



Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Constrained LQR for low-precision data representation[☆]Stefano Longo^{a,1}, Eric C. Kerrigan^{b,c}, George A. Constantinides^b^a Department of Automotive Engineering, Cranfield University, Cranfield, MK43 0AL, UK^b Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, UK^c Department of Aeronautics, Imperial College London, London, SW7 2AZ, UK

ARTICLE INFO

Article history:

Received 5 November 2012

Received in revised form

16 April 2013

Accepted 11 September 2013

Available online xxx

Keywords:

Embedded systems

Control of constrained systems

Predictive control

Optimization

Number representation

ABSTRACT

Performing computations with a low-bit number representation results in a faster implementation that uses less silicon, and hence allows an algorithm to be implemented in smaller and cheaper processors without loss of performance. We propose a novel formulation to efficiently exploit the low (or non-standard) precision number representation of some computer architectures when computing the solution to constrained LQR problems, such as those that arise in predictive control. The main idea is to include suitably-defined decision variables in the quadratic program, in addition to the states and the inputs, to allow for smaller roundoff errors in the solver. This enables one to trade off the number of bits used for data representation against speed and/or hardware resources, so that smaller numerical errors can be achieved for the same number of bits (same silicon area). Because of data dependencies, the algorithm complexity, in terms of computation time and hardware resources, does not necessarily increase despite the larger number of decision variables. Examples show that a 10-fold reduction in hardware resources is possible compared to using double precision floating point, without loss of closed-loop performance.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

It is common practice to optimize the performance of sophisticated algorithms, like online model-based optimization methods, only at a high level of abstraction, viewing the implementation as a different, decoupled problem (Kerrigan, Jerez, Longo, & Constantinides, 2012). This practice might no longer be reasonable due to the fact that the high degree of flexibility of new computer architectures remains largely underexploited (Constantinides, 2009). The standard double- or single-precision floating-point representation may be unnecessarily precise for an application where such precision would be better traded for more important aspects. For a control algorithm, these aspects could be computational speed, processor cost and reduced power consumption. To motivate this point further, we have plotted in Fig. 1 the resources needed on a Virtex-6 FPGA for an implementation of the predictive control

algorithm of Rao, Wright, and Rawlings (1998). Working in single precision (23 bits), rather than double (52 bits), already gives a substantial improvement, but reducing the number of mantissa bits even further allows one to reduce the total hardware resources by one order of magnitude and do the computations almost twice as fast. The ability to reliably run an optimization algorithm in a low precision platform is also motivated by the fact that most microprocessors in embedded systems do not offer any support for double precision floating-point or for floating-point at all (Kerrigan et al., 2012).

In view of the above, we propose a new formulation for the Quadratic Programming (QP) problem (resulting from the constrained LQR problem) that allows one to exploit low precision number representations efficiently (Longo, Kerrigan, & Constantinides, 2013). This formulation is mathematically equivalent to the non-condensed formulation where inputs and states appear as decision variables (Rao et al., 1998). The difference here is that we represent the continuous-time plant model (used for the prediction) in the so-called *delta* (or incremental) form (Goodwin, Middleton, & Poor, 1992; Goodwin, Yuz, Agüero, & Cea, 2010; Middleton & Goodwin, 1986, 1990), and include additional decision variables. The delta form has the virtue of separating the important information stored in the system's transition matrix from quantities of non-comparable size. By doing so, the truncation due to finite precision arithmetic becomes less detrimental than for the more commonly used *shift* form; hence, the algorithm will be numerically more stable. The modern need for the implementation

[☆] This research has been supported by the EPSRC (EP/G031576/1, EP/F041004/1) and FP7/2007-2013 (FP7-ICT-2009-4 248940). The material in this paper was partially presented at the IFAC Conference on Nonlinear Model Predictive Control 2012 (NMPC'12), August 23–27, 2012, Noordwijkerhout, The Netherlands and the 12th Biannual European Control Conference (ECC13), July 17–19, 2013, Zurich, Switzerland. This paper was recommended for publication in revised form by Associate Editor Lalo Magni under the direction of Editor Frank Allgöwer.

E-mail addresses: s.longo@cranfield.ac.uk (S. Longo), e.kerrigan@imperial.ac.uk (E.C. Kerrigan), g.constantinides@imperial.ac.uk (G.A. Constantinides).

