Set $\mathbf{v}_0 \leftarrow \beta_1^{-1} \mathbf{v}_1$
10: Set $\mathbf{v}_1 \leftarrow \mathbf{v}_2$
11: Calculate $\beta_2 = \sqrt{\mathbf{v}_1^T \mathbf{v}_1}$ and $\beta_2^{-1} = 1/\beta_2$
▷ QR Factors and Givens Rotations *(single precision floating point)*
12: $\delta = \gamma_1 \alpha - \gamma_0 \sigma \beta_1$
13: $p_1^{-1} = (\delta^2 + \beta_2^2)^{-1/2}$
14: $p_2 = \sigma_0 \alpha + \gamma_0 \gamma_1 \beta_1$
15: $p_3 = \sigma_0 \beta_1$
16: Set $\gamma_0 \leftarrow \gamma_1$ and $\sigma_0 \leftarrow \sigma_1$
17: Set $\gamma_1 = \delta p_1^{-1}$ and $\sigma_1 = \beta_2 p_1^{-1}$
▷ Solution Update *(single precision floating point)*
18: $\mathbf{w}_0 = p_1^{-1}(\mathbf{v}_0 - p_3 \mathbf{w}_{m2} - p_2 \mathbf{w}_{m1})$
19: $\mathbf{x} \leftarrow \mathbf{x} + \gamma_1 \eta \mathbf{w}_0$
20: $\eta \leftarrow \sigma_1 \eta$
21: **end for**

(b) MINRES

Figure 2. Primal-dual Interior Point and MINRES Algorithms

letting

$$\mathcal{M}_{k,mn} = \begin{cases} \left( \sum_p |\mathcal{A}_{k,mp}| \right)^{-1/2} & \text{if } m = n \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

instead of directly solving $\mathcal{A}_k c_k = b_k$, MINRES is used to solve $(\mathcal{M}_k \mathcal{A}_k \mathcal{M}_k)\, \tilde{c}_k = (\mathcal{M}_k b_k) / \|\mathcal{M}_k b_k\|_2$, and $c_k = \mathcal{M}_k \|\mathcal{M}_k b_k\|_2 \tilde{c}_k$.

## 5. HARDWARE COMPONENT DESIGN

In this section, the methodology used to design the hardware component is described. In contrast to prior FPGA-based MPC designs, where register transfer languages (RTL) such as Verilog [33], VHDL [35, 37, 39, 40], or high level C-like languages (System-C, [36]) have been used, the custom peripheral core (PCORE) presented here is designed using a combination of MathWorks HDL Coder 2012b and Xilinx System Generator for DSP 14.4.

In [67], a complete interior point QP solver was implemented solely using built-in Simulink blocks to enable automatic C-code generation. Elsewhere, Simulink Coder/MATLAB Coder (*née* Real Time Workshop/Embedded MATLAB (EML)) have been used to compile M-code implementations of custom QP/LP solvers to C to accelerate simulation and simplify deployment (e.g. [20, 68, 69]). For FPGA synthesis the process is substantially more complex. It is insufficient



DOI: 10.1002/oca

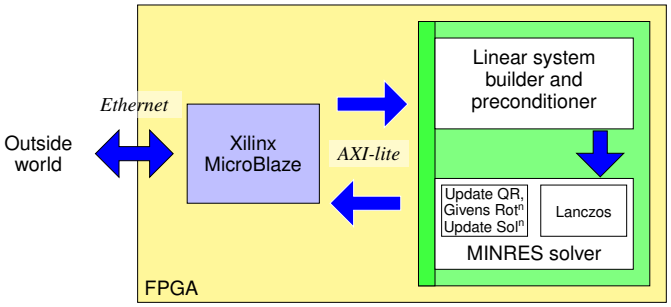



Figure 3. High-level architecture of controller implementation

to generate RTL from the same M-code used for simulation or C generation. Register level timing, numerical representation and parallelism of computation must also be considered to obtain a design suitable for an FPGA. Being required to consider these aspects can also be seen as an advantage, since it gives a fine-grained clock-cycle accurate control of the design, just as with an RTL. Using a Simulink-driven design approach enables high-level visualisation of connectivity between components and rapid simulation.

### *5.1. Architecture of peripheral core*

At a high level, the custom PCORE is split into two major parts. The first component performs steps 6-8 and 10 of the Algorithm in Figure 2(a), building the linear system, whilst the second, more complex component performs step 9. The PCORE is connected to the Xilinx MicroBlaze soft-core processor using the AXI4-lite bus, which presents data and status registers as shared memory locations (Figure 3). The MicroBlaze is clocked at 100 MHz, whilst the custom PCORE is clocked at 200 MHz.

### *5.2. Peripheral Core Interface*

The interface which allows the problem data to be loaded into the custom circuit from the MicroBlaze is implementing using Xilinx System Generator for DSP and Xilinx Platform Studio (XPS). These create the necessary bus interfaces and present the peripheral to the software component on the MicroBlaze as memory mapped registers and FIFO (First In, First Out) buffers.

### *5.3. Linear system builder*

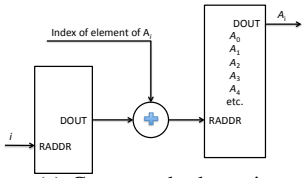
The rôle of the linear system builder component is to build the matrix $\mathcal{A}_k$ and vector $b_k$ and to output their values in a stream, from which the preconditioner (see Section 5.4) can be calculated and applied. The linear system builder is implemented using Xilinx System Generator for DSP since floating point arithmetic is desired, and its task is divided into four key sub-tasks:

1. receive the (row-wise) vectorised matrices $A_i$, $B_i$, $G_i$, $H_i$, $i \in \{0, \ldots, N_{\max} - 1\}$, $F_N$ and $P$, communicated from the MicroBlaze (in row major form), as well as the vectors $[\theta^T, \lambda^T, (\sigma\mu\mathbf{1}./s - \mathrm{diag}(w)g + z)^T]^T$, and $[-h^T, f^T]^T$;
2. compute $G^T \mathrm{diag}(w_k)G + H$ and store this row-wise in a further RAM of size $N_{\max}(n_x + n_u)^2 + n_x^2$;
3. construct a sequence of numbers comprising a representation of $\mathcal{A}_k$;
4. compute $b_k$.

The model matrices are initially stored in four separate dual port RAMs. The allocation of the matrices between the RAMs is described in Table II. Due to the time-varying nature of the model, the model matrices can be re-used between time steps. To facilitate this, the linear system builder also includes two additional shared RAMs. These contain the indices to address the first element of the time varying prediction matrices in the previously mentioned RAMs. Figure 4 explains



DOI: 10.1002/oca




(a) Conceptual schematic

| Elements | Description |
|---|---|
| 0 to $N_{\max}-1$ | Address of first element of $A_0, \dots A_{N_{\max}}-1$. |
| $N_{\max}$ to $2N_{\max}-1$ | Address of first element of $H_0, \dots H_{N_{\max}}-1$. |
| $2N_{\max}$ | Address of first element of $P$ |
| $2N_{\max}+1$ | Address of first element of $F_N$ |

(b) Contents of addressing shared RAM

Figure 4. Indexing of time varying prediction matrices

the concept of how these shared RAMs are used. Consequently the time varying matrices do not need to be stored in their temporal sequence and can even be repeated (e.g. for LTI cases). For deterministic LTV models, at each time step (apart from the initial one), only a single new model must be calculated, and the indices in these RAMS can be cycled, rather than having to re-write each prediction matrix into a new location.

