NeurIPS2021 Tutorial
Real-Time Optimization for Fast and Complex Control Systems
Part 1

Introduction to Control Systems

Toshiyuki Ohtsuka
Department of Systems Science
Graduate School of Informatics
Kyoto University

Goal of This Tutorial

- To help researchers and engineers in the field of machine learning tackle problems in **control systems**

- Control systems involve **real-time decision making**: a kind of artificial intelligence
- Overview of **control theory** that may be helpful for proper use of machine learning
- Primary focus: **model predictive control (MPC)** based on real-time optimization. MPC can address various control problems beyond such traditional control objectives as regulation and tracking.

Outline

Part 1: Introduction to Control Systems

Part 2: Optimal Control and Model Predictive Control

Part 3: Real-Time Optimization for Model Predictive Control

Part 4: Advanced Topics in Model Predictive Control

Outline of Part 1

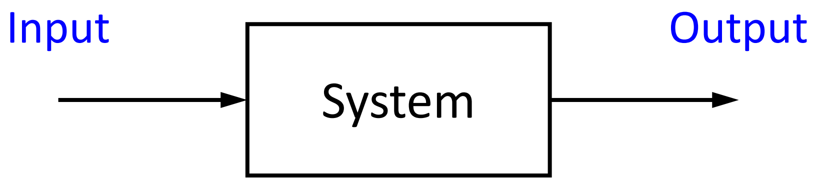
- What is control system?
- Concepts and methods for analysis and design of control systems
  - Mathematical models, modeling, identification, stability, etc.
  - Optimal control, adaptive/learning control, robust control, etc.

What is Control?

- To operate a **system** as desired

What is System?

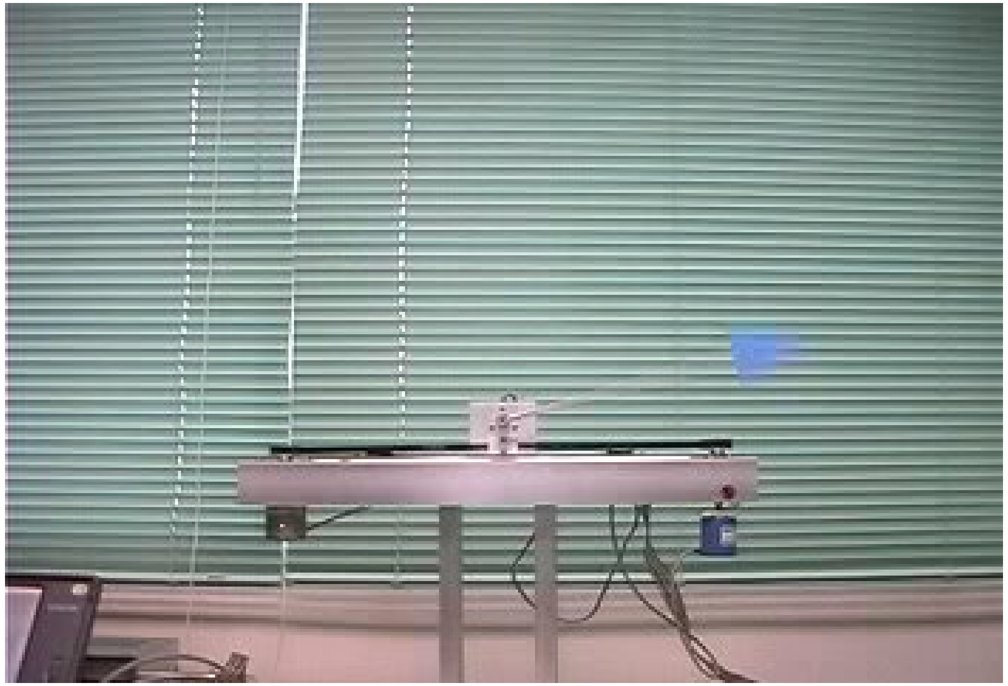
- Something changing dynamically according to inputs



Block Diagram

Input and Output:
Signals (Functions of Time)

Control Systems

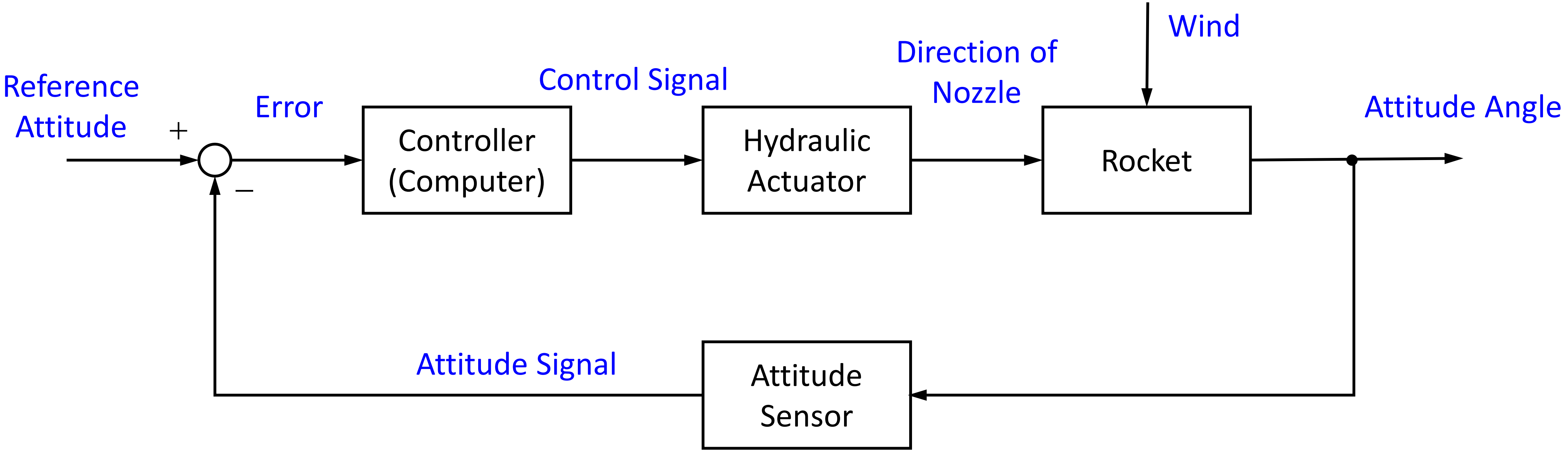


Inverted Pendulum © Toru Asai 2004

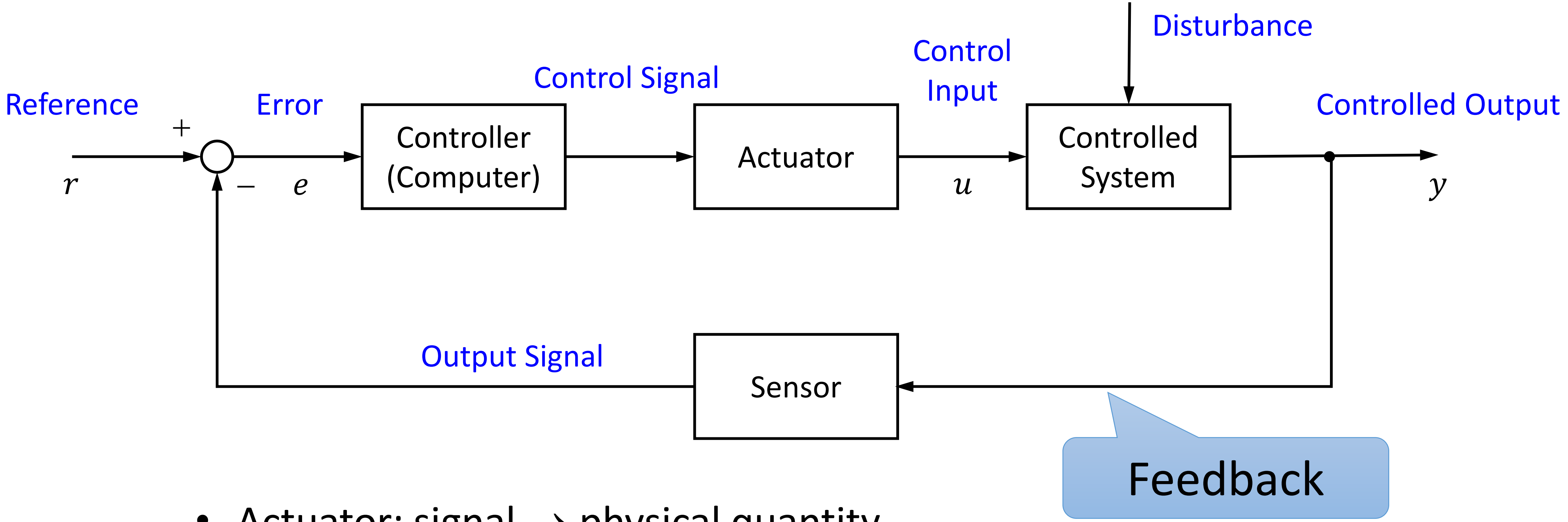
Rocket © JAXA 2014

Systems kept upward by control against gravity

Attitude Control System of a Rocket

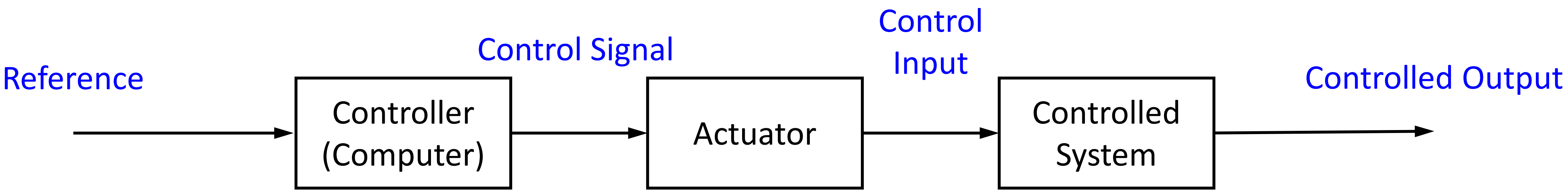


Feedback Control System (Closed-Loop)



- Actuator: signal $\rightarrow$ physical quantity
- Sensor: physical quantity $\rightarrow$ signal
- Actuator/sensor blocks are often omitted.

Feedforward Control System (Open-Loop)



- No sensor
- No disturbance

Control Systems are Everywhere

- Such machines as cars, ships, aircraft, and robots
  - **Inputs:** forces, torques, steering
  - **Outputs:** positions, velocities, directions
- Temperature, environment, economy, and epidemic
  - **Inputs:** heat, gas emissions, monetary policy, mask/vaccine mandate
  - **Outputs:** temperature, atmospheric constituent, money supply, spread rate

Control Engineering

- Methodology to **analyze** and **design** control systems
- Methodology based on mathematical models of control systems: **Control Theory**
- A lot of definitions, **theorems** and proofs: Stability, Controllability, Optimality, etc.

Mathematical Models

- **System:** Mapping from input signal (function of time) to output signal

$$y = P(u), \quad P\text{: Mapping between function spaces}$$

- **Input-Output Model**

$(\dot{\ })$ Time Derivative

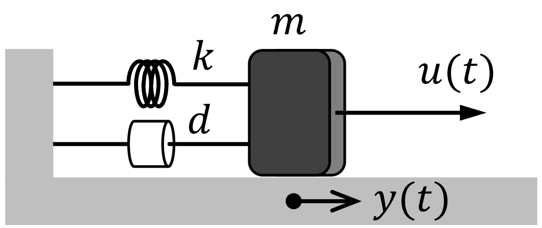
$$y^{(n)}(t) = F\left(y^{(n-1)}(t), \cdots, \dot{y}(t), y(t), u^{(m)}(t), \cdots, \dot{u}(t), u(t)\ , t\right)$$

- **State-Space Representation**

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t), \quad x(t)\text{: Vector of state variables} \\ y(t) &= h(x(t), u(t), t) \end{aligned}$$

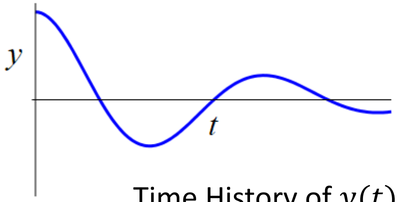
- **Continuous-valued signals**
- Continuous time / Discrete time: differential equations / difference equations
- Stochastic Systems: involves random variables
- Hybrid systems: mixture of continuous dynamics and discrete events

Example: Mass-Spring System



- Input-Output Model ($y(t)$: displacement, $u(t)$: external force)

$$m\ddot{y}(t) + d\dot{y}(t) + ky(t) = u(t)$$

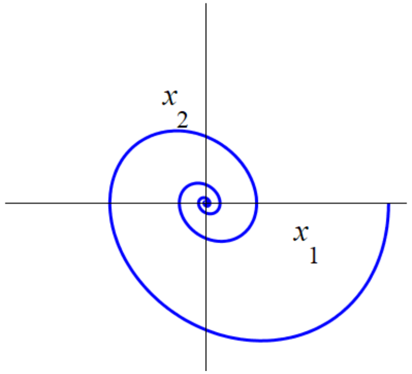


Time History of $y(t)$

- State-Space Representation
  ($x_1(t)$: displacement, $x_2(t)$: velocity, $u(t)$: external force)

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\frac{k}{m}x_1(t) - \frac{d}{m}x_2(t) + \frac{1}{m}u(t) \end{bmatrix}$$

$$y(t) = x_1(t)$$



Trajectory of $\left(x_1(t), x_2(t)\right)$

Linear Time-Invariant (LTI) Systems

- Input-Output Model (Single-Input Single-Output: SISO)

$$a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \cdots + a_1 \dot{y}(t) + a_0 y(t)$$
$$= b_m u^{(m)}(t) + \cdots + b_1 \dot{u}(t) + b_0 u(t)$$

- State-Space Representation (Multiple-Input Multiple-Output: MIMO)

$$\dot{x}(t) = Ax(t) + Bu(t) \quad A, B, C, D\text{: Matrices}$$
$$y(t) = Cx(t) + Du(t)$$

- Transfer Function $y(s) = P(s)u(s)$

$$y(s) = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0} u(s), \quad y(s) = [C(sI - A)^{-1}B + D]u(s)$$

Modeling/Identification

- **Modeling**: Construction of mathematical models based on knowledge
- Model Structures: LTI, Wiener, Hammerstein, Volterra
- Model Transformation: Order Reduction, Structure Simplification
- **Identification**: Construction of mathematical models from data
    - Parametric/Nonparametric
    - Prediction Error Method
    - Subspace Identification
    - Learning Dynamical Systems

L. Ljung: System Identification: Theory for the User, Prentice Hall (1998)
O. Nelles: Nonlinear System Identification, Springer (2001); S. A. Billings: Nonlinear System Identification, Wiley (2013)
K. Fujimoto, J. M. A. Scherpen: Balanced Realization and Model Order Reduction for Nonlinear Systems Based on Singular Value Analysis; SIAM J. Contr. and Optim., 48(7), 4591-4623 (2010)
T. Ohtsuka: Model Structure Simplification of Nonlinear Systems via Immersion; IEEE Trans. Autom. Contr., 50(5), 607-618 (2005)

Analysis

- **Stability**: Input-Output, Lyapunov, Input-to-State
- **Gain**: $||y|| \leq \gamma ||u|| + \beta, \quad ||\cdot||$ norm of a signal
- Passivity, Dissipativity
- LTI: Frequency Response $G(j\omega)$ $(j = \sqrt{-1})$, Bode Plot, Vector Locus
- Controllability/Reachability (Existence of Input Signal for Given Initial/Terminal State)
- Observability (Uniqueness of Initial State for Given Output Signal)
- Invariance of a Set/Manifold (Unreachability, Safety Guarantee)

Stability Analysis

- LTI: Routh/Hurwitz Criterion, Nyquist Criterion, Eigenvalues
- **Lyapunov Function**: Let $x = 0$ be an equilibrium point of $\dot{x}(t) = f(x(t))$. If there is a continuously differentiable function $V(x)$ in a neighborhood D of $x = 0$ such that $V(0) = 0, V(x) > 0$ in $D - \{0\}$ and $\dot{V}(x(t)) < 0$ in $D - \{0\}$ then $x = 0$ is asymptotically stable.
- **Convex Optimization** to Find $V(x)$: Linear Matrix Inequalities (LMI), Sum-of-Squares (SOS) Programming

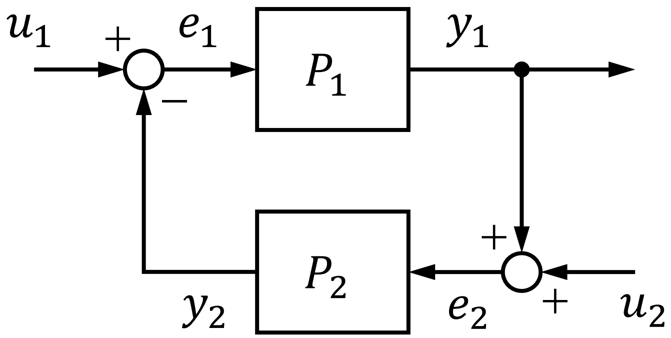
Stability can be checked without solving differential equations!

S. Boyd, et al.: Linear Matrix Inequalities in Systems and Control Theory, SIAM (1994)
D. Henrion, A. Garulli (Eds.): Positive Polynomials in Control, Springer (2005)

Stability Analysis

- **Small Gain Theorem**: Suppose two systems P_1 and P_2 have finite gains γ_1 and γ_2. If $\gamma_1\gamma_2 < 1$ holds then their feedback connection also has a finite gain as a system with input (u_1, u_2) and output (e_1, e_2).
- **Passivity Theorem**: If two systems P_1 and P_2 are passive then their feedback connection is also passive.

Stability can be guaranteed without detailed models!

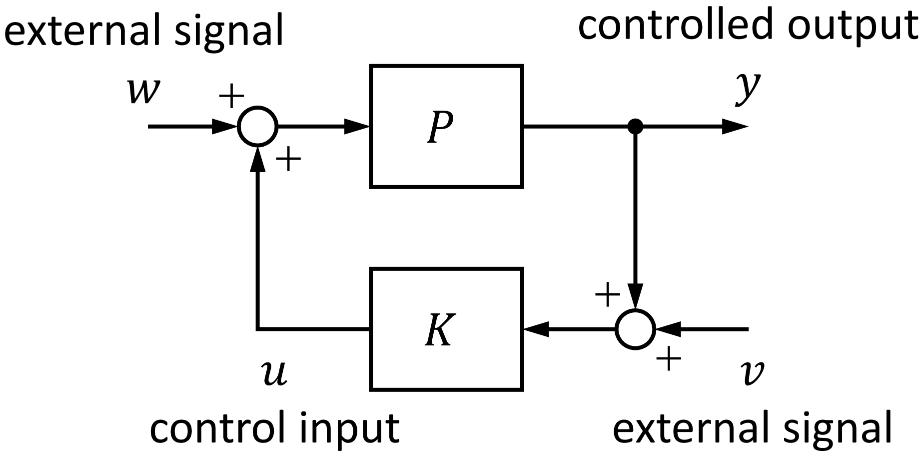


Feedback Connection

H. K. Khalil: Nonlinear Control, Pearson (2015)

Control Design

- For a given system $y = P(u)$, find a controller (a system) $u = K(y)$ so that design specifications are satisfied.
- Not always but often formulated as a constrained optimization problem.



Feedback Control System

Control Design Methods

- PID (Proportional-Integral-Derivative), Loop Shaping
- **State Feedback** (+ State Estimation)
  - Pole Assignment, Control Lyapunov Function
  - Optimal Control
  - Sliding Mode Control
  - Feedback Linearization
- Adaptive Control, Iterative Learning Control
- Robust Control
- Distributed Control

Optimal Control

Find $u(t)$ (**feedforward**) or $u(x,t)$ (**state feedback**) $(0 \le t \le T)$

minimizing $J = \varphi(x(T),T) + \int_0^T L(x(t),u(t),t)dt$

subject to $\dot{x}(t) = f(x(t),u(t),t), \;\; x(0)$ given

$$C(x(t),u(t),t) = 0$$

$$D(x(t),u(t),t) \le 0$$

$$\psi(x(T),T) = 0, \;\; \chi(x(T),T) \le 0$$

Terminal time T can be either given or free.

M. Athans, P. Falb: Optimal Control, McGraw-Hill College (1966)
A. Bryson and Y.-C. Ho: Applied Optimal Control, Routledge (1975)

Adaptive Control, Iterative Learning Control

- **Adaptive Control**: Parameterized controller $u = K(y;\theta)$ and adaptation law to adjust θ

  e.g.) on-line estimation of unknown parameter θ in the system model

- **Iterative Learning Control**: Iteratively update $u(t)$ $(0 \leq t \leq T)$ to achieve perfect tracking based only on tracking error $e(t)$ $(0 \leq t \leq T)$ with almost no prior knowledge on the system

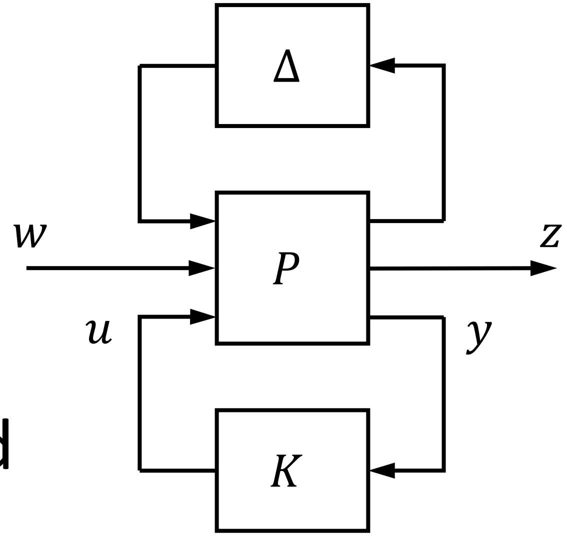
  e.g.) $u_{k+1}(t) = u_k(t) + K\ddot{e}_k(t)$

K. S. Narendra, A. M. Annaswamy: Stable Adaptive Systems, Prentice Hall (1989)
S. Arimoto, et al.: Bettering Operation of Robots by Learning; Journal of Robotic Systems, 1(2), 123-140 (1984)

Robust Control

- **Uncertainty**: System Δ in a unit ball $B = \{\Delta: \|\Delta\| < 1\}$
- Δ is uncertain but deterministic
- **Robust Stabilization**: Find a controller K such that the closed-loop system from w to z is stable for any $\Delta \in B$.
- **Robust Performance**: Find a controller K such that performance specifications from w to z are satisfied for any $\Delta \in B$.
- Basic Tool: Small Gain Theorem

Uncertainty



Feedback Control System with Uncertainty

K. Zhou, J. C. Doyle: Essentials of Robust Control, Prentice Hall (1997)

Distributed Control

- Find a controller K utilizing limited information
$$u_i = K_i\left(y_{j_{i1}}, \cdots, y_{j_{im}}\right)$$
- **Multi-agent system**: Global task (formation, consensus, coverage, etc.) by a set of systems (agents) with distributed control
- **Decentralized Control**: No communication between controllers
$$u_i = K_i(y_i)$$

G. Antonelli: Interconnected Dynamic Systems: An Overview on Distributed Control; IEEE Contr. Sys. Magazine, 33(1), 76-88 (2013)
R. R. Negenborn, J. M. Maestre: Distributed Model Predictive Control: An Overview and Roadmap of Future Research Opportunities; IEEE Contr. Sys. Magazine, 34(4), 87-97 (2014)

Summary

- **Control systems** involve real-time decision making, a kind of artificial intelligence.
- Control systems are everywhere from machines to environment and society.
- **Control theory** provides mathematical tools for analysis and design of control systems.
- **Mathematical models** of systems play crucial roles in control theory. However, there are some nice methods to deal with qualitative characteristics and uncertainties without detailed models.

