Sensitivity-based multistep MPC for embedded systems*

Vryan Gil Palma[1], Andrea Suardi[2]

Abstract— **In model predictive control (MPC), an optimization problem is solved every sampling instant to determine an optimal control for a physical system. We aim to accelerate this procedure for fast systems applications and address the challenge of implementing the resulting MPC scheme on an embedded system with limited computing power. We present the sensitivity-based multistep MPC, a strategy which considerably reduces the computing requirements in terms of floating point operations (FLOPs), compared to a standard MPC formulation, while fulfilling closed-loop performance expectations. We illustrate by applying the method to a DC-DC converter model and show how a designer can optimally trade off closed-loop performance considerations with computing requirements in order to fit the controller into a tightly constraint embedded system.**

I. INTRODUCTION

Over the recent decades, MPC has garnered increased attention as it has proven to be an important tool in control of nonlinear systems in modern technological applications. The optimization problem needed to be solved at each time step results in a high computational expense and computational latency. Computationally costly MPC algorithms used to be implemented using highly powerful computing systems (i.e. server, desktop, industrial PCs) in order to meet real-time requirements. Nowadays, researchers are addressing the challenge to make MPC algorithms less computationally demanding without sacrificing the control performance to cater to systems with fast dynamics.

Fast schemes range from the off-line low-complexity explicit MPC [1] to the on-line strategies (see, e.g., [5], [21]). In addition, further reduction of the computational complexity can be achieved using strategies that uses obtained controls for extended period of time. For instance, the move blocking strategy [4] fixes the control inputs as constant over several time steps while the *multistep MPC* (see, e.g., [8]) uses an open-loop control for several time steps thus reducing the number of optimizations performed. However, these approaches come with the diasadvantage of reduced robustness of the closed-loop solution against perturbations.

An *updated multistep MPC* is introduced in [10] where an update strategy to the multistep MPC based on re-optimizations on shrinking horizons is proposed and analyzed giving a straightforward approach to provide a coping mechanism to counteract the perturbations and enhance controller performance. Robust performance improvements due to re-optimization are rigorously quantified in [10]. Now in this paper, we consider the *sensitivity-based multistep MPC* which is a particular MPC variant that allows further savings in terms of computational load that uses sensitivity analysis in a specific way (see [16] and compare with other MPC strategies that also use sensitivity approach, e.g., [23], [22] and [17].) We show that this sensitivity-based control is a linear approximation of the re-optimization-based control and therefore, the analysis of the updated multistep MPC carries over to the sensitivity-based multistep MPC.

Along with the development of sophisticated algorithms, digital electronics have advanced during the last years. Nowadays, modern embedded systems feature high numerical computing power (e.g. 1GFlops for each core on an ARM Cortex-A9) with low power consumption (<1Watt) and cost ($). This allows the implementation of computationally heavy control schemes for fast dynamical systems at low cost. This provides high performance control techniques to new application domains demanding tight real-time requirements. Still, for a fixed price and/or size of an embedded hardware, which determine/s its capability and limitation, a researcher-designer faces yet a trade-off decision between low computing cost and high performance.

The paper aims to present an MPC controller that fulfills both control performance and low computing complexity requirements and highlight its potential for controller design on embedded computing systems. Based on the setting and basic concepts in Section II, we present various MPC algorithms in Section III. As a case study, the MPC schemes are tested to control a DC-DC converter in Section IV. We show not only is the sensitivity-based control a less costly alternative to re-optimization, we also show how matrix structures can be exploited to obtain the sensitivities much more efficiently. Numerical results and a trade-off analysis on cost and performance are presented in Section V.

II. PRELIMINARY SETUP

Consider the nonlinear discrete time control system

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

where x is the state and u is the control value. The state space X and the control space U are vector spaces and for a given state constraint set $\mathbb{X}$ and control constraint sets $\mathbb{U}$, $x \in \mathbb{X}$ we require $x \in \mathbb{X} \subseteq X$ and $u \in \mathbb{U}(x) \subseteq U$. Let the notation $x_u(\cdot, x_0)$ (or briefly $x_u(\cdot)$) denote the state trajectory steered by control sequence $u(\cdot)$ having initial state x_0. We refer to (1) as the nominal system.

Given a time-dependent feedback law $\mu : \mathbb{X} \times \mathbb{N} \to \mathbb{U}$, we obtain the feedback-controlled system

$$x(k+1) = f(x(k), \mu(x(\tilde{k}), k)) \tag{2}$$

where the state at time instant $k+1$ relies on the state at k and the feedback depending on a certain state at $\tilde{k} \leq k$, where the feedback plays the role of a control for the system. We refer to (2) as the nominal closed-loop system.