The second sub-task requires computation of $N_{\max}(n_x+n_u)^2$ dot products per PD-IP iteration. To achieve a throughput of one scalar addition per clock cycle within each dot product (after an initial latency), elementwise multiplication is performed in single precision floating point using a pipelined IP-core, then converted to 64-bit signed fixed point with a 32-bit fractional part [34]. The accumulated result is transformed back to single precision floating point.

The matrix $\mathcal{A}_k$ is very sparse and has a well-defined structure. Instead of storing all of the zeros explicitly, the third subtask generates a sequence comprising the rows of a compact block representation of $\mathcal{A}_k$:

$$\left[\begin{array}{cccc} H_{\Phi xx,0} & H_{\Phi xu,0} & -I & A_0^T \\ H_{\Phi ux,0} & H_{\Phi uu,0} & 0 & B_0^T \\ \hline H_{\Phi xx,1} & H_{\Phi xu,0} & -I & A_1^T \\ H_{\Phi ux,1} & H_{\Phi uu,0} & 0 & B_1^T \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline H_{\Phi xx,N} & 0 & -I & [F_N^T, 0] \\ \hline -I & 0 & 0 & 0 \\ \hline A_0 & B_0 & -I & 0 \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline F_N & 0 & 0 & 0 \end{array}\right]. \tag{16}$$

This is less generic than interleaving the primal, dual and slack variables corresponding to each stage of the prediction horizon forming a banded matrix [59]. However with this application-specific storage structure, computing the dot products of the rows of (16) with appropriately sized vectors with a throughput of one dot product per clock cycle requires a bank of only $(3n_x+n_u)$ multipliers in parallel, in contrast to $(2n_u+4n_x-1)$ if a band structure had been used. This sequence of numbers is presented on an output of the linear system builder, and at this point is used to construct a matrix preconditioner (Section 5.4).

Table II. RAM allocations

| RAM | Contents |
|---|---|
| 1 | $A_i$, $B_i$, $H_i$, $P$, $F_N$ |
| 2 | $G_i$ |
| 3 | $[\theta^T, \lambda^T, (\sigma\mu\mathbf{1}./s - Wg + z)^T, w^T]^T$ |
| 4 | $[-h^T, f]^T$ |



DOI: 10.1002/oca

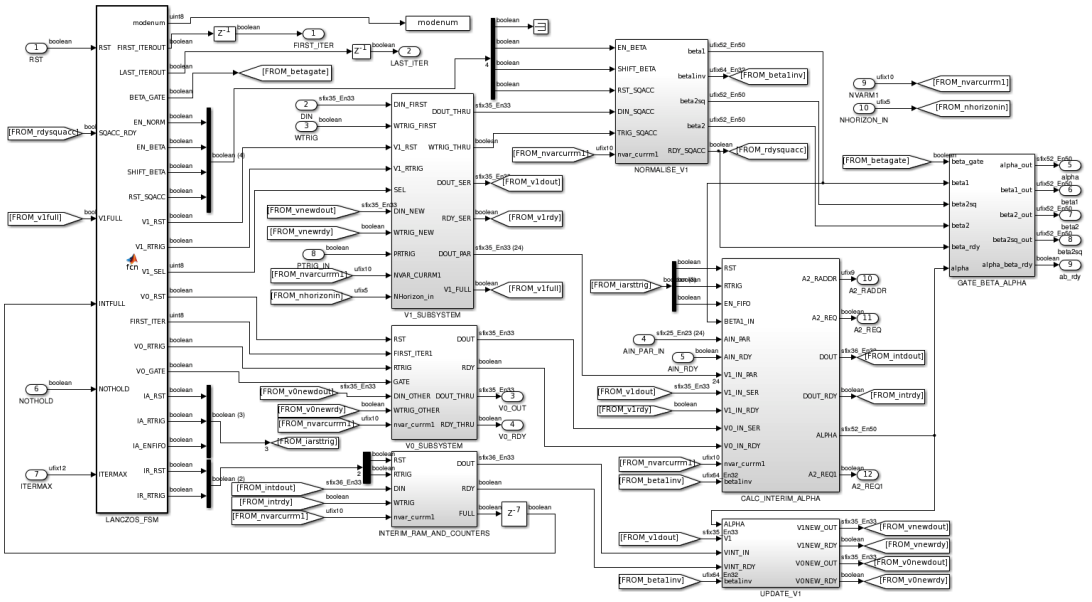



Figure 5. Simulink implementation of Lanczos algorithm, for implementation using HDL Coder

### 5.4. *Preconditioner*

The preconditioner (15) is computed from the row-wise sequence of the elements of (16) generated during the third subtask of the linear system builder, and stored simultaneously in a single-port RAM and an independent dual-port RAM. During the fourth subtask of the linear system builder (computing $b_k$), the preconditioner is simultaneously applied to $b_k$ and to the elements of the sequence constituting the matrix $\mathcal{A}_k$. A sequence of the preconditioned elements of $b_k$ is converted to a fixed-point data type, and output to the core MINRES solver, whilst the sequence of elements of $\mathcal{A}_k$ are converted to a fixed point representation and written to a bank of block RAMs, each of which contains a single column of the preconditioned version of (16).

### 5.5. *MINRES Solver (Lanczos)*

The circuit implementing the Lanczos algorithm for this application is implemented graphically using Mathworks HDL Coder 2012b (Figure 5).

As identified in [70], the key operation within the Lanczos process is a matrix-vector multiplication, $(\mathcal{M}_k\mathcal{A}_k\mathcal{M}_k)v$, performed at each iteration. Here, this is implemented using a bank of $(3n_x + n_u)$ multipliers, a bank of $(3n_x + n_u)$ single port RAMs comprising the columns of the preconditioned fixed point copy of (16) and a bank of $2n_x$ dual port RAMs and $n_u$ single port RAMs, containing appropriately ordered elements of $\mathbf{v}_1$ (in step 5 of MINRES), allowing a throughput of one dot-product per clock cycle. Unlike System Generator for DSP (and hence why it is used for this component) Simulink HDL coder transparently handles element-wise vector operations and automatically creates the tree-reduction structure for adding the products together, and perform register balancing between tree stages. However, to efficiently use the $25 \times 18$-bit hardware multiplier units (DSP48E) on the FPGA and to avoid timing issues at the implementation stage, the operands are split and long multiplication is performed in stages, with pipeline registers (delay blocks) inserted in between. Using the notation (u/s)Fix$xx_yy$ to denote an (un)signed $xx$-bit number with $yy$-bit fractional part, a data type of sFix25_23 is used for the matrix and sFix35_33 for the elements of $v$. This uses 2 DSP48Es per vector element.