NeurIPS2021 Tutorial
Real-Time Optimization for Fast and Complex Control Systems
Part 2

Optimal Control and Model Predictive Control

Toshiyuki Ohtsuka
Department of Systems Science
Graduate School of Informatics
Kyoto University

Outline

Outline of Part 2

- Optimal control problem, Euler-Lagrange Equations (ELE), Hamilton-Jacobi-Bellman Equation (HJBE), and numerical solution methods
- Model Predictive Control (MPC): problem formulation and difficulties, computation and stability

Optimal Control Problem

Find $u(t)$ (feedforward) or $u(x,t)$ (state feedback) $(0 \leq t \leq T)$

minimizing $J = \varphi(x(T),T) + \int_0^T L(x(t),u(t),t)dt$

subject to $\dot{x}(t) = f(x(t),u(t),t), \quad x(0)$ given

We omit the constraints except for the state equation for simplicity. They can be handled by penalty functions or barrier functions. The terminal time T is assumed to be given.

M. Athans, P. Falb: Optimal Control, McGraw-Hill College (1966)

A. Bryson and Y.-C. Ho: Applied Optimal Control, Routledge (1975)

Euler-Lagrange Equations (ELE): Stationary Conditions for Optimal Trajectory

$$\dot{x}(t) = f(x(t), u(t), t), \quad x(0) \text{ given}$$
$$\dot{\lambda}(t) = -\nabla_x H(x(t), u(t), \lambda(t), t), \quad \lambda(T) = \nabla_x \varphi(x(T), T)$$
$$\nabla_u H(x(t), u(t), \lambda(t), t) = 0$$

Nonlinear Two-Point Boundary-Value Problem (TPBVP)

where

$H(x, u, \lambda, t) = L(x, u, t) + \lambda^{\mathrm{T}} f(x, u, t)$: Hamiltonian

$\lambda(t)$: costate or adjoint variable

Solution $u(t)$: Candidate of optimal feedforward control for given $x(0)$

Numerical Solution of ELE

- **Iterative search in the space of functions** $u(t)$ (single shooting) or $(x(t), \lambda(t), u(t))$ (multiple shooting)
- Gradient methods: Steepest descent, conjugate gradient method
- Newton's method
- Quasi-Newton method
- Some special structures are exploited
- Time horizon is normally discretized for numerical solution
- Computationally demanding!

A. Bryson and Y.-C. Ho: Applied Optimal Control, Routledge (1975)
J. T. Betts: Practical Methods for Optimal Control Using Nonlinear Programming, 3rd Ed., SIAM (2020)

Hamilton-Jacobi-Bellman Equation (HJBE)

Value Function

$$V(x,t) = \min_{u[t,T]} \left[\varphi(x(T)) + \int_t^T L(x(t),u(t),t)dt \right]$$

HJBE

$$-\nabla_t V(x,t) = \min_u H(x,u,\nabla_x V(x,t),t), \quad V(x,T) = \varphi(x,T)$$

Nonlinear Partial Differential Equation (PDE)

Optimal Control

$$u_{opt}(x,t) = \arg\min_u H(x,u,\nabla_x V(x,t),t)$$

State Feedback!

Solvable Example: Linear Quadratic Control

Linear system: $\dot{x}(t) = Ax(t) + Bu(t)$

Quadratic cost:

$$J = \frac{1}{2}x^{\mathrm{T}}(T)S_f x(T) + \int_0^T \frac{1}{2}\left(x^{\mathrm{T}}(t)Qx(t) + u^{\mathrm{T}}(t)Ru(t)\right)dt$$

Optimal control: $u_{opt}(x,t) = -\underline{R^{-1}B^{\mathrm{T}}S(t)x}$

State Feedback with
a Time-Varying Gain Matrix

Riccati differential equation:

$$-\dot{S}(t) = A^{\mathrm{T}}S(t) + S(t)A - S(t)BR^{-1}B^{\mathrm{T}}S(t) + Q, \quad S(T) = S_f$$

Value function: $V(x,t) = \frac{1}{2}x^{\mathrm{T}}S(t)x$

Costate in ELE: $\lambda(t) = S(t)x(t)$

Numerical Solution of HJBE

- If $V(x,t)$ can be stored for all (x,t) then HJBE can be solved numerically backward in time starting from $V(x,T) = \varphi(x)$
- However, impractical due to explosive growth of computation and storage for high dimensional systems: **curse of dimensionality**
- Approximate solution methods: power series (Al'Brekht '61; Lukes '69), Galerkin method (Beard et al. '97), interpolation (Kreiselmeier&Birkhölzer '94), neural networks (Goh '93), RBF (Huang et al. '06), GPR (Fujimoto et al. '18), etc.

Infinite-Horizon Problem

Stationary solution to infinite-horizon optimal control problem

$$\text{HJE: } H(x, u(x,\lambda), \lambda) = 0, \quad \lambda = \nabla_x V(x)$$

$$u(x,\lambda) = \arg\min_u H(x,u,\lambda)$$

ELE as a Hamiltonian System (Canonical Equations)

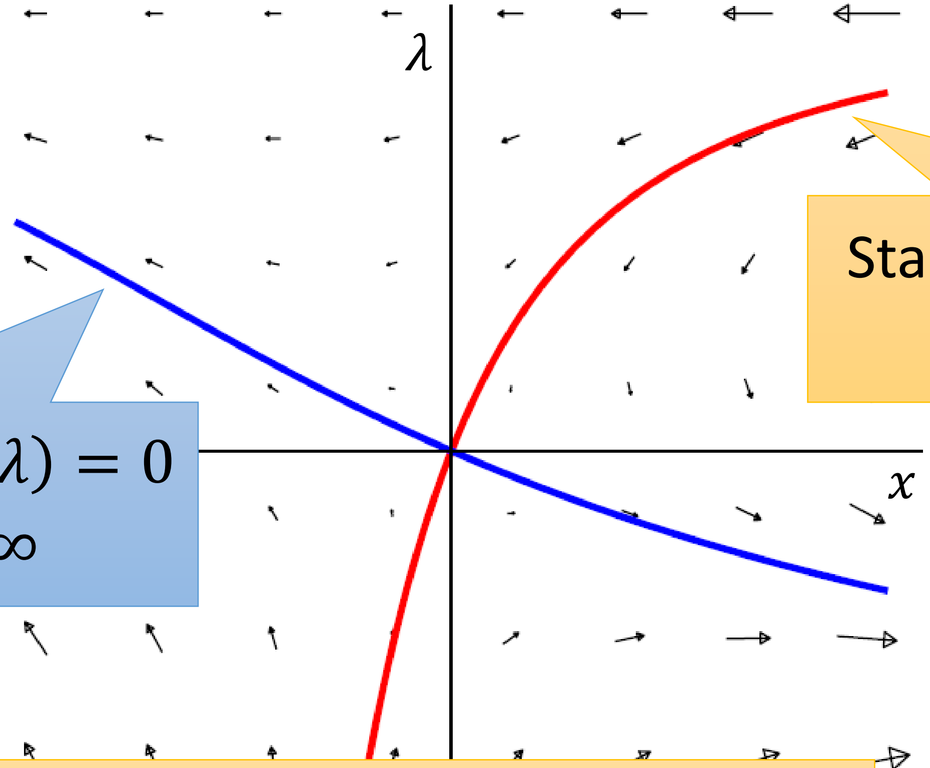
$$\dot{x}(t) = \nabla_\lambda H\big(x(t), \lambda(t)\big)$$

$$\dot{\lambda}(t) = -\nabla_x H(x(t), \lambda(t))$$

where $H(x,\lambda) \coloneqq H(x, u(x,\lambda), \lambda)$

Terminal condition for $\lambda(t)$ $(t \to \infty)$?

Stabile Manifold in Infinite-Horizon Problem



Stable Manifold: $H(x,\lambda) = 0$
and $\big(x(t),\lambda(t)\big) \to 0$

Unstable Manifold: $H(x,\lambda) = 0$
and $\big(x(t),\lambda(t)\big) \to \infty$

Stabilizing solution to infinite-horizon problem
= **Stable manifold** of Hamiltonian system

(van der Schaft '91)

Solution methods to find stable manifold

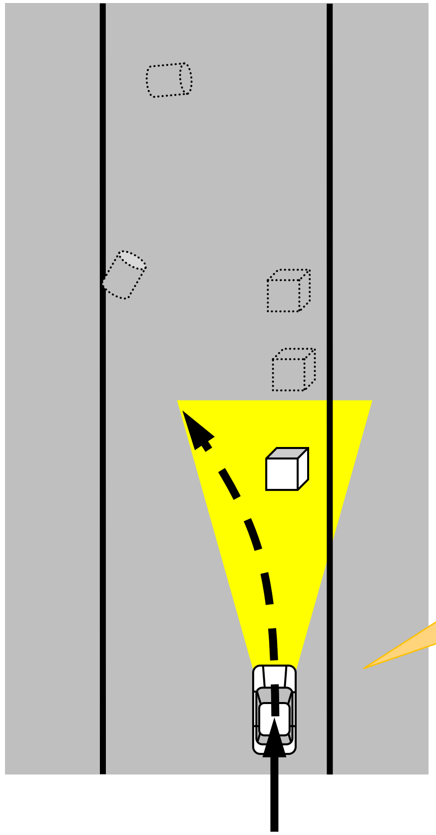
(Sakamoto&van der Schaft '08; O '11)

Outline of Part 2

- Optimal control problem, Euler-Lagrange Equations (ELE), Hamilton-Jacobi-Bellman Equation (HJBE), and numerical solution methods
- Model Predictive Control (MPC): problem formulation and difficulties, computation and stability

What is MPC?

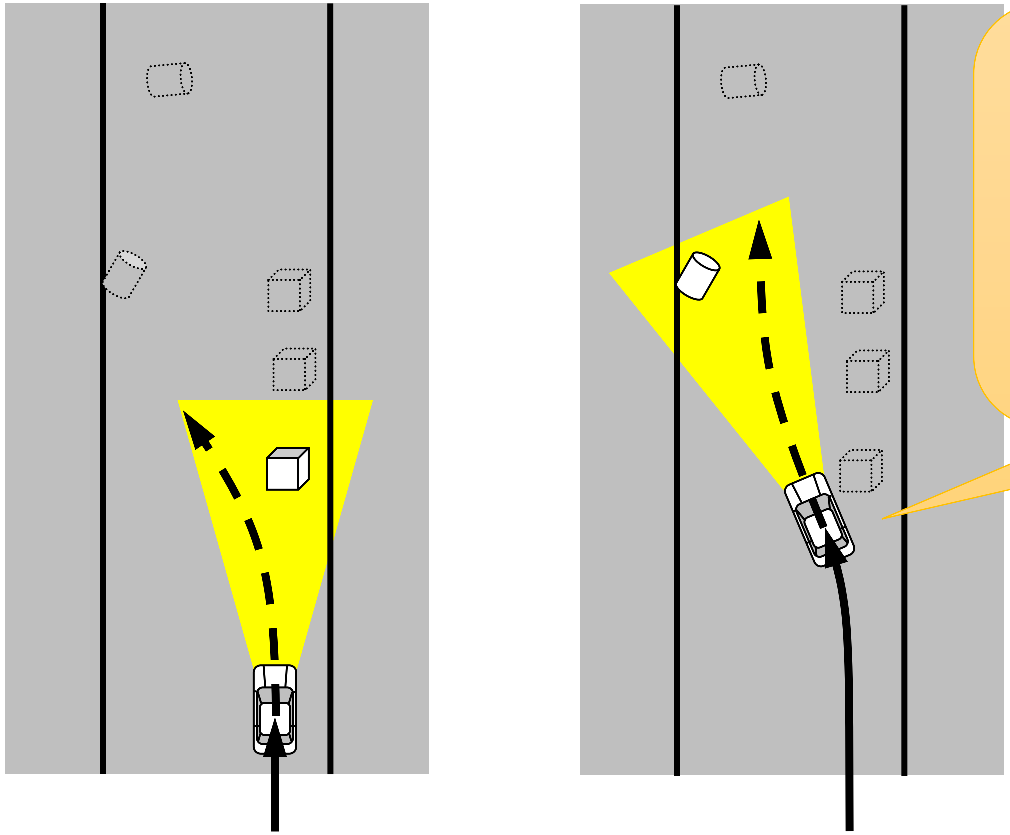
Feedback control by **real-time optimization** of the system response over a finite future.



Current control action is determined by optimization over a foreseeable future.

What is MPC?

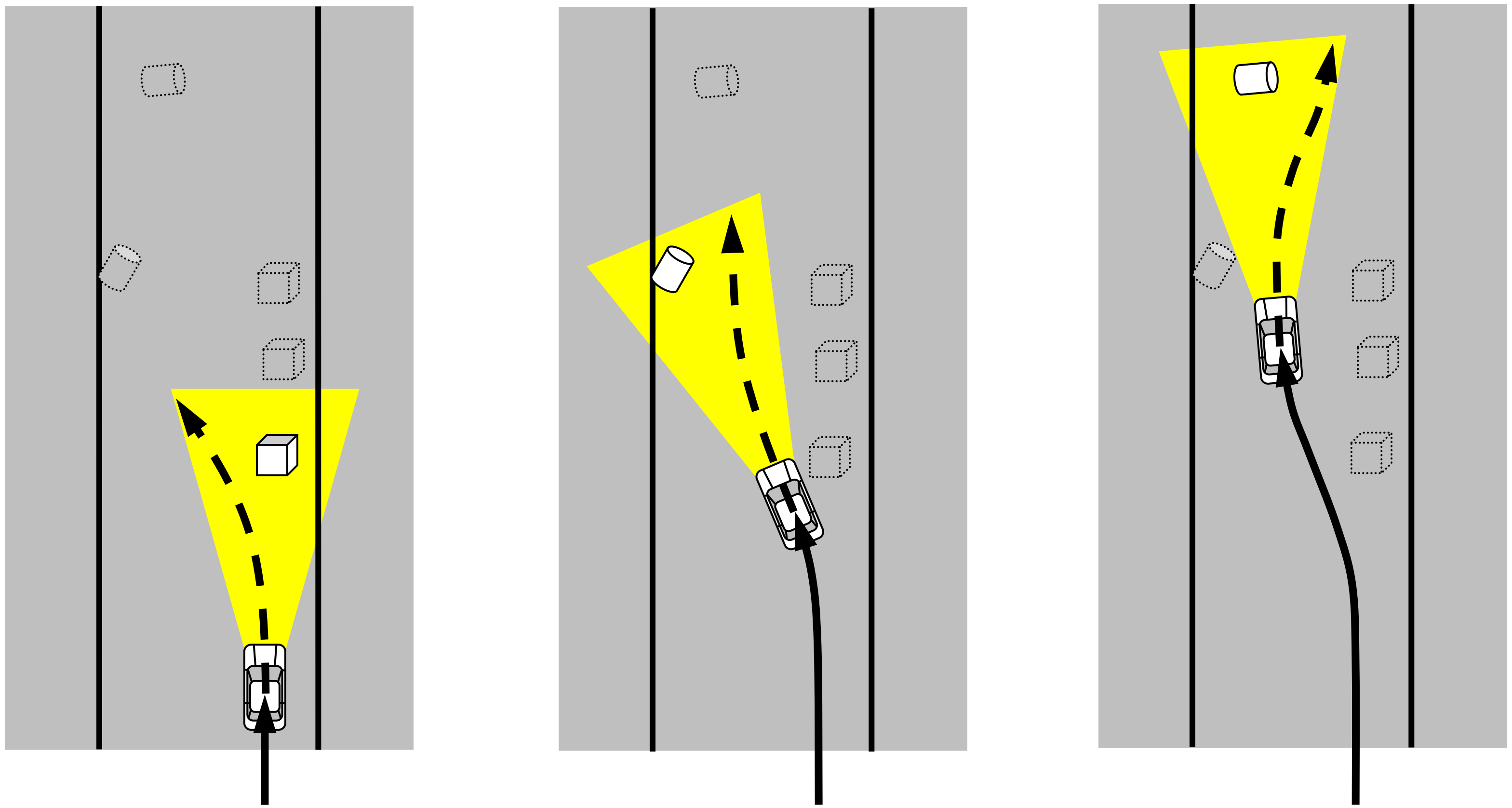
Feedback control by **real-time optimization** of the system response over a finite future.



Control action is updated at each time to utilize new information.

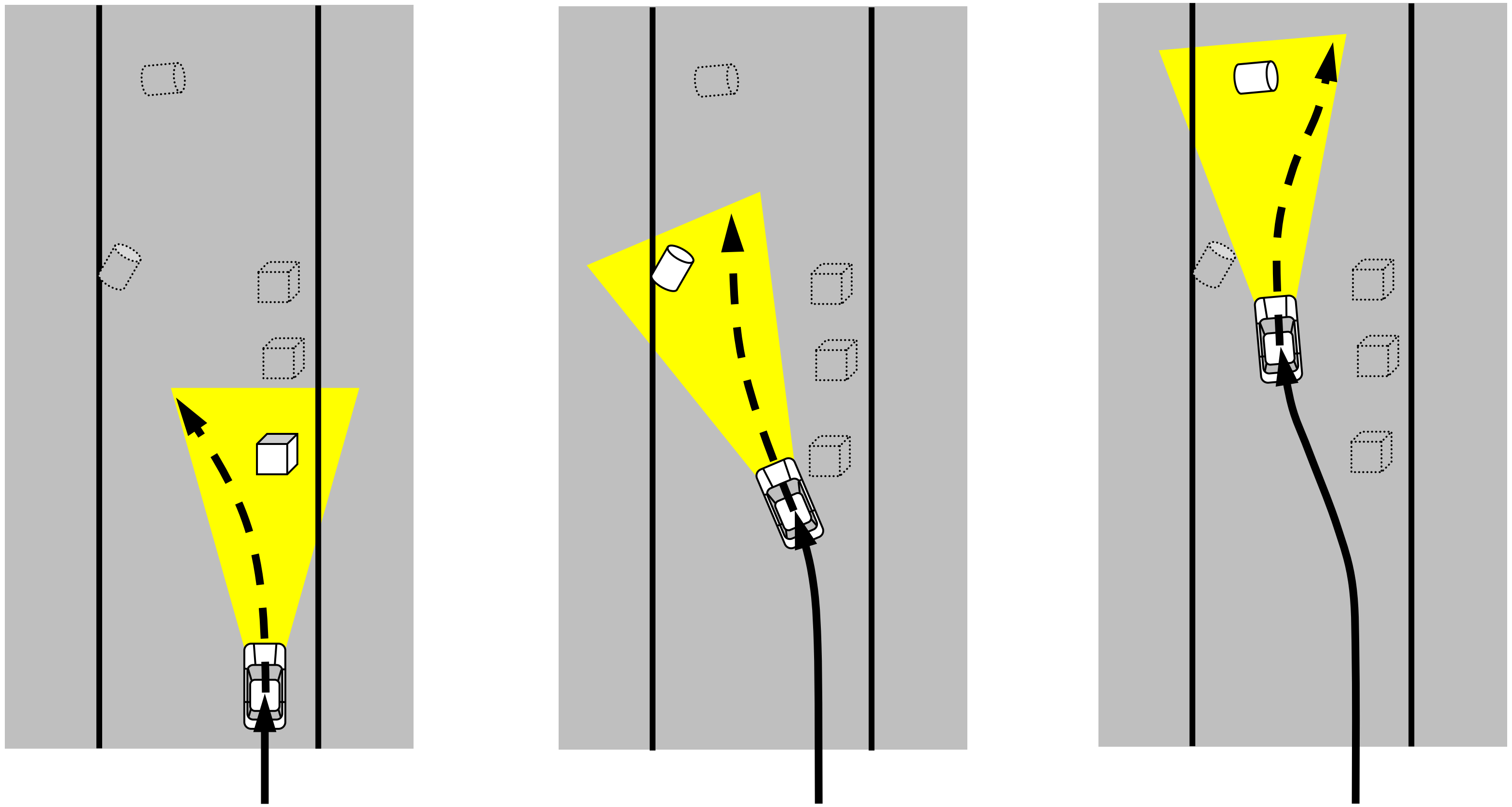
What is MPC?

Feedback control by **real-time optimization** of the system response over a finite future.

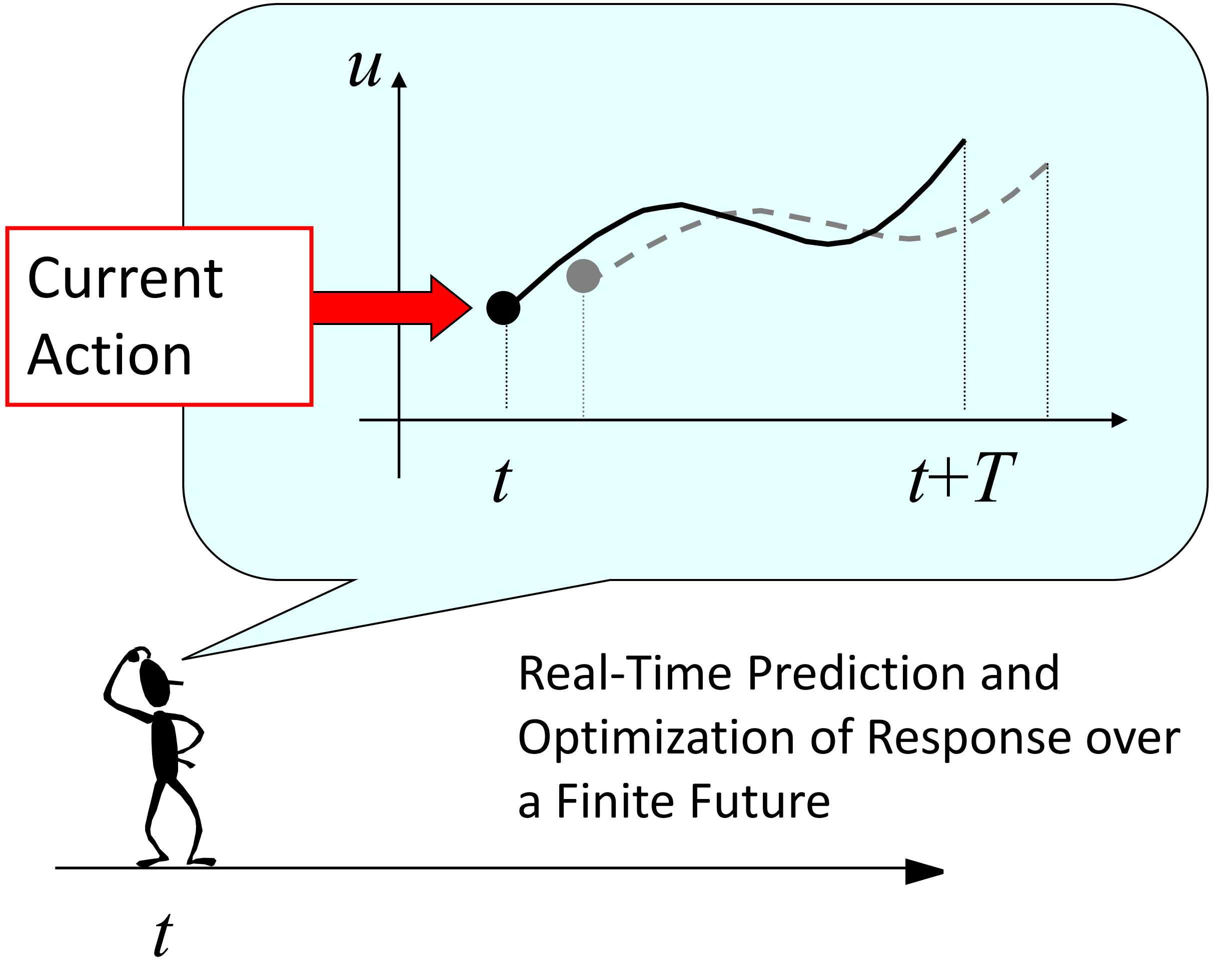


What is MPC?