The following problem motivates the synthesis of the MPC scheme: find an optimal control in feedback form for the infinite horizon optimal control problem (OCP)

$$\min_{u(\cdot)\in\mathbb{U}^\infty(x_0)} J_\infty\left(x_0, u(\cdot)\right)$$

where the objective function is given by

$$J_\infty\left(x_0, u(\cdot)\right) := \sum_{k=0}^{\infty} \ell\left(x_u(k, x_0), u(k)\right)$$

which is an infinite sum of stage costs $\ell : \mathbb{X} \times \mathbb{U} \to \mathbb{R}_0^+$ along the trajectory with initial value x_0 driven by the

*This research is supported by the European Union under the 7th Framework Programme FP7-PEOPLE-2010-ITN, Grant agreement number 264735-SADCO

[1]V. G. Palma is with the Chair of Applied Mathematics, University of Bayreuth, 95447 Germany vryan.palma@uni-bayreuth.de

[2]A. Suardi is with the Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, UK a.suardi@imperial.ac.uk

control sequence $u(\cdot) \in \mathbb{U}^{\infty}(x_0)$. This type of objective function is often related to feedback stabilization problems, see, e.g., [11], [19]. The objective is minimized over all infinite admissible control sequences $u(\cdot) \in \mathbb{U}^{\infty}(x_0)$. Let the optimal value function be given by

$$V_{\infty}(x_0) := \inf_{u(\cdot)\in\mathbb{U}^{\infty}(x_0)} J_{\infty}\left(x_0, u\right)$$

and the infinite horizon closed-loop performance of a given time-dependent feedback μ be given by

$$J_{\infty}^{\mathrm{cl}}(x_0,\mu) := \sum_{k=0}^{\infty} \ell\left(x_{\mu}(k,x_0), \mu(x_{\mu}(\tilde{k},x_0),k)\right)$$

which is the infinite sum of costs along the trajectory driven by the feedback law. Given an initial state, we aim to find a feedback law μ such that $J_{\infty}^{\mathrm{cl}}(x_0,\mu) = V_{\infty}(x_0)$.

In the general nonlinear setting, however, directly solving this problem is typically difficult. For this reason, we circumvent this problem by considering instead the following finite-horizon minimization problem

$$\min_{u(\cdot)\in\mathbb{U}^{N}(x_0)} J_N\left(x_0, u(\cdot)\right) \qquad \mathcal{P}_N(x_0)$$

for an objective function

$$J_N\left(x_0, u(\cdot)\right) := \sum_{k=0}^{N-1} \ell\left(x_u(k,x_0), u(k)\right)$$

representing a cost associated with an initial state x_0, a control sequence $u(\cdot)$ and optimization horizon N. We minimize over all finite control sequences $u(\cdot) \in \mathbb{U}^N(x_0)$ with N elements. We define the optimal value function associated with the initial state value x_0 by

$$V_N(x_0) := \inf_{u(\cdot)\in\mathbb{U}^{N}(x_0)} J_N\left(x_0, u(\cdot)\right)$$

In our discussion, we assume there exists a control sequence $u^*(\cdot) \in \mathbb{U}^N(x_0)$ for which $V_N(x_0) = J_N\left(x_0, u^*(\cdot)\right)$ where $u^*(\cdot)$ is called the optimal control sequence.

III. MPC ALGORITHMS

To form the feedback law μ, we consider the following receding horizon strategies:

Algorithm 3.1: **(Multistep or m-step MPC)**

(1) Measure the state $x(k) \in \mathbb{X}$ of the system at time instant k

(2) Set $x_0 := x(k)$ and solve the finite horizon problem $\mathcal{P}_N(x_0)$. Let $u^*(\cdot) \in \mathbb{U}^N(x_0)$ denote the optimal control sequence and define the time-dependent m-step MPC feedback

$$\mu_{N,m}(x(k), k+j) := u^*(j), \quad j = 0,\ldots,m-1 \quad (3)$$

(3) Apply the control values $\mu_{N,m}(x(k), k+j)$, $j = 0,\ldots,m-1$, to the system, set $k := k+m$ and go to (1)

If $m = 1$, we recover the standard MPC scheme and by increasing m, optimization is performed less often resulting in a lower computational cost. Algorithm 3.1 gives rise to a feedback law $\mu_{N,m}$ which, under appropriate conditions (see, e.g., [8] or [10]), gives a suboptimal solution to the infinite horizon problem and renders the system asymptotically stable.

We may also consider the updated multistep feedback MPC which, similar to the standard MPC, entails performing optimization every time step, but unlike the standard MPC wherein we perform optimization over full horizon N, we re-optimize over shrinking horizons.

Algorithm 3.2: **(Updated m-step MPC)**

(1) Measure the state $x(k) \in \mathbb{X}$ of the system at time instant k

(2) Set $j := k - \lfloor k \rfloor_m$ where $\lfloor k \rfloor_m$ denotes the largest integer multiple of m less than or equal to k, $x_j := x(k)$ and solve the finite horizon problem $\mathcal{P}_{N-j}(x_j)$. Let $u^*(\cdot) \in \mathbb{U}^N(x_0)$ denote the optimal control sequence and define the updated MPC feedback

$$\hat{\mu}_{N,m}(x(k), k) := u^*(0) \quad (4)$$

(3) Apply the control value $\hat{\mu}_{N,m}(x(k), k)$ to the system, set $k := k+1$ and go to (1)

Remark 3.3: In the nominal setting, due to the dynamic programming principle [2], the feedback-controlled systems (2) generated by $\mu_{N,m}(x(k), k)$ and $\hat{\mu}_{N,m}(x(k), k)$, respectively, coincide. Hence, comparison of both schemes will only be meaningful in the perturbed setting.

