FRAM EVALUATION AS UNIFIED MEMORY FOR CONVEX OPTIMIZATION ALGORITHMS

Cimini Gionata[1], *Bemporad Alberto*[2], *Ippoliti Gianluca*[1] *and Longhi Sauro*[1]

[1] Università Politecnica delle Marche, Via Brecce Bianche 12, 60131, Ancona, Italy
[2] IMT - Institute for Advanced Studies, Piazza San Ponziano 6, 55100 Lucca, Italy
email: g.cimini@univpm.it, alberto.bemporad@imtlucca.it, g.ippoliti@univpm.it, s.longhi@univpm.it

ABSTRACT

Ferroelectric Random Access Memory (FRAM) by Texas Instruments (TI) is a non-volatile memory which allows lower power and faster data throughput compared to other non-volatile solutions. These features have accelerated the interest in this technology as the future of embedded unified memory, in particular in data logging, remote sensing and Wireless Sensor Network (WSN). The application of Model Predictive Control (MPC) in WSN has gained lot of attention in the last years and it requires solving convex optimization problems in real-time. In this paper several convex optimization algorithms have been implemented and compared on a FRAM-based MSP-EXP430FR5739 node by TI, to evaluate its suitability in extending the potentialities of onboard volatile Static Random Access Memory (SRAM) for embedded optimization-based control.

1. INTRODUCTION

Thanks to the recent advances in a variety of engineering disciplines, such as system miniaturization, wireless communication and signal processing, WSNs have gained a lot of attention, in particular in the field of smart environments. A sensor node is usually battery-powered and combines sensing, computation and communication capabilities. Nowadays energy consumption remains as a major obstacle for full deployment and exploitation of WSNs in very different application fields [1, 2]. For this reason lot of researchers are focused on the low-power circuit design techniques or the power management approaches to improve node's power consumption, minimizing for example the communication's requirements [3] or focusing on adding external power sources to the node [4], [5].

Among these approaches, efforts have been made to develop memories for this kind of applications, as the FRAM by TI, which is competitive with emerging new memory technologies, but is already commercially available and considered as the memory of the future due to its ideal properties [6]. FRAM is a non-volatile memory which combines the speed, ultra-low-power, endurance and flexibility of SRAM with the reliability and stability of flash memory; furthermore it allows random access to each individual bit for both read and write. FRAM technology is particularly suited in WSNs because, compared to standard flash memory, it has more than $100x$ faster write speed while having less than half the memory access energy per bit resulting in much higher throughput and less energy per stored bit [7].

In this paper the performance of the FRAM has been evaluated, focusing on its flexibility in terms of programming and on its write speed. The benchmark consists in the implementation of convex optimization algorithms on MSP-EXP430FR5739 Experimenter Board by TI, a development platform for data logging applications, energy harvesting, wireless sensing and automatic metering infrastructure. The aim of the work is to verify the possibility to extend the capabilities of on-board volatile memory with a non-volatile support, when the problems to be solved are larger than the storage space in the volatile support. The choice for the algorithms to implement comes from their application in MPC, where the control problem requires solving Quadratic Programming (QP) in real time. The investigation of MPC for the FRAM sensor node is motivated by the recent studies in the applicability of MPC in the context of WSNs, where the technique has been used for power control to guarantee a specified Quality of Service (QoS) adapting the transmission power level which has to be minimized [8]. Furthermore, MPC has been adopted in WSN for other purposes such as resources' allocation and power management in energy harvesting [9]. More generally MPC can be thought as a way of expanding WSN nodes with real-time decision capabilities. The need for a QP solver to be executed at each sampling period has increased the interest in algorithms suited for embedded applications where the simplicity to code is a key feature, together with proved guarantees on worst-case execution time and robustness to low precision arithmetic. For these reasons, different algorithms have been developed in the last years and some of them, suitable for embedded application, have been implemented and evaluated on the device: Dual Gradient Projection (DGP), the Accelerated Dual Gradient-Projection (GPAD) [10], Parallel Quadratic Programming (PQP) [11] and Alternating Direction Method of Multipliers (ADMM) [12]. The four algorithms have been compared in terms of applicability on the FRAM support, evaluating the speed of a complete SRAM execution and a complete FRAM one which, in addition to the above mentioned benefits, permits the storage of a larger problem on the selected device ($1Kb$ of SRAM and $16KB$ of FRAM). Furthermore the performance of the algorithms have been detailed in terms of data stored, Time Per Iteration (TPI) and average total execution time with worst case evaluation. The paper is organized as follows: in Section 2 the FRAM technology is detailed; Section 3 defines the formulation of the problem to be solved together with the algorithms tested which are briefly summarized. Finally in Section 4 the performance of the algorithms are compared.