The Lanczos process also requires a reciprocal square root operation to be performed. This is implemented as an integer square root operation followed by calculation of the reciprocal of the result. Due to the word lengths involved (sFix52_50), the Simulink built-in blocks generate a design with many layers of logic between registers even if automatic register balancing is used in the



DOI: 10.1002/oca

Table III. FPGA Resource Usage Breakdown

| Component | Register (FF) | Look-up Table (LUT) | Hardware Multiplier (DSP48E) | Block RAM (BRAM) |
|---|---|---|---|---|
| MicroBlaze | 19714 | 17633 | 167 | 47 |
| Lanczos | 9322 | 9622 | 89 | 4 |
| Remainder of PCORE | 15712 | 13975 | 78 | 32 |
| Total | 44748 | 41230 | 173 | 83 |
| % of Virtex-6 LX240T FPGA | 15% | 27% | 23% | 20% |

Table IV. Estimated power consumption of implementation on Xilinx ML605 Evaluation Board

| On-Chip | Power Consumption | | | |
|---|---|---|---|---|
| | Total | *MicroBlaze* | *PCORE (other)* | *PCORE (Lanczos)* |
| Clocks | 0.643 W | – | – | – |
| Signals | 0.100 W | 0.021 W | 0.043 W | 0.037 W |
| Logic | 0.466 W | 0.178 W | 0.085 W | 0.203 W |
| IOs | 1.959 W | – | – | – |
| Leakage | 3.526 W | – | – | – |
| Total | 6.694 W | – | – | – |

synthesis tools. These operations need only be performed on scalars and therefore do not need to be pipelined, so to circumvent this limitation, a standard integer division algorithm and an integer square root algorithm are implemented as state machines using M-code. The result of this operation is stored as type sFix64_32.

### *5.6. MINRES solver (other stages)*

Outside of the Lanczos process, the remainder of the algorithm is implemented using floating point arithmetic since the bounds on the magnitudes are not known *a priori*. This is designed using Xilinx System Generator for DSP to instantiate pipelined IP-cores performing floating point operations and type conversions, with the HDL description of the Lanczos process imported as a "black box". A single precision (32-bit) floating point data type is used in preference to the more common double precision (64-bit) representation, this decision being motivated by FPGA resource usage. Unlike in the preliminary design [24], steps 12 to 17 of the MINRES algorithm are computed using a state machine rather than simply chaining arithmetic components in series as a (mostly empty) pipeline. This reduces the number of dedicated hardware multipliers used in the FPGA.

### *5.7. System resource and power usage estimates*

Optimisations are carried out by the toolchain during synthesis and implementation which do not necessarily preserve the design hierarchy. Consequently an exact partition is not possible, but an approximate high-level FPGA resource breakdown analysed using Xilinx PlanAhead is presented in Table III. Approximately 1/4 of the resources of the mid-range Virtex-6 LX 240T FPGA are used by the design. The implication is that the design could also fit comfortably inside a smaller, lower power device, or that the remaining resources could be used for another purpose.

Power consumption is also estimated using Xilinx Power Analyser (XPA) to determine the allocation of consumption between the components. The results of this are presented in Table IV, with signal and logic power consumption also partitioned between that consumed by the MicroBlaze, the fixed-point Lanczos component and the remainder of the custom PCORE. It should be noted that whilst the values of the figures using this particular FPGA are disappointingly high, as identified in the analyses of [24] using a smaller or more recent generation FPGA will substantially reduce power consumption, particularly in terms of leakage. A lower clock frequency and configuration of the synthesis and implementation toolchain to emphasise power optimisation



DOI: 10.1002/oca

Table V. Sensor noise magnitudes

| Range | State | $3\sigma$ | Units |
|---|---|---|---|
| $r \leq 100$ m | $x, y, z$ | 3.5 | cm |
| | $\dot{x}, \dot{y}$ | 7.0 | mm s$^{-1}$ |
| | $\dot{z}$ | 9.0 | mm s$^{-1}$ |
| 100 m $< r \leq 5$ km | $x, y, z$ | $-2.55 + 0.0605r$ | cm |
| | $\dot{x}, \dot{y}$ | $5.10 + 0.0190r$ | mm s$^{-1}$ |
| | $\dot{z}$ | $7.14 + 0.0186r$ | mm s$^{-1}$ |
| 5 km $< r$ | $x, y, z$ | $0.10r$ | cm |
| | $\dot{x}, \dot{y}, \dot{z}$ | $0.03r$ | mm s$^{-1}$ |

* $r$ is the numerical value of the Euclidean distance separating target and chaser, measured in metres.

(which will also limit the feasible clock frequency) will also reduce power consumption, as will integration into a less general system than the ML605 evaluation board.

## 6. CLOSED LOOP SIMULATION

An FPGA-in-the-loop setup is used to verify the controller, with the chaser and target spacecraft simulated using Simulink, communicating with the FPGA-based MPC controller using UDP/IP over Ethernet.

### *6.1. Simulator and scenario*

A nonlinear model of the chaser and target spacecraft are modelled as point masses orbiting a spherical central body with uniform mass distribution. The trajectories of their respective orbital elements are modelled using Gauss' equations [48, 71]. Attitude dynamics are not considered. The chaser's relative navigation values (i.e. plant state) are computed based on its own position in an inertial body-centred body fixed (BCBF) reference frame centred on the central body. Perfect knowledge of the chaser's own position is assumed. Relative navigation is based on the relative position of the target in a LVLH (local vertical, local horizontal) cartesian reference frame centred on the chaser. The measurements are corrupted by Gaussian white noise with the standard deviations given in Table V. This is transformed into the BCBF frame and the estimated relative position in the BCBF frame is used to compute the estimated absolute position of the target in the BCBF reference frame, from which the target's orbital elements can be computed. The relative BCBF position measurements are also used to compute the relative position in a cartesian local orbital frame centred on the target, from which the relative position in the cylindrical orbital frame (CRF) is also computed. These transformations are standard (e.g. [48]) and are therefore not explicitly presented here.

Clearance to "leave" a given holding point is given once the computed solution to the VH-MPC has given an optimal solution of $N_{\text{opt}} \leq 3$ for 6 consecutive time steps. (The triggering horizon length must be greater than unity due to the terminal equality constraint, and the additive uncertainty.) This decision logic is implemented as part of the simulator rather than the control solution.

### *6.2. Tuning the number of iterations*

Since a fixed upper bound on the number of interior point iterations $I_{IP}$ and the number of MINRES iterations $I_{MR}$ is imposed, appropriate values for these parameters need to be determined.

