Approximate LASSO Model Predictive Control for Resource Constrained Systems

Approximate LASSO Model Predictive Control for Resource Constrained Systems

Yun Wu, João F. C. Mota, Andrew M. Wallace
The School of Engineering and Physical Sciences
Heriot-Watt University
Edinburgh, UK
{y.wu, j.mota, a.m.wallace}@hw.ac.uk

***Abstract*—LASSO MPC is a popular method for solving optimal control problem in receding horizon. It is challenge to deploy LASSO MPC on resource constrained system, such as embedded platform, due to the intensive memory and computation cost along the growing horizon length. By exploiting the reduced precision approximation technique on a Lean Proximal Gradient Descent (LPGD), we demonstrate the approximate implementation of the LPGD on reconfigurable device, such as Field Programmable Gate Array (FPGA). An approximate core synthesis infrastructure is developed to simulate and generate LPGD for LASSO MPC. The outcomes shows comparable control consequence with significant improvements on both logic and memory cost reduction as well as the power consumption saving, enabling promising solution for implementing LASSO MPC on resources constrained system.**

***Index Terms*—Model Predictive Control, LASSO, Proximal Gradient Descent, Approximate Computing, Field Programmable Gate Array**

I. Introduction

Model Predictive Control (MPC) is a popular technique to solve optimal control problems in discrete time. Its robustness, stability, and theoretical guarantees have made it widely adopted by industry [1]. Given a dynamical model of a system and an estimate of its current state, MPC computes the next states and inputs by minimizing a cost function that balances between achieving a desired state in a predefined time-horizon and the energy to do so. As energy is often expressed as a quadratic form, MPC typically entails the application of (nonzero) input signals to the system at all time steps.

To reduce the number of times that inputs are applied, the work in [2] proposed *Lasso MPC* which, inspired by results on sparse regularization, regularizes the energy term with an ℓ_1-norm penalty. This leads to temporally sparse input signals.

Despite having many desirable features, (Lasso) MPC requires an iterative procedure to solve an optimization problem at each time step. This not only makes its execution in real-time challenging (as the number of required iterations is unknown a priori), but also demands significant memory and computation, which can be limiting on resource-constrained systems, such as embedded hardware.

In such systems, one often has to trade-off accuracy for power savings by using approximate computing (AC) techniques [3]. One example is reduced precision (RP), in which data is represented with fewer bits than desired throughout the entire computational stack [4]. As arbitrary precision arithmetic is not supported in many modern processors, a flexible hardware architecture enabling RP is demanding, such as reconfigurable platform using Field Programmable Gate Array (FPGA).

Work supported by EPSRC Grant number EP/S000631/1 and the MOD University Defence Research Collaboration (UDRC) in Signal Processing.

Our goal. We aim to implement an efficient solution to Lasso MPC and deploy it on a Field Programmable Gate Array (FPGA), with the goal of achieving real-time performance [5]. Our optimization algorithm of choice is the proximal gradient descent (PGD) [6], which is simple enough to analyze the effects of different RP strategies on accuracy, logic and memory resources, and power consumption.

Contributions. We summarize our contributions as follows:

- We apply PGD to Lasso MPC which, to the best of our knowledge, has never been done before, and analyze its performance. The application of accelerated versions of PGD [6] should be immediate.
- We introduce an approximate core synthesis infrastructure.
- Using our infrastructure, we then conduct a detailed study of the effects of AC on our algorithm.

II. Background

We briefly explain MPC, Lasso MPC, and then review different number representations and their use in RP.

State-space models. We consider state-space representations of linear time-invariant (LTI) discrete systems. That is, at each time t, such systems are completely described by a state vector $x[t] \in \mathbb{R}^n$, which is known to evolve as

$$x[t+1] = Ax[t] + Bu[t]\,, \qquad t = 0, 1, \dots\,, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are given matrices (assumed known), and $u[t] \in \mathbb{R}^m$ is the set of inputs at time t.

