16-665 Robot Mobility (Fall 2022)

Model Predictive and Adaptive Control

Sebastian Scherer & Micah Corah

October 27, 2022

(Based on Nathan Michael's slides 2020, 2019)

Carnegie Mellon
THE ROBOTICS INSTITUTE

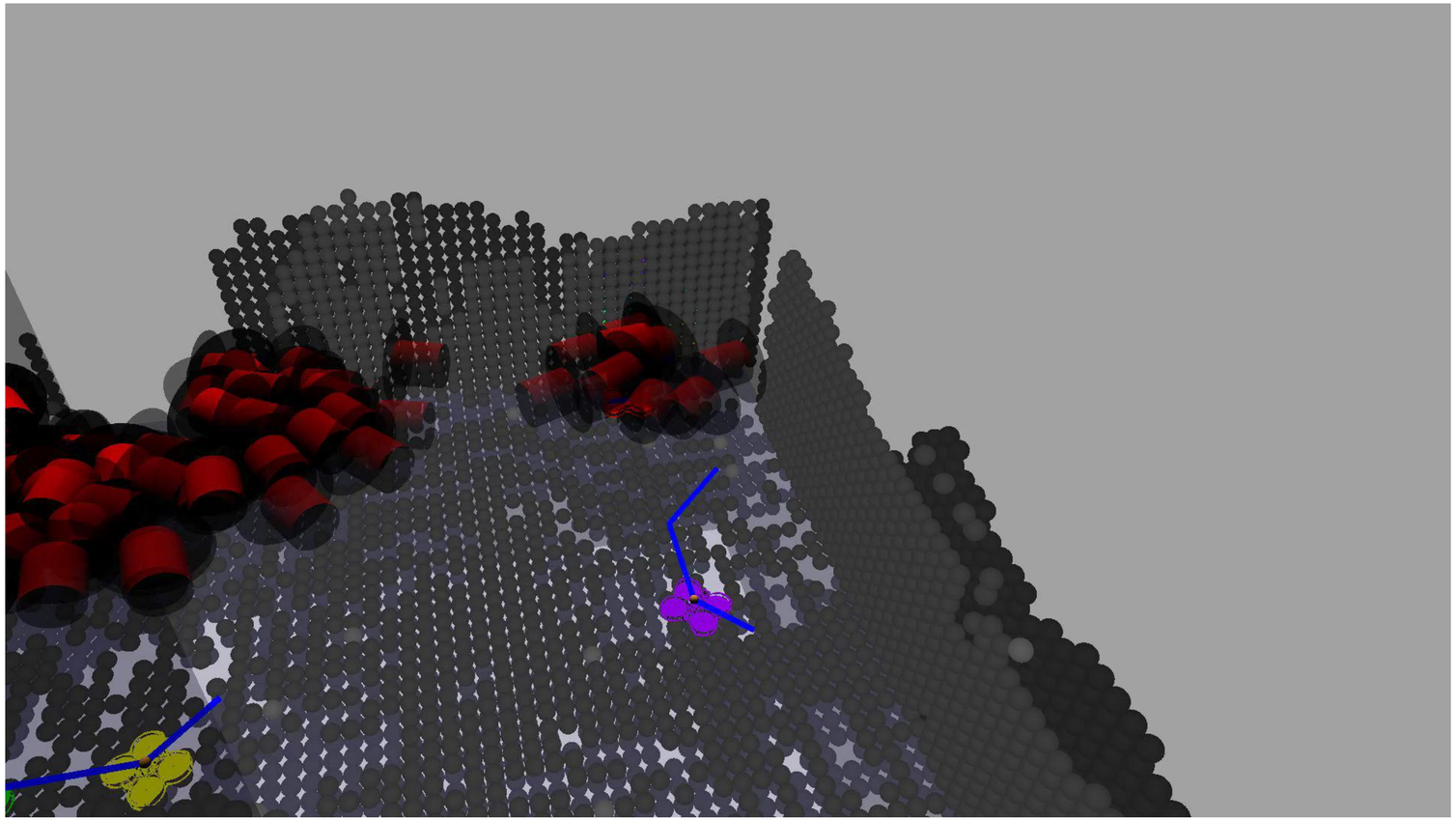
Lectures

- Oct 11: Introduction to Aerial Robotics: Challenges and Current State of the Art
- Oct 13: Multi-rotor and Fixed-Wing Dynamic Models
- Oct 25: Linear Observers and Control
- Oct 27: Model Predictive and Adaptive Control
- Nov 1: Trajectory Generation and Tracking
- Nov 3: Trajectory Generation 2