¹ Tel.: +44 0 1234 758581; fax: +44 0 1234 758259.

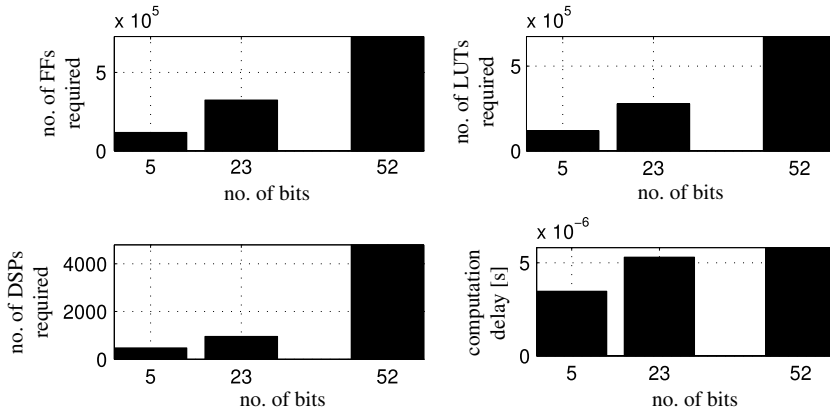


Fig. 1. FPGA resources required for double (52 bits), single (23 bits) and custom (5 bits) precision for data representation of a model predictive control algorithm (6 states, 2 inputs and horizon length of 200 steps). Reducing the number of bits (only the mantissa bits are considered here) used for data representation significantly reduces the hardware resources (Flip-Flops, Look-Up-Tables, Digital-Signal-Processing units) required and the algorithm computational delay (calculated at a clock frequency of 200 MHz and assuming 10 interior-point iterations and a fully parallel implementation).

of fast *constrained* LQ solvers in inexpensive hardware (Wills, Bates, Fleming, Ninness, & Moheimani, 2008) justifies this hardware–algorithm co-design approach and makes it necessary to revive the use of discretization methods that are numerically more stable.

It will be shown that, for some examples, it is possible to achieve good closed-loop performance with only 5 bits for the mantissa, which translates into fewer hardware resources needed, faster computation and less power consumption. More interestingly, it will be shown that, even with the same number of bits for data representation and despite the number of decision variables in the QP becoming larger, (i) the computational cost of solving this problem still scales linearly with the horizon length, (ii) the data dependencies of the algorithm are such that it can be implemented at essentially no extra hardware costs (hardware adders and multipliers are reused efficiently) and (iii) the algorithm is at least as fast as its shift equivalent. In fact, the algorithm in delta domain is normally faster because the better numerical stability leads to fewer interior-point iterations when an interior-point method is used, i.e. one has faster convergence to the solution.

2. Constrained LQR problem formulation in delta domain

Consider the continuous-time LTI plant model

$$\dot{x}(t) = A_c x(t) + B_c u(t), \tag{1}$$

where $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ and (A_c, B_c) is a stabilizable pair. The control input $u(\cdot)$ is a signal created by a sample-and-hold element for a sampling period h such that $u(t) = u(ih)$ for all $t \in [ih, ih+h)$ and $i \in \mathbb{N}_0$ is the sampling instant. The solution of the sampled-data model is given by

$$x(ih + h) = \underbrace{e^{A_c h}}_{\triangleq A_\delta} x(ih) + \underbrace{\int_0^h e^{A_c(h-\tau)} B_c d\tau}_{\triangleq B_\delta} u(ih), \tag{2}$$

where i denotes the sampling instant and the matrix exponential $e^{A_c h}$ is defined by the matrix series

$$e^{A_c h} \triangleq I + A_c h + \frac{(A_c h)^2}{2!} + \frac{(A_c h)^3}{3!} + \dots \tag{3}$$