Model predictive control (MPC). Given a finite time-horizon $T \in \mathbb{N}$, a desired final state $x_f \in \mathbb{R}^n$, and an estimate of the current state $x_0 \in \mathbb{R}^n$, MPC attempts to compute a minimal energy state-trajectory such that the final state $x[T]$

is as close as possible to the desired one, x_f. This can be formulated as an optimization problem:

$$\begin{array}{ll}\min\limits_{\overline{x},\overline{u}} & F(x[T]) + \sum_{t=0}^{T-1} \ell(x[t],u[t]) \\ \text{s.t.} & x[t+1] = Ax[t] + Bu[t]\,, \quad t = 0,\dots,T-1 \\ & x[0] = x_0\,,\end{array} \tag{2}$$

where $(\overline{x},\overline{u}) := (\{x[t]\}_{t=0}^{T}, \{u[t]\}_{t=0}^{T-1}) \in \mathbb{R}^{n(T+1)} \times \mathbb{R}^{mT}$ is the optimization variable, $F : \mathbb{R}^n \to \mathbb{R}$ a function that penalizes deviations of the final state $x[T]$ from the desired one, and $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ a function that measures "energy" at each time instant. Notice that the first set of constraints is exactly (1), and the second set reflects the current state.

The functions F and ℓ are often quadratic forms. An example assuming $x_f = 0$ would be

$$F(x) = x^\top P x \tag{3a}$$

$$\ell(x,u) = x^\top Q x + u^\top R u\,, \tag{3b}$$

where $P, R \succ 0$ are positive definite matrices and $Q \succeq 0$ is positive semidefinite. With this choice, the variable $\overline{x}$ in (2) can be eliminated. To see why, first write (1) as [7, Ch. 8]

$$\underbrace{\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[T] \end{bmatrix}}_{=:\overline{x}} = \underbrace{\begin{bmatrix} I_n \\ A \\ \vdots \\ A^T \end{bmatrix}}_{=:\overline{A}} x_0 + \underbrace{\begin{bmatrix} 0 & \cdots & 0 \\ B & \cdots & 0 \\ \\ AB & \ddots & 0 \\ A^{T-1}B & \cdots & B \end{bmatrix}}_{=:\overline{B}} \underbrace{\begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[T-1] \end{bmatrix}}_{=:\overline{u}}, \tag{4}$$

where I_n is the identity matrix in $\mathbb{R}^n$. Replacing the resulting equation into the cost of (2) and manipulating, we obtain

$$\underset{\overline{u}}{\text{minimize}} \;\; \frac{1}{2}\overline{u}^\top \big(\overline{B}^\top \overline{Q}\,\overline{B} + \overline{R}\big)\overline{u} + (\overline{B}^\top \overline{Q}\,\overline{A}\,x_0)^\top \overline{u}\,, \tag{5}$$

where $\overline{R} := I_T \otimes R$, and $\overline{Q}$ is the diagonal concatenation of $I_T \otimes Q$ and P ($\otimes$ denotes the Kronecker product). Problem (5) is unconstrained quadratic, and thus has a closed-form solution. In MPC [7], whenever (5) is solved, only the first input $u[0]$ is applied to the system, the resulting state is measured, and (5) is solved again using x_0 as the current state.

Lasso MPC. Although (5) has a closed-form solution, it typically yields dense (i.e., non-sparse) inputs, which can lead to over-actuated systems. To encourage sparse inputs, [2] proposed Lasso MPC, which adds an ℓ_1-norm penalty $\lambda\|\overline{u}\|_1$, with $\lambda > 0$, to (5) [or, equivalently, $\lambda\|u\|_1$ to (3b)]. Noticing that the objective of (5) can be written as $\frac{1}{2}\|H\overline{u} - y\|_2^2 - \frac{1}{2}\|y\|_2^2$, with $H := (\overline{B}^\top \overline{Q}\,\overline{B} + \overline{R})^{1/2}$ and $y := -H^{-1}\overline{B}^\top \overline{Q}\,\overline{A}x_0$, the resulting problem is

$$\underset{\overline{u}}{\text{minimize}} \;\; \frac{1}{2}\|H\overline{u} - y\|_2^2 + \lambda\|\overline{u}\|_1\,, \tag{6}$$

which has format of Lasso [8]. Notice, however, that the matrix H in (6) is square. So, instead of regularizing the problem as in classical Lasso, the ℓ_1-norm term enforces sparse inputs at the cost of possibly not reaching (or delaying it) the desired state. Notice also that because we eliminated the state variable $\overline{x}$ from (2) to (5), the system dynamics are always (implicitly) satisfied.

Reduced precision (RP). Some platforms require representing arithmetic numbers with short binary codes, thereby reducing their accuracy. There are three main categories of number representations: floating point [9], [10], Q fixed point format [11], and universal numbers (Unum) [12].