A General Framework for Feedback Control of Nonlinear Dynamical Systems



Receding Horizon



OCP in MPC (Nonlinear MPC; NMPC)

State Equation and Constraint

$C \leq 0 \Leftrightarrow C + v^2 = 0$

$$\dot{x}(t) = f(x(t), u(t), t), \quad C(x(t), u(t), t) = 0$$

Performance Index

Receding Horizon

$$J = \varphi(x(t+T), t+T) + \int_{t}^{t+T} L(x(\tau), u(\tau), \tau) d\tau$$

NMPC as **State Feedback**

Initial Value of Optimal Control over $t \leq \tau \leq t + T$

$$u(t) = u_{opt}(t; x(t), t)$$

Parameterized OCP in MPC

Fixed Horizon over $0 \leq \tau \leq T$

$$\frac{dx^*(\tau;t)}{d\tau} = f(x^*(\tau;t), u^*(\tau;t), \tau + t), \quad x^*(0;t) = x(t)$$

$$C(x^*(\tau;t), u^*(\tau;t), \tau + t) = 0$$

Actual State at t

$$J = \varphi(x^*(T;t), T + t) + \int_0^T L(x^*(\tau;t), u^*(\tau;t), \tau + t)d\tau$$

Actual Input to the System

$$u(t) = u^*_{opt}(0;t)$$

Depends on initial condition $x^*(0;t) = x(t)$ (actual state)

History of MPC

- Early work (Coales&Noton '56), (Propoi '63), (Merriam '64)
- LQR variant (Kleinmann '70), (Thomas '75), (Kwon&Pearson '77)
- Process Control:
  PFC (Richalet et al. '78), DMC (Cutler&Ramaker '80), QDMC (Garcia&Morshedi '86),
  GPC (Clarke et al. '87)
- Stability of NMPC (Chen&Shaw '82), (Mayne&Michalska '90), (Chen&Allgöwer '98), (Jadbabaie et al. '01), etc.
- Numerical methods (90s-): mp-QP, RTO, etc.

J. M. Maciejowski: Model Predictive Control with Constraints, Prentice Hall (2000)
T. Ohtsuka: Research Trend of Nonlinear Model Predictive Control; Systems, Contr. & Info., 61(2), 42-50 (2017) (in Japanese)

Parameterized ELE in MPC

$$\frac{dx^*(\tau;t)}{d\tau} = f(x^*(\tau;t), u^*(\tau;t), \tau + t), \quad x^*(0;t) = x(t)$$

Actual State at t

$$\frac{d\lambda^*(\tau;t)}{d\tau} = -\nabla_x H(x^*(\tau;t), u^*(\tau;t), \lambda^*(\tau;t), \tau + t),$$

$$\lambda^*(T;t) = \nabla_x \varphi(x^*(T;t), T + t)$$

$$\nabla_u H(x^*(\tau;t), u^*(\tau;t), \lambda^*(\tau;t), \tau + t) = 0$$

Nonlinear TPBVP

Actual Input to the System

$$u(t) = u^*_{opt}(0;t)$$

HJBE in MPC

Value Function

$$V(x,\tau;t) = \min_{u[\tau,T]} \left[\varphi(x^*(T;t), T+t) + \int_{\tau}^{T} L(x^*(\tau';t), u^*(\tau';t), \tau'+t) d\tau' \right]$$

HJBE

$$-\nabla_\tau V(x,\tau;t) = \min_u H(x, u, \nabla_x V(x,\tau;t), \tau+t), \quad V(x,T;t) = \varphi(x, T+t)$$

Nonlinear PDE

Optimal Control

$$u^*_{opt}(x,\tau;t) = \arg\min_u H(x, u, \nabla_x V(x,\tau;t), \tau+t),$$
$$u_{MPC}(x,t) = u^*_{opt}(x,0;t)$$

In a time-invariant case (time t does not appear explicitly):

$$u_{MPC}(x) = u^*_{opt}(x,0)$$

Difficulty in MPC: Solution Methods

ELE

- Open-loop solution for a given state
- Iterative methods: gradient methods, Newton's method (computationally demanding for real-time implementation)

HJBE

- Closed-loop solution (state feedback)
- Explosive growth of computation and storage for high dimensional systems (curse of dimensionality)

Implementation of MPC for **fast and complex nonlinear systems** with sampling periods of the order of **milliseconds** is challenging!

Difficulty in MPC: Closed-Loop Stability

Closed-loop stability of (time-invariant) MPC with a finite horizon?

Key Idea: Value function $V_{MPC}(x) = V(x, 0)$ can be a Lyapunov function.

From HJBE

$$\dot{V}_{MPC}(x) = -L\big(x, u_{MPC}(x)\big) - \nabla_\tau V(x, 0)$$
$$u_{MPC}(x) = u_{opt}(x, 0)$$

Methods for Guaranteeing Stability

- Terminal constraint $x^*(T;t) = 0$
- Terminal constraint $x^*(T;t) \in \Omega$, a stabilizing feedback to make Ω invariant, and a bound on the infinite horizon cost in Ω
- Control Lyapunov function (CLF) as the terminal penalty $\varphi(x)$ and a feedback $u = k(x)$ such that

$$\frac{\partial \varphi(x)}{\partial x} f\big(x, k(x)\big) \leq -L(x, k(x))$$

Stability guarantee of MPC is challenging!

D. Q. Mayne, et al.: Constrained Model Predictive Control: Stability and Optimality; Automatica, 36(6), 789-814 (2000)

Some Variants of MPC

- Robust MPC (Constraint satisfaction under uncertainty)
- Stochastic MPC (Open-loop/closed-loop optimization, chance constraint)
- Adaptive MPC
- Learning MPC (Kabzan et al. ‘19; Gros&Zanon ’20; Rosolia&Borrelli ‘20)
  Cf. MPC in Learning: MPC-guided policy search (Zhang et al. ‘16)
- Distributed MPC
- Output MPC
- Moving Horizon Estimation (Fitting model and measurement over a finite past)

J. B. Rawlings, et al.: Model Predictive Control: Theory, Computation, and Design, 2nd Ed., Nob Hill Publishing (2017)
A. Mesbah: Stochastic Model Predictive Control: An Overview and Perspectives for Future Research; IEEE Control Systems Magazine, 36(6), 34-44 (2016)

Summary

- MPC: State feedback control by real-time optimization over a finite future
- General framework for feedback control of nonlinear systems
- Difficulties: Solution methods and stability guarantee
- Implementation of MPC for **fast and complex nonlinear systems** with sampling periods of the order of **milliseconds** is challenging!
- There are some variants of MPC.

References (1/2)

- E. G. Al'Brekht: On the Optimal Stabilization of Nonlinear Systems; J. Appl. Math. Mech., 25(5), 1254–1266 (1961)
- D. L. Lukes: Optimal Regulation of Nonlinear Dynamical Systems; SIAM J. Contr., 7(1), 75–100 (1969)
- R. W. Beard, G. N. Saridis, J. T.Wen: Galerkin Approximations of the Generalized Hamilton-Jacobi-Bellman Equation; Automatica, 33(12), 2159–2177 (1997)
- G. Kreiselmeier, T. Birkhölzer: Numerical Nonlinear Regulator Design; IEEE Trans. Autom. Contr., 39(1), 33–46 (1994)
- C. J. Goh: On the Nonlinear Optimal Regulator Problem; Automatica, 29(3), 751–756 (1993)
- C.-S. Huang, S. Wang, C. S. Chen, Z.-C. Li: A Radial Basis Collocation Method for Hamilton-Jacobi-Bellman Equations; Automatica, 42(12), 2201-2207 (2006)
- K. Fujimoto, H. Beppu, Y. Takaki: Numerical Solutions of Hamilton-Jacobi Inequalities by Constrained Gaussian Process Regression; SICE J. Contr., Meas., & Sys. Integration, 11(5), 419-428 (2018)
- A. J. van der Schaft: On a State Space Approach to Nonlinear H_∞ Control; Syst. Contr. Lett., 16(1), 1–8 (1991)
- N. Sakamoto, A. J. van der Schaft: Analytical Approximation Methods for the Stabilizing Solution of the Hamilton-Jacobi Equation; IEEE Trans. Autom. Contr., 53(10), 2335–2350 (2008)
- T. Ohtsuka: Solutions to the Hamilton-Jacobi Equation with Algebraic Gradients; IEEE Trans. Autom. Contr., 56(8), 1874-1885 (2011)
- J. F. Coales, A. R. M. Noton: An On-Off Servo Mechanism with Predicted Change-Over; Proc. IEE, Part B, 103(10), 449–460 (1956)
- A. I. Propoi: Application of Linear Programming Methods for the Synthesis of Automatic Sampled-Data Systems; Avtomatika i Telemekhanika, 24(7), 912–920 (1963)
- C. W. Merriam, III: Optimization Theory and the Design of Feedback Control Systems, McGraw-Hill (1964)
- D. J. Kleinman: An Easy Way to Stabilize a Linear Constant System; IEEE Trans. Autom. Contr., 15(6), 692 (1970)
- Y. A. Thomas: Linear Quadratic Optimal Estimation and Control with Receding Horizon; Electronics Letters; 11(1), 19-21 (1975)
- W. H. Kwon, A. E. Pearson: A Modified Quadratic Cost Problem and Feedback Stabilization of a Linear System; IEEE Trans. Autom. Contr., AC-15(5), 838-842 (1977)

References (2/2)

- J. Richalet, A. Rault, J. L. Testud, J. Papon: Model Predictive Heuristic Control: Applications to Industrial Processes; Automatica, 14(5), 413–428 (1978)
- C. R. Cutler, B. L. Ramaker: Dynamic Matrix Control - A Computer Control Algorithm; Proc. Joint Autom. Contr. Conf., paper WP5-B (1980)
- C. E. Garcia, A. M. Morshedi: Quadratic Programming Solution of Dynamic Matrix Control (QDMC); Chem. Eng. Comm., 46(1-3), 73-87 (1986)
- D. W. Clarke, C. Mohtadi, P. S. Tuffs: Generalized Predictive Control - Part I. The Basic Algorithm; Automatica, 23(2), 137-148 (1987)
- D. W. Clarke, C. Mohtadi, P. S. Tuffs: Generalized Predictive Control - Part II. Extensions and Interpretations; Automatica, 23(2) 149-160 (1987)
- C. C. Chen, L. Shaw: On Receding Horizon Feedback Control; Automatica, 18(3), 349-352 (1982)
- D. Q. Mayne, H. Michalska: Receding Horizon Control of Nonlinear Systems; IEEE Trans. Autom. Contr., 35(7), 814-824 (1990)
- H. Chen, F. Allgöwer: A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability; Automatica, 34(10), 1205-1217 (1998)
- A. Jadbabaie, J. Yu, J. Hauser: Unconstrained Receding-Horizon Control of Nonlinear Systems; IEEE Trans. Autom. Contr., 46(5), 776-783 (2001)
- J. Kabzan, L. Hewing, A. Liniger, M. N. Zeilinger: Learning-Based Model Predictive Control for Autonomous Racing; IEEE Robotics & Autom. Lett., 4(4), 3363-3370 (2019)
- S. Gros, Zanon: Data-Driven Economic NMPC using Reinforcement Learning; IEEE Trans. Autom. Contr., 65(2), 636-648 (2020)
- U. Rosolia, F. Borrelli: Learning How to Autonomously Race a Car: A Predictive Control Approach; IEEE Trans. Contr. Sys. Tech., 28(6), 2713-2719 (2020)
- T. Zhang, G. Kahn, S. Levine, P. Abbeel: Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search; Proc. 2016 IEEE Int. Conf. Robotics & Autom., 528–535 (2016)

NeurIPS2021 Tutorial
Real-Time Optimization for Fast and Complex Control Systems
Part 3

Real-Time Optimization for Model Predictive Control

Toshiyuki Ohtsuka
Department of Systems Science
Graduate School of Informatics
Kyoto University

Outline

Part 1: Introduction to Control Systems

Part 2: Optimal Control and Model Predictive Control

Part 3: Real-Time Optimization for Model Predictive Control

Part 4: Advanced Topics in Model Predictive Control

Outline of Part 3

- Real-Time Optimization (RTO) Algorithm for Nonlinear MPC (NMPC)
- Automatic Code Generation Tool

Solution Methods of MPC

- **Off-Line Methods (Explicit MPC)**
  - (Bemporad et al. ’02), (Zeilinger et al. ‘11), etc.

- **On-Line Methods (Real-Time Optimization; RTO)**
  - (O ‘97), (O ‘04) **Real-Time Algorithm as ODE**
  - (Diehl et al. ‘05), (Alamir ‘06), (DeHaan&Guay ‘07), (Zavala&Biegler ‘09), (Graichen ‘12), (Stella et al. ‘17), etc.
  - (Deng&O ‘19) **Parallel Algorithm**

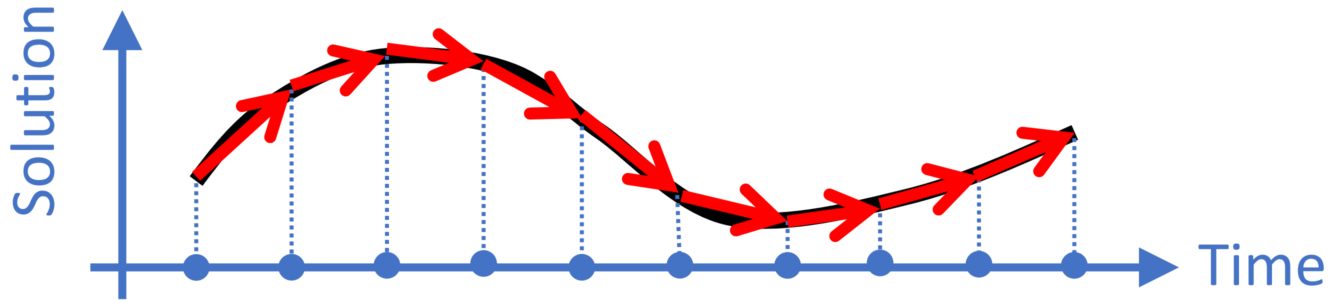
I. J. Wolf, W. Marquardt: Fast NMPC Schemes for Regulatory and Economic NMPC – A Review; J. Process Contr., 44, 162-183 (2016)

Numerical Algorithm for NMPC

An iterative search of an optimal solution within a short sampling period may fail to converge.

However…

The small change in the optimal solution during the short sampling period can be traced without an iterative search!



Numerical Algorithm for NMPC

An iterative search of an optimal solution within a short sampling period may fail to converge.

However…

The small change in the optimal solution during the short sampling period can be traced without an iterative search!

Only one linear equation at each sampling time: Continuation/GMRES

(O '04)

Discretization of the Horizon

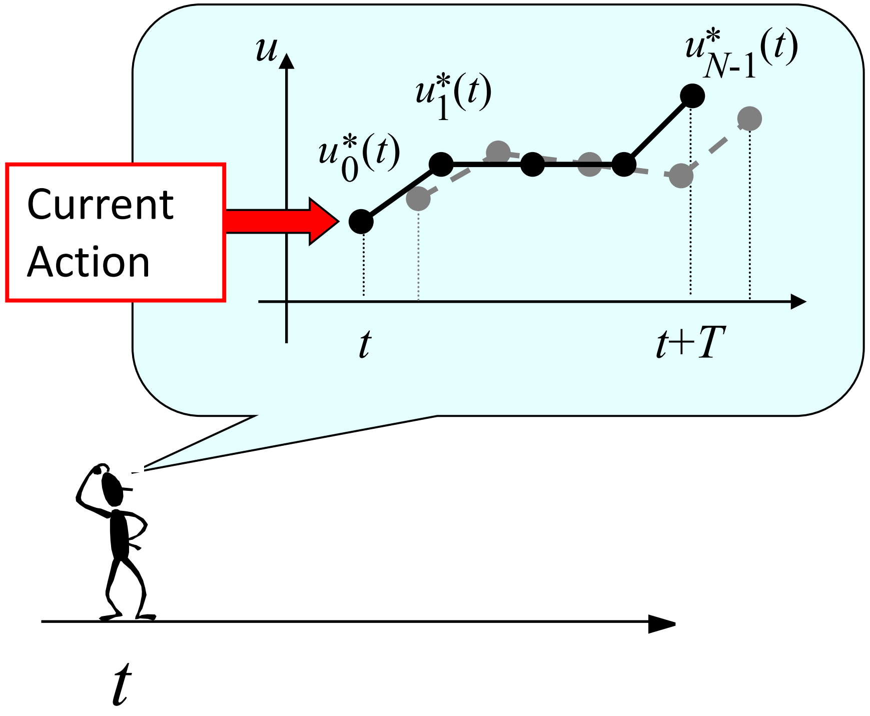
Input Sequence over the Horizon at Time t:

$$U(t) = [u_0^{*T}(t) \quad \mu_0^{*T}(t) \quad \cdots \quad u_{N-1}^{*T}(t) \quad \mu_{N-1}^{*T}(t)]^T$$

Actual Input: $u(t) = u_0^*(t)$, Lagrange Multiplier: $\mu_i^*(t)$

Discretization Step $\Delta\tau = \frac{T}{N}$

$\neq$ Sampling Period Δt



Discretized ELE

$$x_{i+1}^*(t) = x_i^*(t) + f\big(x_i^*(t), u_i^*(t)\big)\Delta\tau, \quad x_0^*(t) = x(t)$$

$$\lambda_i^*(t) = \lambda_{i+1}^*(t) + \nabla_x H\big(x_i^*(t), u_i^*(t), \lambda_{i+1}^*(t), \mu_i^*(t)\big)\Delta\tau$$

$$\lambda_N^*(t) = \nabla_x \varphi(x_N^*(t))$$

$$\nabla_u H\big(x_i^*(t), u_i^*(t), \lambda_{i+1}^*(t), \mu_i^*(t)\big) = 0$$

$$C\big(x_i^*(t), u_i^*(t)\big) = 0$$

Hamiltonian: $H(x^*, u^*, \lambda^*, \mu^*) = L(x^*, u^*) + \lambda^{*T} f(x^*, u^*) + \mu^{*T} C(x^*, u^*)$

State and costate are functions of $U(t)$ and $x_0^*(t) = x(t)$

Nonlinear Equation for the Input Sequence

Future Input Sequence

Current State

$$F(U(t), x(t), t) = \begin{bmatrix} \nabla_u H\left(x_0^*(t), u_0^*(t), \lambda_1^*(t), \mu_0^*(t)\right) \\ C\left(x_0^*(t), u_0^*(t)\right) \\ \vdots \\ \nabla_u H\left(x_{N-1}^*(t), u_{N-1}^*(t), \lambda_N^*(t), \mu_{N-1}^*(t)\right) \\ C\left(x_{N-1}^*(t), u_{N-1}^*(t)\right) \end{bmatrix} = 0$$

State and costate are functions of $U(t)$ and $x_0^*(t) = x(t)$

Continuation Method

Condition for Stabilizing $F(U(t), x(t), t) = 0$

$$\frac{d}{dt}F(U(t), x(t), t) = -\zeta F(U(t), x(t), t) \quad (\zeta > 0)$$

Linear Equation for $\dot{U}(t)$:

$$\frac{\partial F}{\partial U}\dot{U} = -\zeta F - \frac{\partial F}{\partial x}\dot{x} - \frac{\partial F}{\partial t}$$

Update of Optimal Solution $U(t)$:

$$U(t + \Delta t) = U(t) + \dot{U}(t)\Delta t$$

No Iterative Search

Continuatin/GMRES Method

1) At each time t, measure the state $x(t)$
2) Solve a linear equation for $\dot{U}(t)$:
$$\frac{\partial F}{\partial U}\dot{U} = -\zeta F - \frac{\partial F}{\partial x}\dot{x} - \frac{\partial F}{\partial t}$$
3) Update $U(t)$ by $U(t+\Delta t) = U(t) + \dot{U}(t)\Delta t$
4) Go to Step 1)

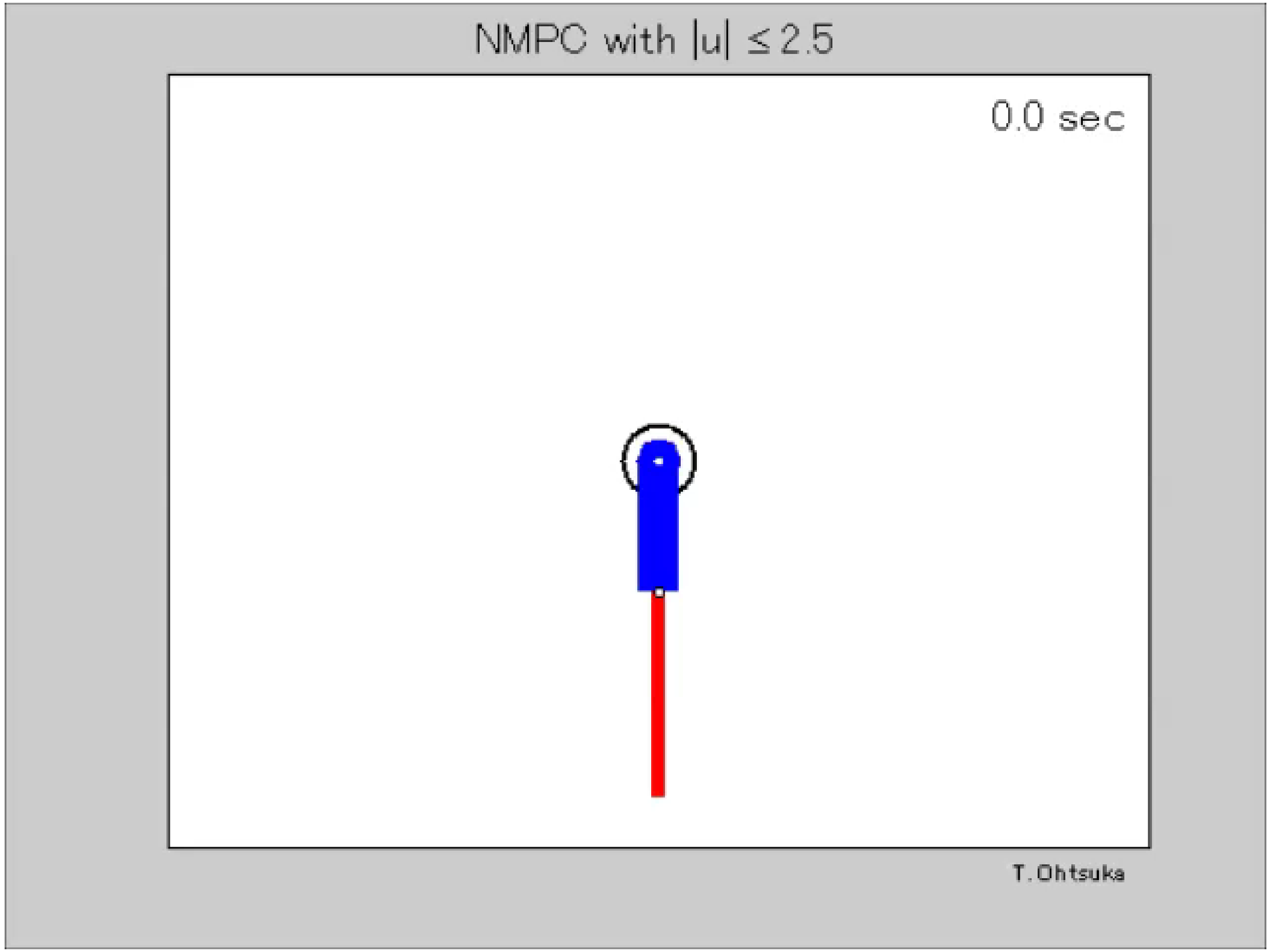
Jacobian-Free

$\frac{\partial F}{\partial U}$ unnecessary

- Only one linear equation at each time
- Efficient linear equation solver **GMRES**
- Horizon length: $T(0) = 0, T(t) \to T_f$ (easy to find $U(0)$)

(O ’04)

Two-Link Arm



- Free Elbow Joint and Constrained Shoulder Torque
- Computational Time per Update: 0.5ms (CPU: Core 2 Duo 1.2GHz)

(O '12)

Underactuated Hovercraft

- Fixed On-Off Thrusters, No Lateral Thrust
- Sampling Period 1/120 s (CPU: Athlon 900 MHz)

(Seguchi&O ’03)

