Sensitivity-based multistep feedback MPC: from algorithm to hardware design

Vryan Gil Palma

Chair of Applied Mathematics, University of Bayreuth
joint work with Lars Grüne, Matthias Gerdts, Eric Kerrigan, Andrea Suardi

SADCO Doctoral Days
11 June 2013, ENSTA ParisTech

SEVENTH FRAMEWORK PROGRAMME
MARIE CURIE ACTIONS

ITN SADCO
Initial Training Network
Sensitivity Analysis for Deterministic Controller Design

UNIVERSITÄT BAYREUTH

Contents

- Introduction to MPC
- Motivations for the project
- MPC Problem Formulation + Sensitivity Analysis = SBMF
- Results & Future Work

Model Predictive Control (MPC)

- an algorithm to find $\mu : X \to U$ such that x^* is asymptotically stable for the feedback controlled system

$$x_\mu(n+1) = f(x_\mu(n), \mu(x_\mu(n)))$$

Model Predictive Control (MPC)

- an algorithm to find $\mu : X \to U$ such that x^* is asymptotically stable for the feedback controlled system

$$x_\mu(n+1) = f(x_\mu(n), \mu(x_\mu(n)))$$

- solves an optimal control problem (OCP) every sampling instant to determine a sequence of input moves that controls the system in an (approximately) optimal manner

Model Predictive Control (MPC)

- an algorithm to find $\mu : X \to U$ such that x^* is asymptotically stable for the feedback controlled system

$$x_\mu(n+1) = f(x_\mu(n), \mu(x_\mu(n)))$$

- solves an optimal control problem (OCP) every sampling instant to determine a sequence of input moves that controls the system in an (approximately) optimal manner

- advantages include its capability to handle constraints on variables and the flexibility for the formulation of objective function and process model

Basic MPC Algorithm

At each time t_n, $n = 0, 1, 2, \ldots$

Basic MPC Algorithm

At each time t_n, $n = 0, 1, 2, \ldots$

1. Measure state $x(n)$ of the system.

Basic MPC Algorithm

At each time t_n, $n = 0, 1, 2, \ldots$

1. Measure state $x(n)$ of the system.
2. Set $\tilde{x}_0 = x(n)$ and solve the OCP:

$$\min_{x(t),u(t)} \quad m(x(t_{f_n})) + \int_{t_n}^{t_{f_n}} \ell\,(x(t), u(t))\,dt$$

$$\text{subject to} \quad \begin{aligned} &\dot{x}(t) = f(x(t), u(t)), \quad x(t_n) = \tilde{x}_0 \\ &g(x(t), u(t)) = 0 \\ &h(x(t), u(t)) \geq 0 \end{aligned}$$

Basic MPC Algorithm

At each time t_n, $n = 0, 1, 2, \ldots$

1. Measure state $x(n)$ of the system.
2. Set $\tilde{x}_0 = x(n)$ and solve the OCP:

$$\min_{\substack{x0,\ldots,x_N,\\ u_0,\ldots,u_{N-1}}} \quad m(x_N) + \sum_{k=0}^{N-1} \ell_d\,(x_k, u_k)$$

subject to

$$x_{k+1} = f_d(x_k, u_k), \quad x_0 = \tilde{x}_0$$
$$g_d(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}) = 0$$
$$h_d(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}) \geq 0$$

Basic MPC Algorithm

At each time t_n, $n = 0, 1, 2, \ldots$

1. Measure state $x(n)$ of the system.
2. Set $\tilde{x}_0 = x(n)$ and solve the OCP:

$$\min_{\substack{x0,\ldots,x_N,\\ u_0,\ldots,u_{N-1}}} \quad m(x_N) + \sum_{k=0}^{N-1} \ell_d(x_k, u_k)$$

$$\text{subject to} \quad \begin{aligned} & x_{k+1} = f_d(x_k, u_k), \quad x_0 = \tilde{x}_0 \\ & g_d(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}) = 0 \\ & h_d(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}) \geq 0 \end{aligned}$$

and denote optimal control sequence by $u_0^*, \ldots, u_{N-1}^*$.

Basic MPC Algorithm

At each time t_n, $n = 0, 1, 2, \ldots$

1. Measure state $x(n)$ of the system.
2. Set $\tilde{x}_0 = x(n)$ and solve the OCP:

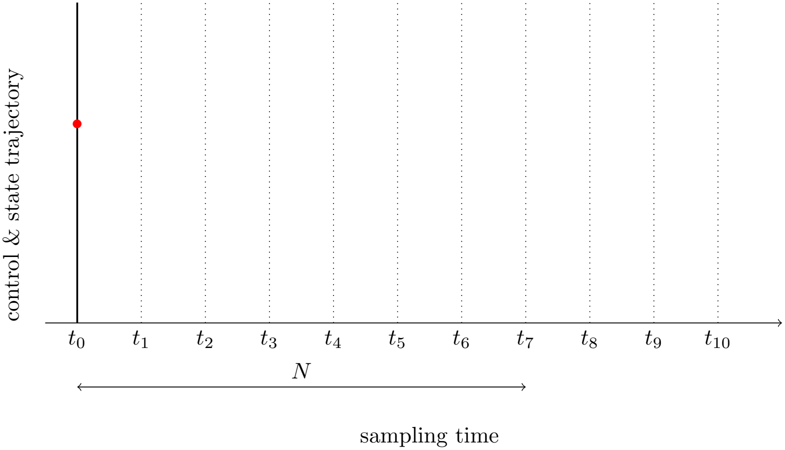
$$\min_{\substack{x0,\ldots,x_N, \\ u_0,\ldots,u_{N-1}}} \quad m(x_N) + \sum_{k=0}^{N-1} \ell_d(x_k, u_k)$$

$$\begin{aligned} \text{subject to} \quad & x_{k+1} = f_d(x_k, u_k), \quad x_0 = \tilde{x}_0 \\ & g_d(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}) = 0 \\ & h_d(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}) \geq 0 \end{aligned}$$