If the product $A_c h$ in (3) results in a matrix with entries much smaller than one, the transition matrix A_δ in (2) will be a matrix where the elements on the diagonal are the summation of 1 with a much smaller number. These coefficients have to be stored in a

computer with finite word length. In finite arithmetic, the coefficients will be truncated, hence some of the information contained in $A_c h + \frac{(A_c h)^2}{2!} + \frac{(A_c h)^3}{3!} + \dots$ – which is where the plant dynamics are represented – will be lost. This numerical problem can be ameliorated by substituting (3) into (2) and rewriting it as

$$x(ih + h) = x(ih) + h \underbrace{\left[A_c + \frac{A_c^2 h}{2!} + \frac{A_c^3 h^2}{3!} + \dots \right]}_{\triangleq A_\delta} x(ih) + h \underbrace{\left[B_c + \frac{A_c h B_c}{2!} + \frac{A_c^2 h^2 B_c}{3!} + \dots \right]}_{\triangleq B_\delta} u(ih), \tag{4}$$

which is mathematically equivalent to (2). However, separating the identity matrix from the summation tail – defined as A_δ in (4) – has the effect of reducing the numerical errors in a finite precision floating-point representation. These matrices can be computed as $A_\delta = \Omega A_c$ and $B_\delta = \Omega B_c$, where

$$\Omega \triangleq \frac{1}{h} \int_0^h e^{A_c \tau} d\tau = \frac{1}{h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \exp \begin{bmatrix} h A_c & h I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{5}$$

By adopting the more convenient notation $x(ih) \triangleq x_i$ (and similarly for other vectors) and introducing a new vector $\delta_i \in \mathbb{R}^{n_x}$, we can rewrite (4) as

$$\delta_i = A_\delta x_i + B_\delta u_i, \tag{6a}$$

$$x_{i+1} = x_i + h \delta_i, \tag{6b}$$

which is in fact the *delta* (or incremental) form of (1) (Goodwin et al., 1992, 2010; Middleton & Goodwin, 1986, 1990), an alternative to the commonly used *shift* form of (2). Associated with the system in (6), consider the discrete-time finite-horizon LQ problem defined by the cost function

$$V(\theta) \triangleq x'_N Q_f x_N + \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}' \begin{bmatrix} Q & M \\ M' & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \tag{7}$$

where N is the number of samples for the prediction horizon, $\begin{bmatrix} Q & M \\ M' & R \end{bmatrix} \geq 0$, $Q = Q' \geq 0$, $R = R' > 0$, $Q_f = Q'_f > 0$ and θ is defined below. These matrices are given to define a controller for an ideal closed-loop performance of the sample-data system. Alternatively, they can be computed via discretization of a continuous-time LQ problem (Åström & Wittenmark, 1997, pp. 411–412), (Levis, Schlueter, & Athans, 1971). For the constrained

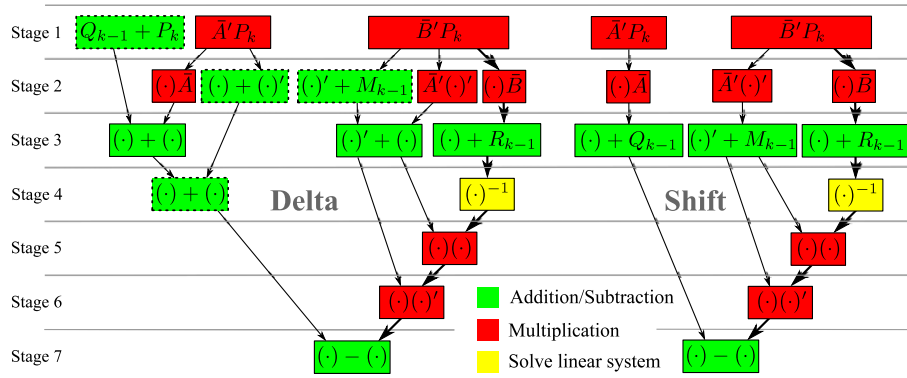


Fig. 2. Data dependencies of the delta- and shift-domain Riccati recursions. For delta, $\bar{A} \triangleq hA_\delta$ and $\bar{B} \triangleq hB_\delta$. For shift, $\bar{A} \triangleq A_s$ and $\bar{B} \triangleq B_s$. Both algorithms have the same latency because they have the same critical path (thicker arrows). At each stage, no more than three matrix additions and three matrix multiplications are performed simultaneously. Hence, although four extra additions are needed for the delta case (dotted boxes), the number of computational resources needed for minimal latency is the same in both cases.