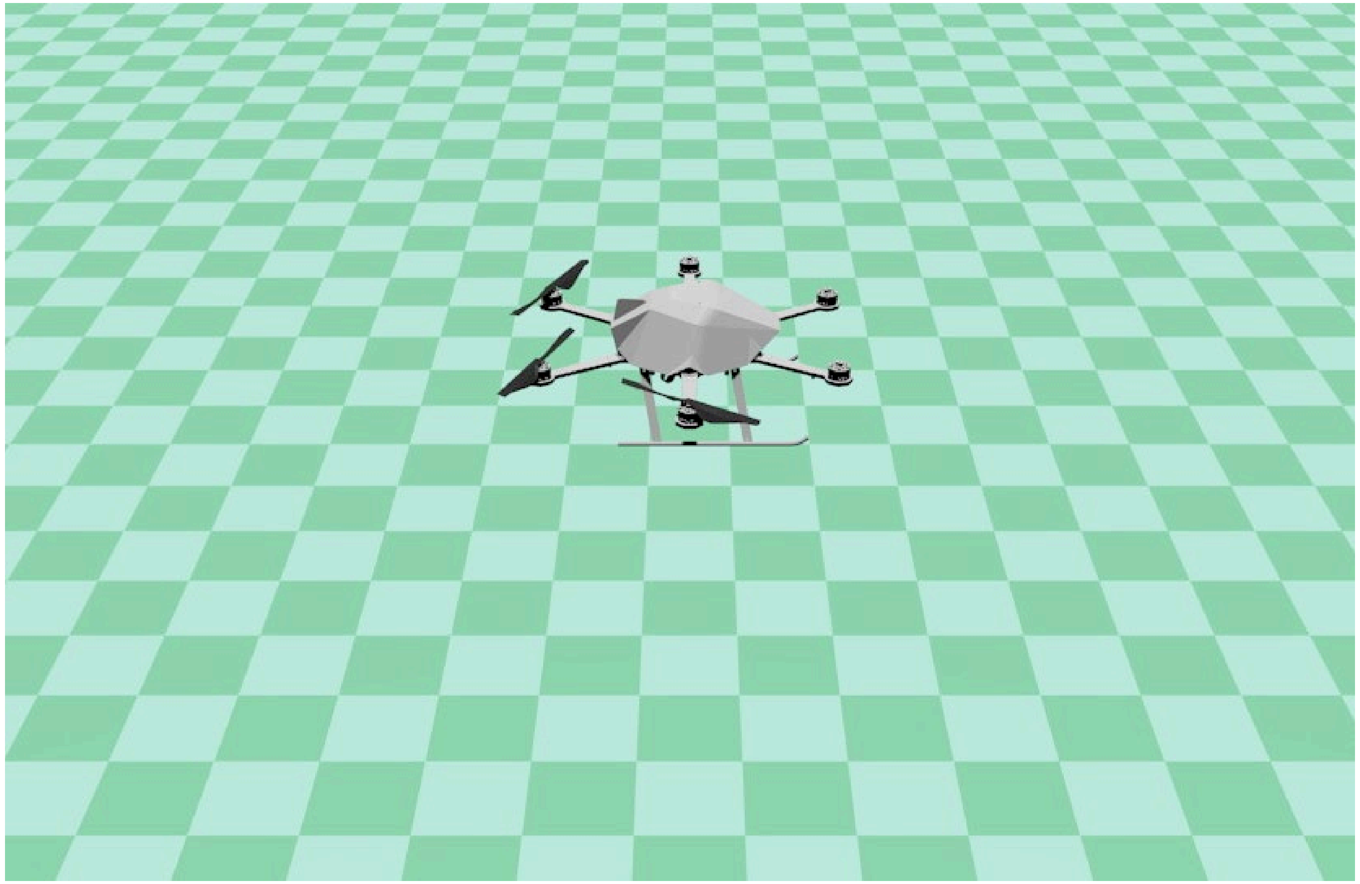
Automatic Ship Maneuvering

© Kawasaki Heavy Industry 2004

- Redundant Actuators
- NMPC for Route Tracking and Thrust Assignment

(Hamamatsu et al. ’08)

Hexacopter with Failed Rotors



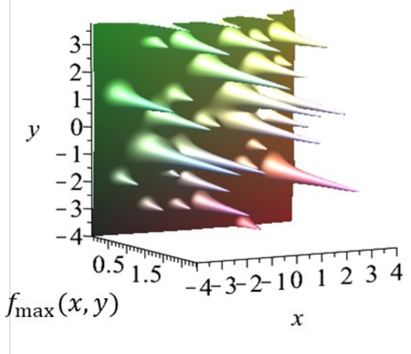
- Nonlinear Model with Quaternions
- Constraints on Thrusts
- Same NMPC for All Failure Patterns with Different Weights
- Computational Time per Update: below 0.2ms

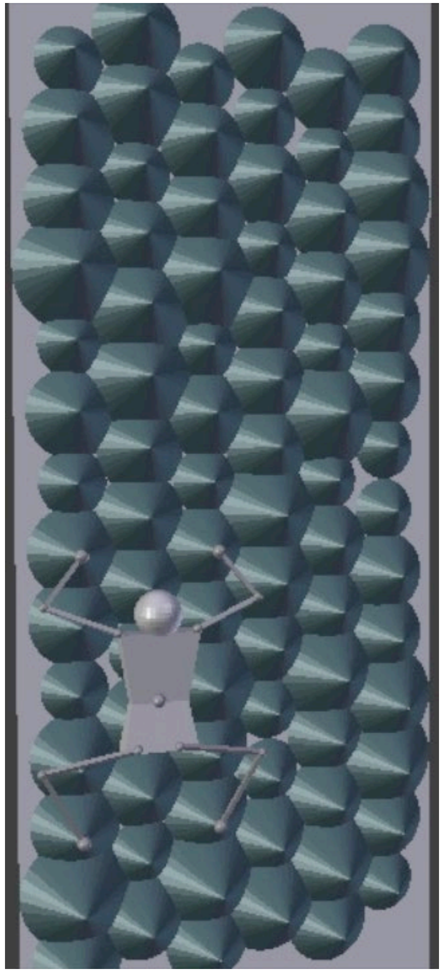
(Aoki et al. ’21)

Climbing Humanoid Robot

- Integrated optimization of path and motions
- Penalty on deviation from reference velocity
- Constraints on force/moment balance
- Constraints on holding forces
- 12 states, 36 inputs (including dummy inputs), and 15 constraints
- Computation time per update < 20 ms (Core i5, 1.8GHz)

Maximum Holding Force on the Wall

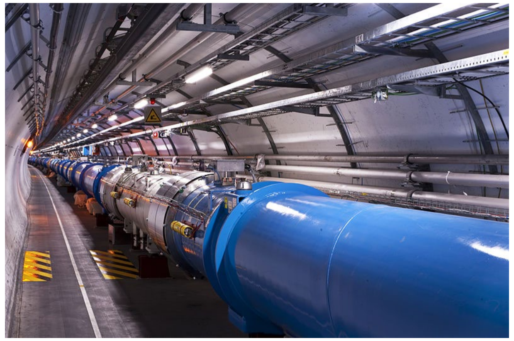




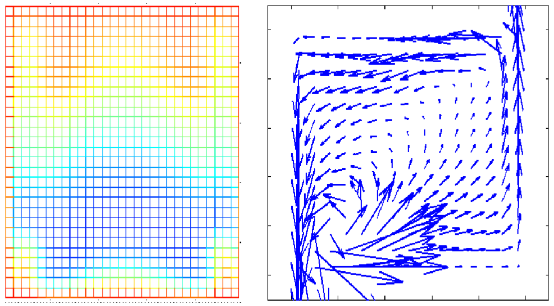
(Omoto et al. ’19)

Other Applications

- Tethered Satellite, Robots, Automobiles, Formation Flight of Helicopters, FOWT, etc.
- Flight Experiment of On-Line Path Generation for an Aircraft
- Temperature Control for Superconducting Magnets in the Large Hadron Collider (LHC)
- Distributed Parameter Systems (Thermofluid Systems)



Maximilien Brice (CERN) (https://commons.wikimedia.org/wiki/File:Views_of_the_LHC_tunnel_sector_3-4,_tirage_2.jpg), „Views of the LHC tunnel sector 3-4, tirage 2“, https://creativecommons.org/licenses/by-sa/3.0/legalcode



T. Ohtsuka, et al.: Practical Applications of Control by Real-Time Optimization, Corona Publishing (2015) (in Japanese)

Other Problems

- Nonlinear Receding Horizon Differential Game
  - Minimizer (Control) and Maximizer (Disturbance) (Hirota et al. ‘17)
  - Nash equilibrium of Multiple Players (Azuma&O ‘11)
- Moving Horizon Estimation (Soneda&O ‘05)
  - Optimal Fitting of a Model and Measurements over a Finite Past

Outline of Part 3

- Real-Time Optimization (RTO) Algorithm for Nonlinear MPC (NMPC)
- Automatic Code Generation Tool

Software Tools for NMPC

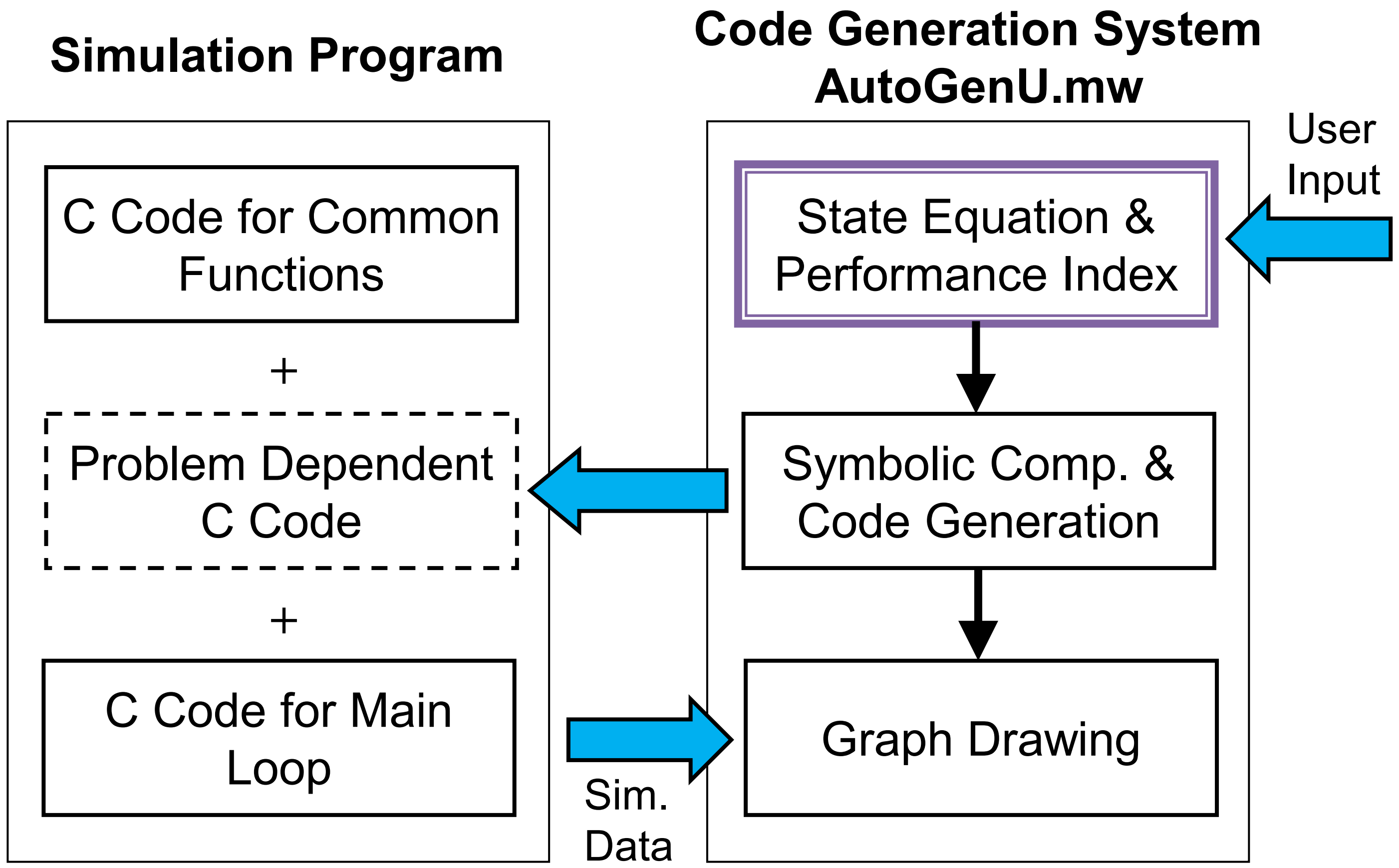
- MPT (Kvasnica et al. ’04), MPT3 (Herceg et al. ‘13)
- AutoGen (O&Kodama ’02)
- **AutoGenU** for Mathematica (O ‘04), Maple (O ‘15), Python Jupyter Notebook (Katayama&O ‘20)
- ACADO (Houska et al. ‘11), acados (Verschueren et al., ‘21)
- CVXGEN (Mattingley&Boyd ‘12), qpOASES (Ferreau et al. ’14)
- FORCES PRO
- GRAMPC (Englert et al. ‘19)
- **ParNMPC** (Deng&O ‘20)

AutoGenU for Maple

- Developed by Cybernet Systems in collaboration with the speaker
- Symbolic computation by Maple
- C code generation
- Integrated compilation, execution, and graph plotting
- Freely available at Maplesoft Application Center website

Automatic Code Generation

C code for simulation is generated from the state equation and performance index specified in **Maple**.



Example: Semi-Active Damper

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ a x_1 + b x_2 u \end{bmatrix}$$

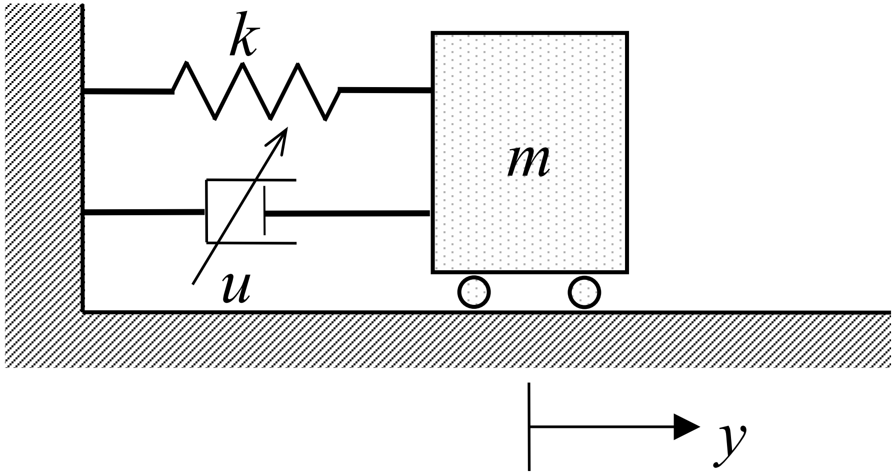
Bilinear

$$0 \le u \le u_{max}$$

$$\Updownarrow$$

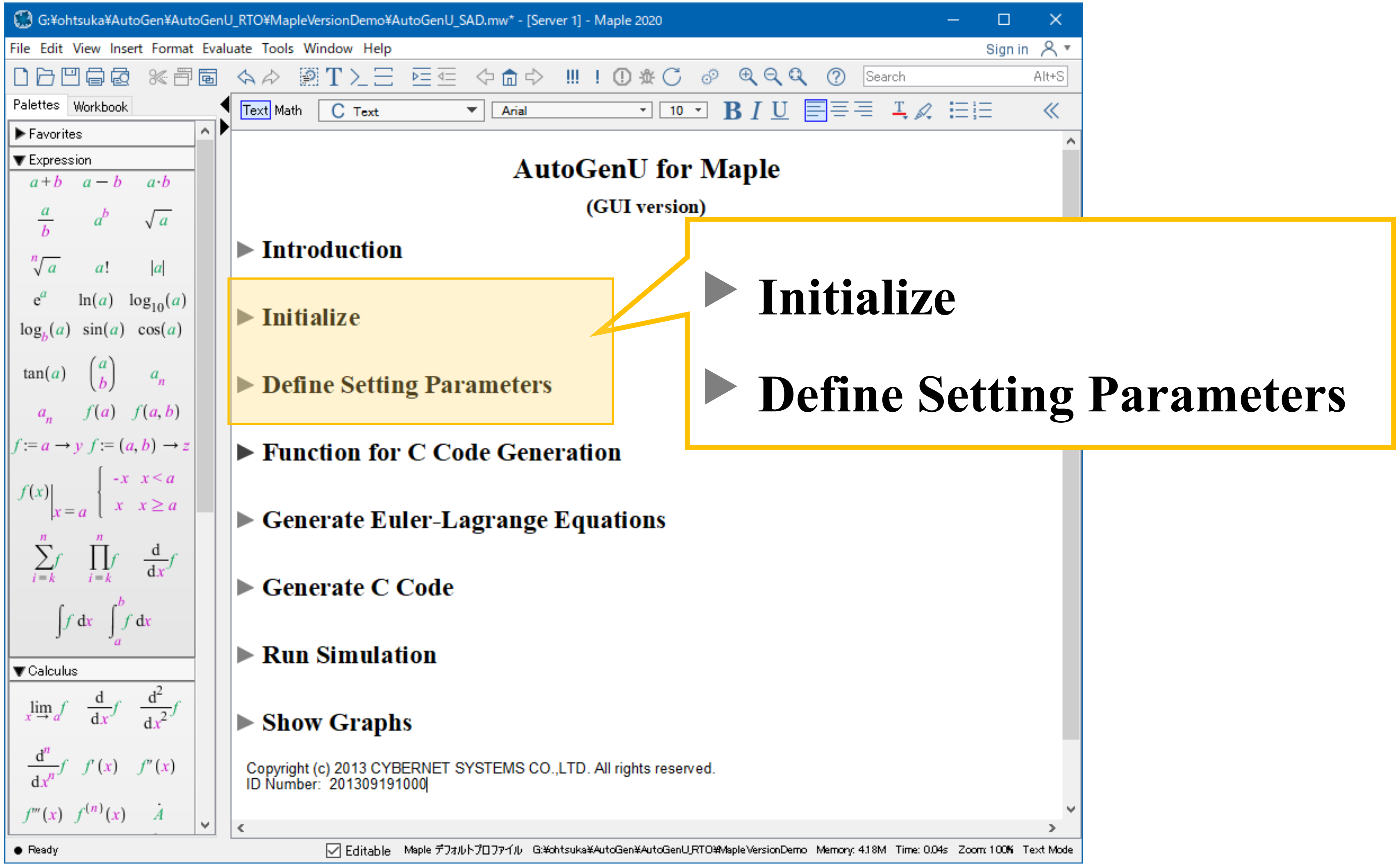
$$\left(u_1 - \frac{u_{max}}{2}\right)^2 + u_2^2 - \frac{u_{max}^2}{4} = 0$$

$$J = \frac{1}{2} x^T(t+T) S_f x(t+T) + \int_t^{t+T} \left(\frac{1}{2} x^T(\tau) Q x(\tau) + \frac{r_1}{2} u_1^2(\tau) - r_2 u_2 \right) d\tau$$



$x_1 = y, x_2 = \dot{y}$,
$u_1 = u, u_2$: Dummy Input

Maple Worksheet AutoGenU.mw



State Equation and Constraint

Input

Input Box

fxu

```
<x[2], a * x[1] + b * x[2] * u[1]>
```

Cxu

```
<(u[1] - umax/2)^2 + u[2]^2 - (umax/2)^2>
```

Result

$$fxu := \begin{bmatrix} x_2 \\ b\,u_1\,x_2 + a\,x_1 \end{bmatrix}$$

$$Cxu :=$$

$$\left[\left(u_1 - \frac{1}{2}\,umax\right)^2 + u_2^2 - \frac{1}{4}\,umax^2 \right]$$

Performance Index

Input

L	`(xv^%T.Q.xv)/2  + r[1]*u[1]^2/2 - r[2]*u[2]`
phi	`(xv^%T.Sf.xv)/2`

Result

$$L := \frac{1}{2} x_1^2 q_1 + \frac{1}{2} x_2^2 q_2 + \frac{1}{2} r_1 u_1^2 - r_2 u_2$$

$$\phi := \frac{1}{2} x_1^2 sf_1 + \frac{1}{2} x_2^2 sf_2$$

User’s Variables and Arrays

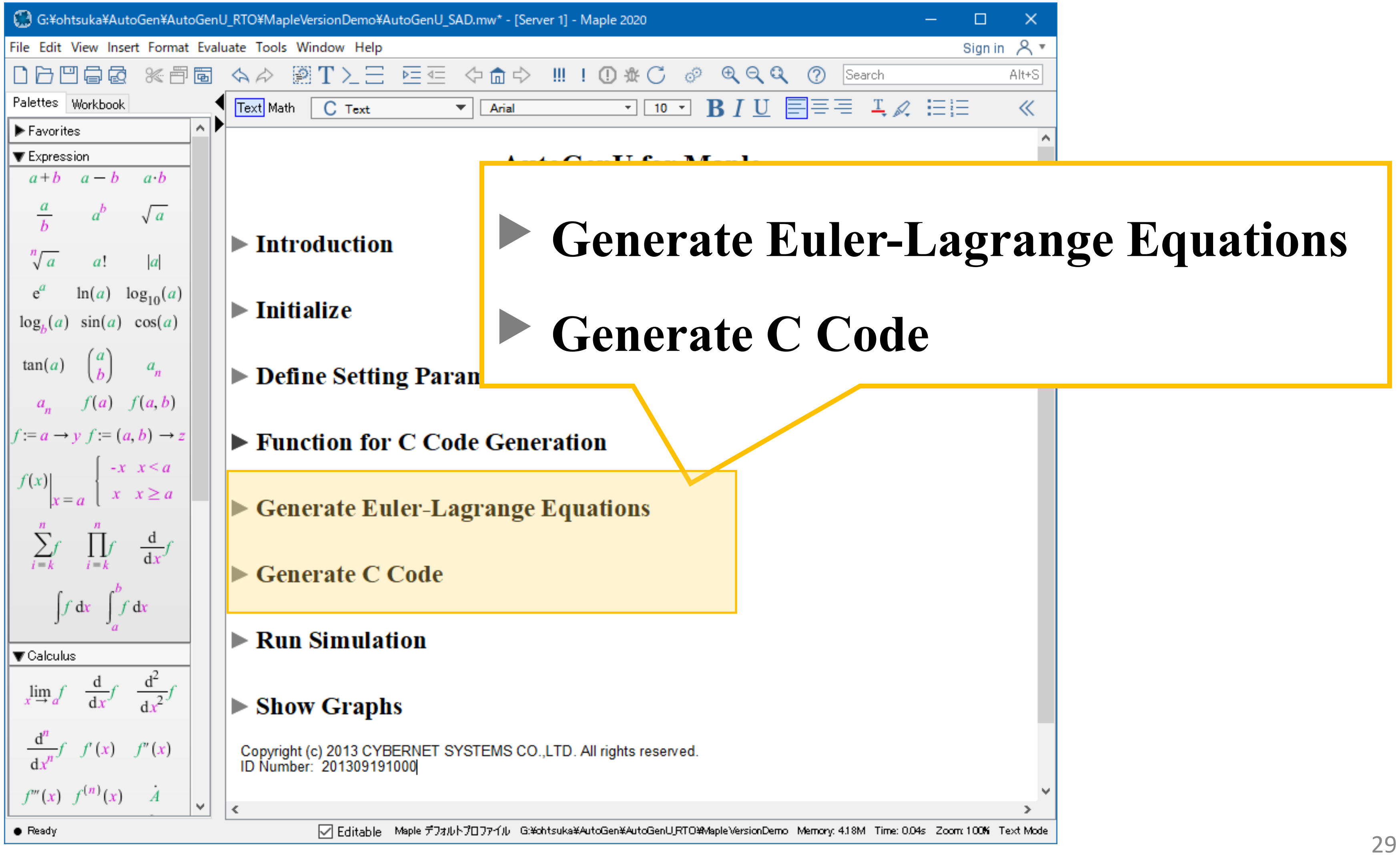
Input

MyVarNames	["a", "b", "umax"]	Variables
MyVarValues	[-1, -1, 1]	
MyArrNames	["q", "r", "sf"]	Arrays
MyArrDims	[dimx, dimu, dimx]	
MyArrValues	[<1,10>, <1,0.01>, <1,10>]	