2. FRAM TECHNOLOGY

While the benefits of FRAM have been known for many years, [6], the mass-production of FRAM-based devices has

978-1-4799-6843-5/14/$31.00 ©2014 IEEE

begun recently due to the improvements in encapsulation, robustness to fatigue and access schemes for high density devices, [13–15]. FRAM stores information using the polarization of ferroelectric thin film placed between two electrodes. The FRAM cell structure is very similar to that used in of DRAM, consisting of One Transistor One Capacitor (1T1C) cell. The dipole orientation, which is the stored information, can be set and reversed by applying voltage across either line. The most important feature of the ferroelectric material is its non-volatility and immunity to magnetic fields. Unlike widely used non-volatile memory technologies such as EEPROM or Flash memory, FRAM does not require a special sequence to write data as random access to each individual bit for both read and write is permitted. The main features of the ferroelectric memory are briefly detailed in the following section.

2.1 Speed and Universality

Normally tens of ms to several ms are required to write flash memory in order to program one data word. In addition, this time does not include the pre-erasing of the segment to be reprogrammed, which takes several *ms*. In contrast, FRAM requires only $100ns$ to program one data word and pre-erasing is not required. Thanks to this fast write, there is virtually no interruption of the program execution and a speed close to the volatile memories such as SRAM allows FRAM to be used as a unified "memory", flexible to be partitioned to the needs of the application, providing more bytes for data (i.e., in data logging application) or for code (i.e., in complex algorithm implementation).

2.2 Low Power and Reliability

Reads and writes in FRAM-based devices occur at low voltage, just $1.5V$, and at very little current; thus a charge pump in not needed to operate, drastically reducing the power consumption as well as the footprint of the device. For this devices TI guarantee 10 years of operation (virtually unlimited write endurance) and data retention at 85°. Furthermore FRAM does not suffer from Soft Error, which are flips of bit states due to external sources.

3. PROBLEM FORMULATION AND IMPLEMENTED ALGORITHMS

For the convenience of the reader, the convex optimization algorithms considered in this paper for solving the linear MPC problems are briefly described. In particular, such algorithms can handle polyhedral constraints and convex quadratic objective function, resulting in a constrained Quadratic Programming (QP) problem of the form:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & V(z) = \frac{1}{2} z^T Q z + q^T z \\ \text{subject to} \quad & g(z) = Az - b \leq 0 \end{aligned} \tag{1}$$

with $z \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^n$. The dual problem of (1) is given by the following formulation:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \Phi(z) = \frac{1}{2} y^T H y + h^T y + \frac{1}{2} W \\ \text{subject to} \quad & y \geq 0 \end{aligned} \tag{2}$$

with

$$H = AQ^{-1}A^T, \quad h = AQ^{-1}q + b, \quad W = q^T Q^{-1} q \tag{3}$$

where $H \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $y \in \mathbb{R}^m$.

Definition 3.1. Let z^* and V^* be the optimal solution and the optimal value for the primal problem (1) and consider two non-negative constants ε^v,ε^g and $\mathscr{Z} = \{z \in \mathbb{R}^n | Az - b \leq 0\}$; we say that $\bar{z} \in \mathbb{R}^n$ is an $(\varepsilon^v, \varepsilon^g)$-optimal solution for (1) if

$$V(\bar{z}) - V^* \leq \varepsilon^v \tag{4a}$$

$$\max_{i \in N_{[1,m_x]}} [g_i(\bar{z})]_+ \leq \varepsilon^g \tag{4b}$$

where $[g_i(\bar{z})]_+ = max\{g_i(\bar{z}), 0\}$.

In the following a summary for each QP solver considered in this paper is provided. The implementations adopted are optimized for speed execution, storing as much data as possible to reduce the Time Per Iteration (TPI). The importance of coding details for energy-aware applications has been addressed in [16].

3.1 Dual Gradient Projection (DGP)

Considering the optimization problem in (1) and its Lagrangian $\mathscr{L}(z,y) = V(z) + y^T g(z)$, the gradient projection algorithm applied to the dual problem consists in solving the following equations at every iterations.

$$z^k = \underset{z}{argmin}\, \mathscr{L}(z, y^k) \tag{5}$$

$$y^{k+1} = \left[y^k + \frac{1}{L} g(z^k) \right]_+ \tag{6}$$

where

$$L = \sqrt{\sum_{i,j=1}^{m} |H_{i,j}|^2} \tag{7}$$

is the Frobenius norm of H. In this paper a fixed-point implementation of DGP has been used, presented in [17] where an inexact DGP method specifically suitable for a fixed-point implementation has been developed together with detailed convergence rate analysis (based on [18]).