$$\Delta\lambda_k = p_{k-1} - P_{k-1}\Delta x_{k-1}, \tag{14d}$$

$$\Delta x_k = \Delta x_{k-1} + h\Delta\delta_{k-1} - r_k^\lambda, \tag{14e}$$

where

$$P_{k-1} \triangleq Q_{k-1} + P_k + h^2 A'_\delta P_k A_\delta + h A'_\delta P_k + h P_k A_\delta - (M_{k-1} + h^2 A'_\delta P_k B_\delta + h P_k B_\delta)(R_{k-1} + h^2 B'_\delta P_k B_\delta)^{-1}(M'_{k-1} + h^2 B'_\delta P_k A_\delta + h B'_\delta P_k), \tag{15a}$$

$$p_{k-1} \triangleq r_{k-1}^x + P_k r_k^\lambda + p_k + h^2 A'_\delta P_k r_{k-1}^\gamma + A'_\delta r_{k-1}^\delta + h A'_\delta P_k r_k^\lambda + h A'_\delta p_k + h P_k r_{k-1}^\gamma - (M_{k-1} + h^2 A'_\delta P_k B_\delta + h P_k B_\delta)(R_{k-1} + h^2 B'_\delta P_k B_\delta)^{-1} \times (r_{k-1}^u + h^2 B'_\delta r_k^\gamma + B'_\delta r_{k-1}^\delta + h B'_\delta P_k r_k^\lambda + h B'_\delta p_k). \tag{15b}$$

Hence, the search direction is calculated as follows: (i) set $p_N \leftarrow Q_f$ and $p_N \leftarrow r_N^x$; (ii) compute P_k and p_k for $k = N, N - 1, \dots, 2$ using the two backward recursions in (15); (iii) compute $\Delta u_0, \Delta\gamma_0, \Delta\delta_0, \Delta\lambda_1$ and Δx_1 using (13); (iv) compute, recursively, $\Delta u_k, \Delta\gamma_k, \Delta\delta_k$ for $k = 2, 3, \dots, N - 1$ and $\Delta\lambda_k$ and Δx_k for $k = 1, 2, \dots, N$ using (14).

While the block reduction manipulations of Rao et al. (1998) yielded the famous Discrete Riccati Difference Equation (DRDE) for time-varying weighting matrices (Rao et al., 1998, Eq. 5.1a), we obtained in (15a), analogously, a DRDE for time-varying weighting matrices in the delta domain. The reader familiar with the work of Rao et al. (1998) will also appreciate more subtle similarities between the recursive equations of both formulations, realizing that the two methods are entirely equivalent in an infinite precision arithmetic system. It is evident, however, that the solution in delta domain requires more operations (matrix additions and multiplications) to be performed at each iteration when compared to the equivalent solution in shift domain.

4. Algorithm properties

Although computing the interior-point search direction requires more operations for the delta formulation, neither the computational time nor the hardware resources utilized significantly increase if care is taken in the implementation. The reason is twofold and lies in the dependencies of the operations.

Proposition 1. *Let the computation of the search direction (12) be implemented in a parallel hardware architecture. Then, when compared to the shift formulation of Rao et al. (1998):*

- (i) *the critical path (the longest non-parallelizable sequence of operations) remains unchanged, hence the time required to calculate the search direction (latency) is the same for both formulations;*
- (ii) *both formulations require the same number of computational hardware resources (adders, multipliers, etc.) for implementation.*

Proof. The most computationally expensive operation of the algorithm is the calculation of matrices P_1, P_2, \dots, P_{N-1} and vectors p_1, p_2, \dots, p_{N-1} using the backward recursions in (15). These have to be obtained before (13) and the forward recursions in (14) can be calculated. For the sake of simplicity, we will only consider the DRDE of (15a) and point out that the same reasoning and procedures can be applied to the whole algorithm by noticing that many intermediate results can be stored in memory and reused.

(i) Let us compare the DRDE in (15a) with the equivalent shift-domain DRDE (Rao et al., 1998)

$$P_{k-1} = Q_{k-1} + A'_s P_k A_s + (M_{k-1} + A'_s P_k B_s) \times (R_{k-1} + B_s P_k A_s)^{-1}(M'_{k-1} + B'_s P_k A_s) \tag{16}$$

and define matrices \bar{A} and \bar{B} to be $\bar{A} \triangleq hA_\delta$ and $\bar{B} \triangleq hB_\delta$ for the delta case and $\bar{A} \triangleq A_s$ and $\bar{B} \triangleq B_s$ for the shift case. The data dependencies of the calculations to compute P_{k-1} for both the delta and the shift formulation are shown as a graph in Fig. 2. At a matrix level, Fig. 2 is an implementation with minimal latency, signifying that all the operations are performed as soon as the data are available (exploiting the parallelism). The seven stages required to compute the solution are determined by the sequence of operations in the *critical path*, which is the path on the right of both graphs indicated by thicker arrows. The critical paths for both the delta and the shift case are the same, hence the time required by both algorithms to complete is the same (although matrices \bar{A} and \bar{B} are different for the two cases, what matters here is their dimensions, which do not change). The search direction is computed recursively in N steps in both formulations.