Simulation Conditions

tsim	20.0	Final Time of Simulation
ht	0.001	Time Step for Simulation
tf	1.0	Horizon Length
x0	<2, 0>	Initial State
kmax	5	No. of Iterations in GMRES
dv	50	No. of Time Steps in Horizon, N

Maple Worksheet AutoGenU.mw



Symbolic Computation for ELE

```
> H:=L + lmdv^%T.fxu + muv^%T.Cxu;
```

$$H := \frac{1}{2}x_1^2 q_1 + \frac{1}{2}x_2^2 q_2 + \frac{1}{2}r_1 u_1^2 - r_2 u_2 + lmd_1 x_2 + lmd_2 (b u_1 x_2 + a x_1) + u_3\left(\left(u_1 - \frac{1}{2}umax\right)^2 + u_2^2 - \frac{1}{4}umax^2\right)$$

Hamiltonian

```
> uv := convert([uv, muv], Vector);
```

$$uv := \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

```
> dimuc := dimu + dimc:
> Hx := Dv(H,xv);
```

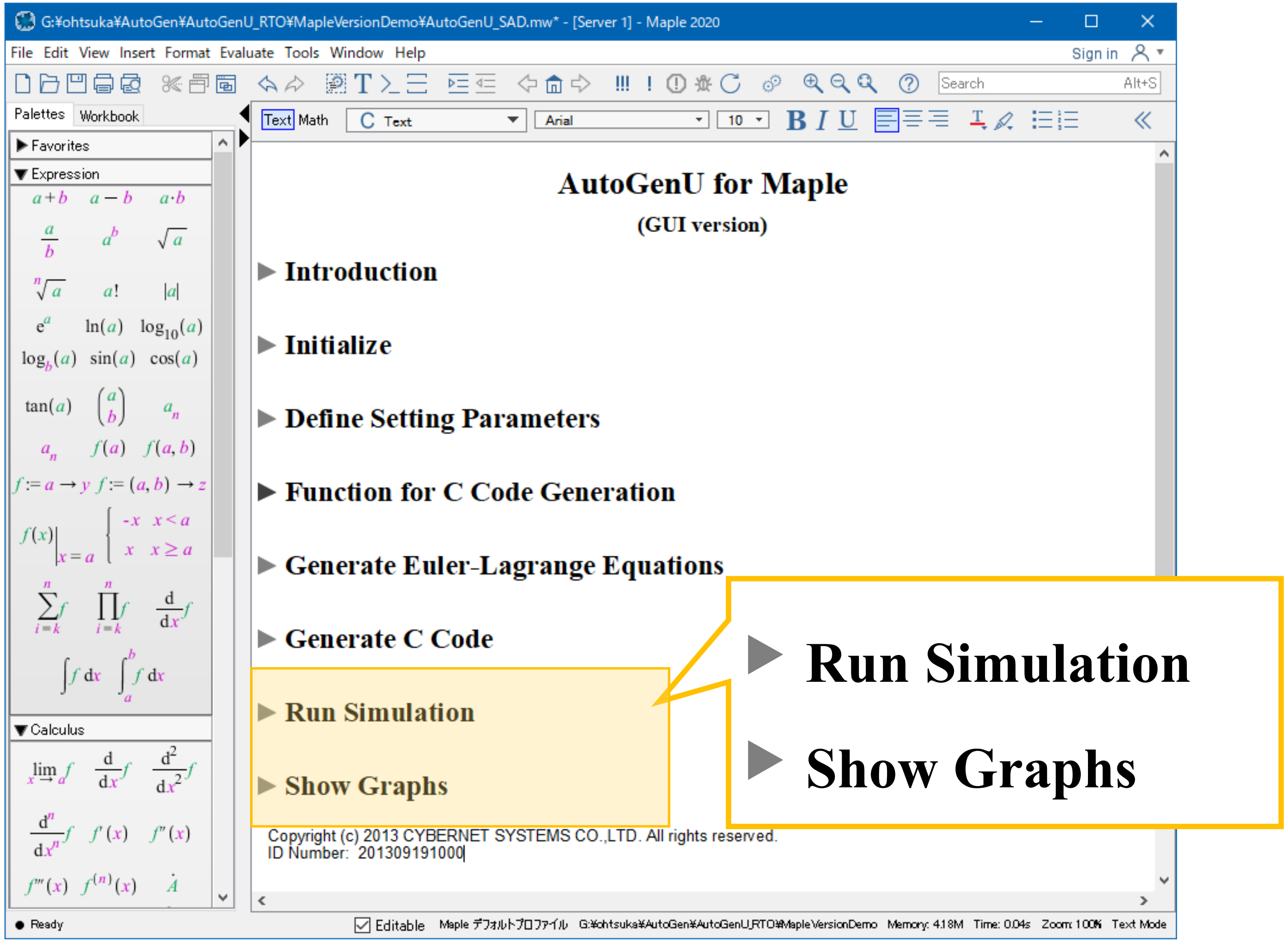
$$Hx := \begin{bmatrix} a\,lmd_2 + q_1 x_1 \\ b\,lmd_2 u_1 + q_2 x_2 + lmd_1 \end{bmatrix}$$

```
> Hu := Dv(H,uv);
```

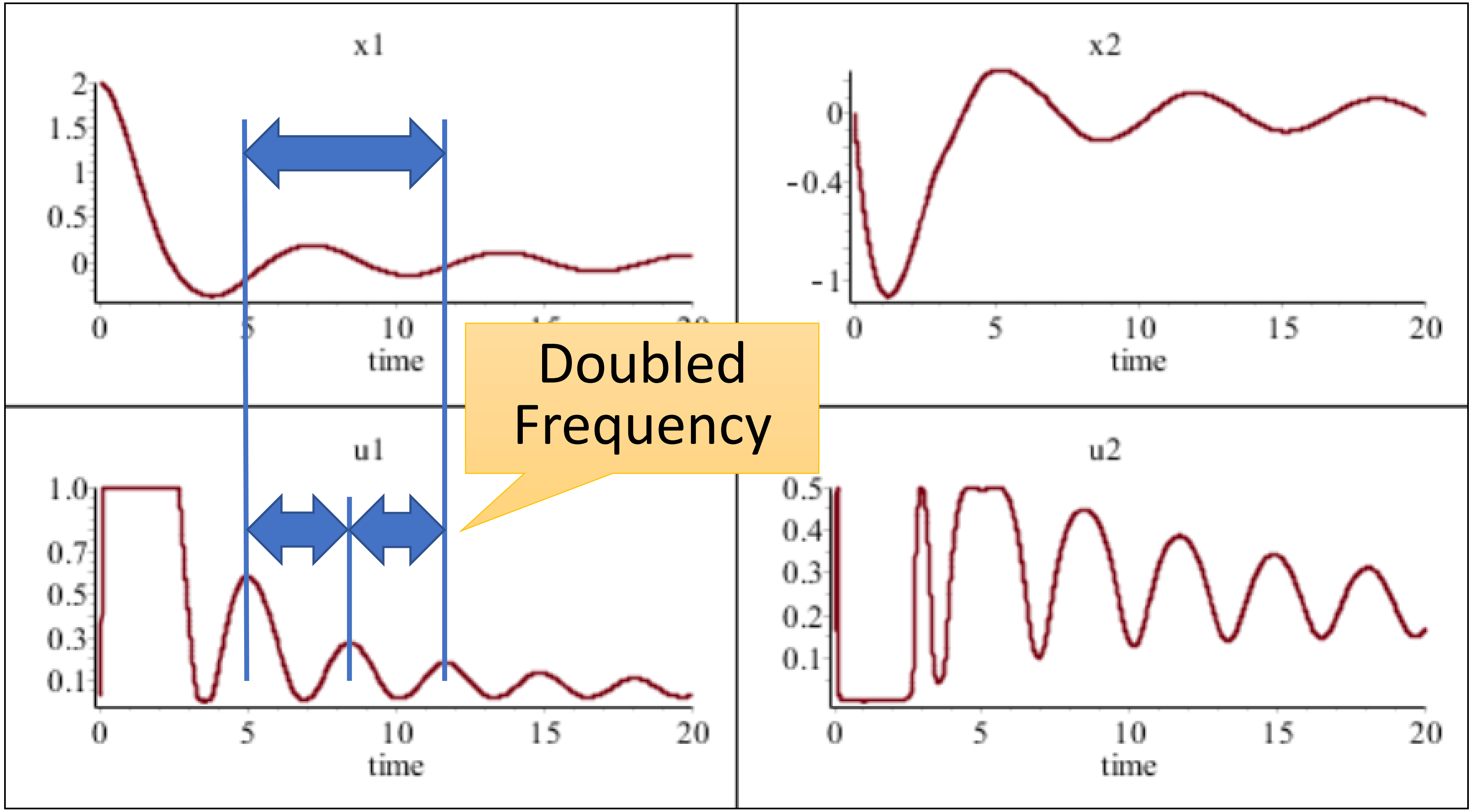
$$Hu := \begin{bmatrix} r_1 u_1 + lmd_2 b x_2 + u_3 (2 u_1 - umax) \\ 2 u_2 u_3 - r_2 \\ \left(u_1 - \frac{1}{2}umax\right)^2 + u_2^2 - \frac{1}{4}umax^2 \end{bmatrix}$$

Partial Derivatives

Maple Worksheet AutoGenU.mw

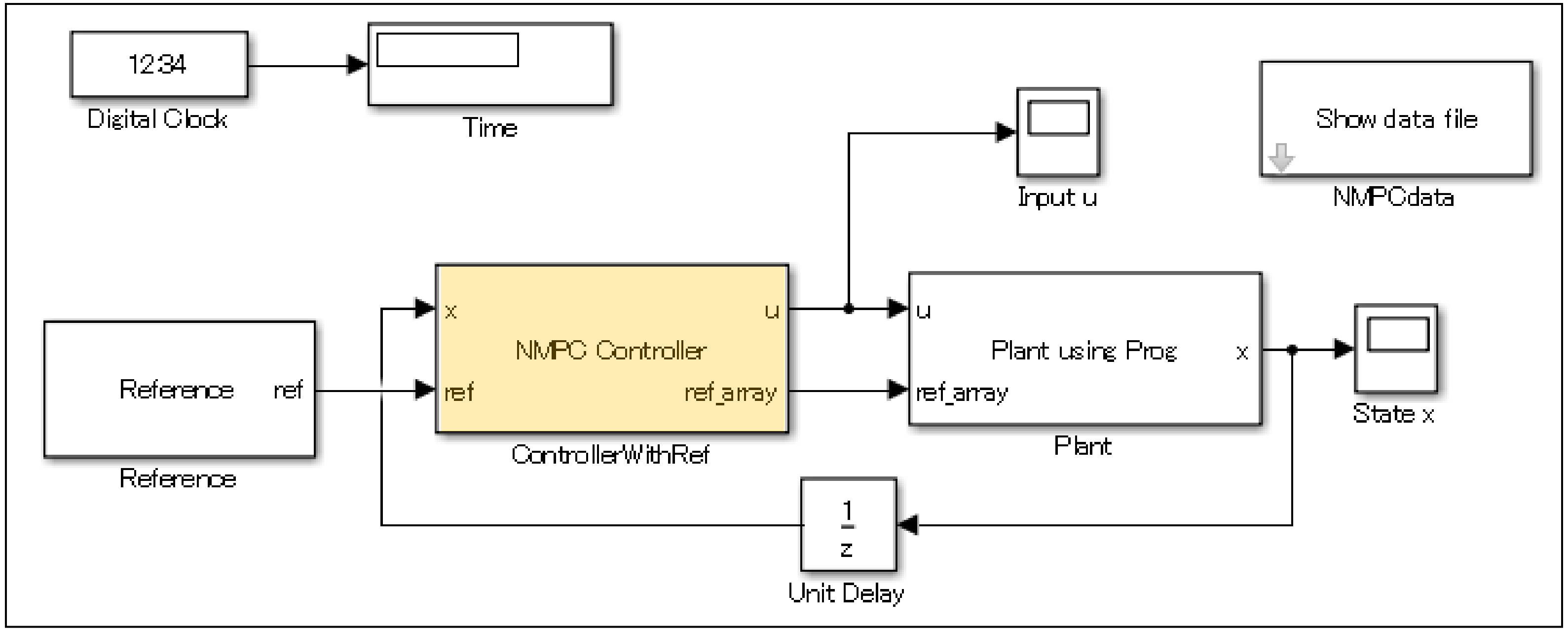


Simulation Result



MPC Design & Connector

- Automatic Generation of a **Simulink Block**
- Contact Cybernet Systems for license.
  Email: infomaple@cybernet.co.jp



Summary

- **Nonlinear model predictive control (NMPC)** is a very general framework for feedback control of nonlinear systems.
- **Real-time optimization (RTO)** is a key component of NMPC.
- **C/GMRES** is a continuation-based real-time algorithm for NMPC with successful applications.
- **AutoGenU for Maple** is a Maple worksheet for automatic C code generation of C/GMRES.
- RTO algorithms can be applied not only to MPC but also to various problems such as estimation, differential games, etc.

URLs

- Maplesoft Application Center: http://www.maplesoft.com/applications/view.aspx?SID=153555
- Maplesoft Webinar: https://www.youtube.com/watch?v=fVsYNjQfYUg
- MPC Design & Connector License: infomaple@cybernet.co.jp
- Mathematica Version: http://www.ids.sys.i.kyoto-u.ac.jp/~ohtsuka/code/autogenu/autogenu.zip
- Python Jupyter Notebook: https://github.com/mayataka/CGMRES

References (1/3)

- A. Bemporad, M. Morari, V. Dua, E. N. Pistikopoulos: The Explicit Linear Quadratic Regulator for Constrained Systems; Automatica, 38(1), 3-20 (2002)
- M. N. Zeilinger, C. N. Jones, M. Morari: Real-Time Suboptimal Model Predictive Control Using a Combination of Explicit MPC and Online Optimization; IEEE Trans. Autom. Contr., 56(7), 1524-1534 (2011)
- T. Ohtsuka, H. A. Fujii: Real-Time Optimization Algorithm for Nonlinear Receding-Horizon Control; Automatica, 33(6), 1147-1154 (1997)
- T. Ohtsuka: A Continuation/GMRES Method for Fast Computation of Nonlinear Receding Horizon Control; Automatica, 40(4), 563-574 (2004)
- M. Diehl, H. G. Bock, J. P. Schlöder: A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control; SIAM J. Contr. Optim., 43(5), 1714-1736 (2005)
- M. Alamir: Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes: A Parameterized Approach for Fast Systems, Springer (2006)
- D. DeHaan, M. Guay: Non-Convex Optimization and Robustness in Realtime Model Predictive Control; Int. J. Robust Nonlinear Contr., 17(17), 1634-1650 (2007)
- V. M. Zavala, L. T. Biegler: The Advanced-Step NMPC Controller: Optimality, Stability and Robustness; Automatica, 45(1), 86-93 (2009)
- K. Graichen, B. Käpernick: A Real-Time Gradient Method for Nonlinear Model Predictive Control; T. Zheng (Ed.): Frontiers of Model Predictive Control, InTech, 9-28 (2012)
- L. Stella, A. Themelis, P. Sopasakis, P. Patrinos: A Simple and Efficient Algorithm for Nonlinear Model Predictive Control; Proc. IEEE 56th Conf. Decision Contr., 1939-1944 (2017)
- H. Deng, T. Ohtsuka: A Parallel Newton-type Method for Nonlinear Model Predictive Control; Automatica, 109, paper 108560 (2019)
- T. Ohtsuka: Mode Predictive Control; Sys., Contr. & Info., 56(6), 310-312 (2012) (in Japanese)
- H. Seguchi, T. Ohtsuka: Nonlinear Receding Horizon Control of an Underactuated Hovercraft; Int. J. Robust Nonlinear Contr., 13(3-4), 381-398 (2003)

References (2/3)

- M. Hamamatsu, H. Kagaya, Y. Kohno: Application of Nonlinear Receding Horizon Control for Ship Maneuvering; Trans. SICE, 44(8), 685-691 (2008) (in Japanese)
- Y. Aoki, Y. Asano, A. Honda, N. Motooka, K. Hoshino, T. Ohtsuka: Nonlinear Model Predictive Control for Hexacopter with Failed Rotors based on Quaternions – Simulations and Hardware Experiments; Mechanical Engineering Journal, 8(5), paper 21-00204 (2021)
- K. Omoto, M. Doi, T. Ohtsuka: Integrated Optimization of Climbing Locomotion for a Humanoid Robot; Proc. 11th IFAC Symp. Nonlinear Contr. Sys., 1043-1048 (2019)
- K. Hirota, Y. Satoh, N. Motooka, Y. Asano, S. Kameoka, T. Ohtsuka: Nonlinear Receding-Horizon Differential Game between a Multirotor UAV and a Moving Object; Proc. 2017 Asian Contr. Conf., 2137-2142 (2017)
- Y. Azuma, T. Ohtsuka: Receding Horizon Nash Game Approach for Distributed Nonlinear Control; Proc. SICE Annual Conf. 2011, 380-384 (2011)
- Y. Soneda, T. Ohtsuka: Nonlinear Moving Horizon State Estimation with Continuation/Generalized Minimum Residual Method; J. Guidance, Contr., Dynamics, 28(5), 878-884 (2005)
- M. Kvasnica, P. Grieder, M. Baotić, M. Morari: Multi-Parametric Toolbox (MPC); R. Alur, G. J. Pappas (Eds.): HSCC 2004, Springer, 448-462 (2004)
- M. Herceg, M. Kvasnica, C. N. Jones, M. Morari: Multi-Parametric Toolbox 3.0; Proc. 12th European Contr. Conf., 502-510 (2013)
- T. Ohtsuka, A. Kodama: Automatic Code Generation System for Nonlinear Receding Horizon Control; Trans. SICE, 38(7), 617-623 (2002)
- T. Ohtsuka: A Tutorial on C/GMRES and Automatic Code Generation for Nonlinear Model Predictive Control; Proc. 14th European Contr. Conf., 73-86 (2015)
- S. Katayama, T. Ohtsuka: An Automatic Code Generator for Nonlinear Model Predictive Control with Jupyter, Proc. 21st IFAC World Congress, paper 236 (2020)
- B. Houska, H. J. Ferreau, M. Diehl: ACADO Toolkit – An Open-Source Framework for Automatic Control and Dynamic Optimization; Optimal Contr. Applications & Methods, 32(3), 298-312 (2011)

References (3/3)

- R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnki, T. Albin, R. Quirynen, M. Diehl: acados – A Modular Open-Source Framework for Fast Embedded Optimal Control; Math. Prog. Comp., (2021) (published online)
- J. Mattingley, S. Boyd: CVXGEN: A Code Generator for Embedded Convex Optimization; Optim. Eng., 13(1), 1-27 (2012)
- H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, M. Diehl: qpOASES: A Parametric Active-Set Algorithm for Quadratic Programming; Math. Prog. Comp., 6, 327-363 (2014),
- FORCES PRO, https://www.embotech.com/products/forcespro/overview/ (accessed Oct. 16, 2021)
- T. Englert, A. Völz, F. Mesmer, S. Rhein, K Graichen: A Software Framework for Embedded Nonlinear Model Predictive Control Using a Gradient-Based Augmented Lagrangian Approach (GRAMPC); Optim. Eng., 20(1), 769-809 (2019)
- H. Deng, T. Ohtsuka: ParNMPC - A Parallel Optimization Toolkit for Real-Time Nonlinear Model Predictive Control; Int. J. Contr. (2020) (published online)

NeurIPS2021 Tutorial
Real-Time Optimization for Fast and Complex Control Systems
Part 4

Advanced Topics in Model Predictive Control

Toshiyuki Ohtsuka
Department of Systems Science
Graduate School of Informatics
Kyoto University

Outline

Part 1: Introduction to Control Systems

Part 2: Optimal Control and Model Predictive Control

Part 3: Real-Time Optimization for Model Predictive Control

Part 4: Advanced Topics in Model Predictive Control

Outline of Part 4

- NMPC with State-Dependent Switches and State Jumps
- Parallel Algorithm for NMPC

S. Katayama, M. Doi, T. Ohtsuka: A Moving Switching Sequence Approach for Nonlinear Model Predictive Control of Switched Systems with State-Dependent Switches and State Jumps; *Int. J. Robust & Nonlinear Contr.*, 30(2), 719-740 (2020)
H. Deng, T. Ohtsuka: A Parallel Newton-type Method for Nonlinear Model Predictive Control; Automatica, 109, paper 108560 (2019)

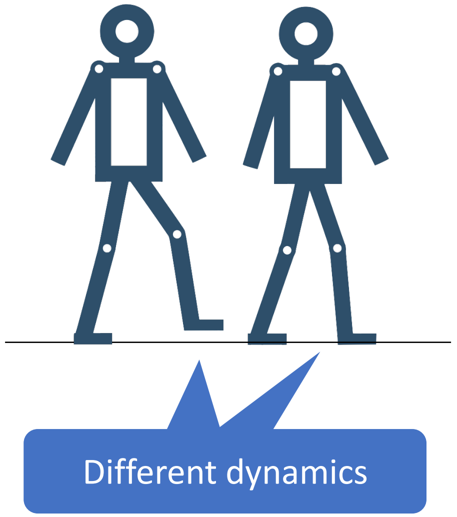
State-Dependents Switches of Dynamics

- Systems with state-dependent switches (SDS)

Controller

Control input

System with SDS

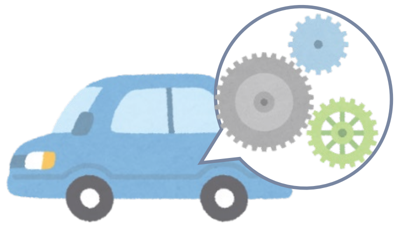


Cf. Systems with state-independent switches (SIS)