For the impulsive and tracking phases separately, simulations have been carried for a range of values of each of these, starting from the initial conditions documented in Table I. For the MINRES iterations, the value of $I_{MR}$ is chosen based on the number of decision variables and a scaling factor $\eta$ such that $I_{MR} = \eta\,(N(2n_x + n_u) + 2n_x + n_T)$, rounded to the nearest integer (where $n_T = n_x$



DOI: 10.1002/oca

Table VI. Control performance varying maximum iteration numbers

(a) Impulsive Phase

| | Completion time (multiples of 1000s) $I_{MR}$ scaling, $\eta$ | | | | $\Delta V$ usage (m/s) $I_{MR}$ scaling, $\eta$ | | | |
|---|---|---|---|---|---|---|---|---|
| $I_{IP}$ | 0.9 | 1.0 | 1.1 | 1.2 | 0.9 | 1.0 | 1.1 | 1.2 |
| 20 | 36.0 | 36.0 | 36.0 | 36.0 | 8.7 | 8.8 | 8.8 | 8.8 |
| 22 | 34.5 | 34.5 | 34.5 | 34.5 | 7.6 | 7.6 | 7.6 | 7.6 |
| 24 | 34.5 | 34.5 | 34.5 | 34.5 | 7.6 | 7.6 | 7.6 | 7.6 |
| 26 | 34.5 | 34.5 | 34.5 | 34.5 | 7.6 | 7.6 | 7.6 | 7.6 |
| 28 | 34.5 | 34.5 | 34.5 | 34.5 | 7.6 | 7.6 | 7.6 | 7.6 |

(b) Tracking phase

| | Completion time (s) $I_{MR}$ scaling, $\eta$ | | | | | | $\Delta V$ usage (m/s) $I_{MR}$ scaling, $\eta$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $I_{IP}$ | 0.1 | 0.2 | 0.4 | 0.8 | 0.9 | 1.0 | 0.1 | 0.2 | 0.4 | 0.8 | 0.9 | 1.0 |
| 12 | 1302 | 1304 | 1304 | 1304 | 1304 | 1304 | 0.66 | 0.68 | 0.69 | 0.69 | 0.69 | 0.69 |
| 14 | 1297 | 1298 | 1298 | 1298 | 1298 | 1298 | 0.61 | 0.62 | 0.62 | 0.62 | 0.62 | 0.62 |
| 16 | 1296 | 1297 | 1297 | 1297 | 1297 | 1297 | 0.61 | 0.62 | 0.62 | 0.62 | 0.62 | 0.62 |
| 18 | 1297 | 1297 | 1297 | 1297 | 1297 | 1297 | 0.61 | 0.61 | 0.61 | 0.62 | 0.62 | 0.62 |
| 20 | 1297 | 1297 | 1297 | 1297 | 1297 | 1297 | 0.61 | 0.61 | 0.61 | 0.61 | 0.61 | 0.61 |
| 22 | 1297 | 1297 | 1297 | 1297 | 1297 | 1297 | 0.61 | 0.61 | 0.61 | 0.61 | 0.61 | 0.61 |

Table VII. Computation time (in milliseconds) for a single variable-horizon controller evaluation with varying maximum iteration numbers

(a) Impulsive phase

| | $I_{MR}$ scaling, $\eta$ | | | |
|---|---|---|---|---|
| $I_{IP}$ | 0.9 | 1.0 | 1.1 | 1.2 |
| 20 | 625 | 654 | 683 | 712 |
| 22 | 686 | 719 | 750 | 783 |
| 24 | 748 | 783 | 818 | 853 |
| 26 | 809 | 848 | 885 | 923 |
| 28 | 865 | 913 | 953 | 990 |

(b) Tracking phase

| | $I_{MR}$ scaling, $\eta$ | | | | | |
|---|---|---|---|---|---|---|
| $I_{IP}$ | 0.1 | 0.2 | 0.4 | 0.8 | 0.9 | 1.0 |
| 12 | 29 | 31 | 35 | 44 | 46 | 48 |
| 14 | 32 | 35 | 40 | 50 | 52 | 55 |
| 16 | 36 | 38 | 44 | 56 | 59 | 61 |
| 18 | 39 | 42 | 49 | 62 | 65 | 68 |
| 20 | 43 | 46 | 54 | 68 | 71 | 75 |
| 22 | 46 | 48 | 56 | 71 | 75 | 78 |

Table VIII. Selected iteration bounds

| | Impulsive phase | | Tracking phase | |
|---|---|---|---|---|
| Symbol | $I_{IP}$ | $\eta$ | $I_{IP}$ | $\eta$ |
| Value | 24 | 1.1 | 20 | 0.4 |

is the number of individual terminal equality constraints). Metrics pertaining to closed-loop control performance for an instance of each phase are presented in Table VI, whilst maximum computation times (measured as time between sending a UDP packet to the FPGA and receiving the response) for each controller evaluation are presented in Table VII. Note, that this corresponds to solution of the variable horizon problem (1), not just a single horizon. The computation times are small in comparison to the sampling periods of $T_s = 300$ s and $T_s = 1$ s, so implementation at much lower clock frequencies would be feasible without adversely affecting control performance (although we do not pursue this here, since the timing results will scale directly).

### *6.3. Closed-loop end-to-end simulation analysis*

Based on the previous analysis, and allowing a conservative additional margin, the parameters presented in Table VIII are chosen to evaluate the control performance of both mission phases