3.2 Accelerated Dual Gradient-Projection Algorithm (GPAD)

In [10] an algorithm based on the fast gradient projection method of [19] has been proposed and applied to the QP (1). It is particularly tailored for embedded linear MPC with polyhedral constraints on both inputs and states. An iteration of the GPAD algorithm is:

$$w^k = y^k + \beta^k (y^k - y^{k-1}) \tag{8}$$

$$z^k = \underset{z}{argmin}\, \mathscr{L}(z, w^k) \tag{9}$$

$$y^{k+1} = \left[w * k + \frac{1}{L} g(z^k) \right]_+ \tag{10}$$

$$\beta^{k+1} = \frac{k-1}{k+2} \tag{11}$$

This algorithm guarantee a convergence rate of $O(1/k^2)$ for both dual and primal optimality as well as for primal feasibility.

3.3 Parallel Quadratic Programming (PQP)

The Parallel Quadratic Programming (PQP) method has been developed in [20] for image processing problems in which a quadratic minimization program in the non-negative cone is involved, which is the form of the dual problem in (2). Recently the PQP structure has been exploited for constrained MPC [11]. Given a generic matrix A, let us define the two matrices:

$$[A^-]_{i,j} = max\{0, -[A]_{i,j}\} \tag{12}$$
$$[A^+]_{i,j} = max\{0, [A]_{i,j}\} \tag{13}$$

Furthermore, construct a diagonal matrix $\Omega \in \mathbb{R}^{m\times m}$, computed to guarantee convergence; the choice adopted in [11] is $[\Omega]_{ii} \geq [H^-1]$. By taking $F = -Q^{-1}A^T$ a PQP iteration is:

$$\left[y^{k+1}\right]_i = \frac{\left[(H^- + \Omega)z^k + F^-\right]_i}{\left[(H^+ + \Omega)z^k + F^+\right]_i}[z^k]_i \tag{14}$$

3.4 Alternating Direction Method of Multipliers (ADMM)

ADMM is a simple method for solving large-scale convex optimization problems and is becoming very popular [21]. In fact, ADMM offers a solution comparable to other efficient algorithms in serial regime, but is very popular in distributed optimization, merging the benefits of augmented Lagrangian and dual decomposition: the solutions of local sub-problems are coordinated to find the solution to a larger problem. Consider problem (1) in a version equal to the ADMM standard form [12], introducing a slack variable x:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \hat{V}(z) = \frac{1}{2}z^T Qz + q^T z + \mathscr{F}(x) \\ \text{subject to} \quad & \hat{g}(z,x) = Az - b + x = 0 \end{aligned} \tag{15}$$

where $\mathscr{F}(x)$ is the indicator function in the non-negative orthant. Considering the augmented Lagrangian function in a scaled form,

$$\begin{aligned} L_\rho(z,x,u) = & \frac{1}{2}z^T Qz + q^T z + \mathscr{F}(x) \\ & + \frac{\rho}{2}||Az - b + x + u||_2^2 - \frac{\rho}{2}||u||_2^2 \end{aligned} \tag{16}$$

with $u = y/\rho$, the scaled ADMM iterations consists on separating minimization of (16) over z and x into two different steps and then perform an ascent step on the Lagrange multipliers y. Thus an ADMM iteration consists in:

$$z^{k+1} = \underset{z}{argmin}\ L_\rho(z, x^k, u^k) \tag{17a}$$
$$x^{k+1} = \underset{z}{argmin}\ L_\rho(z^{k+1}, x, u^k) \tag{17b}$$
$$u^{k+1} = u^k + \rho\hat{g}(z,x) \tag{17c}$$

4. RESULTS

In this section the results of the implementation of the algorithms in Section 3 are collected. The experimental setup consists of a MSP-EXP430FR5739 programmed with USB interface via Code Composer Studio 5. The MSP430FR5739 integrated microcontroller has a 16-Bit fixed-point RISC architecture with a system clock up to 24-MHz and a memory composed of 16KB FRAM and 1KB SRAM. The algorithms have been compared in terms of memory allocation and execution time over the two different memories. Random generated QP problems have been used to test the algorithms; different sizes of QP have been tested considering $n \in [2, 10]$ and $m \in [4, 20]$. The experimental setup for the evaluation of performance is shown in Figure (1).