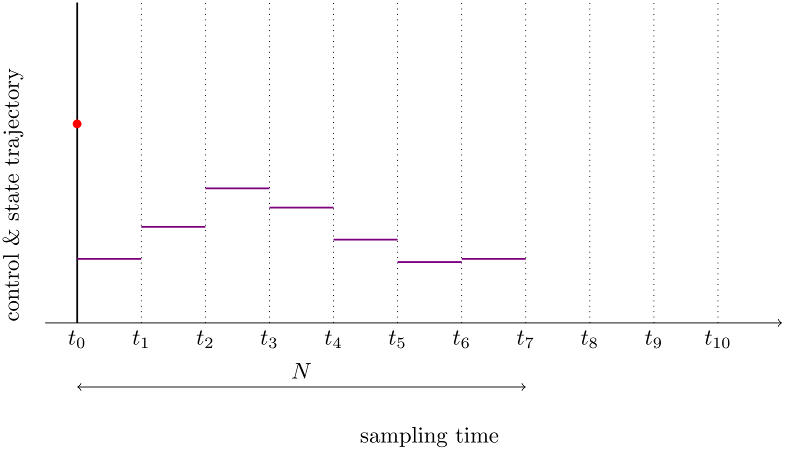
and denote optimal control sequence by $u_0^*, \ldots, u_{N-1}^*$.