(ii) For the delta case, there are four extra matrix additions to be performed, shown in Fig. 2 by the boxes with dotted edges. At each stage of the algorithm, a maximum of three matrix adders and three matrix multipliers are used simultaneously. This is true for both the delta and the shift cases, implying that the same number of hardware blocks is needed for both implementations (of course, in the delta case, the adders will be used more often). If both the recursions in (15) are considered, it can be easily shown that five adders and five multipliers are needed. This is the same as for the analogous shift-domain recursions (Rao et al., 1998, Eq. 5.1). Furthermore, the same adder and multiplier blocks (as well as many

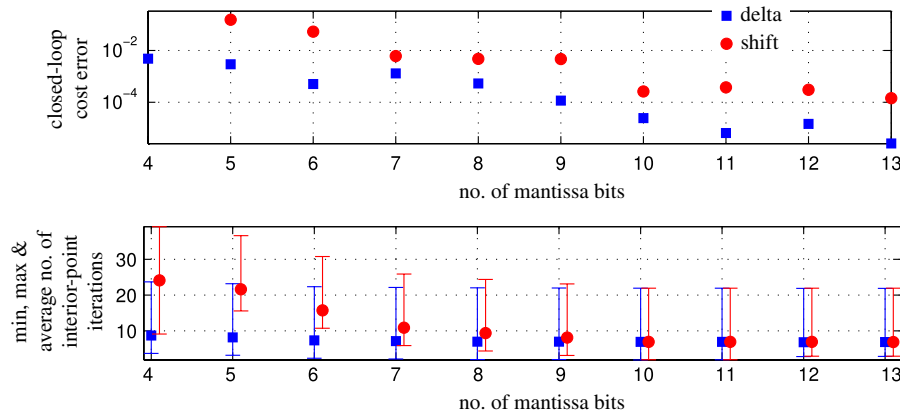


Fig. 3. For the same number of bits, the closed-loop cost error for the delta implementation is lower (top plot) as well as the number of interior-point iterations (bottom plot).

intermediate results) can be re-utilized efficiently for the calculation of (14). ■

Proposition 1 shows that the extra operations required for the delta case do not increase the algorithm latency nor the computational hardware resources utilized. Hence, the numerical advantages given by the delta implementation are obtained at no extra cost from the point of view of resource utilization and computational time. It should also be noted that for multi-input plant models ($n_u > 1$) the bottleneck of the algorithm is the n_u -by- n_u matrix factorization needed in Stage 4 of Fig. 2, which is performed on matrices of the same size in both formulations. If such a factorization is obtained via a Cholesky decomposition, a number of divisions and square roots must be computed (much more expensive to perform than simple additions and multiplications). This motivates even further the use of a reduced number of bits for data representation in order to speed up the execution of divisions and square roots, and therefore the whole algorithm. As a consequence, an implementation of the algorithm in a sequential (rather than parallel) architecture can also have an equal latency and resource utilization if a low enough number of bits is used.

5. Illustrative example

In this section we show, with simulation examples, that the solution from the delta formulation is indeed more numerically accurate than the equivalent shift one; therefore, the same closed-loop performance can be achieved with our formulation using fewer bits in the number representation. Using fewer bits implies increased computational speed and reduced hardware usage. An indication of the improvement in hardware resources utilized will be given for an FPGA from the Xilinx Virtex-6 family (Xilinx, 2013). Furthermore, simulations have shown that the increased numerical accuracy also improves the interior-point algorithm convergence speed, thus reducing the number of interior-point iterations required to completion. We use as a benchmark a spring-mass system with six unitary masses linearly connected by five springs with unitary coefficients with input forces applied to the first and last masses. The input constraints are $\|u_k\|_\infty \leq 0.5$ and the state constraints are $\|x_k\|_\infty \leq 3.5$. The associated performance measure is given by the LQ cost in (7) with matrices Q , R , M and Q_f coming from the discretization, with sampling period $h = 0.01s$, of a continuous-time LQ cost $\int_0^{hN} x(t)'x(t) + u(t)'u(t)dt$. The prediction horizon was selected as $N = 200$.