We consider the evolution described by the perturbed closed-loop system

$$\tilde{x}_{\mu}(k+1) = f(\tilde{x}_{\mu}(k), \mu(\tilde{x}_{\mu}(\tilde{k}), k)) + d(k)$$

where $d(k)$ represents external perturbations. The presence of disturbance acting on the system brings adverse effects on the performance of Algorithm 3.1 since the measured states, in general, deviate from the predicted states $x_{\mu_{N,m}}(j, x_0), j = 1,\ldots,m-1$ since the controller is not able to counteract this deviation for an extended time duration. The use of Algorithm 3.2 addresses this issue as the updates serve as coping mechanism against the perturbations.

It is shown in [10] that the worsening of the suboptimality performance caused by the perturbations is less prominent when using Algorithm 3.2 compared to the nonupdated case in Algorithm 3.1. Furthermore, a significant improvement in suboptimality performance and stability brought about by the updates through re-optimization becomes more pronounced for systems that are unstable and controllable even for larger perturbations. We refer the readers to [10] for the technical details of the comparison between the m-step and the updated m-step MPC schemes.

Compared to standard MPC, the optimal control problems on shrinking horizon needed for the updates are faster to solve than the optimal control problems on full horizon. However, even if costs are gradually reduced at each step via Algorithm 3.2, optimization still nevertheless needs to be carried out at each iteration. To further save costs, for small perturbations, the updates may also be replaced by approximative updates in which re-optimizations are approximated through sensitivity approach.

Remark 3.3 implies that in the nominal setting, by performing the re-optimization on a shrunken horizon using the current state of the system as the initial value, we recover, as a solution, a tail of the optimal solution obtained from full horizon optimization. The current measured state coincides with the predicted state generated by the full horizon optimal control. In the perturbed setting, using the updated m-step MPC, the current measured state we use as the initial value in the re-optimization on a shrunken horizon can be viewed as perturbation of the predicted value that would have been the initial value had there been no perturbations.

This setting allows for an alternative to re-optimization through the use of sensitivity analysis. This enables the approximation of the solution of the updated m-step MPC with the avoidance of solving all optimization problems on shrunken horizon and hence reducing computational cost. We now consider the MPC variant called **sensitivity-based m-step MPC (SBM MPC)** [16] (based on the sensitivity theorem of Fiacco [6], motivated by sensitivity-based strategies in [3], [13], [18], [23], [22]) for which the only optimizations performed are full-horizon optimizations done only every m steps.

Algorithm 3.4: **(SBM MPC)**

```
Assume for the initial time $k$, $k$ is a multiple of $m$.
(1) measure the state $x(k) \in \mathbb{X}$ of the system at time instant $k$
(2) set $j := k - \lfloor k \rfloor_m$, $x_j^m := x(k)$.
      • If $j = 0$, solve $\mathcal{P}_N(x_0^m)$. Store $u_0^*, \ldots, u_{N-1}^*$ and $x_0^*, \ldots, x_N^*$ representing the optimal control sequence and the optimal trajectory, respectively.
      • Define the time-dependent MPC feedback
          $\overline{\mu}_{N,m}(x(k), k) := u_j^* + \frac{\partial u_j}{\partial p_j}(x_j^*)(x_j^{\mathrm{m}} - x_j^*)$   (5)
(3) apply the control values $\overline{\mu}_{N,m}(x(k), k)$ to the system, set $k := k + 1$ and go to (1)
```

In using Algorithm 3.4, we first apply the obtained u_0^* and then we apply corrections on $u_1^*, u_2^*, \ldots, u_{m-1}^*$. Hence, at time instants $1, 2, \ldots, m-1$, instead of optimizing again (i.e., using SQP active-set strategy) as in the standard MPC, or instead of re-optimizing using shrinking horizons as in the updated m-step MPC, in the hopes of reducing the operation costs, we compute the sensitivities

$$\frac{\partial u_1}{\partial p_1}(x_1^*), \frac{\partial u_2}{\partial p_2}(x_2^*), \ldots, \frac{\partial u_{m-1}}{\partial p_{m-1}}(x_{m-1}^*)$$

from appropriate linear systems to detailed shortly and use them as corrective updates.

Observe that at $j = 0$, $x_0^{\mathrm{m}} = x_0^*$, thus the corrective term $\frac{\partial u_j}{\partial p_j}(x_j^*)(x_j^{\mathrm{m}} - x_j^*)$ vanishes, i.e., no update is performed during the first iteration.

Note that the problem $\mathcal{P}_N(x_0^m)$ includes an initial condition constraint $x_0 = x_0^m$. Now for $j = 1, \ldots, N-1$, the tail $u_j^*, \ldots, u_{N-1}^*$ gives the optimal control sequence for $\mathcal{P}_{N-j}(x_j^*)$ which includes an initial condition constraint $x_j = x_j^*$. Let us consider the general problem $\mathcal{P}_{N-j}(p_j)$, $j = 0, \ldots, N-1$ which includes an initial condition constraint $x_j = p_j$ and let the parameter p_j take the value of measured state x_j^{m}. Taking $\mathcal{P}_{N-j}(x_j^{\mathrm{m}})$, let us denote the resulting optimal control sequence as $u_{j,0}^*, \ldots, u_{j,N-j-1}^*$. Then for $j = 1, \ldots, N-1$, the already available information u_j^* from the nominal solution of the problem $\mathcal{P}_{N-j}(x_j^*)$ and the sensitivity differentials $\dfrac{\partial u_j}{\partial p_j}(x_j^*)$ provide $u_{j,0}^*$, i.e., the first element of the optimal control sequence of the *perturbed* problem $\mathcal{P}_{N-j}(x_j^{\mathrm{m}})$ through

$$u_{j,0}^* = u_j^* + \frac{\partial u_j}{\partial p_j}(x_j^*)(x_j^{\mathrm{m}} - x_j^*) + \mathcal{O}\left(\|x_j^{\mathrm{m}} - x_j^*\|^2\right) \quad (6)$$