(Based on Nathan Michael's slides 2020, 2019)

Micah's Work: Receding-Horizon Multi-Robot Exploration

- Receding-horizon approach
  (like MPC)
- Safety/collision-free via terminal-state invariance
  (MPC-like)
- Higher-level perception-aware planning



Lecture Outline

- Quick review of LQR
- LQR Limitations
- Reactive and Predictive Control
- Explicit MPC
- Robust MPC
- Adaptive Control

Introduction to Model Predictive Control (MPC)

- **Optimization approach**
  - Model and simulate system *(need a good model!)*
  - Optimize control inputs w. *constraints* & *costs*
- **Uses**
  - Control complex & nonlinear systems
  - Satisfy strict safety guarantees
  - High or low level control: May control full state or provide reference trajectory to inner-loop control (*short horizon planning*)
- **Cons**
  - Can be slow/expensive, specific optimization problems can be limiting

Review: Linear Quadratic Regulator

LQR Problem Statement:

- Define the cost function:

$$J(x(t), u(t)) = \int_0^\infty (x(t)^\mathrm{T} Q x(t) + u(t)^\mathrm{T} R u(t)) dt$$

where Q and R are:

- *Symmetric*, i.e., $Q = Q^\mathrm{T}$ or $q_{ij} = q_{ji}$.
- *Positive definite*, i.e., $x^\mathrm{T} Q x > 0$ for any $x \in \mathbb{R}^n$ and $u^\mathrm{T} R u > 0$ for any $u \in \mathbb{R}^m$ (or all of the eigenvalues of Q and R are positive).

LQR Limitations

- We want to control systems in complex environments.
  - Safety and reliability are considerations.
    - What about:
      - Nonlinear dynamics?
      - External factors (e.g., wind and wheel slip)?
      - Constraints (e.g., speed limits, actuator saturation)?
- LQR limitations:
  - Can generate large control commands and high velocities.
  - Regulator (the “R” in LQR) - not tracking controller
  - Assumes a known plant model

Reactive versus Predictive Control

- Reactive control strategies such as PD and LQR respond to the current error:

$$e = x - x_d$$

  where x is the current state and x_d is the desired reference.

- PD controller:

$$u = -k_p e - k_d \dot{e} + k_i \int e$$

- LQR controller:

$$u = -Ke$$

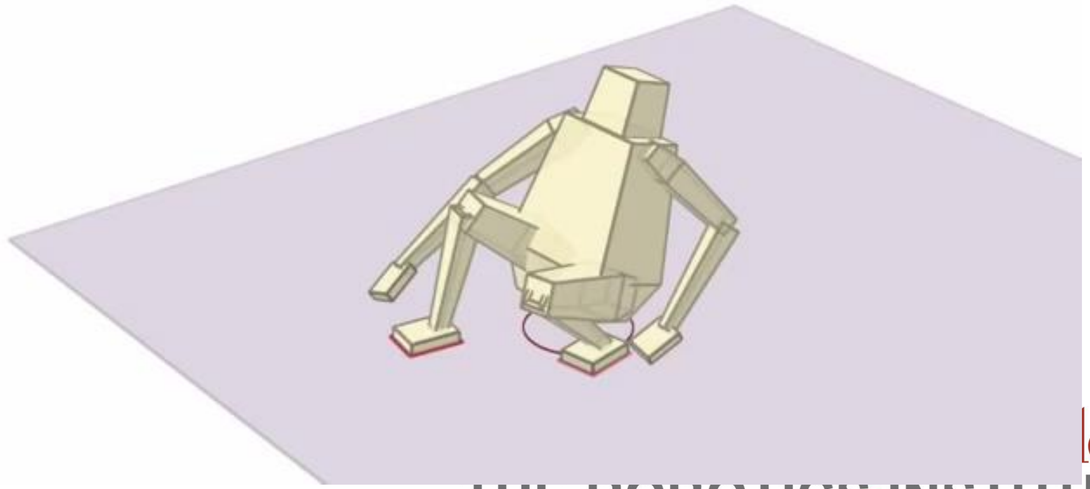
No input without error!

Reactive versus Predictive Control

- Reactive control strategies strive to reduce or *squash* error without consideration of future intended actions.
  - May expend significant effort to reduce current error with consideration of future error (assumes a static reference).
- Predictive control strategies consider future intended actions when computing the control update.