3. Define MPC-feedback $\mu_N(x(n)) := u_0^*$ and use this to generate the next state.

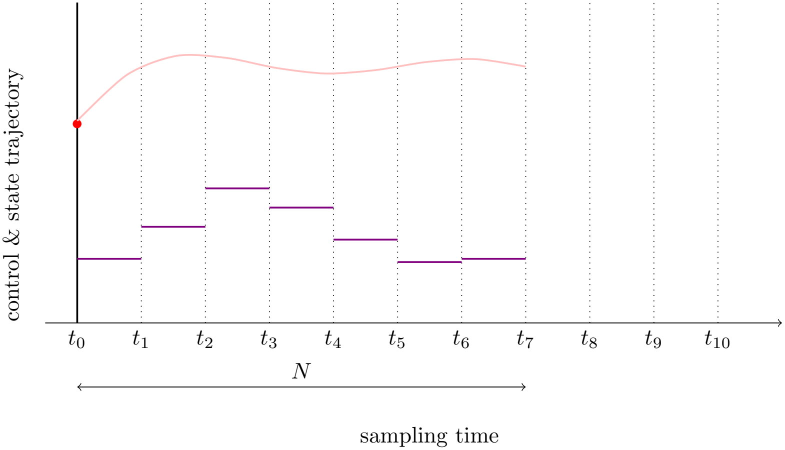
Basic MPC Algorithm



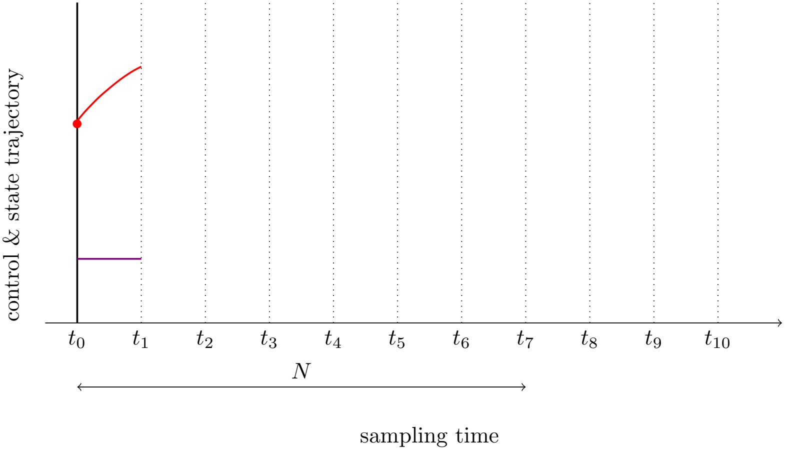
Basic MPC Algorithm



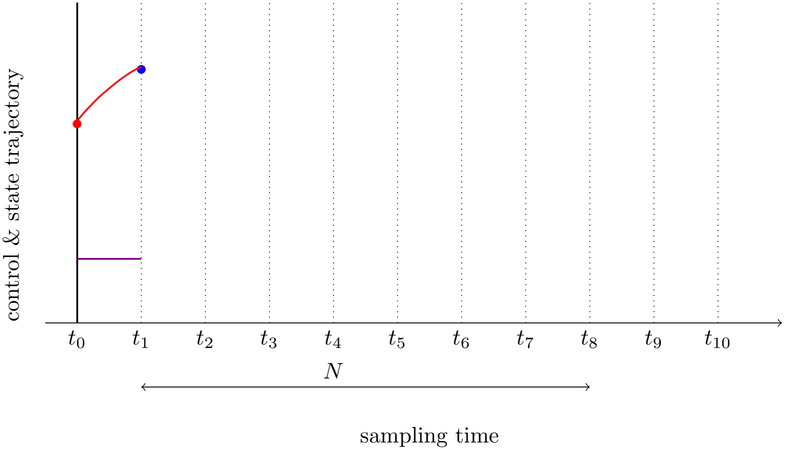
Basic MPC Algorithm



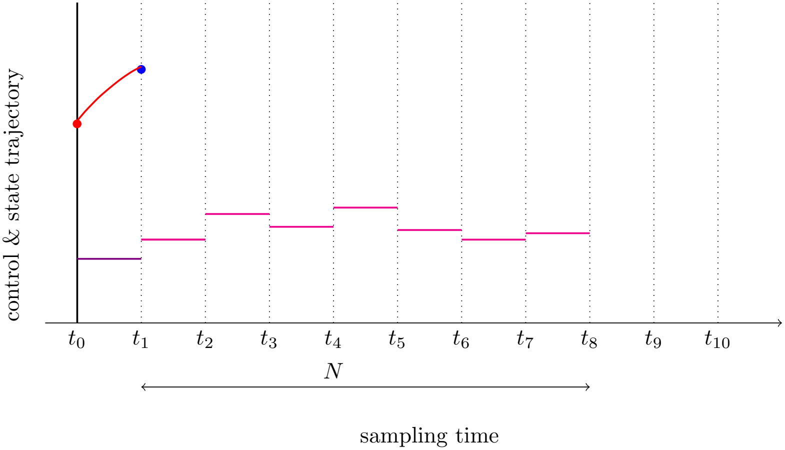
Basic MPC Algorithm



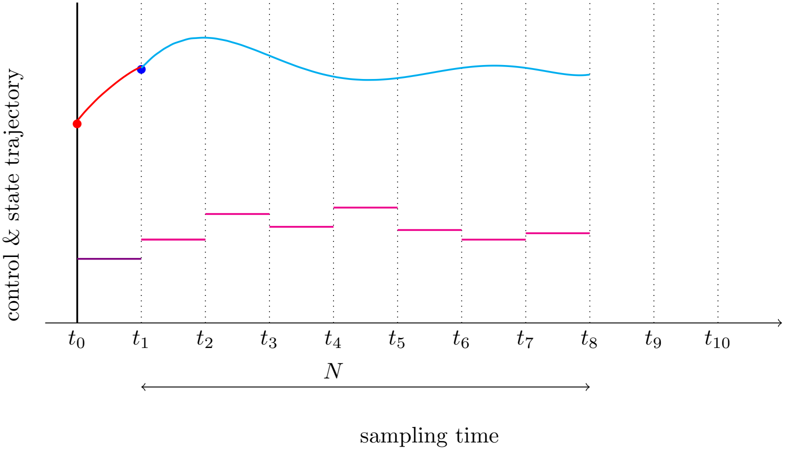
Basic MPC Algorithm



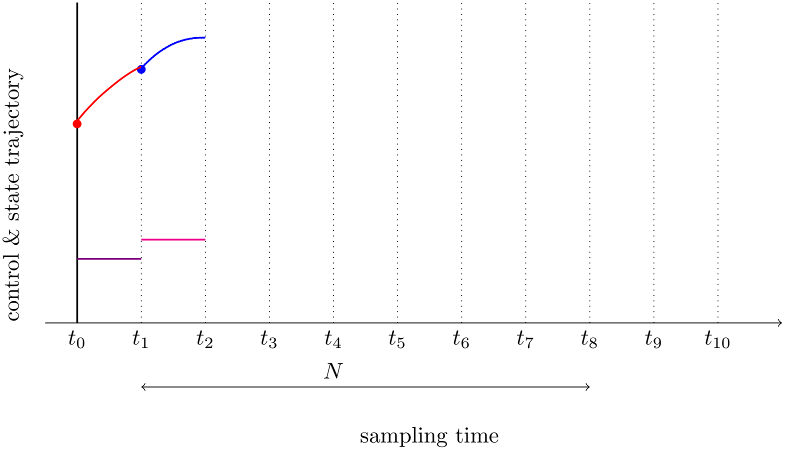
Basic MPC Algorithm



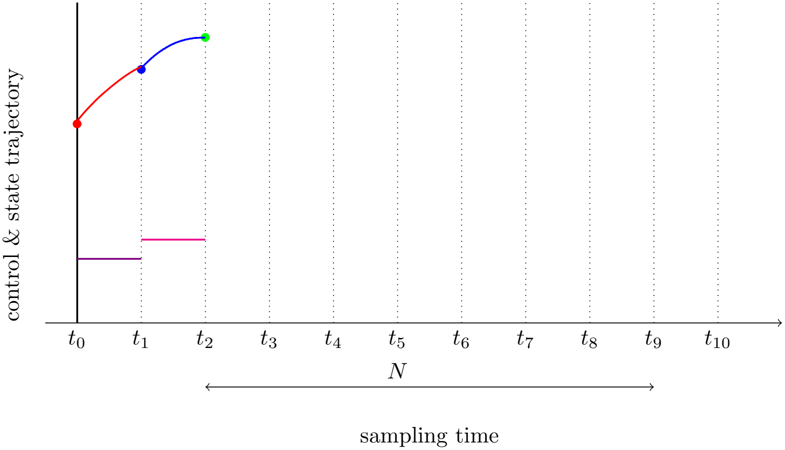
Basic MPC Algorithm



Basic MPC Algorithm



Basic MPC Algorithm



Motivation: Plant and Model 'Closeness'

- MPC - a feedback strategy

feedback if and only if there is uncertainty

---

[1] Åstrom & Murray. *Feedback Systems: An introduction for scientists & engineers*. Princeton, 2008.

Motivation: Plant and Model 'Closeness'

- MPC - a feedback strategy

feedback if and only if there is uncertainty

- a plant vs. mathematical model mismatch!

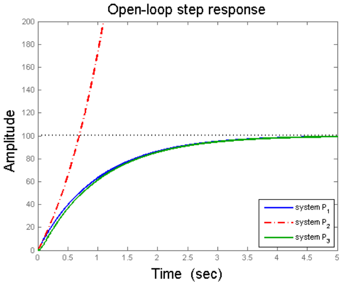
[1] Åstrom & Murray. *Feedback Systems: An introduction for scientists & engineers*. Princeton, 2008.

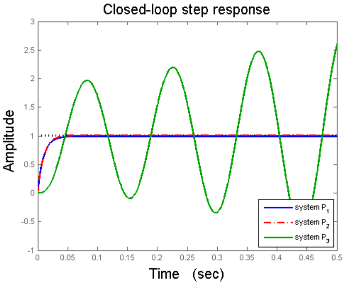
Motivation: Plant and Model 'Closeness'

- MPC - a feedback strategy

feedback if and only if there is uncertainty

- a plant vs. mathematical model mismatch!
- 'closeness' in open-loop $\not\Rightarrow$ 'closeness' in closed-loop[1]

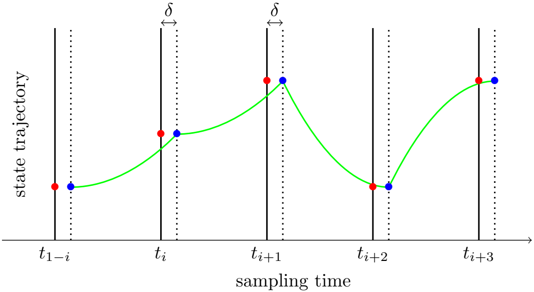




[1] Åstrom & Murray. *Feedback Systems: An introduction for scientists & engineers*. Princeton, 2008.