DOI: 10.1002/oca

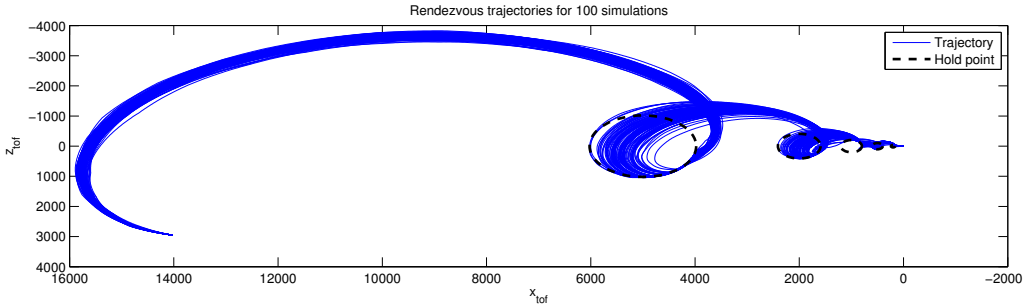



Figure 6. End-to-end rendezvous trajectories for 100 scenarios

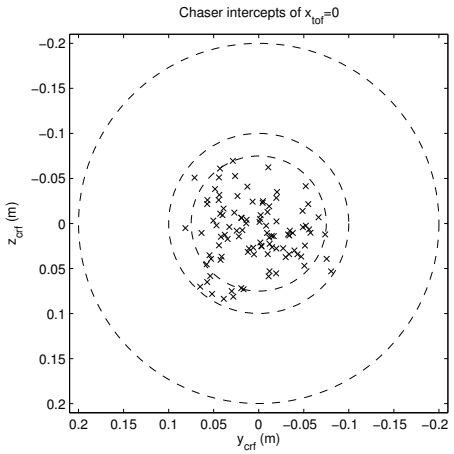



Figure 7. Terminal capture accuracy for 100 scenarios

joined together. One hundred end-to-end simulations have been run from the same initial conditions with varying random number seeds for the additive uncertainties. The trajectories in the $x-z$ plane are shown in Figure 6, whilst the terminal capture accuracy in the $y-z$ plane is shown in Figure 7. All scenarios successfully visit the holding points and intercept the target within a $10$ cm tolerance. Whilst the control action applied is determined by the solution obtained from the FPGA, a PC-based solution is also computed using CPLEX. The distribution of the (base-10 logarithm of the) 2-norm of the error between the control input obtained by solution with CPLEX and that obtained from solution on the FPGA evaluated at each time step of each simulation is presented as a histogram in Figure 8, and compared with 2-norm of the actual applied input. The errors can be seen to be small in comparison to the input magnitudes, providing confidence in the FPGA-based implementation, despite the use of low precision arithmetic and iterative methods.

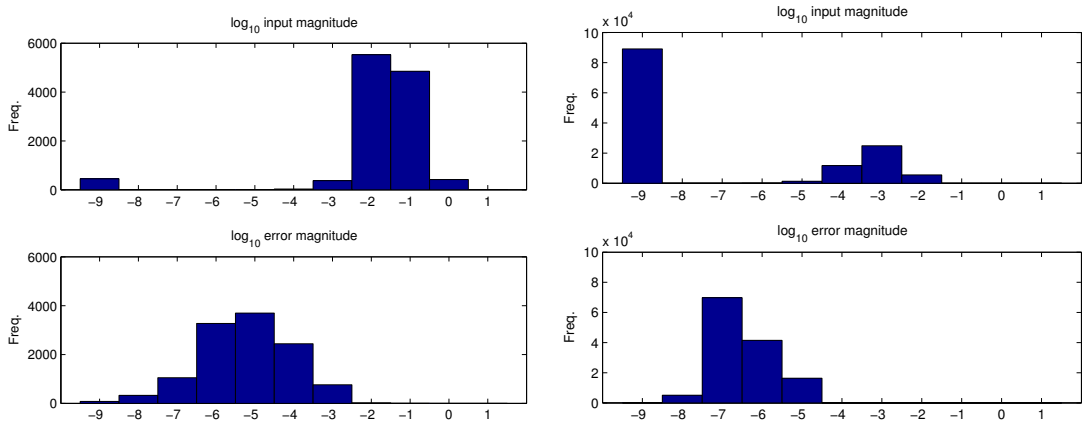
### *6.4. Timing breakdown*

A breakdown of the distribution of compute effort between the different PDIP algorithm phases (Figure 2)(a)) is presented for the final tracking phase (this analysis is simpler with a fixed horizon) in Table IX.

Approximately $2/5$ of the time spent per evaluation of the MPC control law is accounted for by the custom linear system builder and MINRES solver. If this time were doubled or quadrupled (by halving or quartering the clock frequency of the custom circuit), the solution would still be computed within the sampling period $T_s = 1$ s. The shortfall between the total for the corresponding timing in Table VII is accounted for by Ethernet communication time, timing/instrumentation code, and other overhead of the UDP server function.



DOI: 10.1002/oca




(a) Impulsive phase (b) Tracking phase

Figure 8. Magnitude of 2-norm error between FPGA-based MPC computed control and "ideal" PC-based MPC computed control over 100 scenarios ($-9$ includes magnitudes $\leq 10^{-9}$)

Table IX. Detailed computational profiling for tracking phase

| Description | Runs | Time (ms) Once | Time (ms) Total | Alg. Step | Resource used Microblaze | Resource used PCORE | Resource used AXI |
|---|---|---|---|---|---|---|---|
| Compute/transfer LTV model | 1 | 5.01 | 5.01 | – | x | | x |
| Initialise PDIP | 1 | 0.04 | 0.04 | 1 | x | | |
| Calc. $\mu_k$ and $w_k$ | 20 | 0.14 | 2.82 | 3–4 | x | | |
| Calc. residual | 20 | 0.15 | 3.05 | 5 | x | | |
| Transfer to PCORE | 20 | 0.11 | 2.15 | 5 | | | x |
| **Linear system & MINRES** | **20** | **1.00** | **19.94** | 6–9 | | **x** | |
| Transfer $\Delta\theta_k$ | 20 | 0.12 | 2.36 | 9 | | | x |
| Wait for next result | 20 | 0.00 | 0.02 | 10 | x | x | |
| Transfer $G(\theta_k + \Delta\theta_k) - g$ | 20 | 0.08 | 1.50 | 10 | | | x |
| Sanity check for NaN | 20 | 0.17 | 3.30 | – | x | | |
| Calc. $\Delta s_k$, $\Delta z_k$ | 20 | 0.10 | 1.92 | 11–12 | x | | |
| Line search | 20 | 0.27 | 5.44 | 13 | x | | |
| Update solution | 20 | 0.16 | 3.18 | 14 | x | | |
| Check iter. infeas. | 20 | 0.01 | 0.19 | | x | | |
| Check final infeas. | 1 | 0.01 | 0.01 | | x | | |
| Total | | | 50.94 | | | | |

*Figures are rounded to 2 decimal places *after* computing totals!

Table X. Estimated solution time if [39, 41] had been used for an LTI system of the same size

| | Time (ms) $c = 0$ | $c = 20$ | $c = 30$ | $c = 40$ |
|---|---|---|---|---|
| $f_c = 100$ MHz | 55.43 | 62.87 | 66.59 | 70.31 |
| $f_c = 200$ MHz | 27.71 | 31.43 | 33.29 | 35.15 |

The solution time is comparable with that from pure hardware QP solvers for LTI systems. Substituting $n_x$, $n_u$, $I_{IP}$ and $I_{MR}$ and a range of frequency $f_c$ and candidate sequential/parallel stage timing offsets $c$ (n.b. $c$ is not a tuning parameter defined by the problem, but must be obtained experimentally) into Equation 9 of [41] indicates that the speed of the present hybrid implementation is comparable (Table X). A speedup might be obtained (in exchange for additional resource usage and complexity of the custom PCORE) by moving more computation to the PCORE. This would be most efficiently realised through exploiting pipelining and avoidance of communication overhead rather than massive parallelism.