The IEEE 754 floating point arithmetic standard [9] represents a number by using three elements: a sign (1 bit), an exponent (n bits), and a mantissa (m bits). That is,

$$\text{sign} \times \text{mantissa} \times 2^{\text{exponent}}\,. \tag{7}$$

The Q fixed point format uses instead a fixed number of bits to represent the integer and the fractional parts of a number. Specifically, a number is represented by its sign (1 bit), its integer part (t bits), and its fractional part (k bits):

$$\text{sign} \times (2^{\text{integer}} + 2^{-\text{fraction}})\,. \tag{8}$$

Unum arose as an alternative to IEEE 754, and there are several versions. For example, Type III Unum, Posit, represents numbers via their sign (1 bits), regime (g bits), exponent (p bits), and fractional part (c bits):

$$\text{sign} \times (2^{2^p})^{\text{regime}} \times 2^{\text{exponent}} \times \Big(1 + \frac{\text{fraction}}{2^c}\Big)\,. \tag{9}$$

Each of these representations has a different dynamic range. And different dynamic ranges affect not only the performance of the algorithm by limiting the type of operations that can be performed, but also their embedded implementation.

III. Algorithm and Implementation

We now explain the algorithm we use to solve Lasso MPC and its implementation on resource-constrained systems.

A. Proximal Gradient Descent

We consider the Lasso problem in (6) and apply the proximal gradient descent (PGD) algorithm with fixed stepsize [13]. PGD solves problems of the form

$$\underset{u}{\text{minimize}} \;\; f(u) := g(u) + h(u)\,, \tag{10}$$

where $g : \mathbb{R}^q \to \mathbb{R}$ is convex, differentiable, and its gradient is Lipschitz-continuous, i.e., there exists $L > 0$ such that $\|\nabla g(u) - \nabla g(v)\|_2 \le L\|u - v\|_2$, for all u, v. The function $h : \mathbb{R}^q \to \mathbb{R} \cup \{+\infty\}$ is assumed convex and closed. Given a stepsize $\alpha \le 1/L$ and an initial point u^0, PGD solves (10) by iterating (on k)

$$u^{k+1} = \text{prox}_{\alpha h}\big(u^k - \alpha\nabla g(u^k)\big)\,, \tag{11}$$

where the proximal operator of a convex, closed function ϕ at a point u is defined as

$$\text{prox}_\phi(u) := \underset{v}{\arg\min} \;\; \phi(v) + \frac{1}{2}\|v - u\|_2^2\,. \tag{12}$$

It is well known that the iterates produced by PGD satisfy [13]

$$f(u^k) - f(u^\star) \le \frac{\alpha L\|u^0 - u^\star\|_2^2}{2k}\,, \tag{13}$$

where $u^\star$ is a solution of (10), i.e., PGD converges sublinearly.

Application to Lasso MPC. As problem (6) has the format of (10) with[1] $g(u) = (1/2)\|Hu - y\|_2^2$ and $h(u) = \lambda\|u\|_1$, the application of PGD is immediate and yields the soft-thresholding algorithm. Specifically, (11) becomes

$$\begin{aligned} u^{k+1} &= \mathcal{S}_\lambda\Big(u^k - \alpha H^\top (Hu^k - y)\Big) \\ &= \mathcal{S}_\lambda\Big(\big(I_q - \alpha H^\top H\big)u^k + H^\top y)\Big)\,, \end{aligned} \tag{14}$$

where the soft-thresholding operator $\mathcal{S}_\lambda(u)$ applies to component i, for $i = 1, \ldots, q$, the following nonlinearity:

$$\Big[\mathcal{S}_\lambda(u)\Big]_i = \begin{cases} u_i - \lambda & , \ u_i > \lambda \\ 0 & , \ -\lambda \le u_i \le \lambda \\ u_i + \lambda & , \ u_i < -\lambda\,. \end{cases} \tag{15}$$

Parameters and precomputations. PGD (11) and the associated convergence result in (13) apply whenever the stepsize α satisfies $\alpha \le 1/L$, where L is a Lipschitz constant of ∇g. To compute it, we can estimate the maximum eigenvalue of $H^\top H$, $\lambda_{\max}(H^\top H)$, via, e.g., Lanczos's method, or use the structure of H and the matrices that define it:

Lemma 1. *Let $H = (\overline{B}^\top \overline{Q}\,\overline{B} + \overline{R})^{1/2}$, where $\overline{B}$, $\overline{Q}$, $\overline{R}$ are defined in* (4)-(5)*. Also, partition $\overline{B}$ vertically into $\overline{B}_1 \in \mathbb{R}^{nT \times mT}$, which contains the first nT rows of $\overline{B}$, and into $\overline{B}_2 \in \mathbb{R}^{n \times mT}$, which contains the last n rows of $\overline{B}$. Then,*

$$\begin{aligned} \lambda_{\max}(H^\top H) \le\ & \lambda_{\max}(Q)\lambda_{\max}\big(\overline{B}_1\overline{B}_1^\top\big) \\ & + \lambda_{\max}(P)\lambda_{\max}\big(\overline{B}_2\overline{B}_2^\top\big) + \lambda_{\max}(R)\,. \end{aligned} \tag{16}$$

The proof uses the subadditivity of $\lambda_{\max}(\cdot)$, and the properties of the Kronecker product; it is omitted for brevity. The matrices P, Q, and R encode the objective of MPC [cf. (3)], and the matrices $\overline{B}_1$ and $\overline{B}_2$ encode the system dynamics through their dependency on A and B [cf. (1)]. Although the matrices $\overline{B}_1$ and $\overline{B}_2$ have a lot of structure, namely $\overline{B}_1$ is block Toeplitz and $\overline{B}_2$ is the controlability matrix with permuted columns, it is not immediate to obtain a bound on $\lambda_{\max}\big(\overline{B}_1\overline{B}_1^\top\big)$ and $\lambda_{\max}\big(\overline{B}_2\overline{B}_2^\top\big)$ as a function of A and B. These quantities can be accurately estimated via Lanczos's method.

Once α is set as the right-hand side of (16), the matrix $I_q - \alpha H^\top H$ and the vector $H^\top y = \overline{B}^\top \overline{Q}\,\overline{A}\,x_0$ in (14) can be precomputed before the iterations of the algorithm. We stop the algorithm whenever a maximum number of iterations $k_{\max}$ is reached or when $|f(u^{k+1}) - f(u^k)| < \epsilon$.

B. Approximate Optimizer Generator

We now describe a proof-of-concept infrastructure that generates an approximate optimizer for (5) on a reconfigurable device. Fig. 1 shows the workflow of the proposed infrastructure. The optimizer kernel is a user-defined C++ function based on an approximate linear algebra library, which defines basic algebra operations, such as addition, multiplication, inversion, and decomposition, for matrices and vectors. We developed an algebra library for arbitrary precision representations. Fig. 2 shows an example of the kernel using the library with metadata types in C++ template.

[1]In this section, we omit the overline notation for simplicity.

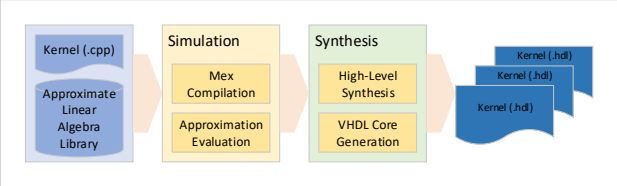


Fig. 1: Infrastructure of the approximate optimizer generator.

```
#include "approxlib.hpp"
... ...
Template<class T, int D1>
void kernel(T input1[D1][D1], T input2[D1],
            T output[D1]){
... ...
mat_vec_mul<T,D1>(... ...);
vec_sub<T,D1>(... ...);
... ...
}
```

Fig. 2: Example of a user-defined kernel.

We solved Lasso MPC under different precisions via an integrated environment with the Matlab MEX API. By compiling the MEX files with the proposed kernel, we can evaluate the functionality and algorithmic performance of the algorithm. After checking the correctness of the kernel via functionality simulation, the kernel is synthesized using high-level synthesis tools, and the approximate core is generated in VHDL.

IV. Experiments

We now describe experiments using Lasso MPC to control the attitude of a spacecraft [14] by ACADO [15]. The dynamic matrices A and B in (1) and the cost matrices P, Q, and R in (3) were set exactly as in [14]. There are four inputs, τ_1, τ_2, τ_3, and τ_4, which correspond to servo controllers, and seven state variables: roll, pitch, yaw, ω_ω, ω_1, ω_2, and ω_3, corresponding to orientation, rotation, and their speed.