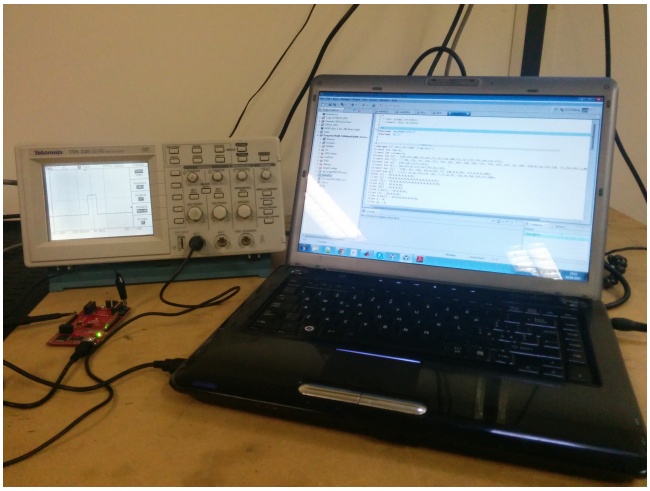


Figure 1: The experimental setup, with MSP-EXP430FR5739 device and oscilloscope to capture the time needed by the different solvers tested.

4.1 Memory Allocation

Considering n, m defined as in equation (1) and the algorithms' implementations as described in Section 3, the memory occupancy relationships in terms of stored data can be compared as in Figure 2. For embedded application, a static memory allocation is preferable and all the algorithms have been coded with this shrewdness. Figure 2 gives a qualitative comparison in terms of bytes allocated by the algorithms in relation with the size of the problem: the horizontal plane in each graph represents the $1Kb$ limit of the SRAM memory of the device; as has been prove in the next section, the size of the problem can be extended involving the FRAM without compromising the performances of the algorithms. Using the proposed implementation Figure 2 shows that the PQP algorithm stores more data than the other implementations which, are seen to be approximately comparable.

4.2 Execution Time

The four algorithms have been tested comparing a complete execution on the SRAM support and a complete execution on the FRAM support. To evaluate the execution time, as we are in the order of microseconds, software-based methods have been avoided and a pin toggle has been used instead with an oscilloscope. By default the CCS compiler stores the variables defined as constants in the FRAM support, and the others in the SRAM; to force a complete execution on SRAM is sufficient to avoid the definition of constant variables, whereas a complete FRAM execution needs different steps. A new area of FRAM must be defined by modifying