DOI: 10.1002/oca

Table XI. Solution time comparison with software solvers for tracking phase

| Location | Solver Solver | Precision | # Iterations | Computation time (ms) Model computation | Total Solve | Mean/iteration |
|---|---|---|---|---|---|---|
| PC | CPLEX | double | – | 11.46 | 7.87 | – |
| PC | FORCES | double | 9 | 11.74 | 1.82 | 0.20 |
| PC | FORCES | single | 20 | – | 4.81 | 0.24 |
| PC | CVXGEN | double | 7 | – | 2.04 | 0.29 |
| PC | CVXGEN | single | 25 | – | 5.90 | 0.23 |
| MicroBlaze | FORCES | double | 8 | – | $\approx$ 77000 | $\approx$ 9625 |
| MicroBlaze | FORCES | single | 20 | – | 9527.85 | 476.39 |
| MicroBlaze | CVXGEN | double | 7 | – | $\approx$ 60000 | $\approx$ 8571 |
| MicroBlaze | CVXGEN | single | 12 | – | 1475.51 | 122.96 |
| **FPGA** | **PDIP/MINRES** | **single/fix** | **20** | **5.01** | **45.94** | **2.30** |

### *6.5. Timing comparison with other tools*

Due to the way the control problem has been formed, opportunities for exact comparison with other embedded solvers are limited. As a representative example, considering the final tracking phase with fixed control horizon, comparison results are presented for the PC-based CPLEX solver [72] and the embedded PDIP QP solvers generated using CVXGEN [73] and FORCES [74] tools in double precision arithmetic and adapted to single precision. The advantages of the proposed design for the variable horizon phase can be inferred by analogy. We only compare with other *uncondensed* MPC methods. (A comprehensive study comparing multiple algorithms, dense and sparse formulations, with different horizon lengths is well beyond the scope of this paper.) (N.B. To obtain results with FORCES with single precision arithmetic, we had to regularise our cost matrices by adding $10^{-6}I$ to the $Q$ and $P$ matrices. This is numerically small compared to the other values involved. Using single precision arithmetic, but an otherwise default parameter configuration, the software-based solvers require more iterations when using single precision arithmetic.)

A sample of the same problem data that triggered the worst-case number of iterations on the PC for FORCES with single precision arithmetic is used to compare the performance of FORCES and CVXGEN on the MicroBlaze. Since the data which causes each solver to struggle is different, and a comparison of the merits of the approaches taken by the software solvers is emphatically not the motivation for this exercise, a "mean time per iteration" is also presented to provide a level playing field.

The key "take home message" from the data presented in Table XI is that even without considering the time to compute the time varying YA prediction model, the 100 MHz MicroBlaze cannot maintain sufficient throughput for the solvers generated using FORCES and CVXGEN to meet the sampling period of 1 s imposed for the continuous thrust phase. However the hybrid software/hardware PDIP/MINRES implementation presented in this work is capable of providing the requisite solution in substantially less than this period. Since FORCES and CVXGEN both use variants of PDIP algorithms, one can make the hypothesis that apart from the linear system construction and solution the types of operations carried out with these will have similar overhead to the hybrid software/hardware implementation, and that the main factor accounting for the differences in timings is therefore the construction and solution of the set of linear equations.

The software-based results using double precision arithmetic (emulated, since the MicroBlaze floating point unit only supports single precision) reinforce the idea that even if custom hardware is not considered, the use of double precision floating point can be an unaffordable luxury in an embedded setting.



DOI: 10.1002/oca

## 7. CONCLUSIONS

This paper has presented an FPGA-based implementation of a controller to perform two phases of a spacecraft rendezvous in an elliptical orbit for which the dynamics are described by a linear time *varying* model. The first phase also employs a variable prediction horizon. By combining software running on a soft-core MicroBlaze processor instantiated on the FPGA with a custom peripheral core designed using high level graphical tools, computation times are obtained that are comparable with those estimated from a pure hardware implementation of a similar algorithm that is restricted to LTI systems with a fixed prediction horizon. The setup is substantially faster than using a purely software-based state-of-the-art embedded QP solver in the same MicroBlaze, attempts at which failed to find a solution in the requisite sampling period. An "FPGA-in-the-loop" setup controlling a nonlinear simulation of the spacecraft demonstrates that control performance of the implemented system equivalent to that obtained using a general purpose QP solver is achieved.

Whilst interior point methods have been applied in the present work and demonstrated to work in a satisfactory manner, it would also be interesting to investigate application of other iterative algorithms that have attracted recent attention in the MPC literature to this problem, and to investigate the effectiveness of each method in handling longer range (and implicitly less well conditioned) rendezvous phases with low precision arithmetic, as well as consideration of a wider variety of constraints.

ACKNOWLEDGEMENT

This work was supported by EPSRC Grant EP/G030308/1, with industrial support from Xilinx, Mathworks, and the European Space Agency, and project collaboration with George Constantinides, Eric Kerrigan, Juan Jerez and Andrea Suardi of Imperial College London.

REFERENCES

1. Maciejowski JM. *Predictive Control with Constraints*. Pearson Education: Harlow, UK., 2002.
2. Camacho EF, Bordons C. *Model predictive control*. Springer-Verlag: London, 2004.
3. Rawlings JB, Mayne DQ. *Model predictive control: Theory and design*. Nob Hill Publishing: Madison, Wisconsin, 2009.
4. Rossi M, Lovera M. A predictive approach to formation keeping for constellations of small spacecraft in elliptic orbits. *Proceedings of the 5th ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, 516, Frascati, Rome, 2003; 209–216.
5. Breger L, How JP. J2-modified GVE-based MPC for formation flying spacecraft. *Proceedings of the 2005 AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, 2005; 158–169, doi:10.2514/6.2005-5833.
6. Larsson R, Berge S, Bodin P, Jönsson U. Fuel efficient relative orbit control strategies for formation flying and rendezvous within PRISMA. *Proceedings of the 29th Annual AAS Guidance and Control Conference*, Breckenridge, CO, 2006.
7. Breger L, How JP. Gauss's variational equation-based dynamics and control for formation flying spacecraft. *Journal of Guidance, Control, and Dynamics* 2007; **30**(2):437–448, doi:10.2514/1.22649.
8. Losa D. High vs low thrust station keeping manever planning for geostationary satellites. PhD Thesis, Ecole Nationale Superieure des Mines de Paris February 2007. URL http://pastel.paristech.org/2163/.
9. Crassidis JL, Markley FL, Anthony TC, Andrews SF. Nonlinear predictive control of spacecraft. *Journal of Guidance, Control, and Dynamics* 1997; **20**(6):1096–1103, doi:10.2514/2.4191.
10. Hegrenæs O, Gravdahl JT, Tondel P. Spacecraft attitude control using explicit model predictive control. *Automatica* December 2005; **41**(12):2107–2114, doi:10.1016/j.automatica.2005.06.015.
11. Myung HS, Bang H. Nonlinear predictive attitude control of spacecraft under external disturbances. *Journal of Spacecraft and Rockets* 2003; **40**(5):696–699, doi:10.2514/2.6896.
12. Wood M, Chen WH, Fertin D. Model predictive control of low earth orbiting spacecraft with magneto-torquers. *Proceedings of the IEEE International Conference on Control Applications*, Munich, Germany, 2006; 2908–2913, doi:10.1109/CACSD-CCA-ISIC.2006.4777100.
13. Wood M, Chen WH. Model predictive control of low Earth orbiting satellites using magnetic actuation. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 2008; **222**(6):619–631, doi:10.1243/09596518JSCE505.
14. Chen X, Wu X. Design and implementation of model predictive control algorithms for small satellite three-axis stabilization. *Proceedings of the International Conference on Information and Automation*, Shenzhen, China, 2011; 666–671, doi:10.1109/ICINFA.2011.5949077.
15. Richards A, How J. Performance evaluation of rendezvous using model predictive control. *AIAA Guidance, Navigation and Control Conference and Exhibit*, Austin, Texas, 2003, doi:10.2514/6.2003-5507.

