Proceedings of the 9th International Symposium on
Dynamics and Control of Process Systems (DYCOPS 2010),
Leuven, Belgium, July 5-7, 2010
Mayuresh Kothare, Moses Tade, Alain Vande Wouwer, Ilse Smets (Eds.)

Lightweight Model Predictive Control intended for embedded applications [⋆]

J. Currie [*] **D.I. Wilson** [**]

[*] *Industrial Information & Control Centre, Auckland University of Technology, Auckland, New Zealand (e-mail:* `jonathan.currie@aut.ac.nz`*).*
[**] *Electrical & Electronic Engineering, Auckland University of Technology, Auckland, New Zealand (e-mail:* `david.i.wilson@aut.ac.nz`*)*

Abstract: The computational demands of Model predictive control (MPC) are well known, and due to its internal constrained optimiser, historically has been ill-suited for embedded controllers designed to tackle high-speed applications. This paper explores the options of developing a low-cost lightweight MPC controller destined for micro-controller or FPGA architectures for modest applications demanding reasonable controller horizons. An object based MPC development tool is introduced and applied to an experimental 4-tank level system to explore the performance of the algorithm.

Keywords: model predictive control, micro-controller, embedded, quadratic programming, 4-tank

1. INTRODUCTION

Model Predictive Control, or MPC, in recent years has clearly garnered the interest of both academia and industry clearly becoming the advanced process control technique of choice as noted in audits Qin and Badgwell (2003), and industrial overviews Wilson and Young (2006). The reasons for its popularity are well known, namely the elegant constraint handling, Garcia et al. (1989), intuitive tuning, and the ability to optimally handle non-square systems.

However current commercial MPC control offerings are hampered by the inherent computationally demanding nature of an optimal controller. Consequently to date MPC applications have been mainly in the chemical processing industries in part due to three reasons. First the industrial plant time constants are relatively slow, in the orders of minutes to hours, second operational constraints on a chemical plant are critical and therefore should be an integral part of the control design, and finally the complexity of the MPC design means that it is more economical to implement it once in a multivariable fashion for a plant-wide or unit-wide application. These reasons may also explain the paucity of MPC applications outside the process control community as also pointed out recently in Dua et al. (2008).

1.1 Suitable applications for embedded MPC

Notwithstanding the computational complexity of MPC, we believe that there is no reason why an MPC could not work adequately on embedded hardware such as micro-controllers or field programmable gate arrays (FPGAs) at high speed. If this is achievable, as recent activity reported in Wang and Boyd (2008) and Lau et al. (2009) seems to suggest, then MPC could be applied to much faster systems such as unmanned vehicles, auto-pilots, intelligent sensors where the same control benefits could open up new opportunities in robotics and intelligent systems.

Our aim is explore the possibility of executing MPC on simple low cost hardware such as micro-controllers or FPGAs at kilo-hertz sampling frequencies. Our immediate interest is in applications involving the control of autonomous flying and underwater vehicles which tend to be characterised by small compact, mildly nonlinear models of between 3 and 10 states typically subject to simple bounded constraints. We term this ‘lightweight’ MPC compared to the large complex commercial MPC applications commonly employed in the process industries.

Currently however, a generic and flexible implementation of MPC requires substantial processing even to achieve sampling rates of around 1kHz, far more for example than classical optimal controllers such as LQG. For smaller, power or weight critical applications (such as those applications mentioned above), we need an alternative, high speed and small footprint processing platform, such as an FPGA. Developing an MPC in embedded hardware is not unique, (see for example Bleris et al. (2006), Dua et al. (2008), Johansen et al. (2007) for FPGA applications and Valencia-Palomo and Rossiter (2009) for PLC), but to date, substantial algorithm modifications such as shortening the prediction horizons, or replacing the online optimisation with a table-lookup are required for it to be practical.

[⋆] Sponsor and financial support from the Industrial Information and Control Centre, AUT University and The University of Auckland, New Zealand.