Predictive Control Example: LQR

- Example of infinite horizon LQR for *behavior discovery*.
- Reference/goal is the desired end-state behavior (e.g., standing up).
- System includes a high dimensional state and constraints (e.g., contact constraints, body constraints).
- *Very* computationally expensive.



Predictive Control Example: LQR

- We can make LQR behave as a predictive controller by making a few changes.
  - Previously discussed the *infinite-horizon* LQR formulation:

$$J(x(t), u(t)) = \int_0^\infty (x(t)^\mathrm{T} Q x(t) + u(t)^\mathrm{T} R u(t)) dt$$

  - Assumes a stationary reference ($x = 0$ in this case) and computes the optimal control strategy to converge to that reference over an infinite horizon.

Predictive Control Example: LQR

- Recall the discrete time formulation of infinite-horizon LQR:

$$J(x[k], u[k]) = \sum_{k=1}^{\infty} \left(x[k]^{\mathrm{T}} Q x[k] + u[k]^{\mathrm{T}} R u[k] \right)$$

- We can consider the implications of future actions (and state evolution) through consideration of constraints:

State equation (from LQR) $x_{k+1} = Ax_k + Bu_k$

State constraint $G_x x_k \leq g_x \longleftarrow g_x \in \mathbb{R}^n$

Input constraint $G_u u_k \leq g_u \longleftarrow g_u \in \mathbb{R}^m$

$G_x \in \mathbb{R}^{n \times n}$ $G_u \in \mathbb{R}^{m \times m}$

Predictive Control Example: LQR

- Constrained infinite-horizon LQR can be solved but solution must adhere to the input and state constraints for all time.
- The finite-horizon LQR problem is similar to the infinite-horizon LQR problem but with a finite summation (looking N steps into the future):

$$J(x[k],\, u[k]) = \sum_{k=1}^{N} \left(x[k]^{\mathrm{T}} Q x[k] + u[k]^{\mathrm{T}} R u[k] \right)$$

- The solution is an optimal controller at each time step:

$$u_1 = -K_1 x_1,\, u_2 = -K_2 x_2,\, \ldots$$

Predictive Control Example: LQR

- We can modify the reference to change over time by changing the formulation to consider a general tracking error:

$$e[k] = x[k] - x_d[k]$$

  - Note: Remarks from earlier lecture wrt tracking considerations still hold.
- The finite-horizon LQR formulation becomes:

$$J(e[k],\, u[k]) = \sum_{k=1}^{N} \left(e[k]^{\mathrm{T}} Q e[k] + u[k]^{\mathrm{T}} R u[k] \right)$$

Predictive Control Example: LQR

- Given a desired reference over time, we can compute the optimal controller for a finite horizon (N steps into the future).
- Step 1: Rewrite future states as a function of the current state and future inputs.

$$x[2] = Ax[1] + Bu[1]$$

$$x[3] = Ax[2] + Bu[2] = A(Ax[1] + Bu[1]) + Bu[2]$$

$$= A^2x[1] + ABu[1] + Bu[2]$$

- Step 2: Write the error $e[k]$ based on Step 1 and the reference over the next N steps.
- Step 3: Solve the discrete finite-horizon LQR problem for $u[1]$.

How do you solve the Problem?

$$\begin{bmatrix} x_0 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} 0 & \cdots & & \\ B & 0 & \cdots & \\ AB & B & 0 & \cdots \\ \vdots & \vdots & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} I \\ A \\ \vdots \\ A^N \end{bmatrix} x_0$$

express as $X = GU + Hx_0$, where $G \in \mathbf{R}^{Nn \times Nm}$, $H \in \mathbf{R}^{Nn \times n}$

$$\begin{aligned} J(U) &= \left\| \mathbf{diag}(Q^{1/2}, \ldots, Q^{1/2}, Q_f^{1/2})(GU + Hx_0) \right\|^2 \\ &+ \left\| \mathbf{diag}(R^{1/2}, \ldots, R^{1/2})U \right\|^2 \end{aligned}$$

=> Big least squares problem

Predictive Control Example: LQR

- The solution to the discrete finite-horizon LQR is actually *N* optimal controllers (for time *k* through *k*+*N*). We will only apply the first one (i.e., *u*[*k*]) and then resolve for the next optimal controller at the next time step (for *u*[*k*+1]).
- We can encode the prior constraints into the formulation:

$$\min_{u[k]} \quad \sum_{k=1}^{N} \left(e[k]^{\mathrm{T}} Q e[k] + u[k]^{\mathrm{T}} R u[k] \right)$$

$$\text{s.t.} \quad x_{k+1} = A x_k + B u_k$$

$$G_x x_k \leq g_x$$

$$G_u u_k \leq g_u$$

subject to (s.t.)