Experimental setup. We considered three different time horizons T: 1, 5, and 10. To assess our implementation of PGD, we compared its solution with the one returned by CVX [16], and used the mean squared error (MSE) as the performance metric.

A. RP Approximation Performance

We implemented the framework described in section III-B under different number representations, each with its own precision. Fig. 3 shows the baseline online simulation using CVX as the Lasso MPC solver while different precision evaluation are performed in (a),(b),(c).The single precision floating point is considered as the baseline for all the comparisons. Notice that only the horizon length 10 is shown in this section due to

limited space, however there are similar process for horizon length with 1 and 5.

(a) Floating Point (FP): In Figure 3 the RP floating point is presented for lean PGD from 6-bit up to 16 bits. As shown, lean PGD with 8 to 10 bits floating point are not comparable to single precision float due to the large MSE while from 12 bits RP floating point similar MSE happens compared to single precision float. Hence, 12 bits floating point is considered as the candidate for RP approximate lean PGD.

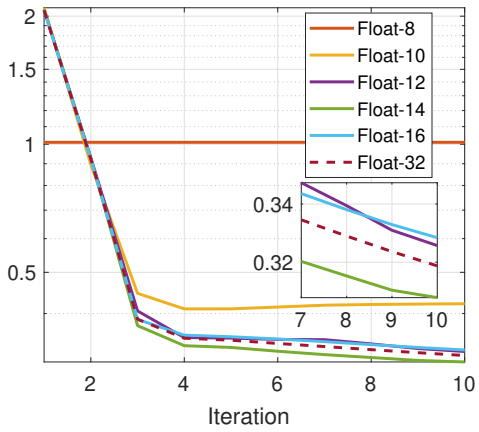


Fig. 3: Floating Point Approximation

(b) Fixed Point (FXP): In Figure 4, the RP fixed point is presented for lean PGD from 20-bit up to 32 bits. As shown, lean PGD with 20 to 24 bits fixed point cannot converge while similar MSE performance happens from 32 bits fixed point. Hence, 32 bits fixed point is considered as the candidate for RP approximate lean PGD in this case.

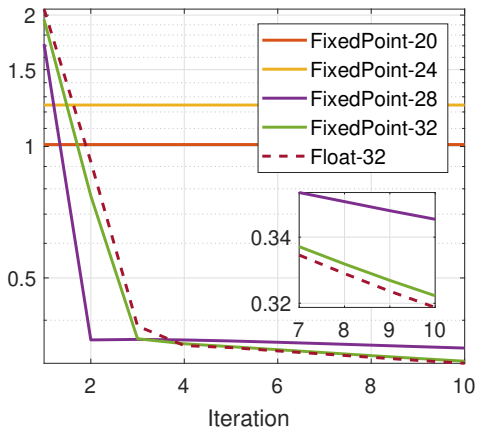


Fig. 4: Fixed Point Approximation

(c) Unum Posit (UP): In Figure 5, the RP unum posit is presneted for lean PGD from 8 to 16 bits. Compared to the single precision float, similar performance come up from 14 bits which makes it the candidate of RP approximate lean PGD as well.

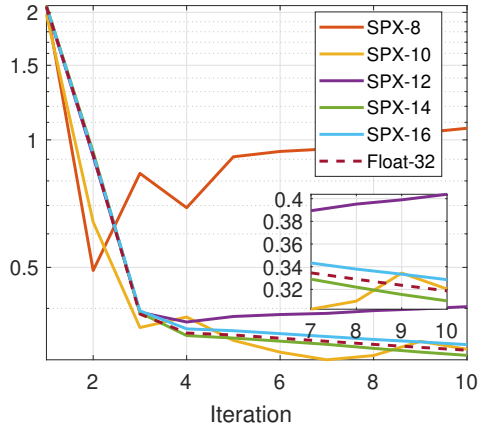


Fig. 5: Unum Posit Approximation

The LASSO MPC simulation is performed using the selected approximate solver to minimize the cost function. Figure 6 shows the comparison of the input and output states of control process in an open-loop simulation for 2s with 0.1s step interval.