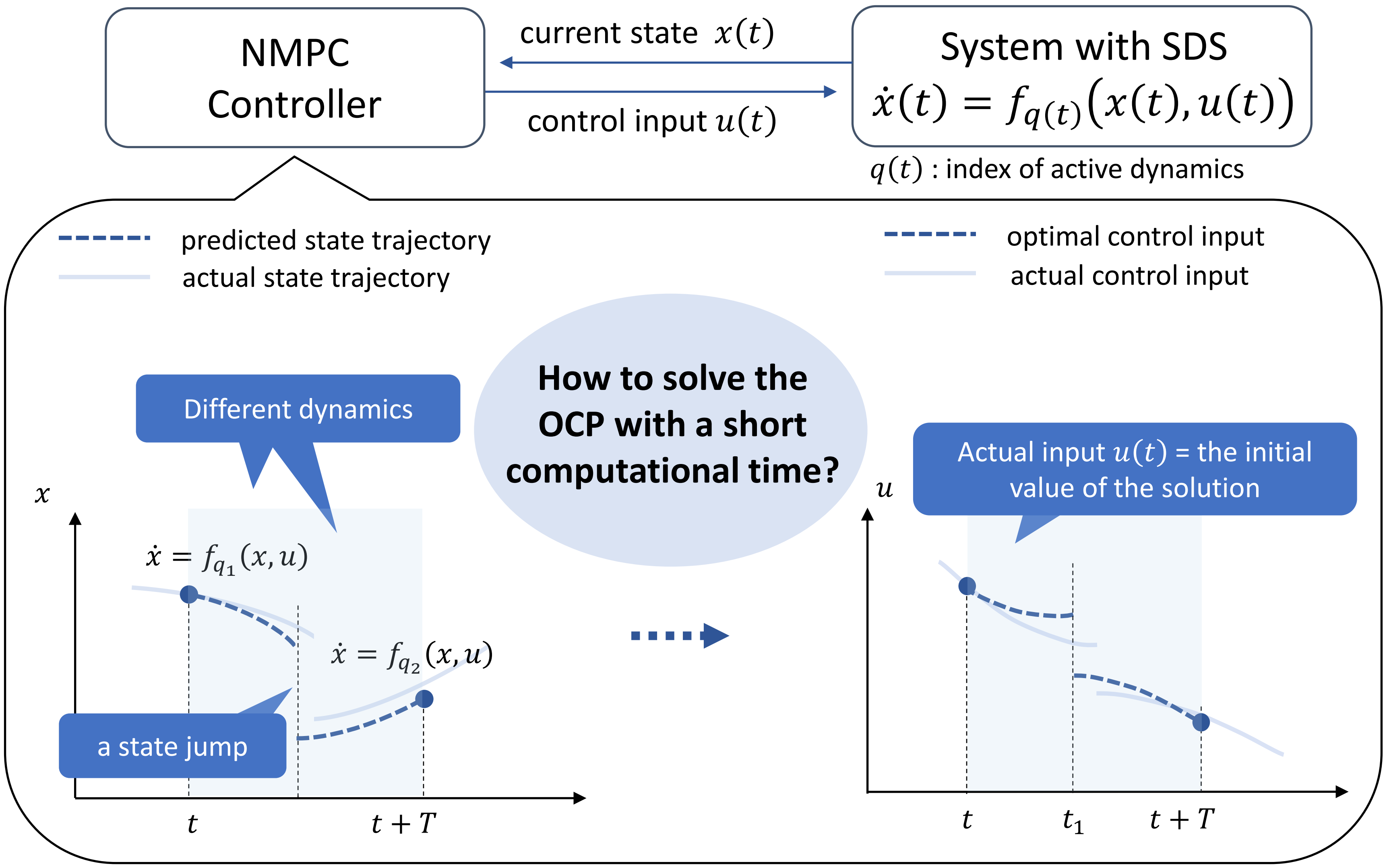
Controller

Control input
Switching signal

System with SIS

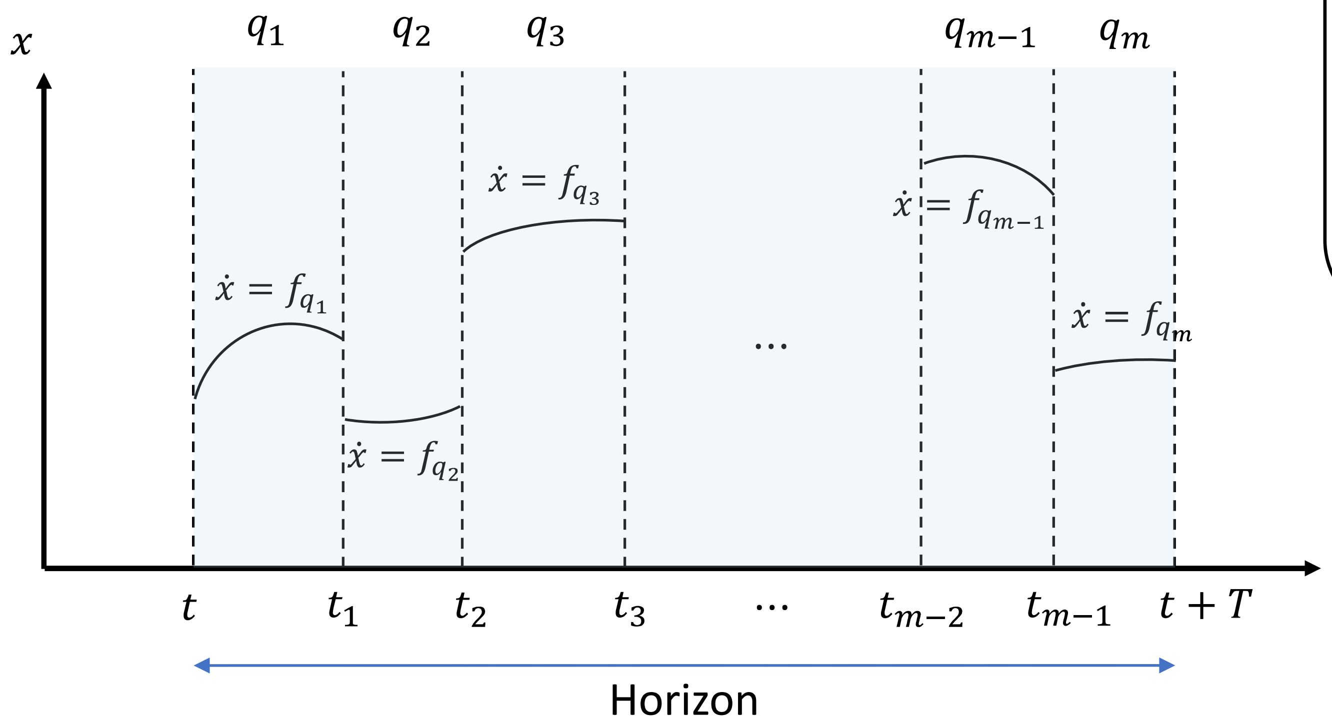


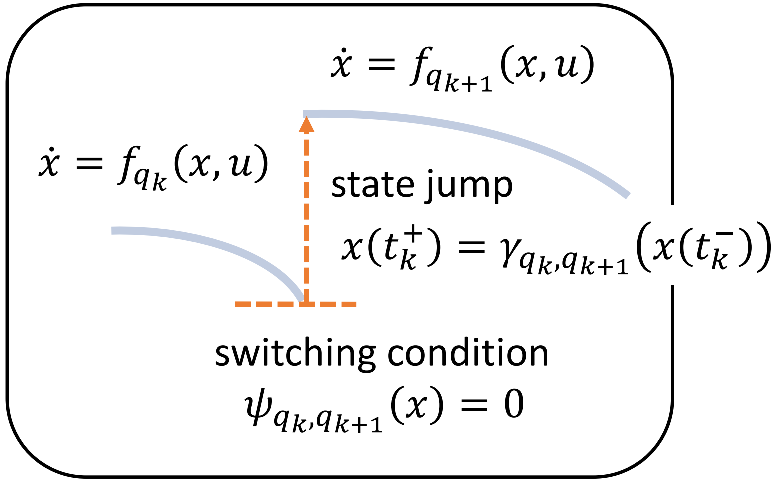
NMPC for Systems with SDS



OCP Formulation

- Switching sequence $\sigma = (q_1, \ldots, q_m)$





OCP Formulation (for a given switching sequence)

Find the control input $u(t')$ $(t \leq t' \leq t+T)$

minimizing

$$J = \varphi_{q_m}\big(x(t+T)\big) + \int_{t_{m-1}}^{t+T} L_{q_m}\big(x(t'), u(t')\big)dt' + \sum_{k}^{m} \int_{t_{k-1}}^{t_k} L_{q_k}\big(x(t'), u(t')\big)dt' + \int_{t}^{t_1} L_{q_1}\big(x(t'), u(t')\big)dt'$$

subject to

$$\frac{d}{dt'}x(t') = f_{q_1}\big(x(t'), u(t')\big) \quad (t \leq t' < t_1)$$

$$\psi_{q_1,q_2}\big(x(t_1^-)\big) = 0,\ x(t_1^+) = \gamma_{q_1,q_2}\big(x(t_1^-)\big)$$

$$\begin{cases} \dfrac{d}{dt'}x(t') = f_{q_k}\big(x(t'), u(t')\big) \quad (t_{k-1} < t' \leq t_k) \\ \psi_{q_k,q_{k+1}}\big(x(t_k^-)\big) = 0, x(t_k^+) = \gamma_{q_k,q_{k+1}}\big(x(t_k^-)\big) \end{cases}$$

$$\frac{d}{dt'}x(t') = f_{q_m}\big(x(t'), u(t')\big) \quad (t_{m-1} < t' \leq t+T)$$

OCP Formulation (for a given switching sequence)

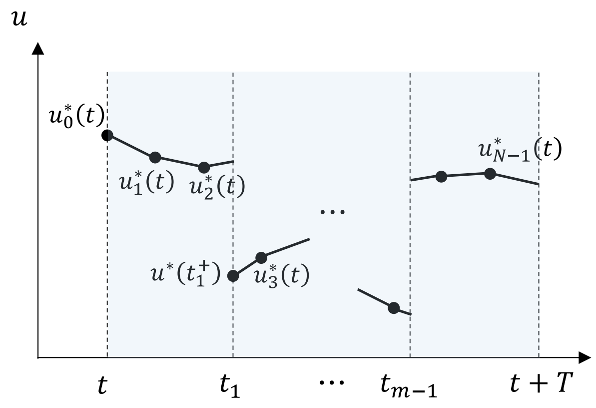
Find the variables to be determined

$$U(t) = \begin{bmatrix} u_0^*(t) \\ \vdots \\ u_{N-1}^*(t) \\ U_{q_1,q_2}(t) \\ \vdots \\ U_{q_{m-1},q_m}(t) \end{bmatrix}$$

Discretized control input sequence on the horizon

$$U_{q_k,q_{k+1}}(t) = \begin{bmatrix} u^*(t_k^+) \\ v_{q_k,q_{k+1}}^*(t) \\ t_k \end{bmatrix}$$

variables with respect to switch from q_k to q_{k+1}



satisfying

$$F(U(t), x(t), t) := \begin{bmatrix} \nabla_u H_{q_1}\left(x_0^*(t), u_0^*(t), \lambda_1^*(t)\right) \\ \vdots \\ \nabla_u H_{q_m}\left(x_{N-1}^*(t), u_{N-1}^*(t), \lambda_N^*(t)\right) \\ F_{q_1,q_2}(U(t), x(t), t) \\ \vdots \\ F_{q_{m-1},q_m}(U(t), x(t), t) \end{bmatrix} = 0,$$

State $x_i^*(t)$ and costate $\lambda_i^*(t)$ are functions of $U(t)$ and $x_0^*(0) = x(t)$ through ELE

Interior boundary conditions associated with switches

$\lambda \in \mathbb{R}^n$: Lagrange multiplier for $\dot{x} = f_q(x,u)$, $H_q(x,u,\lambda) = L_q(x,u) + \lambda^{\mathrm{T}} f_q(x,u)$: Hamiltonian

C/GMRES Method

Nonlinear equation for $U(t)$

Find $U(t)$ satisfying $F(U(t), x(t), t) = 0$

Continuation method $\frac{d}{dt}F = -\zeta F \ \ (\zeta > 0)$

Linear equation for $\dot{U}(t)$

Find $\dot{U}(t)$ satisfying $\frac{\partial F}{\partial U}\dot{U} = -\zeta F - \frac{\partial F}{\partial x}\dot{x} - \frac{\partial F}{\partial t}$

The GMRES method
(a fast Jacobian-free linear solver)

Update $U(t)$ on the basis of $\dot{U}(t)$, e.g., by

$$U(t + \Delta t) = U(t) + \dot{U}(t)\Delta t$$

Only one linear equation per update

How to determine switching sequence?

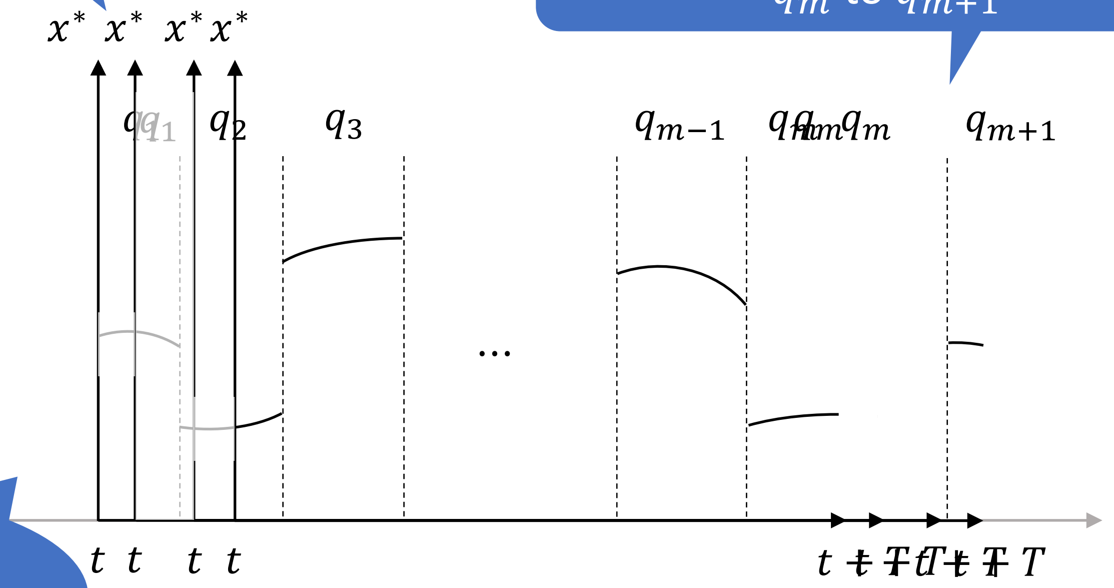
Moving Switching Sequence Approach

Remove $U_{q_1,q_2}(t)$ from $U(t)$

A switch from q_1 to q_2 occurs on the actual system

Modify the solution $U(t)$ taking the additional switch into account

Detect an additional switch from q_m to q_{m+1}

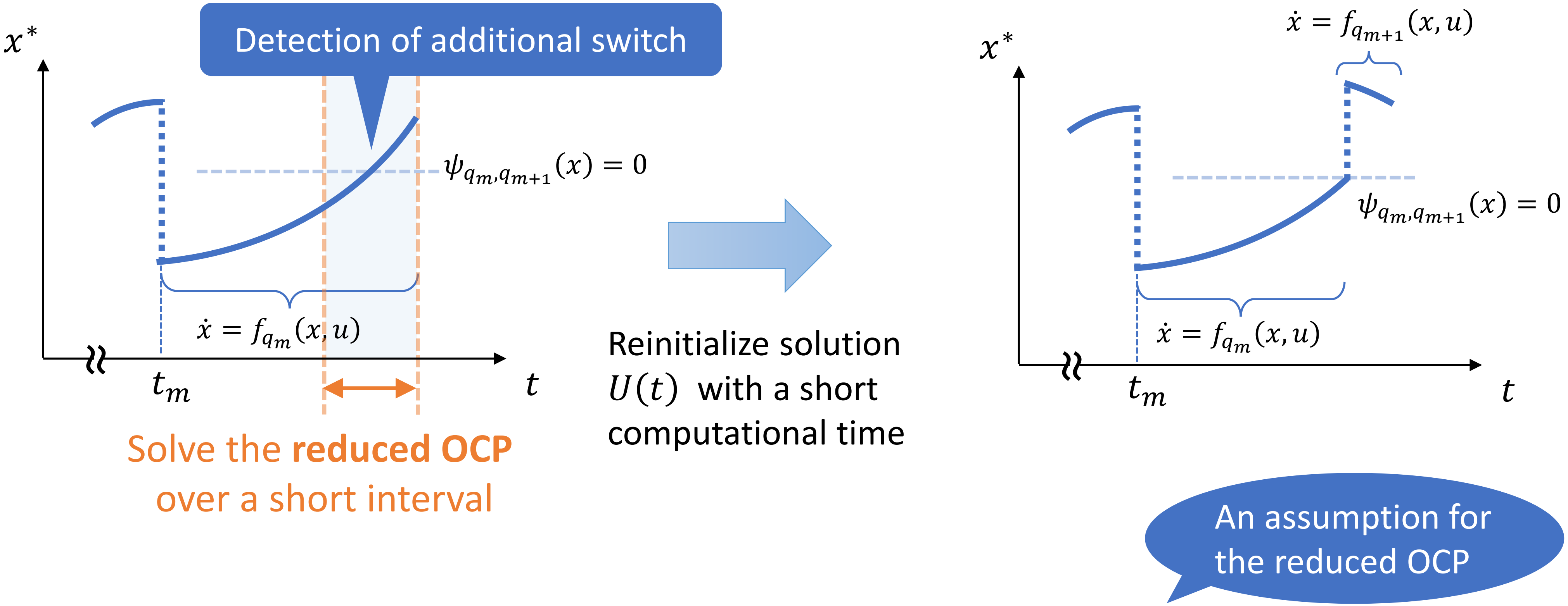


Switching sequence changes continuously

An assumption to justify this approach

Assumption 1 Switching sequence σ does not change except for the first and last elements

Reinitialization at Additional Switch



- **Assumption 2** Difference between the partial derivative of the terminal cost with respect to x before and after reinitialization is sufficiently small
- After reinitializing $U(t)$, we restart the continuation method for the OCP over the entire horizon

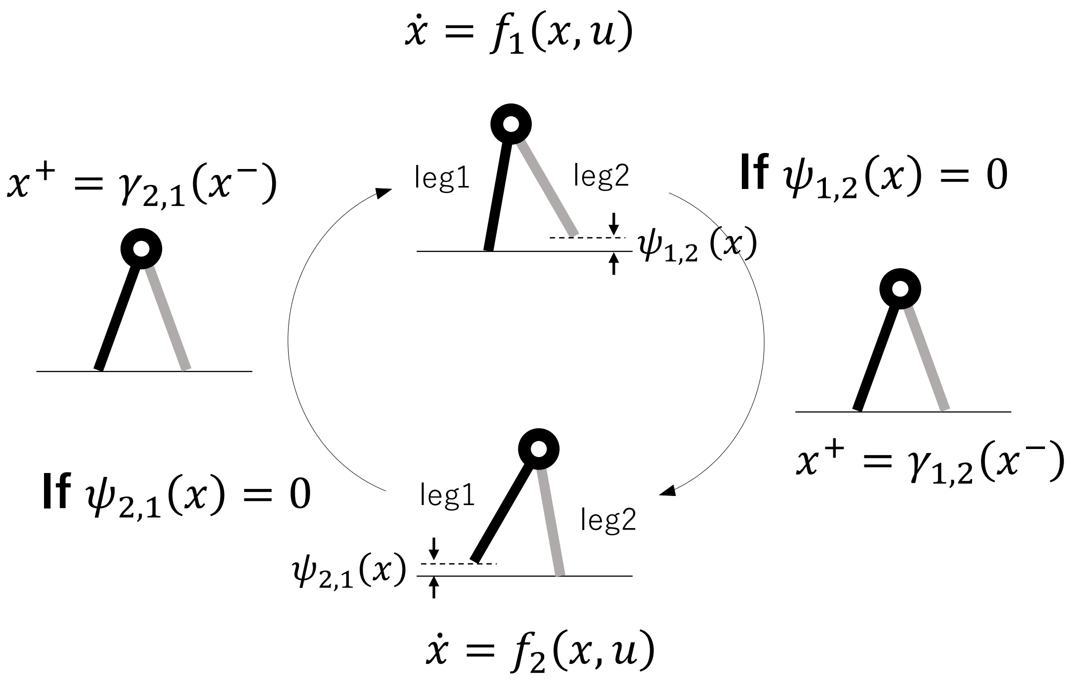
Summary of the Algorithm

At each time t

1. Compute the state trajectory on the horizon based on current $U(t)$ and σ
2. **`If`** an additional switch is detected **`then`**
3. Solve the reduced OCP to reinitialize $U(t)$
4. **`End if`**
5. Update $U(t + \Delta t)$ by the continuation method for the OCP over the entire horizon
6. **`If`** $t_1(t + \Delta t) < t + \Delta t$ **`then`**
7. Remove $U_{q_1,q_2}(t + \Delta t)$ from $U(t + \Delta t)$ and q_1 from σ
8. **`End if`**

Numerical Simulation

- Compass-like walking robot



- Performance index

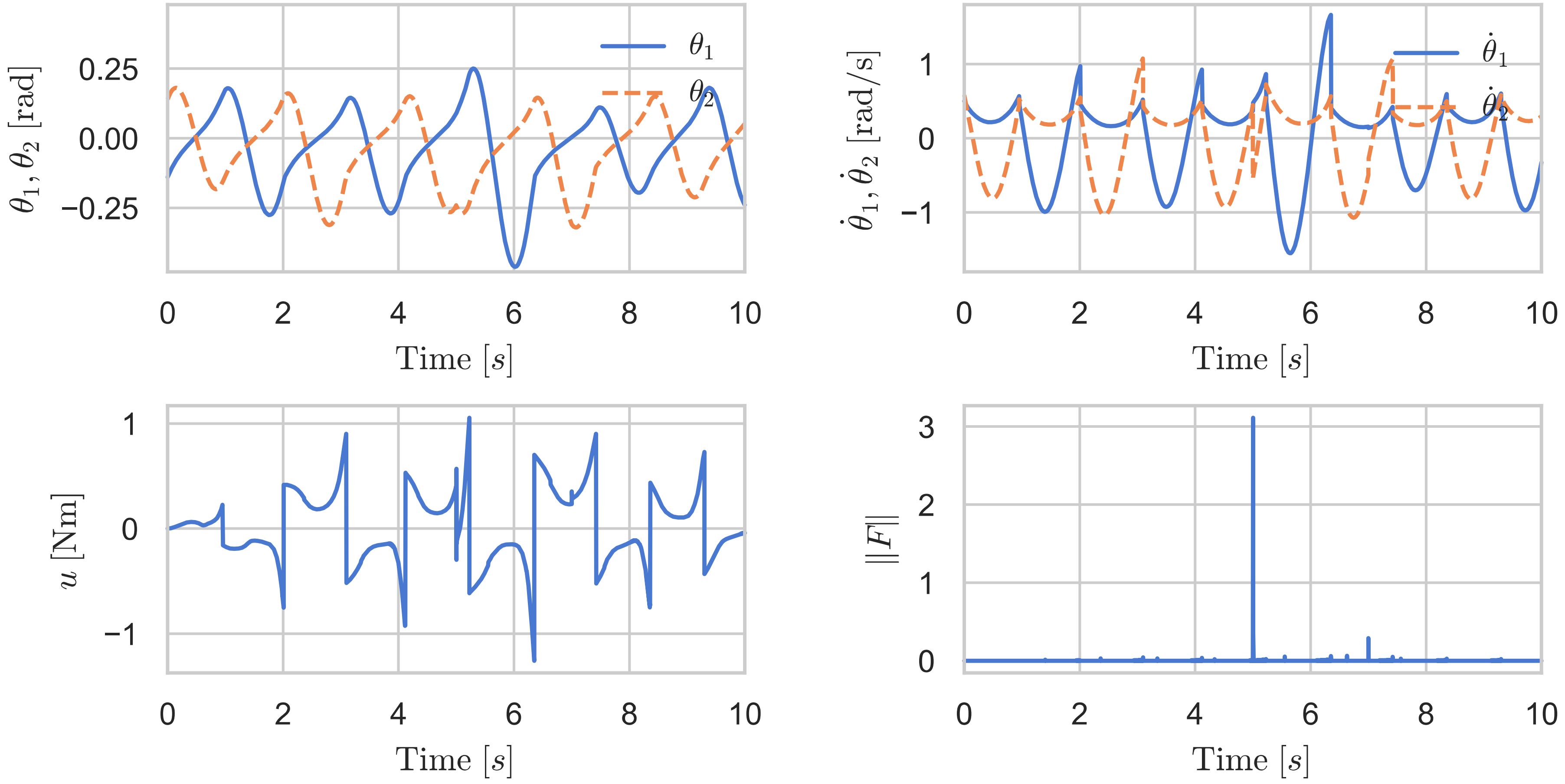
$$L_i(x,u) = \frac{1}{2}a_1\left(\dot{\theta}_i - v_{\mathrm{ref}}\right)^2 + \frac{1}{2}a_2(\theta_1+\theta_2)^2 + \frac{1}{2}ru^2, \qquad i = 1,2$$
$$\varphi_i(x) = 0, \qquad i = 1,2$$

where $a_1, a_2, r, v_{\mathrm{ref}} \in \mathbb{R}$

Numerical Simulation

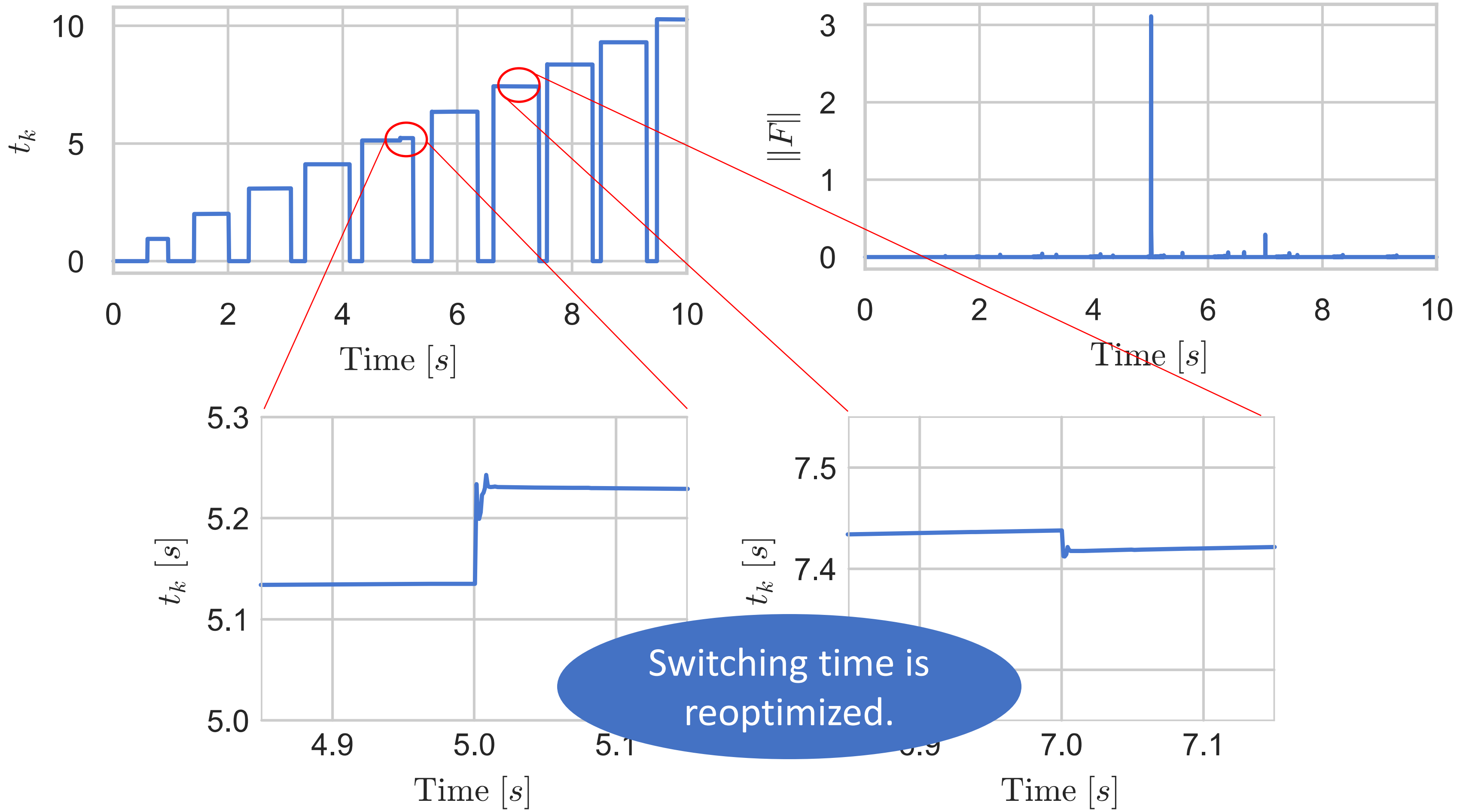
- Impulsive disturbances at 5 s and 7 s
- 0.14 ms per update