$j = 0, \ldots, m-1$. From this we observe that the feedback $\overline{\mu}_{N,m}(x(k), k)$ defined in (5) is a first-order approximation of $\hat{\mu}_{N,m}(x(k), k)$ defined in (4) with an error having an order of magnitude of at most $\|x_j^{\mathrm{m}} - x_j^*\|^2$.

The analysis on the suboptimality performance and stability of the updated m-step MPC carries over to the SBM MPC as presented in great details in [15, Section 6.3] showing that the enhanced robustness induced by performing the shrinking horizon updates (reported in [10]), under certain assumptions, is well-approximated by the sensitivity-based updates.

IV. CASE STUDY: DC-DC CONVERTER

We apply MPC in an electronic circuit process setting. We implement the m-step MPC for a DC-DC converter model motivated by the goal of saving computational costs. We examine a system under perturbation and address the deterioration of performance due to the perturbation by introducing sensitivity-based updates on the controller through the SBM MPC.

A synchronous step-down converter, also referred to as a DC-DC converter, (see modeling in [20] and [7]) is a switching electronic circuit (Figure 1) that converts an input voltage level V_s to satisfy a desired voltage requirement V_o. The setup is comprised of two switches SW_1 and SW_2 cascaded by a second order LC low-pass filter and by an output ohmic load r_0 along with the capacitor C and inductor L internal ESR (r_c) and (r_l).

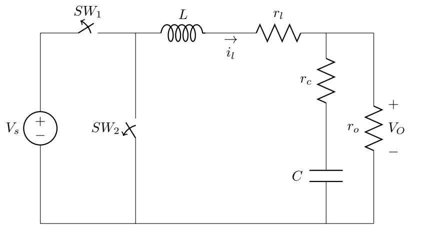

Fig. 1. a DC-DC converter

In this setting, feedback control is used in order to stabilize the output voltage with respect to load, input voltage and component variations. At each switching period T_{SW}, the output voltage and the current flowing in the inductor i_l are measured and used to control the opening and closing time of the two switches. When SW_1 is closed (i.e., at time $d(t) \cdot T_{SW}$, where $d(t) \in [0, 1]$ is the duty cycle), SW_2 is opened and the input power is transferred to the output through the inductor. For the remaining time $(1 - d(t)) \cdot T_{SW}$ of the switching period, the status of the switch are swapped providing a path for the inductor current i_l. This procedure is then repeated.

The described process leads to a set of affine time-invariant continuous-time state-space equations representing the two operating conditions. Defining the state vector as $x(t) := [i_l(t), V_o(t)]^\top$, the system behavior is modeled by

$$\dot{x}(t) = \begin{cases} A_c x(t) + b_c, & kT_s \le t \le (k + d(t))T_s \\ & (SW_1 \text{ is closed}) \\ A_c x(t), & (k + d(t))T_s \le t \le (k+1)T_s \\ & (SW_2 \text{ is closed}) \end{cases} \quad (7)$$

with output voltage given by $V_o(t) := c_c^T x(t)$ and A_c, b_c and c_c given by

$$A_c := \begin{bmatrix} -\frac{r_l}{L} & -\frac{1}{L} \\ \frac{1}{C}\frac{r_o}{r_o+r_c}\left(1 - Cr_c\frac{r_l}{L}\right) & -\frac{1}{C}\frac{1}{r_o+r_c}\left(1 + Cr_c\frac{r_o}{L}\right) \end{bmatrix}$$

$$b_c := \begin{bmatrix} \frac{1}{L} \\ \frac{r_o}{r_o+r_c}\frac{C}{L} \end{bmatrix}, \quad c_c := [0 \ 1]^\top$$

As reported in [20], this hybrid model may not be suitable for control purposes. To address this, a standard state-space averaging method [14] is used resulting in an average continuous-time model that merges the laws of the hybrid model and uses the duty cycle $d(t)$ as an input variable. This gives a nonlinear mathematical model to which linearization around an operating point can be carried out for further simplification of the controller design. This then leads to the state-space average model of the step-down converter (7) given by

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + b_c \cdot d(t) \\ V_o(t) &= c_c^T x(t) \end{aligned} \quad (8)$$

which is a linear system for which the states can be measured straightforwardly. Here, the input is the duty cycle $d(t)$ and the output is the output voltage $V_o(t)$. In addition, constraints arise from the converter topology, e.g., the duty cycle has to be between 0 and 1, and for safety reasons, the inductor current i_l be less than its saturation value i_{lmax}. This therefore implies the need for a controller design that can handle constraints.