The kernel of an MPC is the constrained optimiser which for linear dynamic systems operating inside linear constraints reduces to a quadratic program (QP). While parametric MPC avoids the costly online numeric optimisation replacing it with a table-lookup, Pistikopoulos et al. (2000, 2007), and has been employed in a variety of embedded applications, Dua et al. (2008), the approach does not scale well, so consequently control designers typically employed very simple models with very short time horizons, sometimes just 1 or 2 samples. For some of the applications we had in mind for the embedded MPC, prior testing showed that the horizons needed to be substantially greater than this (as described in section 4), necessitating using the full classical QP, but to streamline and optimise the algorithm to deliver a fast, reliable solution.

A recent example highlighting the difficulties of implementing MPC on FPGA hardware is described in Johansen et al. (2007) where the largest report application is a model helicopter application with 6 states, 2 inputs, 3 outputs and only input constraints. With a prediction horizon of 1, they achieve a loop time of 6.45μs with 202kB of FPGA memory. (To put this in perspective, our intended target FPGA has only 45kB on chip.) Increasing the prediction horizon to 2 (which equates to a QP with 4 decision variables), can be solved in 10μs, which while extremely fast, requires an unrealistic 62MB to store the parametric representation of the system.

The following sections detail our investigation into the implementation of MPC on low-cost hardware such as micro-controllers, DSPs, and FPGAs using a desktop PC-based giga-hertz processor as a benchmark.

2. THE QP FORM OF THE MPC

Our implementation of the MPC follows the now standard approach that transforms a linear discrete state-space model with possible linear constraints into a quadratic optimisation program. Essentially the MPC algorithm delivers a future sequence of control moves, $\Delta\mathbf{u}$, over the immediate future control horizon, N_c, such that the cost function

$$J = \sum_{j=1}^{N_p} ||\hat{\mathbf{y}}_{k+j|k} - \mathbf{y}^\star_{k+j|k}||^2 + \sum_{j=1}^{N_c} ||\lambda \Delta\mathbf{u}_{k+j|k}||^2 \tag{1}$$

is minimised. The vectors $\hat{\mathbf{y}}, \mathbf{y}^\star$ are the estimated future output of the plant, and future setpoints, and λ is a weighting factor. The prediction, N_p, and control, N_c, horizons are important tuning parameters. The objective function is constrained by linear plant dynamics and possible linear output, input, and input rate constraints. With some algebraic manipulation (see Maciejowski (2002), Rossiter (2003) for the details), the objective function and constraints can be re-written as

$$\begin{aligned} \min_{\Delta\mathbf{u}} J &= \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{f}^T\Delta\mathbf{u} \\ \text{subject to: } &\mathbf{A}\Delta\mathbf{u} \le \mathbf{b} \end{aligned} \tag{2}$$

which is a standard quadratic program to be solved each sample time.

For a MIMO system with n states, m inputs and p outputs, the number of decision variables in Eqn. 2 is mN_c. From this it follows that the dimensions of $\mathbf{H}$ are $(mN_c \times mN_c)$, and the dimensions of the constraint matrix $\mathbf{A}$ are $((2pN_p + 4mN_c) \times mN_c)$. Thus for a typical problem in process control, say $n = 10, m = p = 2$, with horizons $N_p = 50, N_c = 25$, the dimensions of $\mathbf{A}$ is (50×400) which admittedly on the large size, are consistent with the recommendations given in (Seborg et al., 2005, p555) for process control applications.

2.1 QP algorithms

A comparison of five QP solvers covering both commercial and public domain; interior point and active set, is given in Currie and Wilson (2009), but suffice to say from our extensive testing, the infeasible interior point algorithm given in Wright (1997), Ling et al. (2008) and, with some algorithmic modifications, is our solver of choice. This supports the findings in a similar study using FPGA hardware, Lau et al. (2009), for the medium scale problems (i.e. number of decision variables exceeding about 9) and the comparisons reported in Bartlett et al. (2000).