16. Breger LS, How JP. Safe trajectories for autonomous rendezvous of spacecraft. *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone, Colorado, 2006, doi:10.2514/6.2006-6584.
17. Breger JS, How JP. Powered safe abort for autonomous rendezvous of spacecraft. *AIAA Guidance, Navigation and Control Conference*, 2007, doi:10.2514/6.2007-6860.
18. Breger L, How JP. Safe trajectories for autonomous rendezvous of spacecraft. *Journal of Guidance Control and Dynamics* September–October 2008; **31**(5):1478–1489, doi:10.2514/1.29590.
19. Saponara M, Barrena V, Bemporad A, Hartley EN, Maciejowski J, Richards A, Tramutola A, Trodden P. Model predictive control application to spacecraft rendezvous in Mars Sample Return scenario. *Proceedings of the 4th European Conference for Aerospace Sciences (EUCASS)*, St. Petersburg, Russia, 2011.
20. Hartley EN, Trodden PA, Richards AG, Maciejowski JM. Model predictive control system design and implementation for spacecraft rendezvous. *Control Engineering Practice* July 2012; **20**(7):695–713, doi:10.1016/j.conengprac.2012.03.009.
21. Gavilan F, Vazquez R, Camacho EF. Chance-constrained model predictive control for spacecraft rendezvous with disturbance estimation. *Control Engineering Practice* February 2012; **20**(2):111–122, doi:10.1016/j.conengprac.2011.09.006.
22. Di-Cairano S, Park H, Kolmanovsky I. Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering. *International Journal of Robust and Nonlinear Control* August 2012; **22**(12):1398–1427, doi:10.1002/rnc.2827.
23. Kolmanovsky I, Baldwin M, Erwin RS. Model predictive control of three dimensional spacecraft relative motion. *Proceedings of the American Control Conference*, Montreal, Canada, 2012; 173–178.
24. Hartley EN, Maciejowski JM. Predictive control for spacecraft rendezvous in an elliptical orbit using an FPGA. *Proceedings of the European Control Conference*, Zurich, Switzerland, 2013; 1359–1364.
25. Hartley EN, Gallieri M, Maciejowski JM. Terminal spacecraft rendezvous and capture using LASSO MPC. *International Journal of Control* 2013; **86**(11):2104–2113, doi:10.1080/00207179.2013.789608.
26. Hartley EN. Graphical FPGA design for a predictive controller with application to spacecraft rendezvous. *Proceedings of the IEEE Conference on Decision and Control*, Florence, Italy, 2013; 1971–1976.
27. Richards A, How J, Schouwenaars T, Feron E. Plume avoidance maneuver planning using mixed integer linear programming. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, 2001, doi:10.2514/6.2001-4091.
28. Sauter L, Palmer P. Analytic model predictive controller for collision-free relative motion reconfiguration. *Journal of Guidance, Control, and Dynamics* July–August 2012; **35**(4):1069–1079, doi:10.2514/1.56521.
29. Huang R, Hwang I, Corless M. A new nonlinear model predictive control algorithm using differential transformation with application to interplanetary low-thrust trajectory tracking. *Proceedings of the 2009 American Control Conference*, St. Louis, MO, USA, 2009; 4868–4873, doi:10.1109/ACC.2009.5160555.
30. Starek JA, Kolmanovsky IV. Nonlinear model predictive control strategy for low thrust spacecraft missions. *Optimal Control Applications and Methods* 2014; **35**(1):1–20, doi:10.1002/oca.2049.
31. Bodin P, Noteborn R, Larsson R, Chasset C. System test results from the GNC experiments on the PRISMA in-orbit test bed. *Acta Astronautica* April 2011; **68**(7–8):862–872, doi:10.1016/j.actaastro.2010.08.021.
32. Vouzis PD, Bleris LG, Arnold MG, Kothare MV. A system-on-a-chip implementation for embedded real-time model predictive control. *IEEE Transactions on Control Systems Technology* 2009; **17**(3):1–12, doi:10.1109/TCST.2008.2004503.
33. Yang N, Li D, Zhang J, Xi Y. Model predictive controller design and implementation on FPGA with application to motor servo system. *Control Engineering Practice* 2012; **20**(11), doi:10.1016/j.conengprac.2012.06.012.
34. Chen H, Xu F, Xi Y. Field programmable gate array/system on a programmable chip-based implementation of model predictive controller. *IET Control Theory and Applications* Jul 2012; **6**(8):1055–1063, doi:10.1049/iet-cta.2010.0443.
35. Wills AG, Knagge G, Ninness B. Fast linear model predictive control via custom integrated circuit architecture. *IEEE Transactions on Control Systems Technology* January 2012; **20**(1):59–71, doi:10.1109/TCST.2010.2096224.
36. Ling KV, Wu BF, Maciejowski JM. Embedded model predictive control (MPC) using a FPGA. *Proceedings of the 17th IFAC World Congress*, Seoul, Korea, 2008; 15 250–15 255, doi:10.3182/20080706-5-KR-1001.02579.
37. Wills A, Mills A, Ninness B. FPGA implementation of an interior-point solution for linear model predictive control. *Proceedings of the 18th IFAC World Congress*, Milan, Italy, 2011; 14 527–14 532, doi:10.3182/20110828-6-IT-1002.02857.
38. Mills A, Wills AG, Weller SR, Ninness B. Implementation of linear model predictive control using a field-programmable gate array. *IET Control Theory & Applications* May 2012; **6**(8):1042–1054, doi:10.1049/iet-cta.2010.0739.
39. Jerez JL, Ling KV, Constantinides GA, Kerrigan EC. Model predictive control for deeply pipelined field-programmable gate array implementation: algorithms and circuitry. *IET Control Theory and Applications* 2012; **6**(8):1029–1041, doi:10.1049/iet-cta.2010.0441.
40. Hartley EN, J, Suardi A, Maciejowski JM, Kerrigan EC, Constantinides GA. Predictive control of a Boeing 747 aircraft using an FPGA. *Proceedings of the IFAC conference on Nonlinear Model Predictive Control*, Noordwijkerhout, NL, 2012; 80–85, doi:10.3182/20120823-5-NL-3013.00016.
41. Hartley EN, Jerez JL, Suardi A, Maciejowski JM, Kerrigan EC, Constantinides GA. Predictive control using an FPGA with application to aircraft control. *IEEE Transactions on Control Systems Technology* 2013; (**Article in press**), doi:10.1109/TCST.2013.2271791.
42. Jerez JL, Goulart PJ, Richter S, Constantinides GA, Kerrigan EC, Morari M. Embedded predictive control on an FPGA using the fast gradient method. *Proceedings of the European Control Conference*, Zurich, 2013; 3614–3620.
43. Gallieri M, Maciejowski JM. LASSO MPC: Smart regulation of over-actuated systems. *Proceedings of the American Control Conference*, Montreal, Canada, 2012; 1217–1222.
44. Mattsson D, Christensson M. Evaluation of syntheizable CPU cores. Master's Thesis, Chalmers University of Technology 2004. URL http://www.gaisler.com/doc/Evaluation_of_synthesizable_CPU_