A. Design of the controller

We consider the continuous-time finite horizon LQ problem defined by the cost function

$$\begin{aligned} J_c &= x(T)^\top P_c x(T) \\ &\quad + \int_0^T \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^\top \begin{bmatrix} Q_c & 0 \\ 0 & R_c \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} dt \end{aligned} \tag{9}$$

where $Q_c = I$ and $R_c = 1$ have been arbitrarily chosen, P_c is the solution of continuous Ricatti equation and $T = 40\ \mu s$ is the prediction horizon. We assume zero-order hold. The function (9) represents the nominal closed-loop performance of the continuous-time model (8).

B. Discretization

We discretize the continuous-time model (8) and the continuous weighting matrices $\begin{bmatrix} Q_c & 0 \\ 0 & R_c \end{bmatrix}$ in (9) using the sample time T_s and zero-order hold approximation on the input. Let u_k denote the discrete domain counterpart of the input $d(t)$ in (8). Due to sampling, (8) is transformed into

$$x_{k+1} = Ax_k + bu_k$$

where $A := e^{A_c T_s}$, $b := \left(\int_0^{T_s} e^{A_c \tau} d\tau\right) b_c$ and u_k is a constant control between sampling instants. The corresponding sampled-data cost function is given by

$$J_{T_s} = x_N^\top P x_N + \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

where $N = \lceil T/T_s \rceil$ is the number of samples for the prediction horizon T.

C. MPC problem formulation

The MPC problem is defined by the core optimization problem solved at each time instant given by

$$\begin{aligned} \min_{x_k, u_k} \quad & x_N^\top P x_N + \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ \text{s.t.} \quad & x_0 = [\alpha, \beta]^\top \\ & x_{j+1} = Ax_j + bu_j \\ & [0,0]^\top \leq x_{j+1} \leq [i_{\text{lmax}}, V_s]^\top \\ & 0 \leq u_j \leq 1 \\ & j = 0, 1, \ldots, N-1 \end{aligned} \tag{10}$$

We gauge the performance of the algorithm through the closed-loop cost function

$$\begin{aligned} J_{\text{cl}} &= x_{N_T}^\top P x_{N_T} \\ &\quad + \sum_{k=0}^{N_T - 1} \begin{bmatrix} x_k \\ \mu(x_k) \end{bmatrix}^\top \begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix} \begin{bmatrix} x_k \\ \mu(x_k) \end{bmatrix} \end{aligned} \tag{11}$$

for simulation time $N_T = \lceil T_T / T_s \rceil$ where T_T is the simulation time and μ is the MPC feedback (namely, $\mu_{N,m}$ and $\overline{\mu}_{N,m}$.)

D. Matrix structures

Defining the optimization variable

$$\begin{aligned} z := \Big[& x_0^{(1)}\ x_0^{(2)}\ u_0 \mid x_1^{(1)}\ x_1^{(2)}\ u_1 \mid x_2^{(1)}\ x_2^{(2)}\ u_2 \mid \ldots \\ & \ldots \mid x_{N-1}^{(1)}\ x_{N-1}^{(2)}\ u_{N-1} \mid x_N^{(1)}\ x_N^{(2)} \ \Big]^\top \end{aligned}$$

the objective function has the form $\min_z \frac{1}{2} z^\top H z$ where H is block diagonal with N blocks of $\begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix}$ and a block of P. The equality constraints composed of $2 \cdot (N+1)$ equations can be written as

$$\begin{bmatrix} I_2 & & & & & \\ -A & -B & I_2 & & & \\ & & & \ddots & & \\ & & & -A & -B & I_2 \end{bmatrix} z = \begin{bmatrix} \alpha \\ \beta \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

which is of the form $C_{\text{eq}} z = d_{\text{eq}}$. The inequality constraints giving $(2+1) \cdot 2 \cdot N$ inequalities can be written as

$$\left[\begin{array}{cccccc} 0 & 0 & 1 & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \\ \hline 0 & 0 & -1 & & & \\ & & & -1 & & \\ & & & & \ddots & \\ & & & & & -1 \end{array}\right] z + \left[\begin{array}{c} 0 \\ \hline u_{\text{ub}} \\ x_{\text{ub}}^{(1)} \\ x_{\text{ub}}^{(2)} \\ \hline \vdots \\ \hline u_{\text{ub}} \\ x_{\text{ub}}^{(1)} \\ x_{\text{ub}}^{(2)} \end{array}\right] \geq 0$$

which we can write in the form $Cz \leq d$.

This shows that the problem (10) can be written in the form

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^\top H z \\ \text{s.t.} \quad & C_{\text{eq}} z - d_{\text{eq}} = 0 \\ & -Cz + d \geq 0 \end{aligned} \tag{12}$$

which is a QP wherein the constant matrix H happens to be the exact Hessian of the Lagrangian function of (10). Solving the optimization problem (12) is straightforward using `quadprog` in Matlab where active-set method can be chosen to solve the problem.

E. Implementing m-step and SBM MPC

For the standard MPC, at each time instant, we solve the problem (10) (or equivalently (12)) i.e., solve for the optimal solution z^* wherein we obtain the open-loop optimal control u^*. We apply u_0^* to the system and generate the next state. For the next time instant, the current state is measured and assigned as x_0 in (10). Then the process is repeated.