2.2 Performance of the MPC algorithm

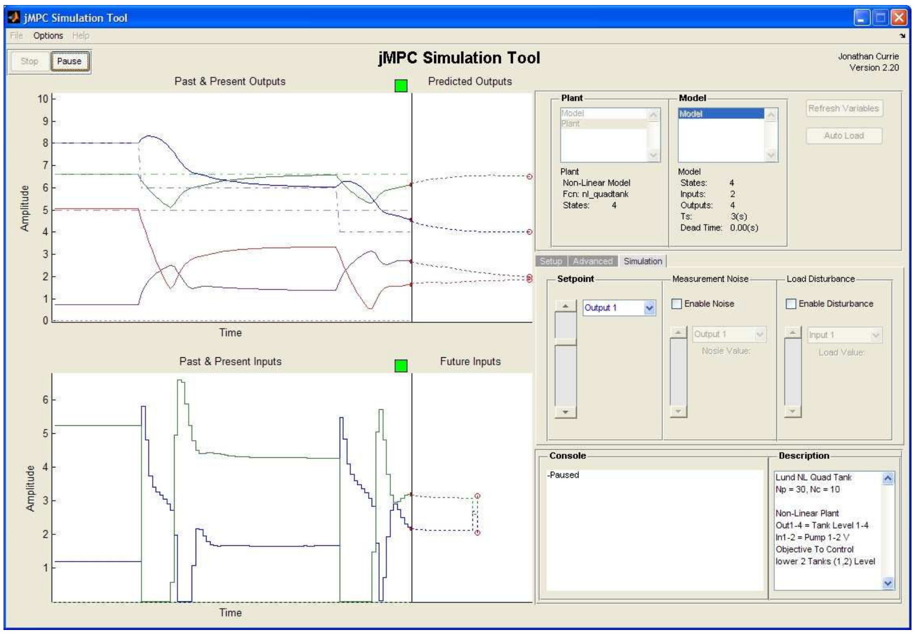
There is a subtle difference between the performance of the algorithm solving a constrained QP, and the overall MPC. One of the interesting features of the MPC controller is that there is a huge difference in computation load between solving a constrained optimisation QP with no prior information, and solving an unconstrained optimisation problem which is simply a matter of solving the unconstrained QP where the Hessian $\mathbf{H}$ can be factored offline. In practice, one pre-computes the Cholesky factor of $\mathbf{H} = \mathbf{R}^T\mathbf{R}$, and then online one must simply execute one forward substitution followed by a backward substitution, or in Matlab notation, `du = R\(R'\f)`.

For cases when the optimisation problem is constrained, then some savings can be made by using the previous optimal solution, appropriately shifted, as an initial start guess. As it turns out, this warm start procedure is not as beneficial as it seems cutting down less than a quarter of the iterations. Fig. 1 shows the near-perfect MPC control and timing breakdown for a second order non-minimum phase example, $G(s) = (-5s+1)/((3s+1)(s+1))$. In this example, the limiting constraint is the rate limiter on Δu, and the controller is allowed to take advantage of future setpoint changes.

In this example, one can see that the seemingly unimportant assembly of the vectors $\mathbf{f}$ and $\mathbf{b}$ in Eqn. 2 take around 0.6 ms which is comparable to the 1ms the QP takes to solve the constrained optimisation problem using around 10 iterations. We also note that 10 iterations lie at the upper end of the range recommended in Wang and Boyd (2008) after investigating 12 MPC problems with different dimensions.

3. AN MPC GRAPHICAL USER INTERFACE

To aid the design of model-predictive controllers, and to assist new users in MPC, we have developed a graphical user interface (GUI) for the design and testing of MPC controllers. Fig. 2 shows the interface demonstrating the MIMO control of a 4-tank liquid-level control problem.



Plant
Non-Linear Model
Fcn: nl_quadtank
States: 4

Model
States: 4
Inputs: 2
Outputs: 4
Ts: 3(s)
Dead Time: 0.00(s)

-Paused

Lund NL Quad Tank
Np = 30, Nc = 10

Non-Linear Plant
Out1-4 = Tank Level 1-4
In1-2 = Pump 1-2 V
Objective To Control
lower 2 Tanks (1,2) Level

Fig. 2. The MPC graphical user interface showing a simulation of a 4-tank system. Note the depiction of the future input and output predictions.