cores.pdf.
45. Xilinx. *ML605 Hardware User Guide* February 15 2011.
46. Beaty D, Grady M, May L, Gardini B. Preliminary planning for an international Mars Sample Return mission. Report of the International Mars Architecture for the Return of Samples (iMARS) Working Group June 01 2008. URL http://www.esa.int/esaMI/Aurora/SEM1PM808BE_0.html.
47. Kaplan MH. *Modern spacecraft dynamics & control*. Wiley: New York, 1976.
48. Sidi MJ. *Spacecraft dynamics and control: A practical engineering approach*. Cambridge University Press: Cambridge, 1997.
49. Fehse W. *Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press: Cambridge, 2003.
50. Carter TE. State transition matrices for terminal rendezvous studies: brief survey and new example. *Journal of Guidance Control and Dynamics* January–February 1998; **21**(1):148–155, doi:10.2514/2.4211.
51. Yamanaka K, Ankersen F. New state transition matrix for relative motion on an arbitrary elliptical orbit. *Journal of Guidance Control and Dynamics* 2002; **25**(1):60–66, doi:10.2514/2.4875.
52. Gim D, Alfriend KT. State transition matrix of relative motion for the perturbed noncircular reference orbit. *Journal of Guidance, Control, and Dynamics* 2003; **26**(6):956–971, doi:10.2514/2.6924.
53. Michalska H, Mayne DQ. Robust receding horizon control of constrained nonlinear-systems. *IEEE Transactions on Automatic Control* Nov 1993; **38**(11):1623–1633, doi:10.1109/9.262032.
54. Richards A, How JP. Robust variable horizon model predictive control for vehicle maneuvering. *International Journal of Robust and Nonlinear Control* February 2006; **16**(7):333–351, doi:10.1002/rnc.1059.
55. Richards AG, How JP. Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. *Proceedings of the 2003 American Control Conference*, vol. 5, Denver, Colorado, 2003; 4034–4040, doi:10.1109/ACC.2003.1240467.
56. Rao CV, Rawlings JB. Linear programming and model predictive control. *Journal of Process Control* April 2000; **10**(2–3):283–289, doi:10.1016/S0959-1524(99)00034-7.
57. Ohlsson H, Gustafsson F, Ljung L, Boyd S. Trajectory generation using sum-of-norms regularization. *Proceedings of the 49th IEEE Conference on Decision and Control*, 2010; 540–545, doi:10.1109/CDC.2010.5717368.
58. Gallieri M, Maciejowski JM. Stabilising terminal cost and terminal controller for $\ell_{asso}$-MPC: enhanced optimality and region of attraction. *Proceedings of the European Control Conference*, Zurich, Switzerland, 2013; 524–529.
59. Rao CV, Wright SJ, Rawlings JB. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications* December 1998; **99**(3):723–757, doi:10.1023/A:1021711402723.
60. Wang Y, Boyd S. Fast model predictive control using online optimization. *IFAC World Congress*, Seoul, Korea, 2008; 6974–6997, doi:10.3182/20080706-5-KR-1001.3844.
61. Wang Y, Boyd S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology* 2010; **18**(2):267–278, doi:10.1109/TCST.2009.2017934.
62. Mehrotra S. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization* 1992; **2**(4):575–601, doi:10.1137/0802028.
63. Gertz EM, Wright S. *OOQP User Guide*. Mathematics and Computer Science Division, Argonne National Laboratory October 2001. URL http://pages.cs.wisc.edu/~swright/ooqp/, (Updated May 2004).
64. Kerrigan EC, Jerez JL, Longo S, Constantinides GA. Number representation in predictive control. *Proceedings of the IFAC conference on Nonlinear Model Predictive Control*, Noordwijkerhout, NL, 2012; 60–67, doi:10.3182/20120823-5-NL-3013.00017.
65. Jerez JL, Constantinides GA, Kerrigan EC. Towards a fixed point QP solver for predictive control. *Proceedings of the 51st IEEE Conference on Decision and Control*, Maui, HI, USA, 2012, doi:10.1109/CDC.2012.6427015.
66. Jerez J, Constantinides G, Kerrigan E. A low complexity scaling method for the Lanczos kernel in fixed-point arithmetic. *IEEE Transactions on Computers* 2013; (**Article in press**), doi:10.1109/TC.2013.162.
67. Richards A, Stewart W, Wilkinson A. Auto-coding implementation of model predictive control with application to flight control. *Proceedings of the European Control Conference*, Budapest, Hungary, 2009; 150–155.
68. Kogel M, Findeisen R. Fast predictive control of linear systems combining Nesterov's gradient method and the method of multipliers. *Proceedings of the 50th Conference on Decision and Control and European Control Conference*, 2011; 501–506, doi:10.1109/CDC.2011.6160688.
69. Binet G, Krenn R, Bemporad A. Model predictive control applications for planetary rovers. *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Turin, Italy, 2012.
70. Boland DP, Constantinides GA. An FPGA-based implementation of the MINRES algorithm. *Proceedings of the IEEE International Conference on Field-Programmable Logic and Applications*, Heidelberg, 2008; 379–384, doi:10.1109/FPL.2008.4629967.
71. Walker MJH, Ireland B, Owens J. A set modified equinoctial orbit elements. *Celestial Mechanics* 1985; **36**(4):409–419, doi:10.1007/BF01227493.
72. CPLEX. *IBM ILOG CPLEX V12.1 User's Manual for CPLEX* 2009. URL http://www-01.ibm.com/software/integration/optimization/cplex/.
73. Mattingley J, Boyd S. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering* 2012; **13**(1):1–27, doi:10.1007/s11081-011-9176-9.
74. Domahidi A, Zgraggen A, Zeilinger MN, Jones CN. Efficient interior point methods for multistage problems arising in receding horizon control. *Proceedings of the IEEE Conference on Decision and Control*, Maui, HI, USA, 2012; 668–674, doi:10.1109/CDC.2012.6426855.