First, we compare the variation of closed-loop performance when the QP solver is implemented (in floating-point) using the proposed formulation (Section 3) and the formulation of Rao et al.

(1998) (shift). As a performance metric we use

$$\Gamma \triangleq \sum_{k=0}^{N_{\text{sim}}-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}' \begin{bmatrix} Q & M \\ M' & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad (17)$$

where the system is left to evolve from initial conditions $x_0 = [0 \ 0 \ 3.5 \ 3.5 \ 0 \ 0]'$, $u_0 = [0 \ 0]'$ and with $N_{\text{sim}} = 20/h$ (selected to be long enough to allow the system to reach steady-state). A cost, Γ_{52} , has been evaluated as in (17) for a shift implementation in double precision and for an implementation with reduced precision for data representation, which we call Γ_{low} . An error, called *closed-loop cost error*, has been defined as $\|\Gamma_{\text{low}} - \Gamma_{52}\|/\|\Gamma_{52}\|$. The top plot of Fig. 3 shows how the closed-loop cost error varies for a range of mantissa bits for the two Riccati recursion-based implementations. This error, for the delta implementation, is always lower than for the shift implementation, especially for a small number of bits. As the number of bits grows, the difference between the two formulations becomes negligible (hence results are shown only up to 13 bits). The bottom plot of Fig. 3 shows the minimum, maximum and average number of interior-point iterations required for the duality gap to be less than 10^{-5} (set as a tolerance to stop the interior-point iterations). The delta formulation requires on average fewer iterations to reach the predefined threshold, implying that faster convergence can be achieved.

Fig. 4 shows some closed-loop trajectories where only a 5-bit mantissa was used for the number representation for the numerical solution of the QP. Even for such a low precision, the trajectories of the delta implementation almost completely overlap with the shift double-precision (52 bits) one. However, a traditional (shift) 5-bit implementation response becomes practically unacceptable. This is caused by numerical errors, which are primarily due to the errors in the evaluation of the Riccati equation, as has also been observed in Goodwin et al. (1992). Consequently, the number of interior-point iterations is also much lower for the delta formulation, as shown in Fig. 5.

Finally, we comment on the memory resources required by each implementation. We consider an FPGA from the Virtex-6 family and assume that the data words are stored in look-up tables (LUTs). We also assume that 8 bits are used for the exponent part of the floating-point numbers. Fig. 6 shows the number of LUTs required by both algorithms for different numbers of mantissa bits. The numbers of LUTs for a shift implementation in single and double precision are shown in Fig. 1. As expected, the delta implementation always requires slightly more LUTs, and hence slightly more resources. However, it is sufficient to reduce the number of mantissa bits by at most two to allow the delta implementation to utilize fewer resources than the shift one. Reducing the mantissa bits

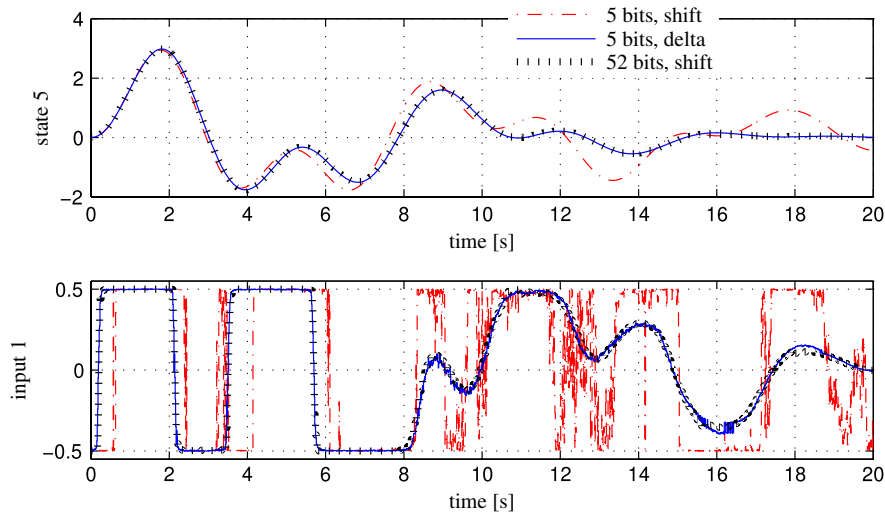


Fig. 4. While the performance of a 5-bit shift implementation is very poor, a 5-bit delta implementation produces trajectories that almost perfectly overlap the ones from a 52-bit shift implementation.