Numerical Simulation



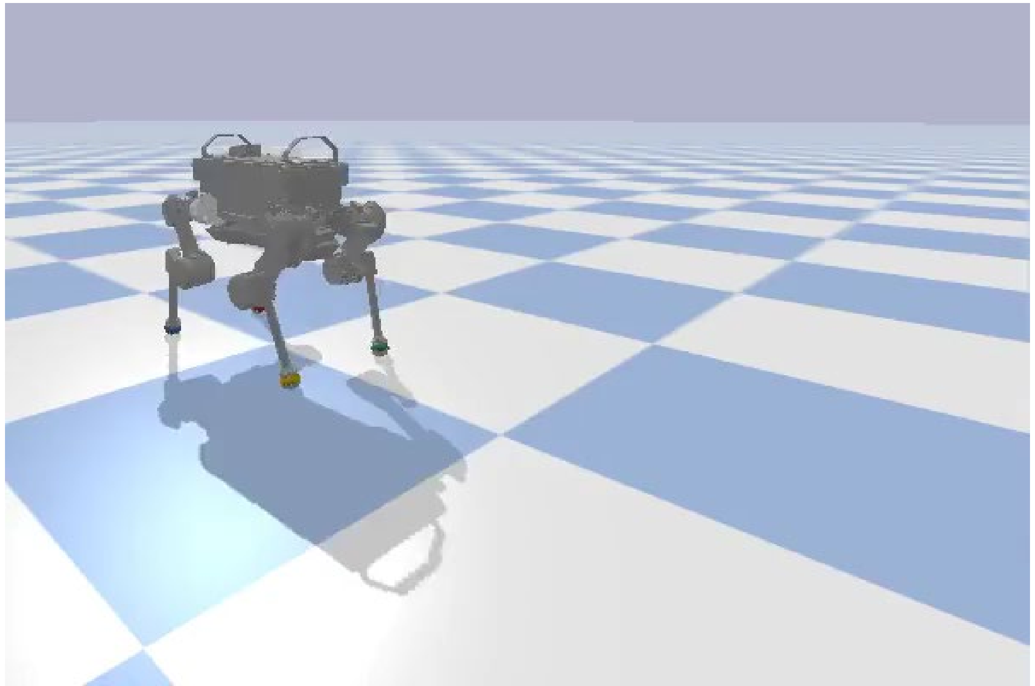
Impulsive disturbances at 5 s and 7 s

Numerical Simulation

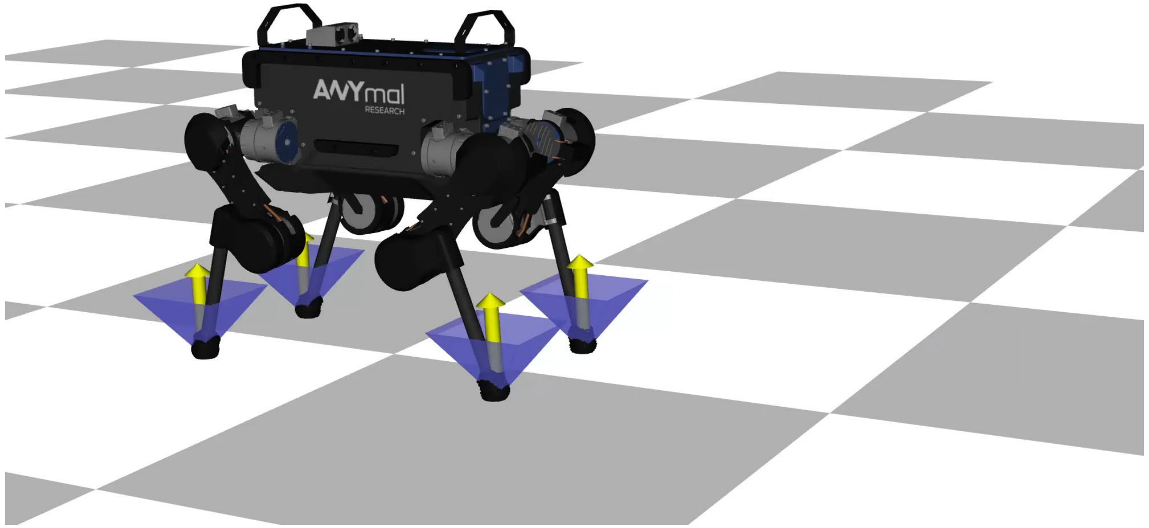


Impulsive disturbances at 5 s and 7 s

More Complex System



Trotting by NMPC
(1.2ms per update of input)



Jump by OCP
(0.3 s in total for optimization)

S. Katayama: https://github.com/mayataka/robotoc

Summary

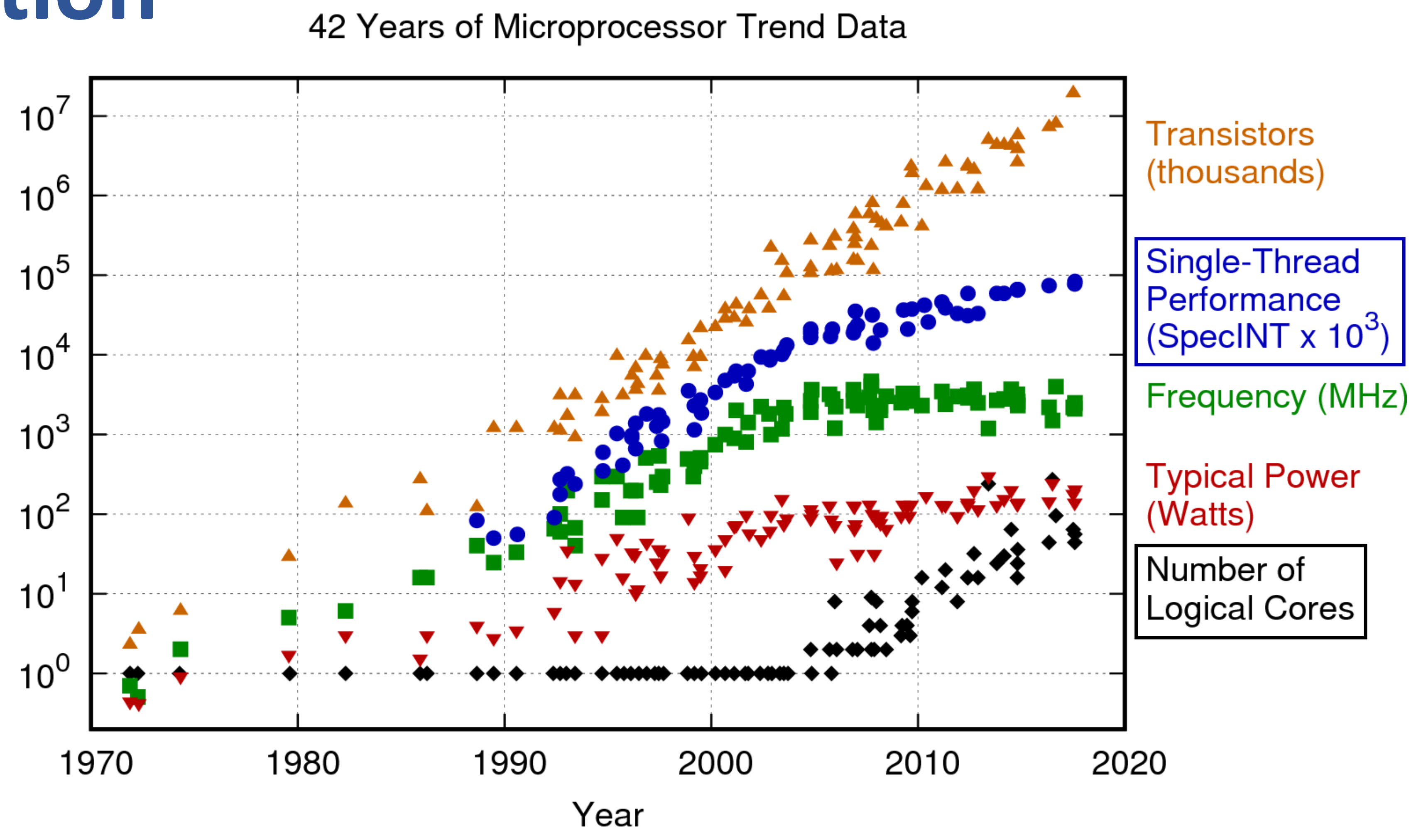
- RTO algorithm for NMPC of nonlinear systems with **state-dependent switches and state jumps**
  - ✓Based on an assumption that the **switching sequence is invariant except for the both ends of the horizon**
  - ✓Solve a reduced OCP to reinitialize the solution when an additional switch is detected at the end of the horizon
- Succeeded in controlling a compass-like walking robot even when there are disturbances
- Ongoing work: Modifications, extensions, and software tools

Outline of Part 4

- NMPC with State-Dependent Switches and State Jumps
- Parallel Algorithm for NMPC

S. Katayama, M. Doi, T. Ohtsuka: A Moving Switching Sequence Approach for Nonlinear Model Predictive Control of Switched Systems with State-Dependent Switches and State Jumps; *Int. J. Robust & Nonlinear Contr.*, 30(2), 719-740 (2020)
H. Deng, T. Ohtsuka: A Parallel Newton-type Method for Nonlinear Model Predictive Control; Automatica, 109, paper 108560 (2019)

Motivation



Parallel computing is a trend, however…

Motivation

Existing toolkits (algorithms) for NMPC:

- AutoGenU (C/GMRES)
- ACADO/acados (RTI/SQP)
- VIATOC (Gradient proj.)
- GRAMPC (Aug. Lagrangian & Gradient proj.)
- FORCES Pro (Interior-point)
- ……

Low degrees of parallelism

Speedup:
(Amdahl's law)

$$S(C) = \frac{1}{(1-p) + \frac{p}{C}}$$

Non-parallelizable part $1-p$ | Parallelizable part p

$\downarrow$ Parallelized on C cores

Non-parallelizable part $1-p$ | p/C

Motivation

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L\left(u(\tau), x(\tau), p(\tau)\right) d\tau \\
\text{s.t.} \quad & x(0) = \bar{x}_0, \\
& \dot{x}(\tau) = f\left(u(\tau), x(\tau), p(\tau)\right), \ \ \tau \in [0, T], \\
& C(u(\tau), x(\tau), p(\tau)) = 0, \ \ \tau \in [0, T], \\
& G(u(\tau), x(\tau), p(\tau)) \geq 0, \ \ \tau \in [0, T]
\end{aligned}
$$

Goal: Highly parallelizable algorithm & efficient toolkit

Key idea: Dividing the NMPC problem into subproblems along the prediction horizon

NMPC Problem

Inequality constraint elimination:

- To an equality constraint by introducing a dummy variable

$$G(u,x,p) \geq 0 \quad \longrightarrow \quad G(u,x,p) = v^2$$

- To an equality constraint and an additional cost (interior-point method)

$$G(u,x,p) \geq 0 \quad \longrightarrow \quad \begin{array}{l} G(u,x,p) = v \\ v \geq 0 \end{array} \quad \longrightarrow \quad \begin{array}{l} G(u,x,p) = v \\ -\rho \sum_j \log v_j \ \ (\rho > 0) \end{array}$$

Equality-constraint NMPC problem:

$$\begin{aligned} \min_{x(\cdot),u(\cdot)} & \int_0^T L\left(u(\tau), x(\tau), p(\tau)\right) d\tau \\ \text{s.t.} \quad & x(0) = \bar{x}_0, \\ & \dot{x}(\tau) = f\left(u(\tau), x(\tau), p(\tau)\right), \ \tau \in [0,T], \\ & C(u(\tau), x(\tau), p(\tau)) = 0, \ \tau \in [0,T] \end{aligned}$$

Couplings in NMPC Problem

- Parallelization is difficult because of the **couplings** introduced by the differential equation:

$$\dot{x}(\tau) = f\left(u(\tau), x(\tau), p(\tau)\right), \ \ \tau \in [0, T]$$

- Consider the optimization problem without the differential constraint:

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \ & \int_0^T L\left(u(\tau), x(\tau), p(\tau)\right) d\tau \\
\text{s.t.} \quad & x(0) = \bar{x}_0, \\
& \dot{x}(\tau) = f\left(u(\tau), x(\tau), p(\tau)\right), \ \ \tau \in [0, T], \\
& C(u(\tau), x(\tau), p(\tau)) = 0, \ \ \tau \in [0, T]
\end{aligned}$$

It can solved in parallel for each time stamp

Discretization for NMPC Problem

Generally, the discretized NMPC problem is solved:

Variables: $u(t),\ t \in [0,T]$

$x(t),\ t \in [0,T]$

$$\min_{x(\cdot),u(\cdot)} \int_0^T L\left(u(\tau), x(\tau), p(\tau)\right) d\tau$$

s.t. $x(0) = \bar{x}_0,$

$\dot{x}(\tau) = f\left(u(\tau), x(\tau), p(\tau)\right),\ \tau \in [0,T],$

$C(u(\tau), x(\tau), p(\tau)) = 0,\ \tau \in [0,T]$

Variables: $U = (u_1, u_2, \cdots, u_N)$

$X = (x_0, x_1, x_2, \cdots, x_N)$

$$\sum_{i=1}^{N} L(u_i, x_i, p_i)$$

$x_0 = \bar{x}_0$

Which disc. method should we use?

$C(u_i, x_i, p_i) = 0,\ i \in \{1, \cdots, N\}$

Different discretization methods lead to different problem structures/couplings:

- Forward Euler method $x_i = x_{i-1} + hf(u, x_{i-1}, p)$
- Backward Euler method $x_i = x_{i-1} + hf(u, x_i, p)$
- Runge-Kutta method
- ……

Discretization method with a minimum coupling?

Discretization with Reduced Couplings

Reverse-time discretization method:

$$\dot{x}(\tau) = f\left(u(\tau), x(\tau), p(\tau)\right), \ \tau \in [0, T]$$

$$\downarrow$$

$$x_{i-1} + \mathcal{F}(u_i, x_i, p_i) = 0, \ i \in \{1, \cdots, N\}$$

e.g., backward Euler method: $x_{i-1} \overbrace{- x_i + hf(u_i, x_i, p_i)}^{\mathcal{F}(u_i, x_i, p_i)} = 0$

Discretized problem (*N*-stage optimal control problem)

$$\min_{X,U} \sum_{i=1}^{N} L(u_i, x_i, p_i)$$

$$\begin{aligned} \text{s.t.} \quad & x_0 = \bar{x}_0, \\ & x_{i-1} + \mathcal{F}(u_i, x_i, p_i) = 0, \quad i \in \{1, \cdots, N\}, \\ & C(u_i, x_i, p_i) = 0, \quad i \in \{1, \cdots, N\} \end{aligned}$$

where: $X = (x_0, x_1, x_2, \cdots, x_N)$

$U = (u_1, u_2, \cdots, u_N)$

The couplings are "*reduced*" by using the reverse-time discretization method

KKT Conditions with Linear Couplings

- Karush-Kuhn-Tucker (KKT) conditions (necessary conditions for optimality):

Linear couplings

$$\begin{bmatrix} x_{i-1} + \mathcal{F}(u_i, x_i, p_i) \\ C(u_i, x_i, p_i) \\ \mathcal{H}_u^T(\lambda_i, \mu_i, u_i, x_i, p_i) \\ \lambda_{i+1} + \mathcal{H}_x^T(\lambda_i, \mu_i, u_i, x_i, p_i) \end{bmatrix} = 0, \ i \in \{1, \cdots, N\}$$

where: λ (**costate**) and μ : Lagrange multipliers

$$\mathcal{H}(\lambda, \mu, u, x, p) := L(u, x, p) + \lambda^T \mathcal{F}(u, x, p) + \mu^T C(u, x, p)$$

with initial and terminal conditions:

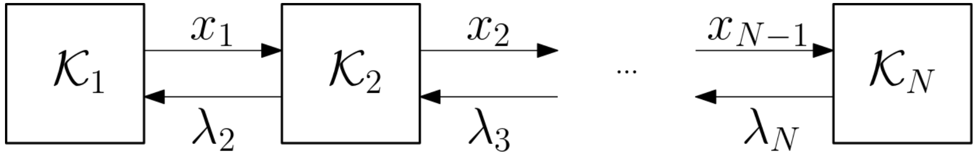
$$x_0 = \bar{x}_0 \quad \lambda_{N+1} = 0$$

- Compact form of the KKT conditions:

$$\mathcal{K}_i(x_{i-1}, s_i, \lambda_{i+1}) = 0, \ i \in \{1, \cdots, N\}$$
$$s_i := (\lambda_i, \mu_i, u_i, x_i)$$

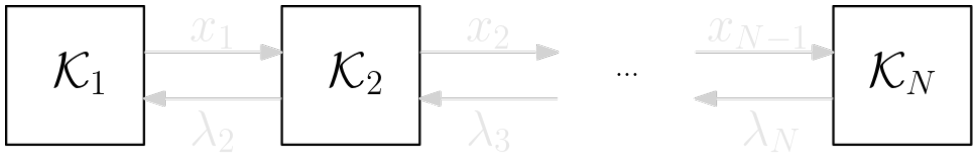
or

$$\mathcal{K}(S) = 0$$



Parallel Method: Motivation

Assume that the optimal coupling variables are known in advance:



Then, the KKT conditions can be solved in parallel:

$$\mathcal{K}_i(x_{i-1}^*, s_i, \lambda_{i+1}^*) = 0, \;\; i \in \{1, \cdots, N\}$$

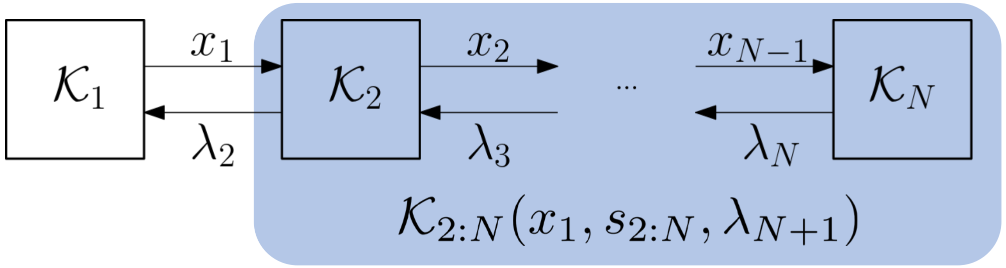
Single-stage problem

$$\begin{aligned} \min_{x_i, u_i} \; & L(u_i, x_i, p_i) + (\lambda_{i+1}^*)^T x_i \\ \text{s.t.} \quad & x_{i-1}^* + \mathcal{F}(u_i, x_i, p_i) = 0, \\ & C(u_i, x_i, p_i) = 0 \end{aligned}$$

However, it is impossible to know the coupling variables (x_{i-1}^* and λ_{i+1}^*) in advance!
Can we estimate them?