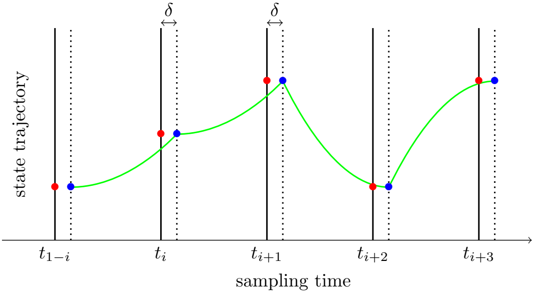
Motivation: Real-time Iteration

- solving an OCP can be very computationally intensive.
- once the current system state is measured, the OCP must be solved *on the fly!*



Motivation: Real-time Iteration

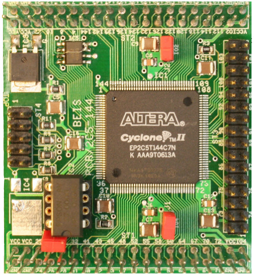
- solving an OCP can be very computationally intensive.
- once the current system state is measured, the OCP must be solved *on the fly!*

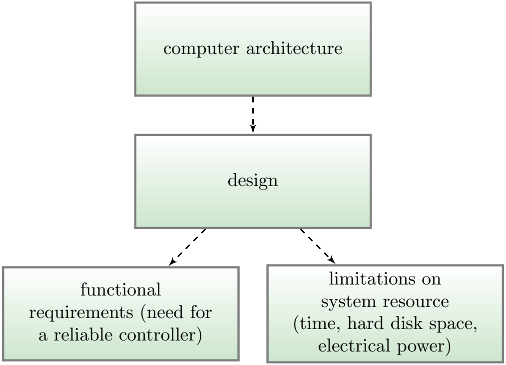


- δ is called the computational delay or latency. How can latency be reduced?

Motivation: Computer Architecture

the development of an algorithm $+$ the design of a machine





Motivation: Computer Architecture

- The power consumed by a CPU obeys the following relation

$$P \sim \alpha \ C \, V^2 \ f$$

Motivation: Computer Architecture

- The power consumed by a CPU obeys the following relation

$$P \sim \alpha \; C \, V^2 \; f$$

Motivation: Computer Architecture

- The power consumed by a CPU obeys the following relation

$$P \sim \alpha \; C \, V^2 \; f$$

power consumption

capacitance

voltage

clock frequency

- How can power consumption be reduced?

Motivation: Computer Architecture

- suppose computing a control action takes K operations

Motivation: Computer Architecture

- suppose computing a control action takes K operations
- the time the computation takes is

$$\frac{K}{\mathsf{IPC} \cdot f}$$

where IPC is the average number of instructions executed per clock cycle across the computation of the control action

Motivation: Computer Architecture

- suppose computing a control action takes K operations
- the time the computation takes is

$$\frac{K}{\mathsf{IPC} \cdot f}$$

  where IPC is the average number of instructions executed per clock cycle across the computation of the control action
- suppose τ is the time required in to compute control action

Motivation: Computer Architecture

- suppose computing a control action takes K operations
- the time the computation takes is

$$\frac{K}{\mathsf{IPC} \cdot f}$$

where IPC is the average number of instructions executed per clock cycle across the computation of the control action
- suppose τ is the time required in to compute control action
- what happens if $\frac{K}{\mathsf{IPC} \cdot f} < \tau$?

Motivation: Computer Architecture

- suppose computing a control action takes K operations
- the time the computation takes is

$$\frac{K}{\mathsf{IPC} \cdot f}$$

  where IPC is the average number of instructions executed per clock cycle across the computation of the control action
- suppose τ is the time required in to compute control action
- what happens if $\frac{K}{\mathsf{IPC} \cdot f} < \tau$? high performance but wastes power!

Motivation: Computer Architecture

- suppose computing a control action takes K operations
- the time the computation takes is

$$\frac{K}{\mathsf{IPC} \cdot f}$$

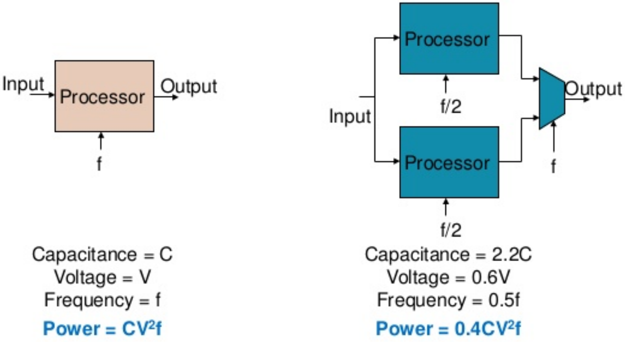
where IPC is the average number of instructions executed per clock cycle across the computation of the control action

- suppose τ is the time required in to compute control action
- what happens if $\dfrac{K}{\mathsf{IPC} \cdot f} < \tau$? high performance but wastes power!
- an ideal frequency $f_{\mathsf{ideal}} = \dfrac{N}{\mathsf{IPC} \cdot \tau}$

lower clock frequency $\implies$ less power consumption

Motivation: Computer Architecture

- A key could be computer architectures that allow for parallelization and pipelining techniques.