Predictive Control Example: LQR

- Let us decode this mathematical statement.

$$\begin{aligned} \min_{u[k]} \quad & \sum_{k=1}^{N} \left(e[k]^{\mathrm{T}} Q e[k] + u[k]^{\mathrm{T}} R u[k] \right) \\ \text{s.t.} \quad & x_{k+1} = A x_k + B u_k \\ & G_x x_k \leq g_x \\ & G_u u_k \leq g_u \end{aligned}$$

- It reads: "Find the $u[k]$ that minimizes the cost function from LQR while also ensuring that the state equation and state/input constraints hold."

Predictive Control Example: LQR

- The statement is similar to the discrete finite horizon LQR problem but differs in the introduction of the constraints (so we could call it the constrained discrete finite horizon LQR problem).

$$\begin{aligned} \min_{u[k]} \quad & \sum_{k=1}^{N} \left(e[k]^{\mathrm{T}} Q e[k] + u[k]^{\mathrm{T}} R u[k] \right) \\ \text{s.t.} \quad & x_{k+1} = A x_k + B u_k \\ & G_x x_k \leq g_x \\ & G_u u_k \leq g_u \end{aligned}$$

- That name is too long - so we call it *Model Predictive Control*.

Model Predictive Control

- The general form of Model Predictive Control (MPC) is (from the prior slide):

$$\begin{aligned} \min_{u[k]} \quad & \sum_{k=1}^{N} \left(e[k]^{\mathrm{T}} Q e[k] + u[k]^{\mathrm{T}} R u[k] \right) \\ \text{s.t.} \quad & x_{k+1} = A x_k + B u_k \\ & G_x x_k \leq g_x \\ & G_u u_k \leq g_u \end{aligned}$$

- It is an optimization problem that you solve for $u[k]$ at *each time step*.

MPC: Interactive Activity

Discuss in pairs (2 min)

- How would you/have you applied MPC or similar techniques?
- Key questions or challenges regarding applying MPC

Afterward: Questions, discuss as a class (2 min)

Model Predictive Control: Optimization Problems

- MPC requires an optimization solution at each time step
- Optimization problems take many forms depending on the nature of the objective function and constraints
- LQR objective is a sum of quadratic terms (“LQ” in LQR) and constraints are linear inequalities
  - This forms a quadratic program
- Optimization programs (minimize subject to…)
  - **LP:** Linear Program, linear objective and constraints
  - **QP:** Quadratic Program, quadratic objective and linear constraints
  - **QPQC:** Quadratically Constrained QP, QP w. quadratic constraints

Model Predictive Control: Considerations for Optimization

- Different programs (QP,LP) lead to different considerations or guarantees
  - Feasibility: *Is there a solution?*
  - Optimality: *Will I get an optimal solution?* (locally, globally)
- Some programs/problems *friendly*, easier to solve than others
- Convex programs: *Includes LPs and QPs of interest*
  - Can typically guarantee globally optimal solutions efficiently
  - Limited expressiveness: *Cannot solve path planning problems directly*

Model Predictive Control: Optimization and Implementation

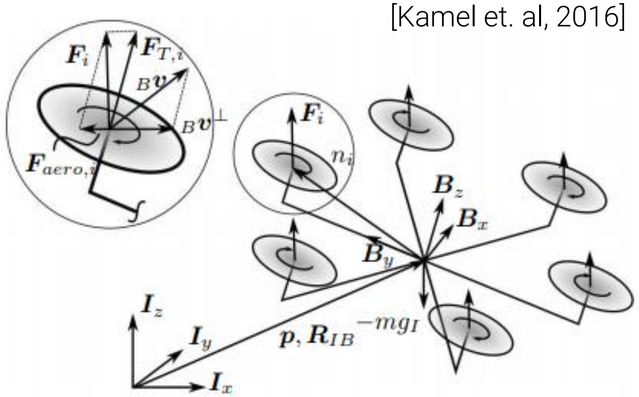
- Many frameworks/libraries exist for solving these optimization problems
  - qpOASIS: open source QP solver
  - CPLEX: closed source package (IBM)
  - Matlab: quadprog
  - Many problems can be solved reasonably quickly on your computers
  - Computation, time, and memory constraints make this harder
    - Hard to do fast low-level control with complex constraints