To reduce further the computational cost, we can use the m-step MPC in which we use the first m elements of the optimal control sequence u^*.

Since longer control horizon may reduce robustness due to external perturbations, a remedy for this issue is incorporating a sensitivity strategy through the SBM MPC wherein updates are performed on the entries of the m-step feedback before being injected to the system to generate the next state and the process is repeated to the remaining succeeding entries of the m-step feedback before finally performing the next optimization solving the next NLP problem at time m. In using SBM algorithm, we first apply the obtained u_0^* and instead of optimizing again at time instants $j =$

$1, 2, \ldots, m-1$, we apply corrections on $u_1^*, u_2^*, \ldots, u_{m-1}^*$ using the sensitivity-based update rule (5). It is at the time instant m, where we solve an optimization problem again.

To solve the required updating/correcting sensitivities, we need to construct and solve the systems

$$\begin{bmatrix} \nabla^2_{z^j z^j} \mathcal{L}^j(z^{j*}, \eta^*, x_j{}^*) & \nabla_{z^j} C_{\mathcal{A}^j}(z^{j*}, x_j{}^*) \\ \nabla_{z^j} C_{\mathcal{A}^j}(z^{j*}, x_j{}^*)^\top & 0 \end{bmatrix} \begin{bmatrix} \dfrac{\partial z^j}{\partial p_j}(x_j^*) \\ \dfrac{\partial \eta_{\mathcal{A}^j}}{\partial p_j}(x_j^*) \end{bmatrix}$$

$$= - \begin{bmatrix} \nabla^2_{z^j p_j} \mathcal{L}^j(z^{j*}, \eta^*, x_j{}^*)^\top \\ \nabla_{p_j} C_{\mathcal{A}^j}(z^{j*}, x_j{}^*)^\top \end{bmatrix} \quad (13)$$

for $j = 1, \ldots, m-1$. Consequently, by computing the sensitivities $\dfrac{\partial z^j}{\partial p_j}(x_j^*), j = 1, \ldots, m-1$, we obtain $\dfrac{\partial u^j}{\partial p_j}(x_j^*), j = 1, \ldots, m-1$. If we denote the problem formulation (10) (or (12)) by $\mathcal{P}_N(p_0)$ where $p_0 = [\alpha, \beta]^\top$, computing the sensitivities $\dfrac{\partial u^j}{\partial p_j}(x_j^*), j = 1, \ldots, m-1$ by (13) requires solving a sequence of systems for $j = 1, \ldots, m-1$ corresponding to the OCPs $\mathcal{P}_{N-j}(p_j)$ of decreasing horizon and adjusting parametric value.

It is worth mentioning that in this formulation, due to the nice structure of the matrices resulting from the OCP (10) (i.e., the involved Hessian and Jacobian matrices), adding the fact that these resulting matrices are constant matrices, the sequence of systems (13) can easily and immediately be constructed.

The exact Hessian $\nabla^2_{z^j z^j} \mathcal{L}^j(z^{j*}, \eta^*, x_j^*)$ of the Lagrangian function of $\mathcal{P}_{N-j}(p_j)$ evaluated at $p_j = x_j^*$ has the same form but smaller in size as H (i.e., the corresponding Hessian for $\mathcal{P}_N(p_0)$ with $p_0 = [\alpha, \beta]^\top$). It has $N - j$ blocks of $\begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix}$ and a block of P. The submatrix $\nabla_{z^j} C_{\mathcal{A}^j}(z^{j*}, x_j^*)^\top$ denoting the Jacobian of the active constraints are obtained appropriately from the active constraints of $\mathcal{P}_N(p_0)$ with $p_0 = [\alpha, \beta]^\top$. This shows that the coefficient matrix of the linear system corresponding to the OCP $\mathcal{P}_{N-j}(p_j)$ at $p_j = x_j^*$ can be constructed through the submatrices of the coefficient matrix solved for $P_N(p_0)$ at $p_0 = [\alpha, \beta]^\top$ which is already available. Finally, the right-hand side is a zero matrix except for the identity I_2 appearing in $\nabla_{p_j} C_{\mathcal{A}^j}(z^{j*}, x_j{}^*)^\top$ corresponding to $x_j - p_j$.

V. Case Study Numerical Results

We consider a low-power (2 Watt) step-down converter setup with the following design parameters: $V_s = 6$ V, $r_l = 15.5$ mΩ, $V_o = 1$ V, $i_{lmax} = 4$ A, $r_o = 500$ mΩ, $C = 68$ μF, $L = 1.5$ μH and $r_c = 1.5$ mΩ.

We formulate different m-step and SBM MPC controllers by varying the sampling frequency $f_s \in [300\text{kHz}, 400\text{kHz}]$ (where $f_s := 1/T_s$) and the number of steps $m \in \{1, 2, ...10, 11\}$. Closed-loop simulations are performed in Matlab in order to measure the controller closed-loop performance and the required computing power in terms of FLOPs.

A. Closed-loop performance

For each m-step or SBM MPC scheme, we perform 10^3 simulations of the plant evolution of different initial values (using a set of random and uniformly distributed feasible initial state values) and evaluate the closed-loop function (11). These values are then averaged and assigned to the scheme.