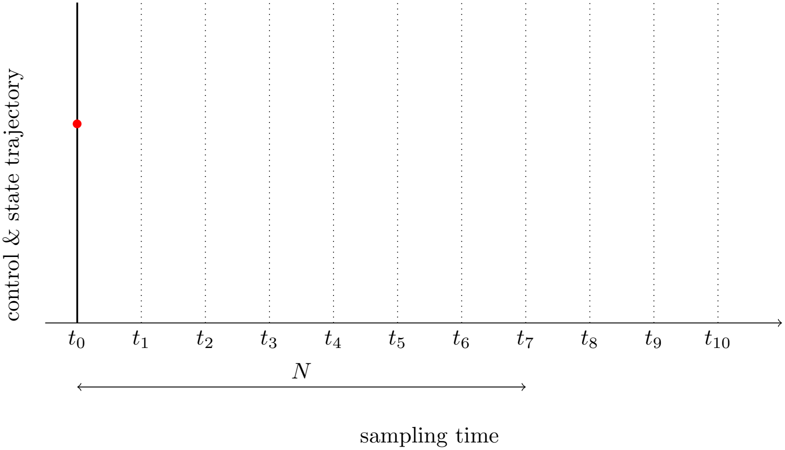
2

[2] Ian Phillips. *Energy Efficient Computing*. 2013

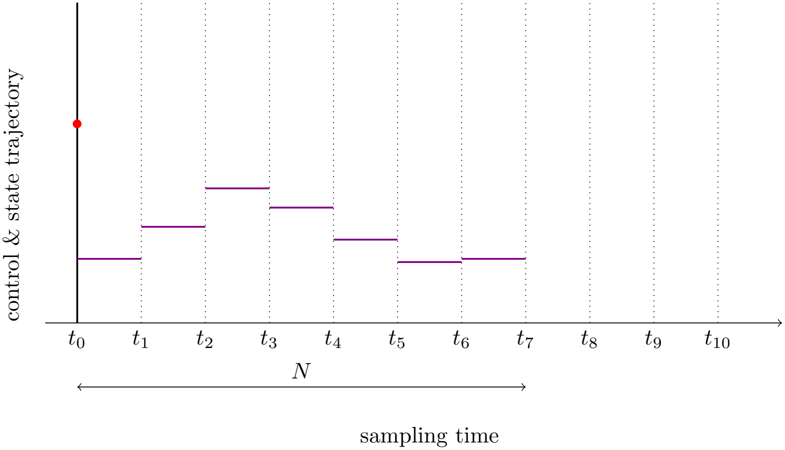
Multistep Feedback (MF) MPC

- a straightforward approach to reduce computational cost
- multistep feedback
- using more than just the first element of the resulting sequence of input moves computed from the OCP at a sampling time