Model Predictive Control

- Software frameworks exist to solve the optimization programs.
  - qpOASIS: open source QP solver
  - CPLEX: closed-source optimization package
  - Matlab's `quadprog`
- Many optimization problems can be solved quickly on a reasonable computer.
- Challenges arise when computational resources are constrained (consider small robots with limited computers).
  - Leads to a problem as controllers must update quickly for fast systems.

Implementation: MPC for Multirotors

[Kamel et. al, 2016]



State $\mathbf{x} = \begin{pmatrix} \mathbf{p}^\top & \mathbf{v}^\top & \phi & \theta & \psi \end{pmatrix}^\top$ ← Position, Velocity, Attitude

Control $\mathbf{u} = \begin{pmatrix} \phi_{cmd} & \theta_{cmd} & \dot{\psi}_{cmd} & T \end{pmatrix}^\top$ ← Roll, Pitch, Yaw Rate, Thrust

Optimize

$$\mathbf{u}^* = \arg\min_{\mathbf{u},\mathbf{x}} \int_{t=0}^{T} \left(||\mathbf{x}(t) - \mathbf{x}_{ref}(t)||_Q^2 + ||\mathbf{u}(t) - \mathbf{u}_{ref}(t)||_R^2 \right) \mathrm{d}t + ||\mathbf{x}(T) - \mathbf{x}_{ref}(T)||_P^2$$

Ensure Q and P are positive definite

Subject to dynamics

$$\dot{\mathbf{p}} = \mathbf{v}$$

$$\dot{\mathbf{v}} = \frac{1}{m} \left(R_{IB} \sum_{i=0}^{N_r} F_{T,i} - R_{IB} \sum_{i=0}^{N_r} F_{aero,i} \right) + \begin{pmatrix} 0 & 0 & -g \end{pmatrix}^\top$$

$$\dot{\phi} = \frac{1}{\tau_\phi} \left(k_\phi \phi_{cmd} - \phi \right)$$

$$\dot{\theta} = \frac{1}{\tau_\theta} \left(k_\theta \theta_{cmd} - \theta \right)$$

First order closed-loop attitude response

$$\dot{\psi} = \dot{\psi}_{cmd}$$

$$\mathbf{u}(t) \in \mathbb{U}$$

$$\mathbf{x}(0) = \mathbf{x}(t_0)$$

Implementation: MPC for Multirotors

- Usually, problems are nonlinear and non-convex (such as the quadrotor control problem).
- The solutions are usually generated using sequential quadratic programming (SQP) techniques.
- ACADO is a commonly-used toolkit for dynamic optimization and optimal control: https://acado.github.io/
- The MPC problem can be defined intuitively using ACADO's API and then after compilation, it produces optimized auto-generated C++ header/source files that can be used in applications

ACADO code for multirotor MPC

ACADO can be used for:

- Optimal Control
- Dynamic optimization
- Robust optimization
- State/parameter estimation

Also has a MATLAB interface, and autogenerates C++ code for optimization

https://acado.github.io/

```
DifferentialState                p_x, p_y, p_z;
DifferentialState                roll, pitch, yaw;
DifferentialState                v_x, v_y, v_z;
Control                                          roll_cmd, pitch_cmd, yawrate_cmd, T;
DifferentialEquation             f;
Function                         h, hN;

// System Dynamics
f << dot(p_x)    == v_x;
f << dot(p_y)    == v_y;
f << dot(p_z)    == v_z;
f << dot(roll)   == (roll_gain*roll_cmd - roll)/roll_tau;
f << dot(pitch)  == (pitch_gain*pitch_cmd - pitch)/pitch_tau;
f << dot(yaw)    == yawrate_cmd;
f << dot(v_x)    == ((cos(roll)*cos(yaw)*sin(pitch) + sin(roll)*sin(yaw))*T - dragacc1);
f << dot(v_y)    == ((cos(roll)*sin(pitch)*sin(yaw) - cos(yaw)*sin(roll))*T - dragacc2);
f << dot(v_z)    == (-g + cos(pitch)*cos(roll)*T);

// Running cost vector consists of all states and inputs.
h << p_x << p_y << p_z << v_x << v_y << v_z << roll << pitch << yaw << roll_cmd << pitch_cmd << yawrate_cmd << T;

// End cost vector consists of all states (no inputs at last state).
hN << p_x << p_y << p_z << v_x << v_y << v_z;

OCP ocp( t_start, t_end, N );
ocp.subjectTo( f );
ocp.minimizeLSQ( Q, h, r );
ocp.minimizeLSQEndTerm( QN, hN, rN );
```