Figure 2 shows the trend of the performance of the algorithm along increasing sampling frequency f_s for varying multistep m both for m-step and SBM MPC. The scheme with $m = 1$ gives the standard MPC where we solve an OCP at every sampling instant. As expected, this gives the best performance where the feedback is able to react to the disturbance at each time step. Also shown is that higher sampling frequency yields better closed-loop performance since faster reaction implies faster disturbance rejection.

Furthermore, the closed-loop performance worsens upon using higher value of m (in solid lines). This is as expected since the system runs in open loop for a longer time causing further propagation of the deviation between the measured and the predicted states. However, improvement is achieved through the use of the sensitivity updates. Unlike the m-step feedback law, SBM MPC uses the perturbation magnitude and the sensitivity information to allow the controller to react to this measured and predicted state deviation. As seen in Figure 2 (in dashed lines), the performance profiles get closer to that of the standard MPC although it is not clear which of the SBM schemes performs the best.

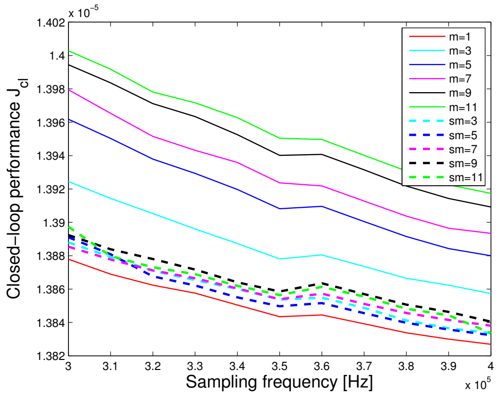

Fig. 2. Performance J_{cl} for varying sampling frequency f_s. The symbol m stands for the number of steps of the m-step MPC while sm for the SBM MPC.

B. Computing Power

Figure 3 shows the trend in the amount of FLOPs of the algorithm along increasing sampling frequency for varying multistep m both for MF and SBM MPC. The standard MPC ($m = 1$) requires the most number of iterations. The number is divided by m as m increases and additional amount is added if sensitivity updates are performed. Note that Figure 3 shows the worst-case scenario FLOPs requirement, i.e., with maximum number of active inequality constraints. In the reality, the number of active constraints is significantly much less than the maximum possible. This means that the FLOPs represented in the dashed lines must be significantly much lower than those represented in the red plot. The SBM MPC requires significantly less computing power compared to standard MPC, but requires more compared to an m-step approach when $m > 1$. In addition, by increasing the sampling frequency f_s, the measured FLOPs increase for any controller. This is related to the discretization step (see Section IV-B) in the sense that increasing f_s means increasing the prediction horizon N and therefore the problem size and computational complexity.

C. Pareto Optimality Analysis

As shown in Figures 2 and 3, the closed-loop performance and computing power requirements are strongly correlated:

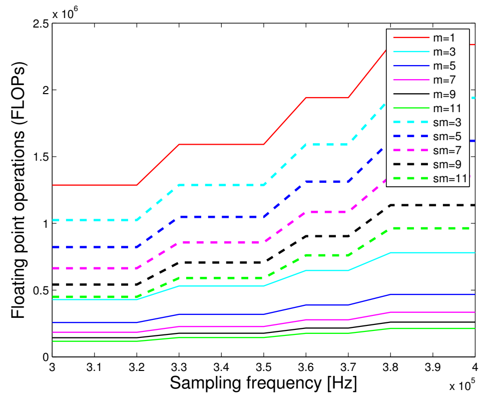

Fig. 3. FLOP for varying sampling frequency f_s and various m-step MPC and sm for the SBM MPC

(i) increasing the sampling frequency f_s and decreasing the number of multistep m lead to controllers with lower J_{cl} (i.e., better closed-loop performance) and higher computing power requirement; (ii) similarly, decreasing f_s and using higher multistep m yield controllers with worse closed-loop performance and limited computing power. This results in the design trade-off between closed-loop performance and computing power. We analyze these trade-offs and present them in terms of Pareto optimality and efficiency (for a single point solution) or compromise solutions (see tutorial in [12]). Figure 4 shows the Pareto frontier, thus the design trade-off between closed-loop performance J_{cl} and computing power in terms of FLOPs. On one extreme, the points in red represent the m-step schemes with higher value of m which we observe to be less computationally demanding algorithms, while on the other extreme is the MPC scheme with $m = 1$ which is the one with the highest computing requirements but with the best closed-loop performance (indicated by the lowest J_{cl}). Moreover, the points in blue represent the SBM MPC schemes which we observe to be the algorithms compromising a 'balance' between the two opposing objectives of having a good algorithm performance and being computationally less demanding. This suggests a great potential for the suitability of the scheme for embedded systems with limited computing power.