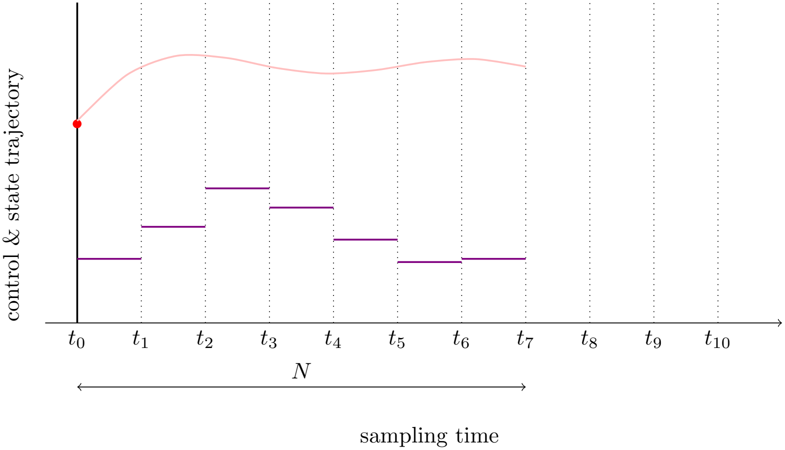
Multistep Feedback (MF) MPC



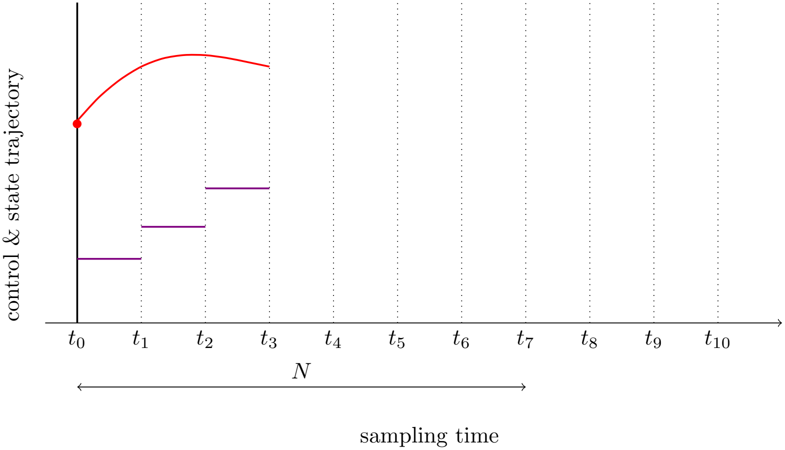
Multistep Feedback (MF) MPC



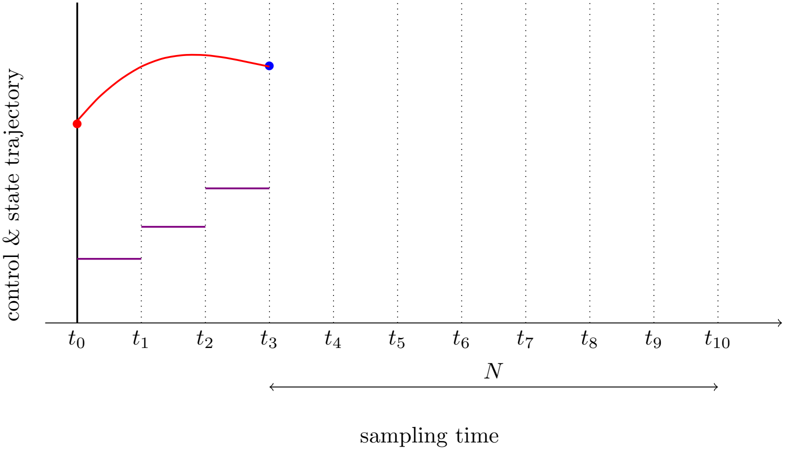
Multistep Feedback (MF) MPC



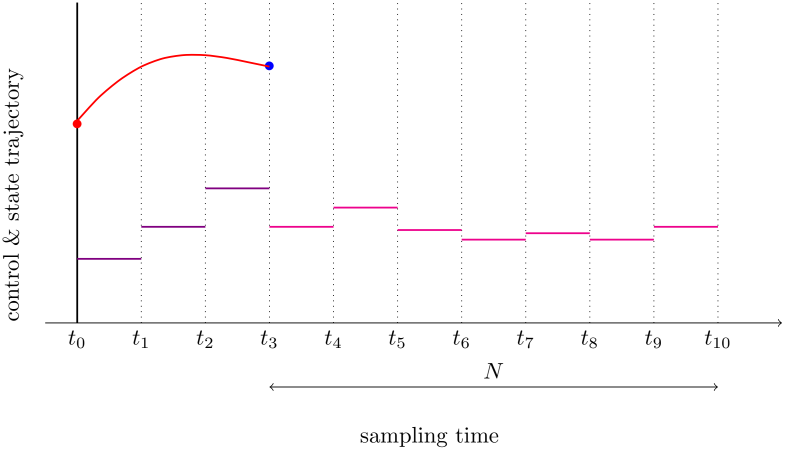
Multistep Feedback (MF) MPC



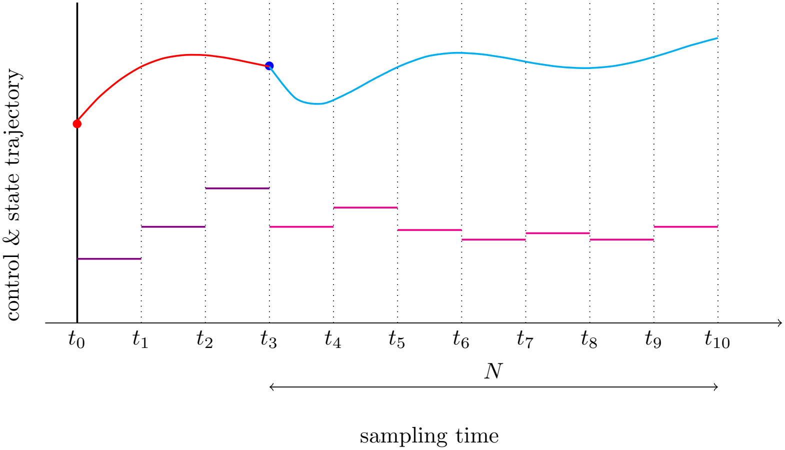
Multistep Feedback (MF) MPC



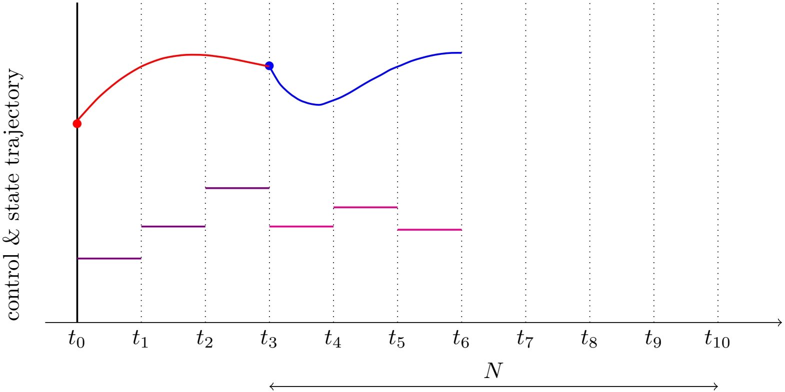
Multistep Feedback (MF) MPC



Multistep Feedback (MF) MPC

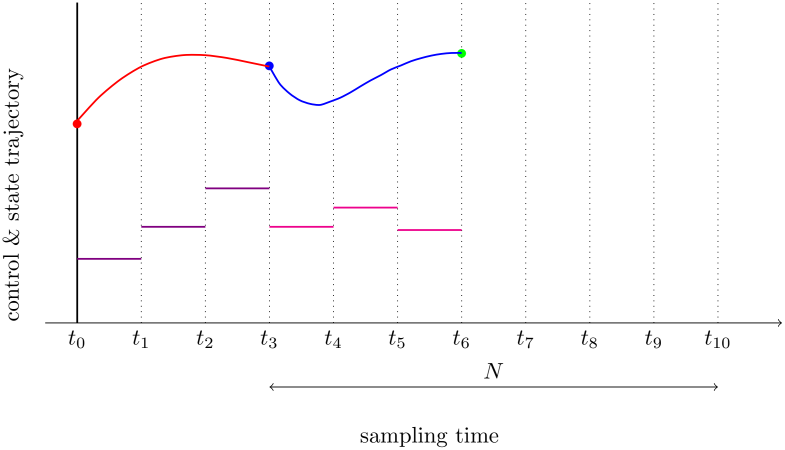


Multistep Feedback (MF) MPC



sampling time

Multistep Feedback (MF) MPC



Multistep Feedback (MF) MPC

- the multistep predictions and control moves would be based on old information
- adverse effects of unmeasured disturbances $\Longrightarrow$ reduced robustness
- How can robustness be improved?
- Attempt to use sensitivities!

Sensitivity-based Multistep Feedback (SBMF) MPC

- a method based on parametric sensitivity analysis of a nonlinear programming problem (NLP) to calculate approximations of optimal solutions of problems depending on neighboring parameter

- Consider the NLP $\mathcal{P}_N(p)$

$$\min_z f(z,p) \quad \text{s.t.} \quad g(z,p) = 0 \qquad h(z,p) \geq 0$$

where p is a given parameter and $z(p)$ is the optimization variable.