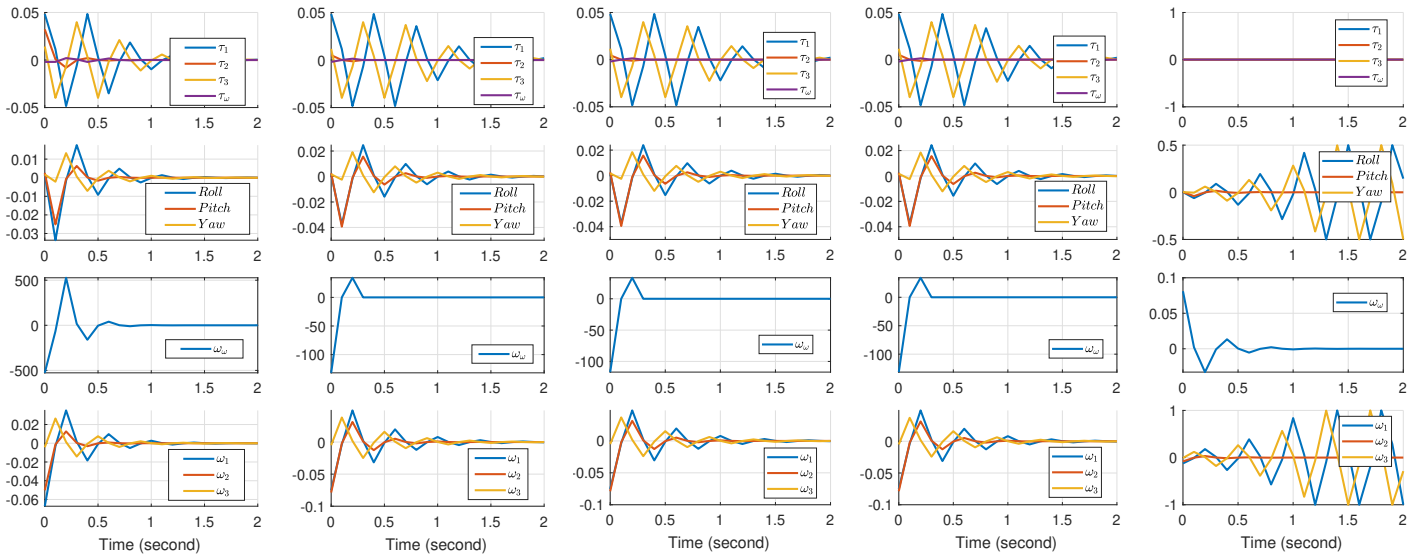
As shown in Fig. 6b-6d, compared with spacecraft attitude control with CVX solver of single precision i Fig. 6a, the approximate solver shows the similar performance across time. However, the approximation with over-reduced precision will cause unstable control outcome as shown in Fig. 6e.

B. Cost Evaluation

By choosing the closest MPC performance of each RP approximation across different precisions as shown in last section, the corresponding costs are evaluated by implementing the lean PGD kernel on Xilinx Ultrascale+ ZCU106 device as shown in Table I. The single precision implemetation are considered as the baseline while the others are compared to it correspondingly. All cost are estimated from high level synthesis using Xilinx Vivado design tool 2019.2 while

Across different horizon length, the approximate fixed point implementation consumes the least logic area, which is about 30% of baseline implementation with single precision floating point. Accordingly, the power consumption is reduced up to 25% when horizon length is 1 while less than half of power is reduced when horizon length is 5 or 10. As the problem size increases from horizon length 1 to 10, the number of bits for fixed point arithmetic is growing from 24 to 32 to maintain the similar performance to single precision implementation. This is due to the fact that dynamical numerical range of linear algebra computations is larger as the size increases.

The approximate floating point implementation does not save as much logic area as fixed point, while as the horizon length increases up to 40% reduction is achieved. Accordingly, up to 20% power reduction is introduced. However, the least number of bits are adopted compared to fixed point im-



(a) CVX (FP-32) (b) FP-12 (c) FXP-24 (d) UP-14 (e) FP-8

Fig. 6: LASSO MPC Simulation

TABLE I: Cost Comparison

	T=1				T=5				T=10			
Precision	FP-32	FP-12	FXP-24	UP12	FP-32	FP-16	FXP-28	UP14	FP-32	FP-14	FXP-32	UP14
LUT ($\times 10^3$)	3.31	2.99	1.21	19.4	3.17	2.16	1.24	10.9	4.01	2.42	1.46	10.6
DSP48E1	30	0	11	12	20	6	10	4	20	6	16	4
BRAM	0	0	0	0	2	2	4	0	5	5	5	8
Clock (MHz)	482	465	443	393	434	401	403	382	370	384	379	382
T (M Inst/sec)	0.42	0.48	0.46	0.41	0.45	0.41	0.42	0.39	0.38	0.39	0.39	0.39
Power (mW)	273	199	68	248	219	220	110	152	250	254	113	148
Bandwidth	100%	14.06%	56.25%	14.06%	100%	25%	76.56%	19.14%	100%	19.14%	100%	19.14%

plementations enabling significant saving on communication bandwidth, up to 14.06%.