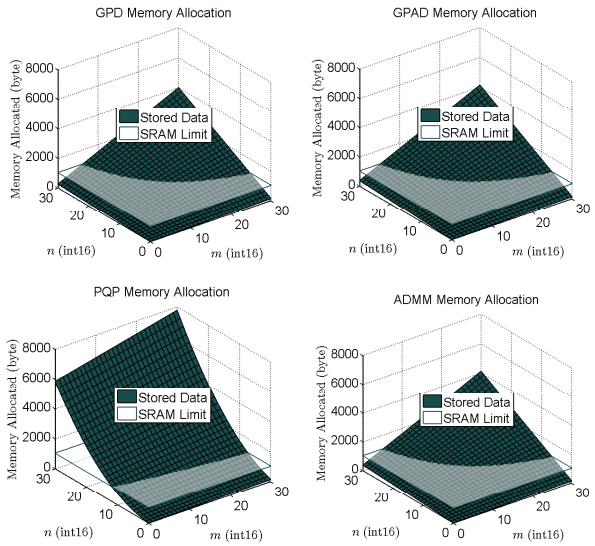


Figure 2: Memory Allocation for the Optimizazion Algorithms

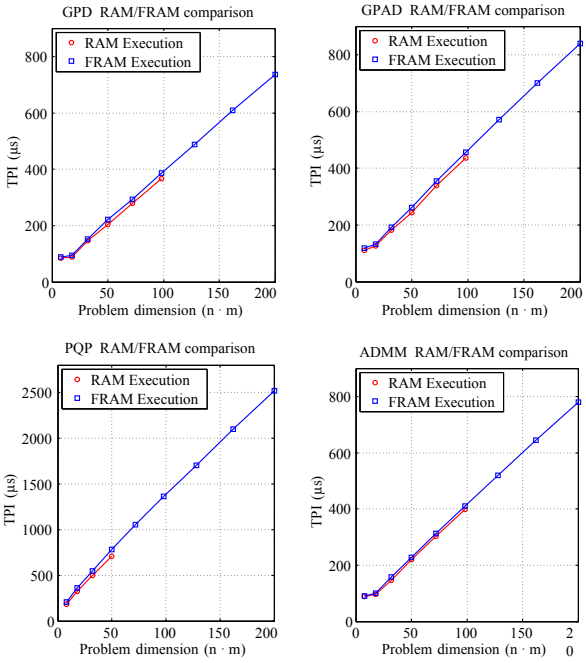


Figure 3: Time Per Iteration Benchmark between SRAM and FRAM Execution: the SRAM plots stop when there is no more space available.

the system memory map and a new memory section has to be created inside the previously defined memory area. Finally the definition of the variables have to be included in an environment as in Code 1. A part from this basic steps, the variables can be used with the simplicity of a SRAM as the writing and reading phase from FRAM support is totally transparent.

Figure 3 shows the Time Per Iteration (TPI) benchmark of the four algorithms. The TPI is a deterministic evaluation of the embedding capabilities of an algorithm. As the selected platform provides a 32-bit Hardware Multiplier (MPY), sums and multiplications are easily performed by the device. GPD, GPAD and ADMM are substantially comparable; GPD and ADMM show equal computation time, GPAD exhibits a slight increase in time and finally PQP shows the worst performances respect to other algorithms (this is due to the necessity of several divisions). Figure 3 shows the comparison between the execution time on SRAM and on FRAM for each algorithm. The results show that the time needed by a complete FRAM execution is comparable to that needed by the volatile memory, thus it is observed that FRAM can be used to deal with larger problems without noticeable decrease in performance. For each graph in Figure 3 the execution on SRAM stops when no more space is available on volatile memory; as expected from memory allocation results, PQP can solve only smaller problems with respect to other algorithms. Figure 4(a) shows the mean time needed (over 1000 testing problems per dimension) to solve QPs of different sizes, considering the FRAM execution; Figure 4(b) presents the worst case with the same testing problems.

The results for both memory allocation and execution time can be summarized as follows:

- A complete execution on the FRAM is comparable with that on SRAM, so the non-volatile memory can be used to store and solve larger problems;
- The PQP algorithm stores a greater amount of data compared to the other three algorithms which are substantially comparable;
- ADMM, GPAD and GPD result to be particularly suited to be executed on the embedded platform, thanks to the simplicity of the operations involved;
- ADMM outperforms the other algorithms (also for the worst case) with a combination of low TPI and the lowest number of iterations needed; only the GPAD remains a good alternative;

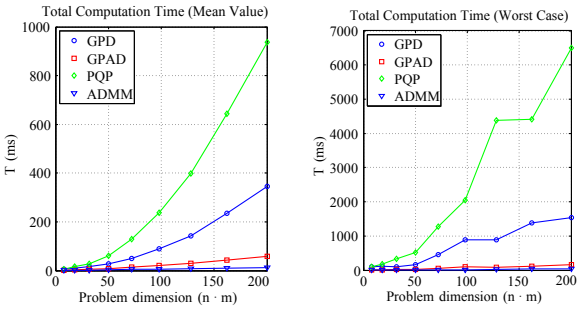
It is important to note that the good performance of the ADMM algorithm is deeply related on the selection of the augmenting term ρ for the Lagrangian. The results presented in this paper are obtained with a good choice of this factor, selected empirically.

Code 1: Example of FRAM variable definition

```
#pragma SET_DATA_SECTION(".fram_vars")
[...variables definition...]
#pragma SET_DATA_SECTION()
```

5. CONCLUSION

In this paper an MSP430 platform by TI with FRAM memory technology has been used for a benchmark of iterative solvers for quadratic programming. The novel non-volatile memory has been tested in terms of execution speed and compared to the standard integrated SRAM, showing competitive results and thus allowing a combined use of volatile and non-volatile memory to increase the stored problem's dimensions. The suitability of the platform for solving optimization problems, for example in MPC applications for WSNs, has been confirmed. The GPD, GPAD, PQP and ADMM algorithms have been implemented in a speed-aware form on the platform and compared from both a memory allocation and an execution time point of view.



(a) Total Optimization Time Comparison (Mean Value)

(b) Total Optimization Time Comparison (Worst case)

Figure 4: Execution time tested on 1000 random QP per dimension with $n = x$, $m = 2x$ and $x \in [2, 10]$.