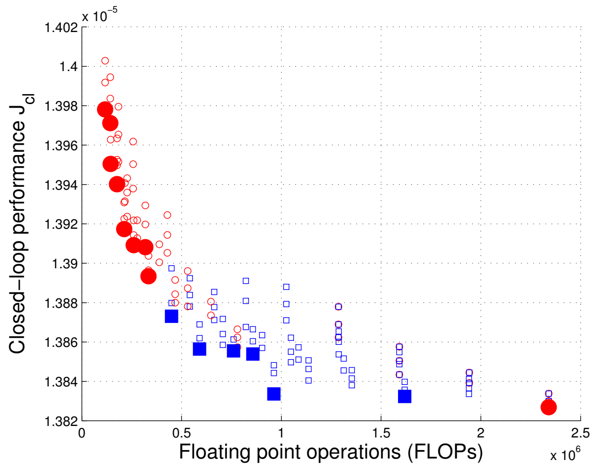

Fig. 4. A Pareto efficiency plot (solid circles and squares forming the Pareto frontier) on a set of feasible options for m-step (red circles) and SBM (blue squares) MPC

VI. CONCLUSION

The SBM MPC, viewed as a less costly approximation of the updated m-step feedback MPC is examined and implemented to control a DC-DC converter. Comparing the standard MPC, m-step ($m > 1$) MPC and SBM MPC schemes, a trade-off analysis, essential for designing and implementing controller on embedded system, is conducted. SBM MPC maintains a compromise between fulfilling control performance and low computational cost requirements.

REFERENCES

[1] A. Bemporad, M. Morari, V. Dua and E.N. Pistikopoulos, *The Explicit Linear Quadratic Regulator for Constrained Systems*, Automatica, 38 (2002), 3–20

[2] D.P. Bertsekas, *Dynamic Programming and Optimal Control. Vol. 1 and 2.* Athena Scientific, Belmont, MA, 1995.

[3] C. Büskens and H. Maurer, *Sensitivity analysis and real-time optimization of parametric nonlinear programming problems*, in M. Grötschel, S. O. Krumke, J. Rambau, eds., Online Optimization of Large Scale Systems, Springer-Verlag, Berlin, 2001, 3–16.

[4] R. Cagienard, P. Grieder, E. Kerrigan and M. Morari, *Move Blocking Strategies in Receding Horizon Control*, Journal of Process Control, 17 (2007), 563–570

[5] H.J. Ferreau, H.G. Bock and M. Diehl, *An online active set strategy for fast parametric quadratic programming in MPC applications*, Proceedings of the IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, Grenoble (2006)

[6] A. V. Fiacco, *Sensitivity analysis for nonlinear programming using penalty methods*, Mathematical Programming, 10 (1976), pp. 287-311.

[7] T. Geyer, G. Papafotiou, R. Frasca, and M. Morari, *Constrained Optimal Control of the Step-Down DC-DC Converter*, IEEE Transactions on Power Electronics, 23 (2008), 2454 –2464

[8] L. Grüne, *Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems*, SIAM Journal on Control and Optimization, 48(2009), 1206–1228.

[9] L. Grüne and V.G. Palma, *On the Benefit of Re-optimization in Optimal Control under Perturbations*, in: Proceedings of the 21st International Symposium on Mathematical Theory of Networks and Systems (MTNS 2014), 2014, 439 - 446 .

[10] L. Grüne and V.G. Palma, *Robustness of performance and stability for multistep and updated multistep MPC schemes*, Preprint, University of Bayreuth, 2014, 30 pages

[11] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*, Springer-Verlag, London, 2011.

[12] R. T. Marler and J. S. Arora, *Survey of multi-objective optimization methods for engineering*, Structural and Multidisciplinary Optimization, 26 (2004), 369–395.

[13] H. Maurer and H.J. Pesch, *Solution Differentiability for Parametric Nonlinear Control Problems with Control-State Constraints*, SIAM Journal on Control and Optimization 32 (1991), 285–309.

[14] R. Middlebrook and S. Cuk, *A general unified approach to modeling switching-converter power stages*, Int. Journal of electronics 42, 6 (1977), 521550.

[15] V. Palma, *Robust Updated MPC Schemes.* PhD thesis, Universität Bayreuth, 2015.

[16] V. Palma and L. Grüne, *Stability, performance and robustness of sensitivity-based multistep feedback NMPC*, Extended Abstract in: Proceedings of the 20th International Symposium on Mathematical Theory of Networks and Systems — MTNS 2012, CD-ROM, Paper No. 68, 4 pages

[17] J. Pannek, J. Michael, M. Gerdts, *A General Framework for Nonlinear Model Predictive Control with Abstract Updates*, arXiv preprint arXiv:1309.1610

[18] H.J. Pesch, *Numerical computation of neighboring optimum feedback control schemes in real-time*, Applied Mathematics and Optimization 5 (1979), 231–252.

[19] J.B. Rawlings and D.Q. Mayne, *Model Predictive Control: Theory and Design.* Nob Hill Publishing, Madison, 2009.

[20] A. Suardi, S. Longo, E. C. Kerrigan and G. A. Constantinides, *Energy-aware MPC co-design for DC-DC converters*, in: Proceedings of European Control Conference 2013, Zürich, 2013, 3608 – 3613

[21] Y. Wang and S. Boyd, *Fast model predictive control using online optimization*, IEEE Transactions on Control Systems Technology, 18 (2010), 267–278

[22] X. Yang, L. T. Biegler, *Advanced-multi-step nonlinear model predictive control*, Journal of Process Control, 23, 7 (2013), 1001 – 1011

[23] V. Zavala and L. Biegler, *The advanced-step NMPC controller: Optimality, stability and robustness*, Automatica, 45 (2009), 86–93