The approximate unum posit is based on the high level synthesis of SoftPosit [**?**]. The significant larger logic area is consumed compared to floating point and fixed point implementation, although similar saving on communication bandwidth due to the low number of bits representation.

Hence, if power and area are the most concern on a resource constrained system, the fixed point lean PGD does provide significant reduction compared to single precision. However, if the communication is the most concern on a resource constrained system, such as a mesh network with edge devices, the RP floating point does show the advantage on the bandwidth savings.

V. Conclusion

In this work, a Lean Proximal Gradient Descent algorithm is proposed with simplified procedure of descending step for solving LASSO MPC control problem. The evidence shows a similar performance to the original standard Proximal Gradient Descent approach. By adopting the reduced precision technique, a approximate core generator is developed to fast prototyping the proximal kernel on reconfigurable device, FPGA. The results show up to 60% logic cost reduction, 80% memory bandwidth saving, and 70% power reduction, which is very promising for deploying LASSO MPC on resource constrained system. The future works include further exploring the approximation effects against online MPC applications as well as different optimization solvers.

References

[1] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," Control Engineering Practice, vol. 11, no. 7, pp. 733 – 764, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066102001867

[2] M. Gallieri and J. M. Maciejowski, "Lasso MPC: Smart regulation of over-actuated systems," in 2012 American Control Conference (ACC), 2012, pp. 1217–1222.

[3] H. B. Barua and K. C. Mondal, "Approximate Computing: A Survey of Recent Trends–Bringing Greenness to Computing and Communication," Journal of The Institution of Engineers (India):

Series B, vol. 100, no. 6, pp. 619–626, 2019. [Online]. Available: https://doi.org/10.1007/s40031-019-00418-8

[4] A. Agrawal, J. Choi, K. Gopalakrishnan, S. Gupta, R. Nair, J. Oh, D. A. Prener, S. Shukla, V. Srinivasan, and Z. Sura, "Approximate computing: Challenges and opportunities," in 2016 IEEE International Conference on Rebooting Computing (ICRC), 2016, pp. 1–8.

[5] K. Ling, B.-F. Wu, and J. Maciejowski, "Embedded model predictive control (mpc) using a fpga," IFAC Proceedings Volumes (IFAC-PapersOnline), vol. 17, 01 2008.

[6] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM J. Im. Sc., vol. 2, no. 1, pp. 183–202, 2009.

[7] F. Borrelli, A. Bemporad, and M. Morari, Predictive Control For Linear And Hybrid Systems. Cambridge University Press, 2017.

[8] R. Tibshirani, "Regression shrinkage and selection via the lasso," J. Royal Statistical. Soc., Series B, vol. 58, no. 1, pp. 267–288, 1996.

[9] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefvre, G. Melquiond, N. Revol, and S. Torres, Handbook of Floating-Point Arithmetic, 2nd ed. Birkhäuser Basel, 2018.

[10] G. Flegar, F. Scheidegger, V. Novaković, G. Mariani, A. E. Tom´ s, A. C. I. Malossi, and E. S. Quintana-Ortí, "Floatx: A c++ library for customized floating-point arithmetic," ACM Transactions on Mathematical Software (TOMS), vol. 45, no. 4, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3368086

[11] S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing. USA: California Technical Publishing, 1997.

[12] J. Gustafson and I. Yonemoto, "Beating floating point at its own game: Posit arithmetic," Supercomputing Frontiers and Innovations, vol. 4, pp. 71–86, 01 2017.

[13] A. Beck and M. Teboulle, Convex Optimization in Signal Processing and Communications. Cambridge University Press, 2010, ch. Gradient-based algorithms with applications to signal-recovery problems.

[14] Øyvind Hegrenæs, J. T. Gravdahl, and P. Tøndel, "Spacecraft attitude control using explicit model predictive control," Automatica, vol. 41, no. 12, pp. 2107 – 2114, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109805002657

[15] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," Optimal Control Applications and Methods, vol. 32, no. 3, pp. 298–312, 2011.

[16] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.