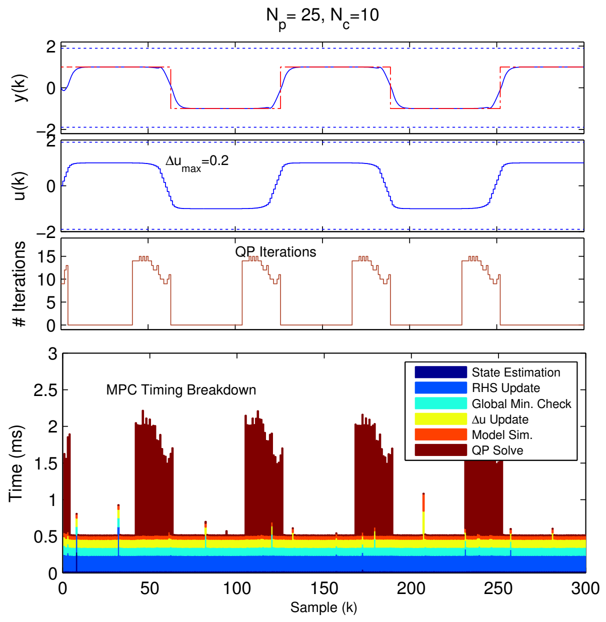


Fig. 1. Timing breakdown of an MPC implementation for a SISO non-minimum phase (inverse response) application.

The GUI design shows the input/output trends on the left, complete with red/green lights showing if constraints are active. The righthand panels allows the user to interactively adjust the various horizons, controller weights, and constraints.

The tool accepts a wide variety of models: linear discrete or continuous (or a mixture), transfer function or state-space, with, or without deadtime. One can even specify a nonlinear plant, although the model internally used within the MPC will be linearised. The tool also allows one to specify a different dynamic system for the plant than for the model thus enabling the exploration of model/plant mismatches.

This is no limit to the size of the models that can be used, but the practicalities of the GUI plotting are such that systems with more than half a dozen i/o rapidly get confusing. However in these cases, one can readily export the data for plotting externally.

The screen capture in Fig. 2 shows not only the output and setpoints, but also the constraints (dotted lines), and the future predictions for both the inputs and outputs to the right of the solid line two thirds along the scrolling trend.

4. A 4-TANK MPC APPLICATION

A deceptively simple, but still nonetheless challenging problem is a four-liquid level tank control problem. The system comprises of two sets of two coupled tanks manufactured by Quanser (`www.quanser.com`) intended as a control benchmark as shown in Fig. 3 and Fig. 4.

When comparing various control schemes (but not MPC) for a similar configuration, Rajanikanth Vadigepalli and Edward P. Gatzke and Francis J. Doyle III (2001) pointed out that such a system can exhibit non-minimum phase behaviour. Furthermore Åkesson (2006) has described an MPC simulation. The underlying dynamics are mildly nonlinear principally due to the square root relation between flow out and level, but also due to the nonlinear pump characteristics. The latter characteristics can of course be easily handled by a static nonlinearity in a Hammerstein configuration, and therefore does not overly effect the behaviour of the MPC. The basic dynamic model structure is given in Åkesson (2006), although we added deadtime established from in situ experiments, and from examining how the A/D and D/A hardware actually performed.