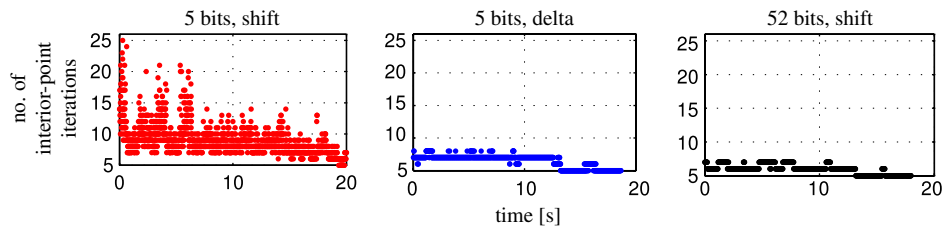


Fig. 5. Because the numerical errors in the delta formulation are lower, the number of iterations required is also lower when compared to the shift implementation.

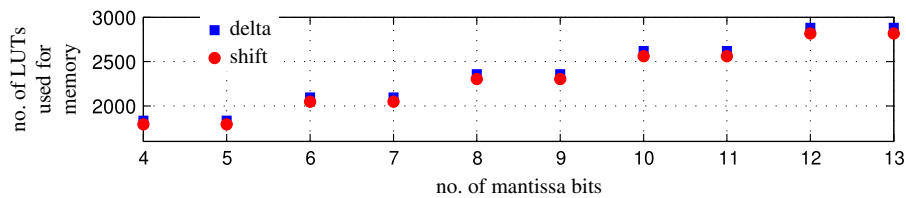


Fig. 6. The delta implementation always requires slightly more memory. However, it is sufficient to reduce the number of mantissa bits by at most two to bring the number of LUTs required by the delta implementation below the number of LUTs required by the shift implementation.

affects the closed-loop performance, but the top plot of Fig. 4 shows that a reduction of two bits still leaves the delta implementation with a better performance than the shift one.

6. Conclusions

We have shown that the superior numerical properties of a delta-domain formulation of a constrained LQR problem can be enjoyed without sacrificing solver complexity in terms of its execution time or computational hardware resources required. For the Riccati recursion approach to solving the linear equations in (12), the number of decision variables in the resulting QP problem is indeed larger when compared to its equivalent shift formulation. However, because of the particular data dependencies in the algorithm, the latency and arithmetic units required for the operations will not increase when care is taken in the implementation. The proposed design procedure gives the designer the flexibility to reduce the number of bits used for data representation in order to increase the computational speed and/or reduce the circuit size. This paper is a contribution toward a larger research goal to develop

model predictive control algorithms with a guarantee that the loss of accuracy due to finite precision effects is minimized or bounded a priori. Future research could investigate the appropriateness of other sampled-data models and/or number representations and an extension of these ideas to the nonlinear case.

References

- Åström, K., & Wittenmark, B. (1997). *Computer-controlled systems* (3rd ed.). Prentice-Hall.
- Constantinides, G.A. (2009). Tutorial paper: parallel architectures for model predictive control. In *Proc. European control conference 2009* (pp. 138–143), Budapest, HU.
- Goodwin, G. C., Middleton, R. H., & Poor, H. V. (1992). High-speed digital signal processing and control. *Proceedings of the IEEE*, 80(2), 240–259.
- Goodwin, G.C., Yuz, J.L., Agüero, J.C., & Cea, M. (2010). Sampling and sampled-data models. In *Proc. American control conference* (pp. 1–20), Baltimore, USA.
- Kerrigan, E.C., Jerez, J.L., Longo, S., & Constantinides, G.A. (2012). Number representation in predictive control. In *Proc. IFAC conf. on nonlinear model predictive control, vol. 4* (pp. 60–67), Noordwijkerhout, the Netherlands.
- Levis, A. H., Schlueter, R. S., & Athans, M. (1971). On the behaviour of optimal linear sampled-data regulators. *International Journal of Control*, 13(2), 343–361.