Examples

- Explicit MPC: *Efficient computation with hard constraints*
- Adaptive control: *Modifying control based on unknown parameters*
- Robust MPC: *Modeling uncertainty w. constraints*

Explicit Model Predictive Control

- Each database entry corresponds to a controller solution with different active constraints.
  - An active constraint means that you are at the constraint boundary (so where the inequality becomes an equality).
- The database approach is called *Explicit* MPC.
  - Create the database offline.
  - Figure out which constraints (if any) are active online and apply the corresponding controller.
  - Controllers will switch at boundaries so care must be used to ensure no loss of stability (e.g., introducing *dwell* times).

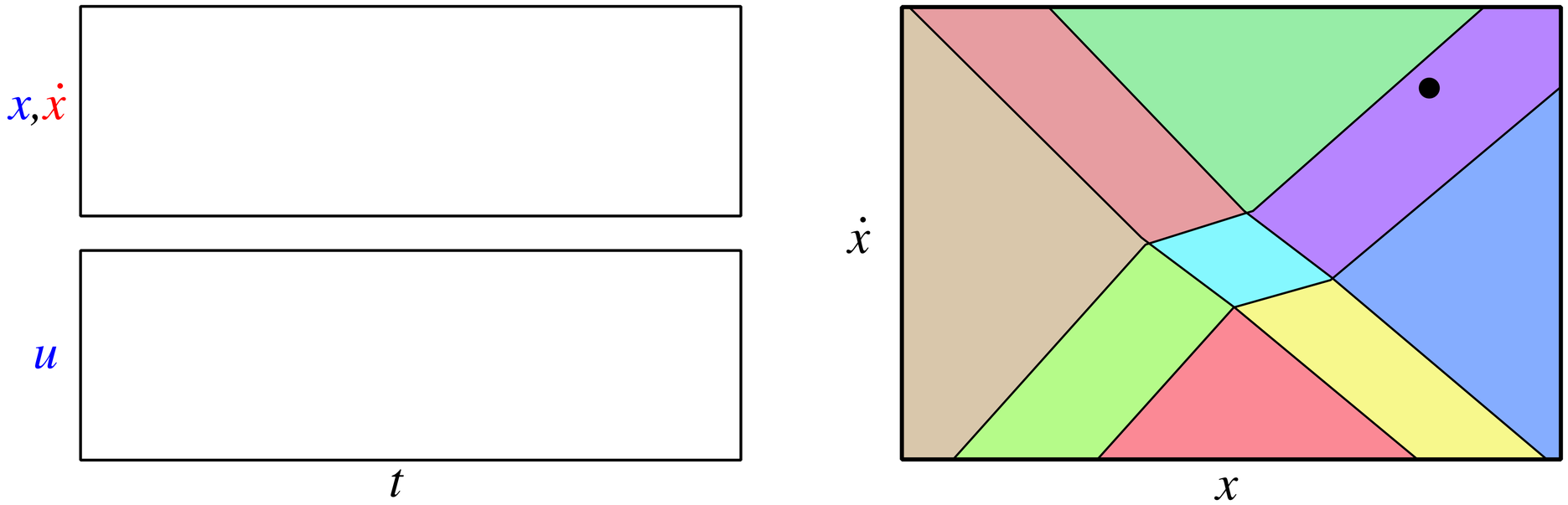
Explicit Model Predictive Control

- Explicit MPC is computationally expensive.
  - Considers all combination of all active constraints.
  - Database grows exponentially in the number of constraints.
    - Often $3^{N*\text{number of constraints}}$

Explicit Model Predictive Control

Example:

- The system starts in the purple region with its corresponding controller.