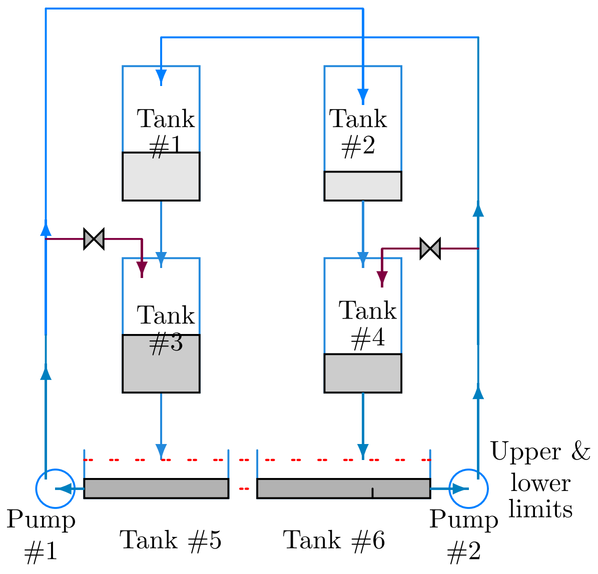


Fig. 3. An experimental 4-tank system. See also Fig. 4

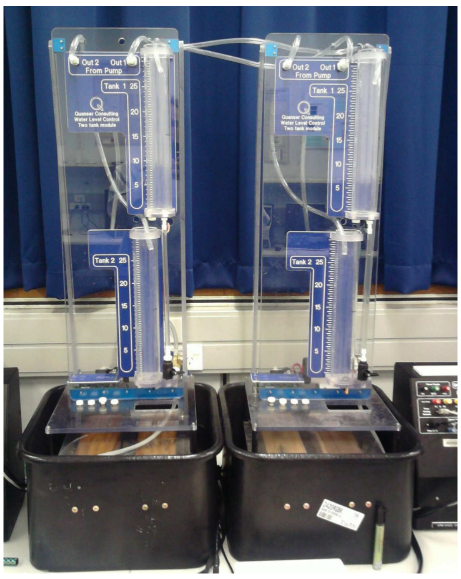


Fig. 4. Two Quanser 2-coupled tanks connected together to form an interacting 4 tank plant.

Given that the system is non-square, (it has only two pump inputs to control the 4 state levels), we can only keep 2 levels at the setpoints. In this case we have chosen the two lower tanks as controlled process variables.

The principle constraints for this application are the upper and lower levels in all four tanks (state constraints), the minimum and maximum pump flows (technically voltages to the pump motors), and the rate of change of the voltage to the pump motors.

4.1 A gigahertz processor implementation using Simulink

The controlled results following in this section were obtained using our MPC algorithm coded as a C Simulink S-function using the LAPACK and BLAS linear algebra libraries. The data acquisition was performed using a National Instrument DAQ card, and the real-time environment was provided using the Simulink execution tool.

Having the MPC algorithm in C means that it is possible to port to other hardware platforms as opposed to a Simulink model, or a Matlab m-file while maintaining control of the algorithm that auto-coding tools typically do not provide.

Fig. 5 shows the controlled response of the lower two tanks to changes in setpoint with control and prediction horizons $N_c = 10$ and $N_p = 40$ and a sampling rate of $T_s = 1$ second. The upper two tank levels are left free, but have upper and lower constraints. For this system, prediction horizons greater than 27s were necessary to stabilise the plant.

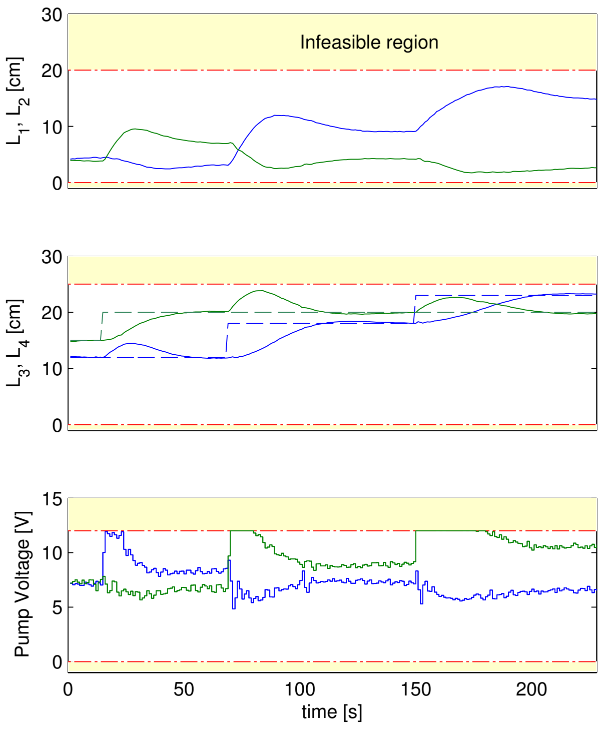


Fig. 5. An MPC response of the 4 tank system showing setpoint response of the lower two tanks.