Sensitivity-based Multistep Feedback (SBMF) MPC

- NLP Sensitivity Theorem *(Fiacco 1976)*
  Under certain assumptions, for p in the some neighborhood of nominal p_0, it holds that

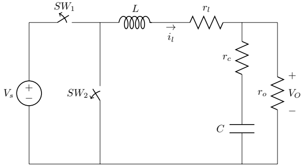
$$\begin{bmatrix} \nabla^2_{zz}\mathcal{L}(z^*,\mu^*,p_0) & \nabla_z C_{\mathcal{A}}(z^*,p_0)^\top \\ \nabla_z C_{\mathcal{A}}(z^*,p_0) & 0 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \dfrac{\partial z}{\partial p}(p_0) \\ \dfrac{\partial \mu_{\mathcal{A}}}{\partial p}(p_0) \end{bmatrix}}_{\text{sensitivity matrix}} = - \begin{bmatrix} \nabla^2_{zp}\mathcal{L}(z^*,\mu^*,p_0)^\top \\ \nabla_p C_{\mathcal{A}}(z^*,p_0)^\top \end{bmatrix}$$

- sensitivities can be used to obtain a linear approximation for the solution of a perturbed OCP

$$\underbrace{u_0^*(x_0)}_{\text{sol of pertubed problem}} = \underbrace{u_0^*(\tilde{x_0})}_{\text{sol of nominal problem}} + \underbrace{\frac{\partial u_0^*}{\partial x_0^*}}_{\text{sensitivity}} \underbrace{(x_0 - \tilde{x_0})}_{\text{perturbation}}$$

An Example Problem: DC-DC Converter

Controller Design



- synchronous stepdown converter - a switching electronic circuit
- consider the LQ problem defined by the cost

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

- continuous-time model

$$\dot{x}(t) = \begin{cases} A_c x(t) + b_c, & kT_s \leq t \leq (k + d(t))T_s \\ A_c x(t), & (k + d(t))T_s \leq t \leq (k+1)T_s \end{cases}$$

An Example Problem: DC-DC Converter

MPC Problem

the core optimization problem solved at each time instant is

$$\min_{x,u} \quad x_N^\top P x_N + \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

$$\begin{array}{rrcll} \text{s.t.} & x_0 & = & [\alpha, \beta]^\top & \\ & x_{j+1} & = & Ax_j + bu_j & j = 0, 1, \ldots, N-1 \\ & [0,0]^\top & \leq & x_{j+1} \leq [i_{l\mathsf{max}}, V_s]^\top & j = 0, 1, \ldots, N-1 \\ & 0 & \leq & u_j \leq 1 & j = 0, 1, \ldots, N-1. \end{array}$$

An Example Problem: DC-DC Converter

Incorporating Sensitivities

- first apply obtained u_0^* and then we apply corrected optimal controls $\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{m-1}$.
- instead of optimizing again at time instants $t_1, t_2, \ldots, t_{m-1}$ (as in single-step MPC), we calculate

$$\frac{\partial u_1^*}{\partial x_1^*}, \frac{\partial u_2^*}{\partial x_2^*}, \ldots, \frac{\partial u_{m-1}^*}{\partial x_{m-1}^*}$$

- SBMF is given by the update rule

$$\tilde{u}_i \quad = \quad u_i^* + \underbrace{\frac{\partial u_i^*}{\partial x_i^*}\left(x_i^{(m)} - x_i^*\right)}_{\text{the update/correction}} \qquad i = 1, \ldots, m-1.$$

- at the time instant t_m, we solve the OCP again

An Example Problem: DC-DC Converter

Incorporating Sensitivities

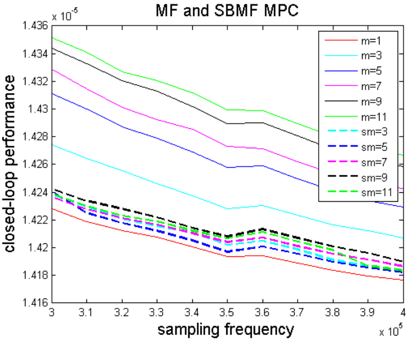
- the sensitivities $\frac{\partial u_1^*}{\partial x_1^*}, \frac{\partial u_2^*}{\partial x_2^*}, \ldots, \frac{\partial u_{m-1}^*}{\partial x_{m-1}^*}$ are computed via

$$\begin{bmatrix} \nabla_{zz}^2 \mathcal{L}_i & \nabla_z C_{\mathcal{A}^i}^\top \\ \nabla_z C_{\mathcal{A}^i} & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial z^i}{\partial x_i}(x_i^*) \\ \frac{\partial \mu_{\mathcal{A}^i}}{\partial x_i}(x_i^*) \end{bmatrix} = - \begin{bmatrix} \nabla_{zp}^2 \mathcal{L}_i^\top \\ \nabla_p C_{\mathcal{A}^i}^\top \end{bmatrix}$$

  for $i = 0, \ldots, m-1$

- solving a sequence of linear systems corresponding to OCPs $\mathcal{P}_{N-i}(x_i^*)$ of decreasing horizon and adjusting parametric initial state value

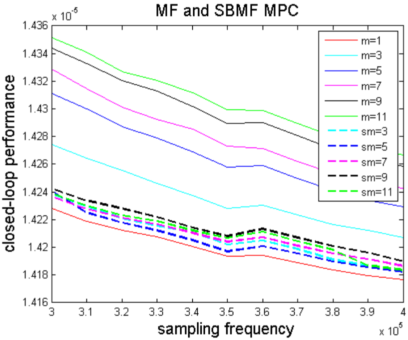
Closed-loop Performance



- $$J_{\mathsf{cl}} = x_{N_T}^\top P x_{N_T} + \sum_{k=0}^{N_T-1} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}^\top \begin{bmatrix} Q & M \\ M^\top & R \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}$$

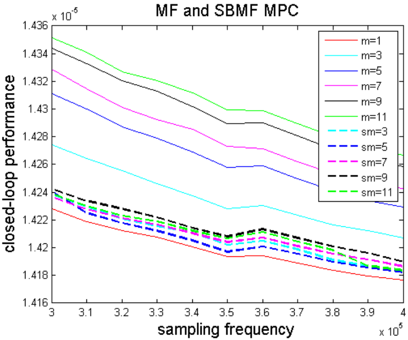
  for discrete simulation time N_T

Closed-loop Performance



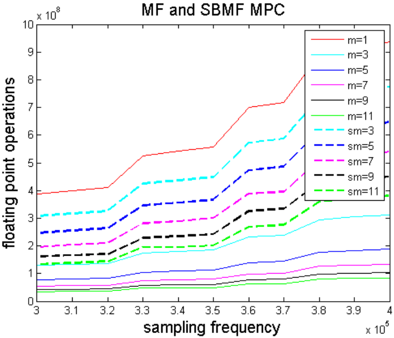
- the performance improves along increasing sampling frequency f_s, suffers by increasing multistep m for MF and improves with SBMF MPC