- Longo, S., Kerrigan, E.C., & Constantinides, G.A. (2013). A predictive control solver for low-precision data representation. In *Proc. European control conference*, Zürich, Switzerland.
- Middleton, R. H., & Goodwin, G. C. (1986). Improved finite word length characteristics in digital control using delta operators. *IEEE Transactions on Automatic Control*, *AC-31*(11), 1015–1021.
- Middleton, R. H., & Goodwin, G. C. (1990). *Digital control and estimation: a unified approach*. Prentice-Hall.
- Rao, C. V., Wright, S. J., & Rawlings, J. B. (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, *99*(3), 723–757.
- Wills, A. G., Bates, D., Fleming, A. J., Ninness, B., & Moheimani, S. O. R. (2008). Model predictive control applied to constraint handling in active noise and vibration control. *IEEE Transactions on Control Systems Technology*, *16*(1), 3–12.
- Wright, S. J. (1993). Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, *77*(1), 161–187.
- Xilinx (2013). www.xilinx.com. April.

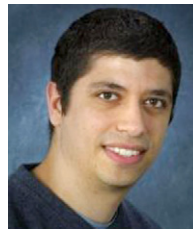


Stefano Longo (B.Sc., Ph.D., MIEEE, MIET) received his M.Sc. in Control Systems from the Department of Automatic Control and System Engineering at the University of Sheffield in 2007 and completed his Ph.D. in Control Systems in the Department of Mechanical Engineering at the University of Bristol in 2011. His Ph.D. thesis was awarded the prestigious *Institution of Engineering and Technology (IET) Control and Automation Prize* for significant achievements in the area of control engineering. In November 2010 he was appointed to the position of Research Associate in the Department of Electrical and Electronic

Engineering at Imperial College London, where he worked in the intersection of the Control & Power and Circuit & Systems Research Groups. He was appointed to the position of lecturer in Vehicles' Electrical and Electronic Systems at Cranfield University in the summer of 2012. His work and his research interests gravitate around the problem of implementing advanced control and estimation algorithms in hardware, where the algorithm design and the hardware implementation are not seen as two separate and decoupled problems, but as a single interconnected one. These ideas have been applied to networked control systems and to parallelizable hardware for constrained optimal control and real-time optimization.



Eric C. Kerrigan joined Imperial College London in 2006, where he has a joint faculty appointment in the Department of Electrical and Electronic Engineering and the Department of Aeronautics. He was a Research Fellow at the Department of Engineering, University of Cambridge, from 2001 to 2005, where he also obtained his Ph.D. in 2001. His current research is in the optimal and robust control of systems with nonlinearities and constraints, as well as the control of distributed parameter systems. The emphasis of his research is on the development of efficient numerical methods and computational architectures for solving the resulting control, estimation, modeling and optimization problems. This research is motivated by and applied to a variety of control problems in aerospace and fluid dynamics, such as skin friction drag reduction, attenuation of buffeting in wind turbine blades and control of pico-satellites. His research is funded by the Engineering and Physical Sciences Research Council, the European Commission and industrial sponsors Xilinx Inc., The Mathworks and the European Space Agency. He is a Member of the IEEE, IET and SIAM and is on the editorial boards of *IEEE Transactions on Control Systems Technology*, *IEEE Control Systems Society Conferences*, *Control Engineering Practice* and *Optimal Control Applications and Methods*.



George A. Constantinides received his M.Eng. degree (with honors) in Information Systems Engineering and his Ph.D. degree from Imperial College London, London, UK, in 1998 and 2001, respectively. Since 2002, he has been with the faculty at Imperial College London, where he is currently a Professor in Digital Systems and Head of the Circuits and Systems research group. Dr. Constantinides is a Fellow of the BCS and a Senior Member of the IEEE. He is an Associate Editor of *IEEE Transactions on Computers* and the *Journal of VLSI Signal Processing*. He was a Program Cochair of the IEEE International Conference on Field Programmable Technology in 2006 and Field Programmable Logic and Applications in 2003, and is a member of the steering committee of the International Symposium on Applied Reconfigurable Computing. He serves on the technical program committees of several conferences, including DAC, FPGA, FPT and FPL.