Fig. 6 shows the challenging situation where the upper tank level constraints are reduced such that they become active. Once the state constraints are active, the QP struggles to find a feasible solution, particularly in the 30 iterations allowed in this implementation. This is a consequence of the infeasible interior point QP algorithm which, if terminated prematurely, may deliver an infeasible solution. Because of this concern, commercial MPC software such as Pavilion implement output constraints as penalties in the objective function as opposed to hard constraints. Notwithstanding in our case using the IIP strategy, that the returned solution is actually only just infeasible and the subsequent control is reasonable under the circumstances.

Fig. 7 shows the effect of varying the control and prediction horizons, in this case with a relatively coarse sample time of $T_s = 3$. Extensive tests indicated that the prediction horizon needed to be larger than 27s ($N_p > 9$ at $T_s = 3$). Note that for prediction horizons of $N_p = 6$ (the leftmost column in Fig. 7), the levels coincidentally (and obtusely) settled at the opposite setpoints.

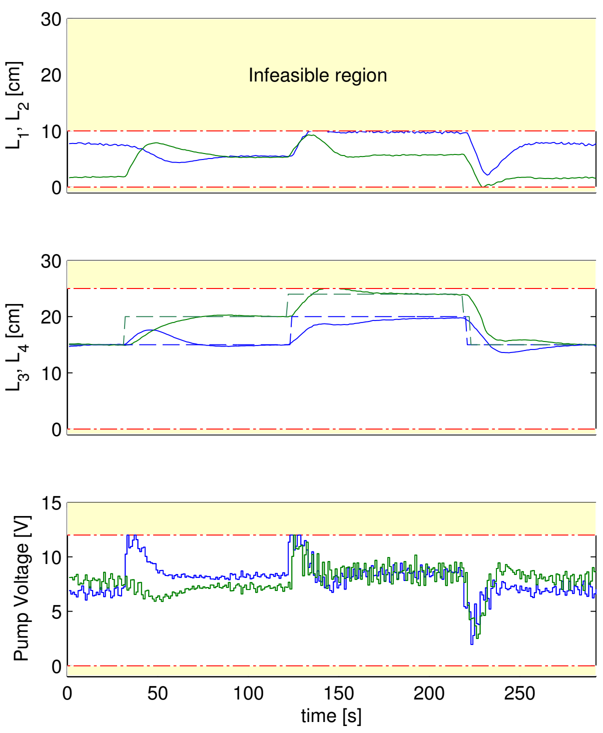


Fig. 6. An MPC response showing active state constraints where the QP could not find a feasible solution, but still delivered adequate control.

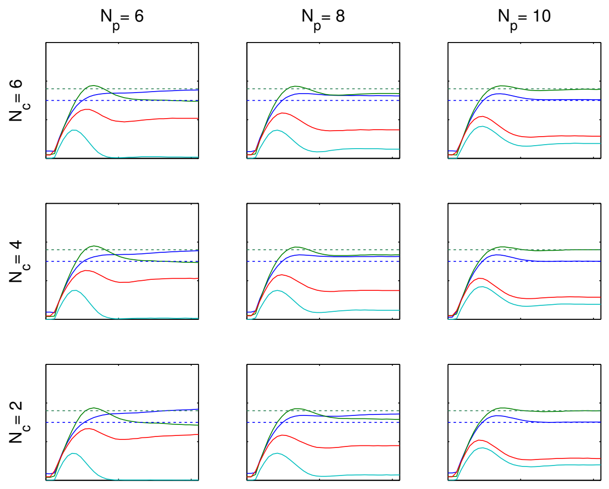


Fig. 7. An MPC response of the 4 tank system showing varying horizon lengths with sample time $T_s = 3$.

The ability of MPC to still maintain reasonable control at coarse sample times is vital because it means that one can implement the MPC in modest hardware since the horizons are relatively short. If we were to sample at the recommended rate as shown in Fig. 5, then the horizons need be $N_p > 27$ which is memory and computationally demanding.