Closed-loop Performance



- the magnitude of perturbation affects the results/trends

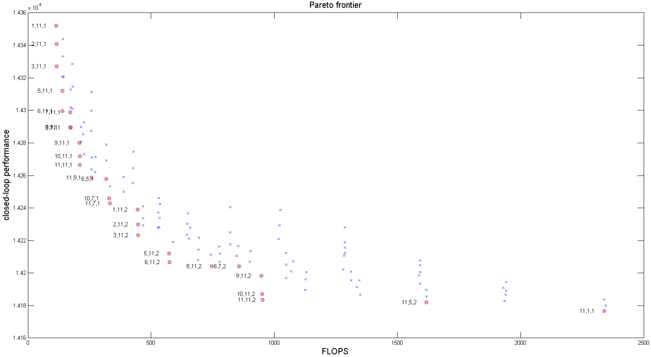
Floating-point Operations (FLOPs)



- for fixed f_s and simulation time length, FLOPs for MF MPC is $\sim \mathcal{O}(N^3)/m$
- for SBMF, some reasonable amount of FLOPs is the cost of better performance

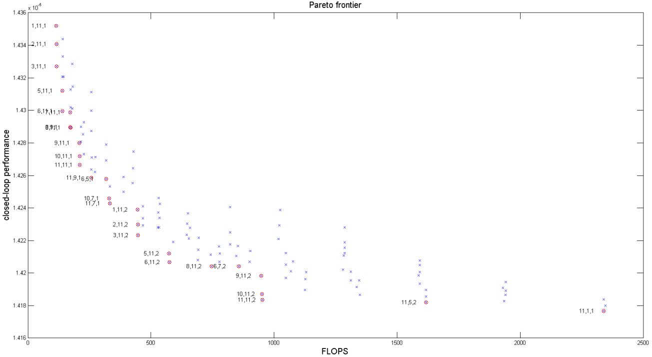
Pareto Efficiency Analysis

- Pareto efficiency is a state of tuning of design parameters in which it is impossible to be better off in a certain criterion without making at being worse off in another criterion



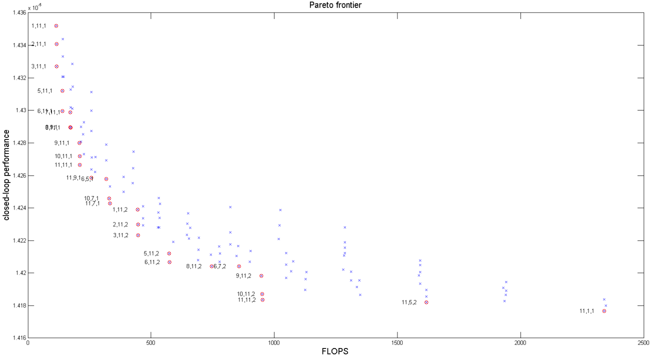
Pareto Efficiency Analysis

- given a set of feasible options, we obtain the Pareto frontier – set of choices that are Pareto efficient



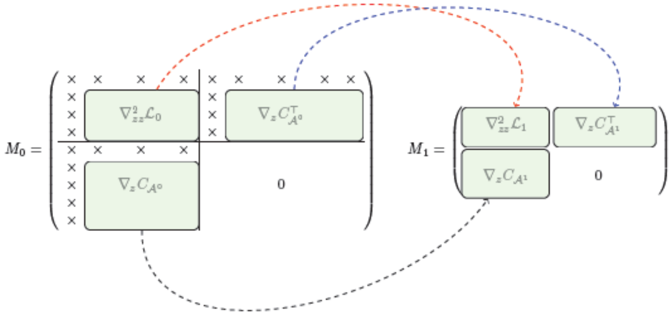
Pareto Efficiency Analysis

- by restricting attention to the set of choices that are Pareto efficient, a control algorithm-hardware designer can analyze and weigh trade-offs within this set



Exploiting Matrix Structures

- The coefficient matrix of the linear system for solving sensitivities for $\mathcal{P}_{N-i}(x_i^*), i = 0, \ldots, m-1$ can be constructed from submatrices of the coefficient matrix solved for $P_N(x_0^*)$
- these are information we get for free!



Conclusion and Outlook

- SBMF shows improvement on performance at a low cost on top of MF MPC
- Algorithm Design + Implementational point of view

Conclusion and Outlook

- SBMF shows improvement on performance at a low cost on top of MF MPC
- Algorithm Design + Implementational point of view

- How can exploiting matrix structures lead to parallelization?

Conclusion and Outlook

- SBMF shows improvement on performance at a low cost on top of MF MPC
- Algorithm Design + Implementational point of view

- How can exploiting matrix structures lead to parallelization?
- How does SBMF MPC become a suitable technique in the real-time setting?

Conclusion and Outlook

- SBMF shows improvement on performance at a low cost on top of MF MPC
- Algorithm Design + Implementational point of view

- How can exploiting matrix structures lead to parallelization?
- How does SBMF MPC become a suitable technique in the real-time setting?
- What is the difference in performance and how much is the extra cost incurred when using nonlinear system dynamics?

Conclusion and Outlook

- SBMF shows improvement on performance at a low cost on top of MF MPC
- Algorithm Design + Implementational point of view

- How can exploiting matrix structures lead to parallelization?
- How does SBMF MPC become a suitable technique in the real-time setting?
- What is the difference in performance and how much is the extra cost incurred when using nonlinear system dynamics?
- Does SBMF, under certain assumptions, yield some kind of asymptotic stability for the system?