Linearity in Costate

Consider the estimation of λ_2^*:



- Solving $\mathcal{K}_{2:N}$ using Newton's method (*k*-th iteration):

$$s_{2:N}^{k+1}(x_1) = s_{2:N}^{k} - \underbrace{\left(\left.\frac{\partial \mathcal{K}_{2:N}}{\partial s_{2:N}}\right|_{(x_1, s_{2:N}^k, \lambda_{N+1})} \right)^{-1}}_{\text{Independent of } x_1} \mathcal{K}_{2:N}(x_1, s_{2:N}^k, \lambda_{N+1})$$

$$\mathcal{K}_{2:N}(x_1, s_{2:N}^k, \lambda_{N+1}) \;\downarrow\; \mathcal{K}_{2:N}(x_1^k, s_{2:N}^k, \lambda_{N+1}) + [x_1 - x_1^k \;\; 0]$$

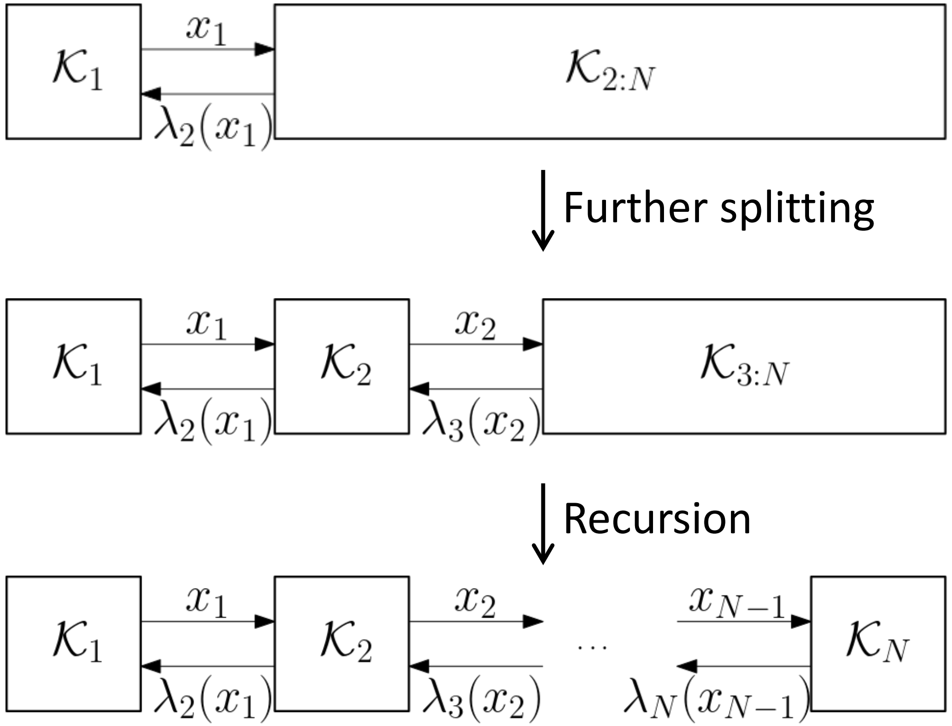
Linear coupling of x_1

- Extracting $\lambda_2(x_1)$ from $s_{2:N}^{k+1}(x_1)$:

$$\lambda_2^{k+1}(x_1) = \lambda_2^k - \Lambda_2^k(x_1 - x_1^k) - d_{\lambda_2}^k \quad \text{(estimation of } \lambda_2^*\text{)}$$

Backward Correction Method

Estimation of the costate variables:



$$\lambda_i^{k+1}(x_{i-1}) = \lambda_i^k - \Lambda_i^k(x_{i-1} - x_{i-1}^k) - d_{\lambda_i}^k$$

Then, we can solve $\mathcal{K}$ in a forward manner

Backward correction method

- Estimating $\lambda_{N:1}^*$ (backward)
- Solving $\mathcal{K}_{1:N}$ (forward)

Backward correction method
= Newton's method

$$S^{k+1} = S^k - \nabla\mathcal{K}(S^k)^{-1}\mathcal{K}(S^k)$$

Backward Correction Method

Backward correction method (structure-exploiting Newton's method):

Algorithm 1 k-th iteration

Input: $\bar{x}_0$, S^k
Output: S^{k+1}
for $i = N$ to 1 **do**

$$\mathcal{K}_i^k \leftarrow \mathcal{K}_i(x_{i-1}^k, s_i^k, \lambda_{i+1}^k)$$
$$J_i^k \leftarrow \nabla_{s_i} \mathcal{K}_i(x_{i-1}^k, s_i^k, \lambda_{i+1}^k)$$

KKT & Jacobian evaluations

$$H_i^k \leftarrow \left(J_i^k - \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+1}^k \end{bmatrix} \right)^{-1}$$
$$\Lambda_i^k \leftarrow \begin{bmatrix} 0 & 0 \\ I_{n_x} & 0 \end{bmatrix} H_i^k \begin{bmatrix} 0 & I_{n_x} \\ 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 \\ d_{\lambda_i}^k \end{bmatrix} \leftarrow \begin{bmatrix} 0 & 0 \\ I_{n_x} & 0 \end{bmatrix} H_i^k \left(\mathcal{K}_i^k - \begin{bmatrix} 0 \\ d_{\lambda_{i+1}}^k \end{bmatrix} \right)$$

Costate λ estimation (backward)

$$\lambda_i^{k+1}(x_{i-1}) = \lambda_i^k - \Lambda_i^k(x_{i-1} - x_{i-1}^k) - d_{\lambda_i}^k$$

Most computationally expensive part & in **serial**

end for
for $i = 1$ to N **do**

$$s_i^{k+1} \leftarrow s_i^k - H_i^k \left(\mathcal{K}_i^k - \begin{bmatrix} 0 \\ d_{\lambda_{i+1}}^k \end{bmatrix} + \begin{bmatrix} \Delta x_{i-1}^k \\ 0 \end{bmatrix} \right)$$

Iteration (forward)

end for

How can we parallelize this algorithm?

Approximation for Parallelization

- Estimation of λ_{i+1}^{*} in the backward correction method:

$$\lambda_{i+1}^{k+1}(x_i) = \lambda_{i+1}^{k} - \Lambda_{i+1}^{k}(x_i - x_i^k) - d_{\lambda_{i+1}}^{k}$$

$$H_i^k \leftarrow \left(J_i^k - \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+1}^{k} \end{bmatrix}\right)^{-1}$$

- Accurate estimation
- Time-consuming (**recursion**)

- A coarse estimation of λ_{i+1}^{*} based on the previous iteration's information:

$$\lambda_{i+1}^{k+1}(x_i) = \lambda_{i+1}^{k} - \Lambda_{i+1}^{k-1}(x_i - x_i^k) - d_{\lambda_{i+1}}^{k}$$

$$H_i^k \leftarrow \left(J_i^k - \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+1}^{k-1} \end{bmatrix}\right)^{-1}$$

- Approximate estimation
- Fast (in **parallel**)

Parallel Newton-type Method

Parallel algorithm:

Algorithm 1 k-th iteration

Input: $\bar{x}_0$, S^k, $\Lambda_{1:N}^{k-1}$
Output: S^{k+1}, $\Lambda_{1:N}^{k}$

for $i = 1$ to N **do in parallel**

$$\mathcal{K}_i^k \leftarrow \mathcal{K}_i(x_{i-1}^k, s_i^k, \lambda_{i+1}^k)$$
$$J_i^k \leftarrow \nabla_{s_i}\mathcal{K}_i(x_{i-1}^k, s_i^k, \lambda_{i+1}^k)$$
$$H_i^k \leftarrow \left(J_i^k - \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+1}^{k-1} \end{bmatrix}\right)^{-1}$$
$$\Lambda_i^k \leftarrow \begin{bmatrix} 0 & 0 \\ I_{n_x} & 0 \end{bmatrix} H_i^k \begin{bmatrix} 0 & I_{n_x} \\ 0 & 0 \end{bmatrix}$$

end for

- KKT & Jacobian evaluations
- Matrix factorization

- In **parallel**
- Degree of parallelism *N*

for $i = N$ to 1 **do**

$$\begin{bmatrix} 0 \\ d_{\lambda_i}^k \end{bmatrix} \leftarrow \begin{bmatrix} 0 & 0 \\ I_{n_x} & 0 \end{bmatrix} H_i^k \left(\mathcal{K}_i^k - \begin{bmatrix} 0 \\ d_{\lambda_{i+1}}^k \end{bmatrix}\right)$$

end for

for $i = 1$ to N **do**

$$s_i^{k+1} \leftarrow s_i^k - H_i^k \left(\mathcal{K}_i^k - \begin{bmatrix} 0 \\ d_{\lambda_{i+1}}^k \end{bmatrix} + \begin{bmatrix} \Delta x_{i-1}^k \\ 0 \end{bmatrix}\right)$$

end for

- In serial
- 2*N* matrix-vector multiplications

ParNMPC Toolkit

ParNMPC (**Par**allel **NMPC**)

- Open source
- Symbolic problem formulation
- Primal-dual interior-point method
- Automatic parallel code generation
  - C/C++
  - lib, dll, mex, exe

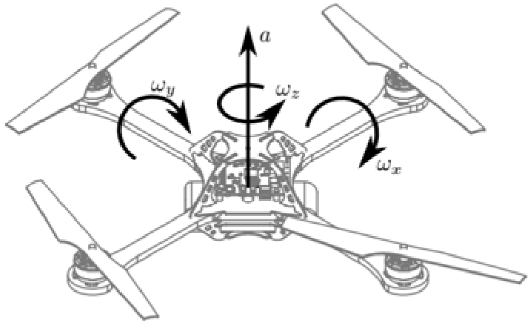
Options (version 1903)

- Hessian approximation
- High-order discretization
- Degree of parallelism configuration
- NLP techniques
  - Regularization
  - Line search
  - Barrier strategy
  - Warm start strategy
  - ……

- Fast
- Numerically robust
- Easy-to-use

H. Deng, T. Ohtsuka: ParNMPC - A Parallel Optimization Toolkit for Real-Time Nonlinear Model Predictive Control; *Int. J. Contr.* (2020)
https://github.com/deng-haoyang/ParNMPC

Numerical Experiment: Quadrotor



$$\ddot{X} = a(\cos\gamma \sin\beta \cos\alpha + \sin\gamma \sin\alpha)$$
$$\ddot{Y} = a(\cos\gamma \sin\beta \sin\alpha - \sin\gamma \cos\alpha)$$
$$\ddot{Z} = a\cos\gamma \cos\beta - g$$
$$\dot{\gamma} = (\omega_X \cos\gamma + \omega_Y \sin\gamma)/\cos\beta$$
$$\dot{\beta} = -\omega_X \sin\gamma + \omega_Y \cos\gamma$$
$$\dot{\alpha} = \omega_X \cos\gamma \tan\beta + \omega_Y \sin\gamma \tan\beta + \omega_Z$$

Quadrotor position control

- 9 states, 4 inputs
- T = 0.5 s, *N* = 24
- Quadratic cost function:

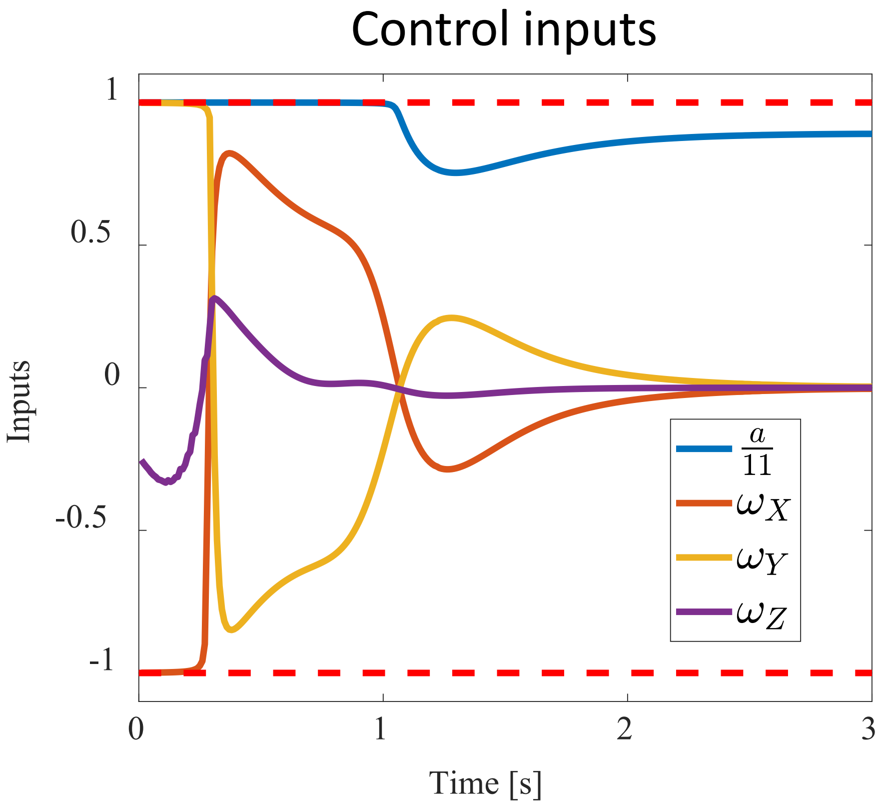
$$L(u, x, p) := \frac{1}{2}\left(\|x - x_r\|_Q^2 + \|u - u_r\|_R^2\right)$$

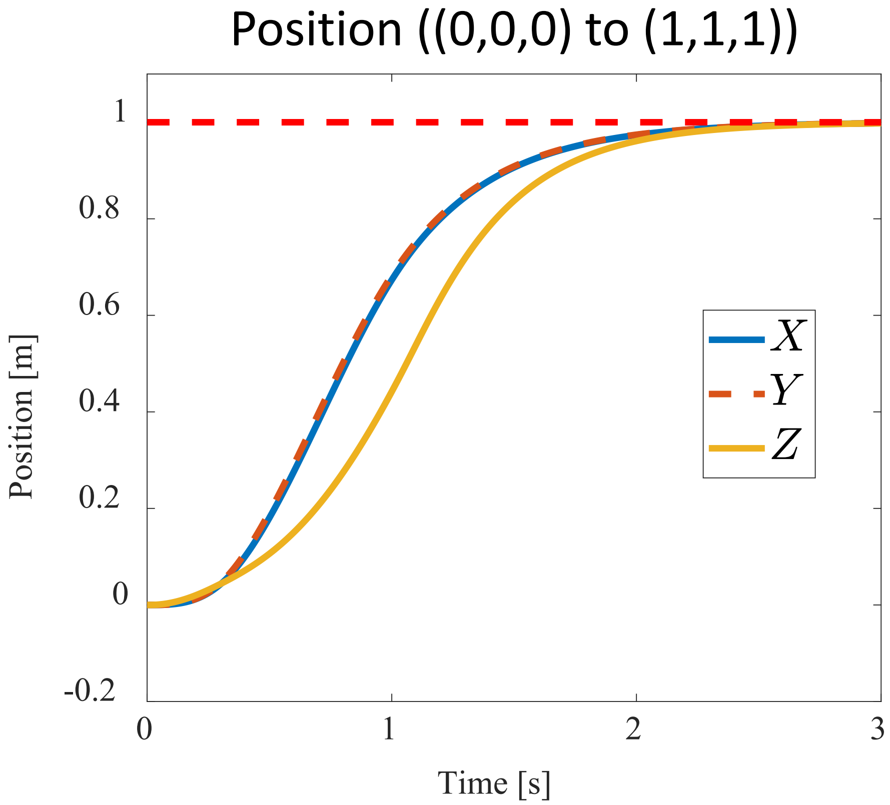
- Input constraints:

$$\begin{bmatrix} 0\ \mathrm{m/s^2} \\ -1\ \mathrm{rad/s} \\ -1\ \mathrm{rad/s} \\ -1\ \mathrm{rad/s} \end{bmatrix} \leq u \leq \begin{bmatrix} 11\ \mathrm{m/s^2} \\ 1\ \mathrm{rad/s} \\ 1\ \mathrm{rad/s} \\ 1\ \mathrm{rad/s} \end{bmatrix}$$

- Intel Core i9-8950HK @2.9 GHz, 6 cores

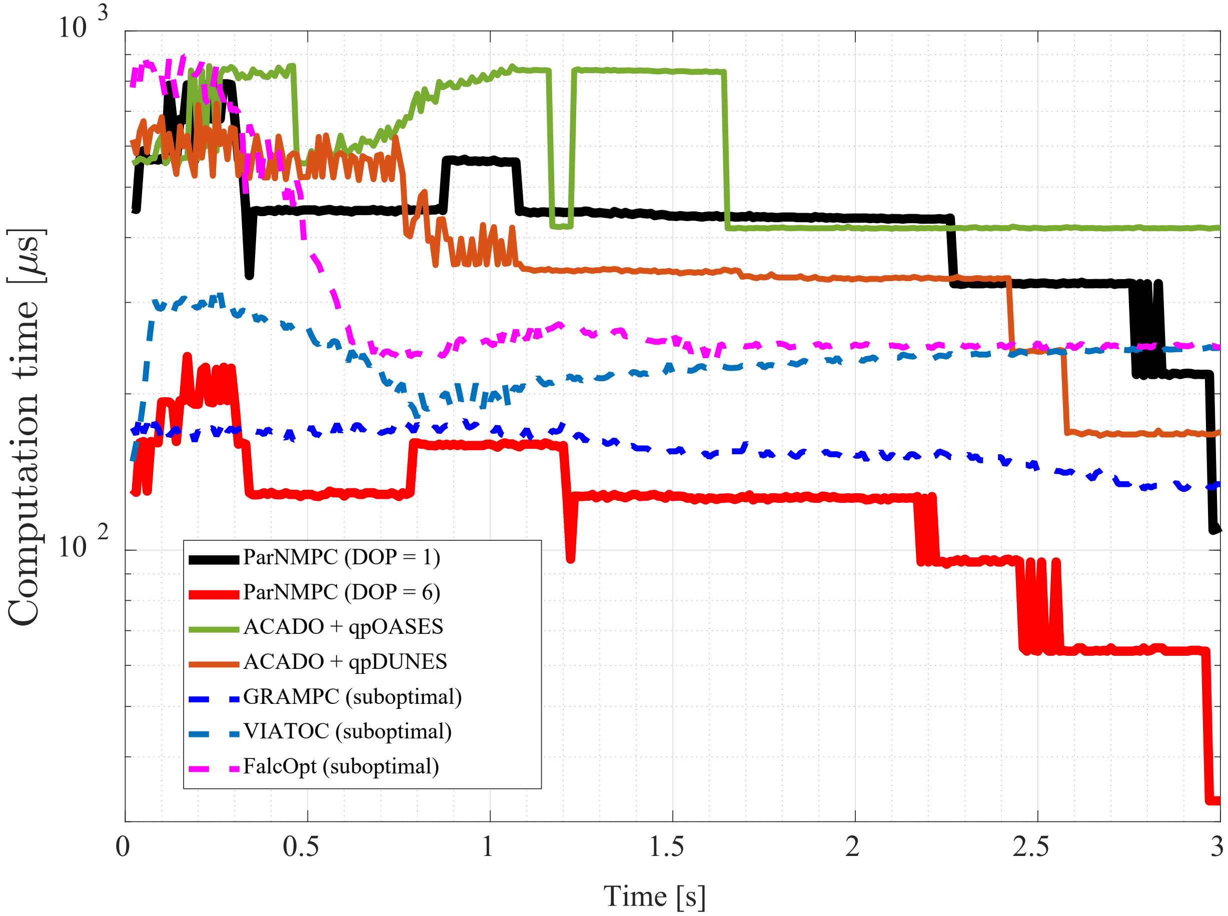
Numerical Experiment: Quadrotor





Numerical Experiment: Quadrotor

Comparison with state-of-the-art solvers



ParNMPC (1 core)

ParNMPC (6 cores)

ACADO + qpOASES

ACADO + qpDUNES

GRAMPC (suboptimal)

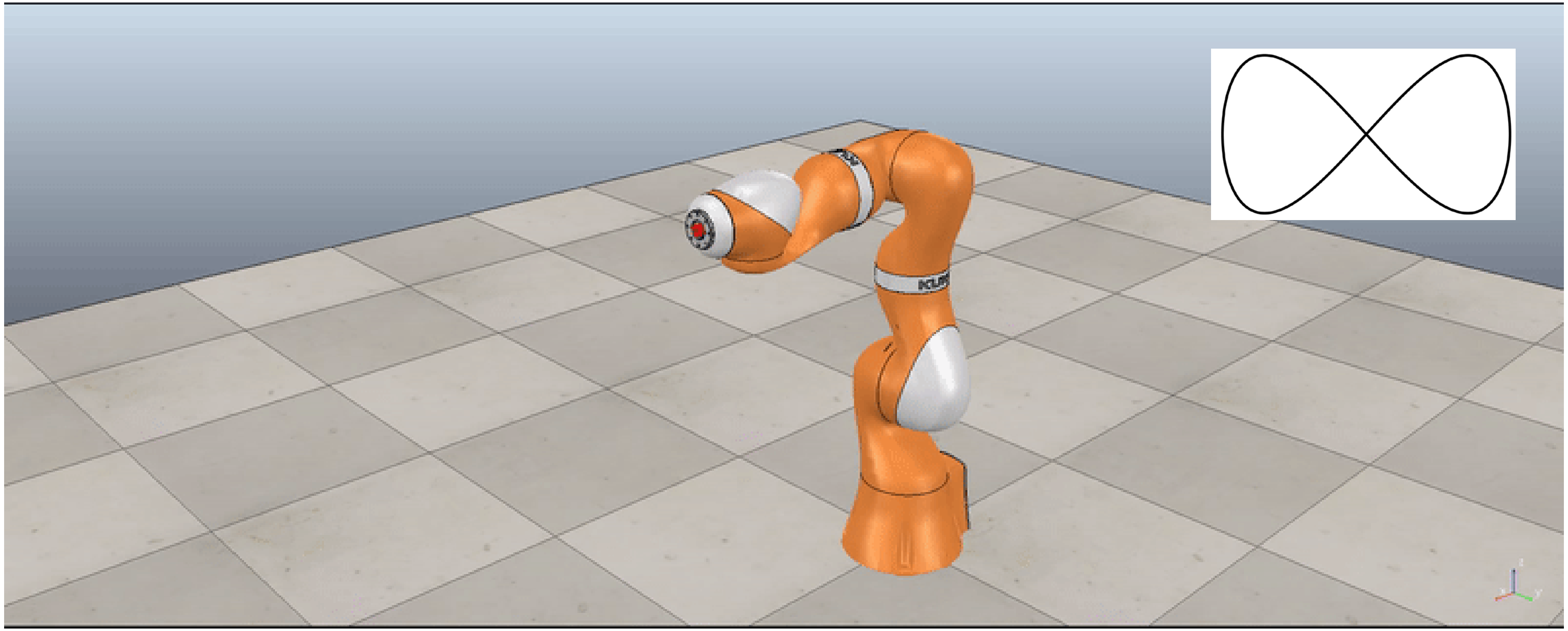
VIATOC (suboptimal)

FalcOpt (suboptimal)

4x speedup (6 cores)

Numerical Experiment: Robot Manipulator

Path tracking control of a 7 DOF robot manipulator:



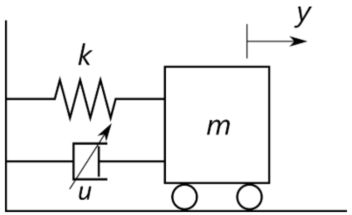
Constraints:
- Input torque
- Angular velocity

Results (T = 1 s, N = 18)
- **240 us/iteration**
- **5.6x** speedup (6 cores)

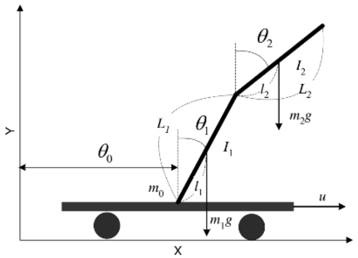
1 kHz real-time NMPC

Other Applications

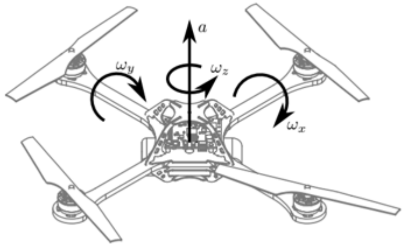
Successful applications of ParNMPC:



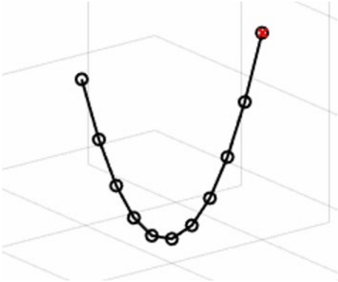
Semi-active damper



Double inverted pendulum

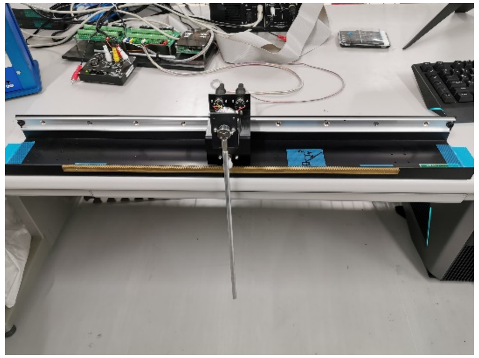


Quadrotor



Chain of masses

Helicopter



Inverted pendulum on a cart

7 DOF robot manipulator

...

Summary

- **Parallel Newton-type method**
  - ✓ Highly parallelizable
  - ✓ Fast rate of convergence
  - ✓ Applicable to general NMPC problems
- **ParNMPC (code generation tool)**
  - ✓ Easy-to-use
  - ✓ Automatic parallel code generation
  - ✓ Large number of applications
- **Future directions**
  - ✓ Reliable parallel computing
  - ✓ FPGA/GPU implementation

Conclusions

- Control systems are everywhere and provide motivations and opportunities for artificial intelligence.
- **Model predictive control (MPC) by real-time optimization** is very powerful as long as mathematical models are available.
- Most control systems have some mathematical models, but it is often difficult to find **appropriate models for control**. It is also often difficult to define **appropriate performance indices**.
- Machine learning would be particularly useful for fine tuning of mathematical models and/or performance indices.
- Application of artificial intelligence would be expected to the entire process of analysis and design of control systems.