5. EMBEDDED HARDWARE OPTIONS FOR AN MPC

Our long-term goal for this project is to explore the possibility of implementing predictive control on small footprint, low power, low-cost hardware. This includes micro-controllers, DSP chips and FPGAs. For benchmarking purposes we also include giga-hertz processing units such as those found in current desktop PCs, although we note that such chips are not price competitive, nor are they complete systems on a chip.

A selection of suitable hardware for the embedded application is given in Table 1. For those chips without onboard flash, one can either use RAM, or if memory is a constraint, one can use external flash.

Table 1. Embedded hardware options considered for embedded MPC applications

Hardware	RAM kB	Flash kB	clock MHz	cost $
TI Delfino C28346	516	–	300	20.85
ARM9 LPC322/50	256	–	266	11.66
PIC32 MX795	128	512	80	9.41
FPGA Spartan 3 XC3S500E	128	–	50	30.94

The ram and flash memory requirements for the 4-tank system are given in Fig. 8 as a function of control and prediction horizons using a sample time of $T_s = 3$. The maximum amount of onboard ram and flash for each of the chips is given in Table 1 for reference.

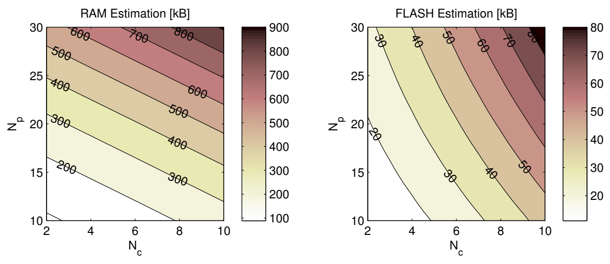


Fig. 8. The RAM and Flash memory requirements for the 4-tank system as a function of control and prediction horizons.

Problem specific constants such as the QP matrices **H** and **A** can be stored in flash (with **H** suitably prefactored), while variables such as the vectors **f**, components of **b**, and the internal variables must be stored in RAM.

An approximate estimate of the maximum achievable sampling rate for each of the 4 chips in Table 1 is given in Fig. 9. We derived these estimates using the following assumptions:

- No memory overhead is considered i.e. moving between registers, for loops, memory addressing
- The QP takes 15 iterations at every sample
- The KKT fails on 1/4 of constraints at every iteration
- Floating point operations on the integer micro take 30× as long

Furthermore, on the FPGA we have parallelised the arithmetic by using three floating point units to perform the matrix and vector operations.

It is interesting to note that the sampling rate is only a very weak function of the prediction horizon, (shown only for the TI chip in Fig. 9, but the others are similar), so provided there is enough memory, long predictions do not overly penalize the performance. Clearly the integer PIC32 micro-controller is completely outclassed by the more dedicated hardware.

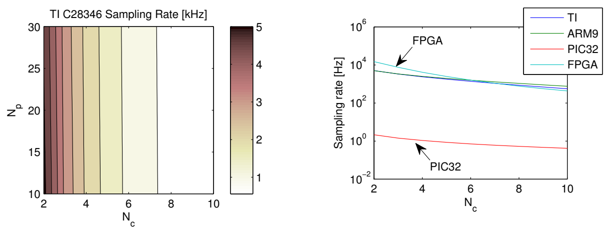


Fig. 9. The approximate maximum achievable sampling rates for the 4 chips.

6. CONCLUSIONS

This paper has demonstrated a lightweight MPC application to an experimental bench-scale 4-tank system and shown that the algorithm works under a wide variety of operating conditions and externally imposed constraints. Furthermore, we have shown that the algorithm is implementable on a variety of candidate embedded systems, that the memory requirements are feasible, and the predicted sampling rates are on par, or even surpass other reported embedded MPC applications.

ACKNOWLEDGEMENTS

Financial support to this project from the Industrial Information and Control Centre, Faculty of Engineering, The University of Auckland, New Zealand is gratefully acknowledged.