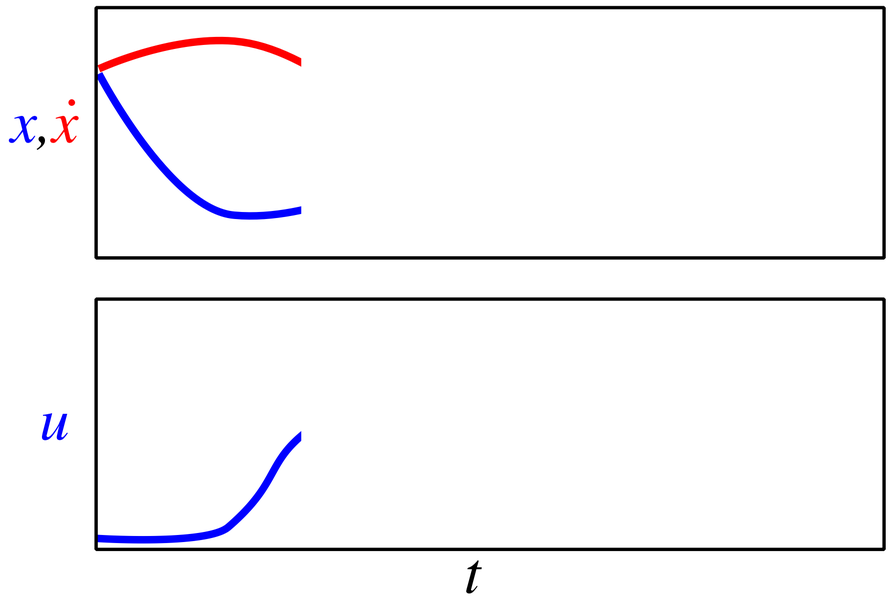


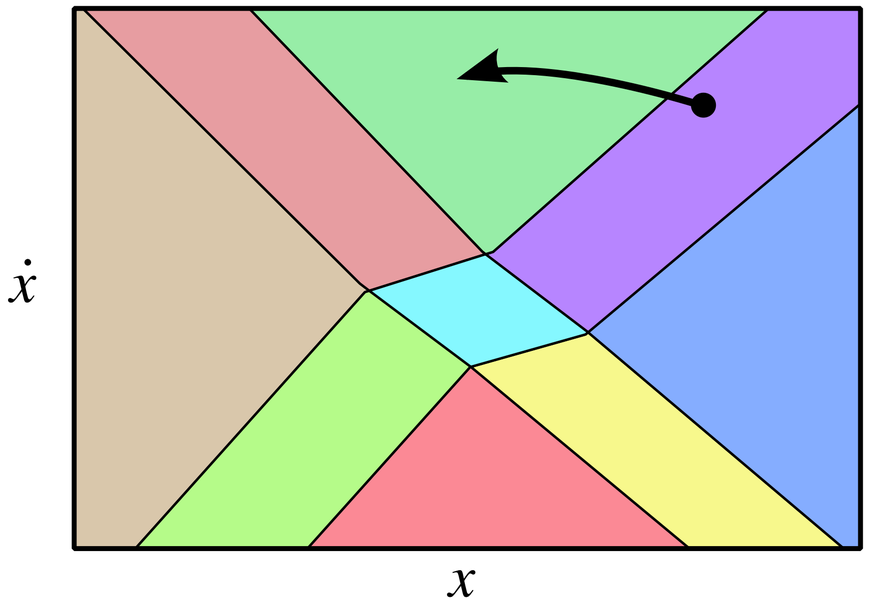
[Adapted from Bemporad et al., 2002]

Explicit Model Predictive Control

Example:

- System changes regions and controllers as it moves.