REFERENCES

[1] A. Seema and M. Reisslein, "Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a Flexi-WVSNP design," *Communications Surveys Tutorials, IEEE*, vol. 13, pp. 462–486, Third 2011.

[2] G. Owojaiye and Y. Sun, "Focal design issues affecting the deployment of wireless sensor networks for intelligent transport systems," *Intelligent Transport Systems, IET*, vol. 6, pp. 432–432, December 2012.

[3] R. Yan, H. Sun, and Y. Qian, "Energy-aware sensor node design with its application in wireless sensor networks," *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, pp. 1183–1191, May 2013.

[4] A. S. Weddell, N. J. Grabham, N. R. Harris, and N. M. White, "Modular plug-and-play power resources for energy-aware wireless sensor nodes.," in *SECON*, pp. 1–9, IEEE, 2009.

[5] D. Riley and M. Younis, "A modular and power-intelligent architecture for wireless sensor nodes," in *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pp. 304–307, Oct 2012.

[6] R. Bailey, G. Fox, J. Eliason, M. Depner, D. Kim, E. Jabillo, J. Groat, J. Walbert, T. Moise, S. Summerfelt, K. R. Udayakumar, J. Rodriquez, K. Remack, K. Boku, and J. Gertas, "FRAM memory technology - advantages for low power, fast write, high endurance applications," in *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, 2005.

[7] A. Baumann, M. Jung, K. Huber, M. Arnold, C. Sichert, S. Schauer, and R. Brederlow, "A MCU platform with embedded FRAM achieving 350na current consumption in real-time clock mode with full state retention and system wakeup time," in *VLSI Circuits (VLSIC), 2013 Symposium on*, pp. C202–C203, 2013.

[8] K. Witheephanich and M. J. Hayes, "On the applicability of model predictive power control to an ieee 802.15.4 wireless sensor network," in *Signals and Systems Conference (ISSC 2009), IET Irish*, pp. 1–8, June 2009.

[9] X. J. Li, X. Shao, K.-V. Ling, and B.-H. Soong, "Application of model predictive control in wireless sensor networks," in *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pp. 1–5, Dec 2011.

[10] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *Automatic Control, IEEE Transactions on*, vol. 59, pp. 18–33, Jan 2014.

[11] S. D. Cairano, M. Brand, and S. A. Bortoff, "Projection-free parallel quadratic programming for linear model predictive control," *International Journal of Control*, vol. 86, no. 8, pp. 1367–1385, 2013.

[12] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Now Publishers, 2011.

[13] Y. Kang and S. Lee, "The challenges and directions for the mass-production of highly-reliable, high-density 1T1C FRAM," in *Applications of Ferroelectrics, 2008. ISAF 2008. 17th IEEE International Symposium on the*, vol. 1, pp. 1–2, 2008.

[14] H. K. Ko, D. Jung, Y. K. Hong, J. Park, Y. Kang, H. Kim, Y. Kang, H. Kim, W. Jung, D. Choi, S. Kim, W. S. Ahn, J.-H. Kim, W. W. Jung, E. Lee, Y. Kang, S. Lee, H. Jeong, and K. Kim, "A novel encapsulation technology for mass-productive 150 nm, 64-mb, 1T1C FRAM," in *Applications of Ferroelectrics, 2007. ISAF 2007. Sixteenth IEEE International Symposium on*, pp. 25–27, 2007.

[15] Z. Jia, G. Zhang, M. ming Zhang, T.-L. Ren, and H. yi Chen, "A novel fatigue-insensitive self-referenced scheme for 1T1C FRAM," in *Memory Workshop (IMW), 2010 IEEE International*, pp. 1–2, 2010.

[16] T. Krueger, "Simple and cheap measurement of FRAM current consumption and performance data," in *Education and Research Conference (EDERC), 2012 5th European DSP*, pp. 53–57, 2012.

[17] P. Patrinos, A. Guiggiani, and A. Bemporad, "Fixed-point dual gradient projection for embedded model predictive control," in *Control Conference (ECC), 2013 European*, pp. 3602–3607, 2013.

[18] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," No. 2011002, 2011.

[19] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[20] M. Brand and D. Chen, "Parallel quadratic programming for image processing," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 2261–2264, 2011.

[21] D. Han and X. Yuan, "Local linear convergence of the alternating direction method of multipliers for quadratic programs," *SIAM Journal on Numerical Analysis*, vol. 51, no. 6, pp. 3446–3457, 2013.