REFERENCES

Johan Åkesson. MPCtools: A toolbox for simulation of MPC controllers in Matlab. Technical report, Lund Institute of Technology, Lund, Sweden, January 2006.

R.A. Bartlett, A. Wächter, and L. T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, pages 4229–4233, Chicago, Illinois, USA, 2000.

Leonidas G. Bleris, Jesus Garcia, Mayuresh V. Kothare, and Mark G. Arnold. Towards embedded model predictive control for system-on-a-chip applications. *Journal of Process Control*, 16(3):255–264, 2006.

Jonathan Currie and David I. Wilson. A Model Predictive Control toolbox intended for rapid prototyping. In Tim Molteno, editor, *16th Electronics New Zealand Conference (ENZCon 2009)*, pages 7–12, Dunedin, New Zealand, 18–20 November 2009. ISBN 978-0-473-16099-9.

P. Dua, K. Kouramas, V. Dua, and E.N. Pistikopoulos. MPC on a chip — Recent advances on the application of multi-parametric model-based control. *Computers & Chemical Engineering*, 32(4-5):754 – 765, 2008.

C.E. Garcia, D.M. Prett, and M. Morari. Model predictive control: Theory and practice — A survey. *Automatica*, 25(3):335–348, 1989.

T. A. Johansen, W. Jackson, R. Schreiber, and P. Tøndel. Hardware synthesis of explicit model predictive controllers. *IEEE Transactions Control Systems Technology*, 15(1):191–197, 2007.

Mark S.K. Lau, S.P. Yue, K.V. Ling, and J. M. Maciejowski. A comparison of interior point and active set methods for FPGA implementation of model predictive control. In *European Control Conference 2009*, pages 156–161, Budapest, Hungary, 23–26 August 2009. EUCA. ISBN 978-963-311-369-1.

K.V. Ling, B.F. Wu, and J.M. Maciejowski. Embedded Model Predictive Control (MPC) using a FPGA. In *IFAC Proceedings of the 17th World Congress*, pages 15250–15255, Seoul, South Korea, 6–11 July 2008.

J. M. Maciejowski. *Predictive Control with Constraints.* Prentice–Hall, 2002.

Efstratios N. Pistikopoulos, Vivek Dua, Nikolaos A. Bozinis, Alberto Bemporad, and Manfred Morari. On-line optimization via off-line parametric optimization tools. *Computers & Chemical Engineering*, 24(2–7):188–188, 2000.

Efstratios N. Pistikopoulos, Michael C. Georgiadis, and Vivek Dua. Parametric programming & control: from theory to practice. In Valentin Plesu and Paul Serban Agachi, editors, *17th European Symposium on Computer Aided Process Engineering*, volume 24 of *Computer Aided Chemical Engineering*, pages 569–574. Elsevier, 2007.

S. Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, July 2003.

Rajanikanth Vadigepalli and Edward P. Gatzke and Francis J. Doyle III. Robust Control of a Multivariable Experimental Four-Tank System. *Industrial & Engineering Chemistry Research*, 40(8):1916–1927, 2001.

J. A. Rossiter. *Model-Based Predictive Control: A Practical Approach.* CRC Press, 2003.

Dale E. Seborg, Thomas F. Edgar, and Duncan A. Mellichamp. *Process Dynamics and Control.* Wiley, 2 edition, 2005.

G. Valencia-Palomo and J.A. Rossiter. Auto-tuned predictive control based on minimal plant information. In *International Symposium on Advanced Control of Chemical Processes ADCHEM 2009*, pages 582–587, Istanbul, Turkey, 12–15 July 2009. International Federation of Automatic Control.

Y. Wang and S. Boyd. Fast model predictive control using online optimization. In *Proceedings of the IFAC World Congress*, pages 6974–6997, Seoul, Korea, July 2008.

David I. Wilson and Brent R. Young. The Seduction of Model Predictive Control. *Electrical & Automation Technology*, pages 27–28, Dec/Jan 2006. ISSN: 1177-2123.

Stephen Wright. Applying new optimization algorithms to model predictive control. In *Fifth International Conference on Chemical Process Control CPC V*, pages 147–155. CACHE Publications, 1997.