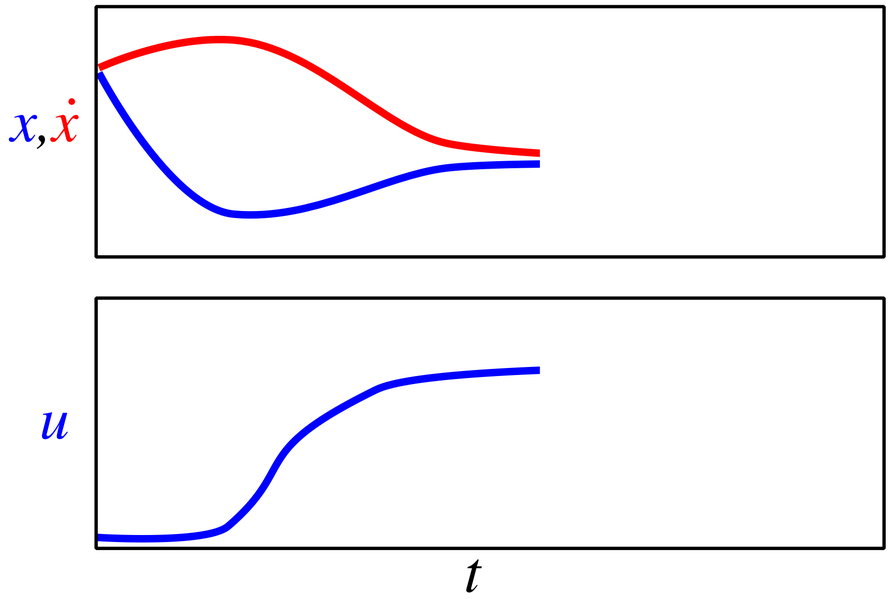


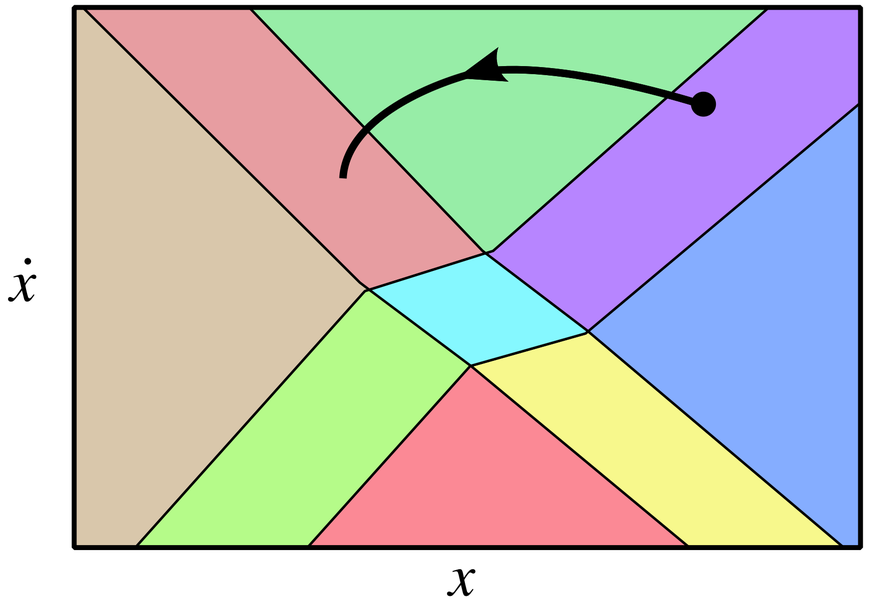
[Adapted from Bemporad et al., 2002]

Explicit Model Predictive Control

Example:

- Each transition requires a database search (which can be expensive).



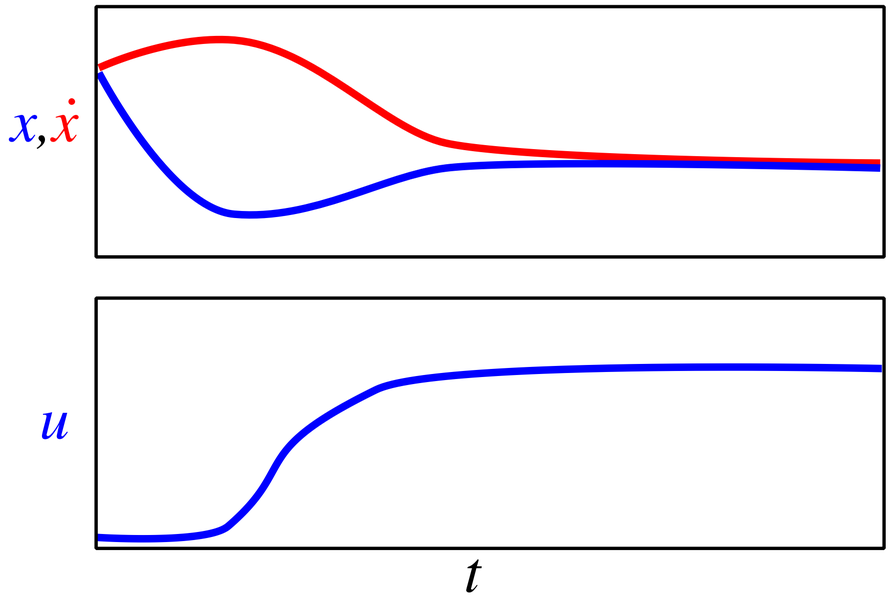


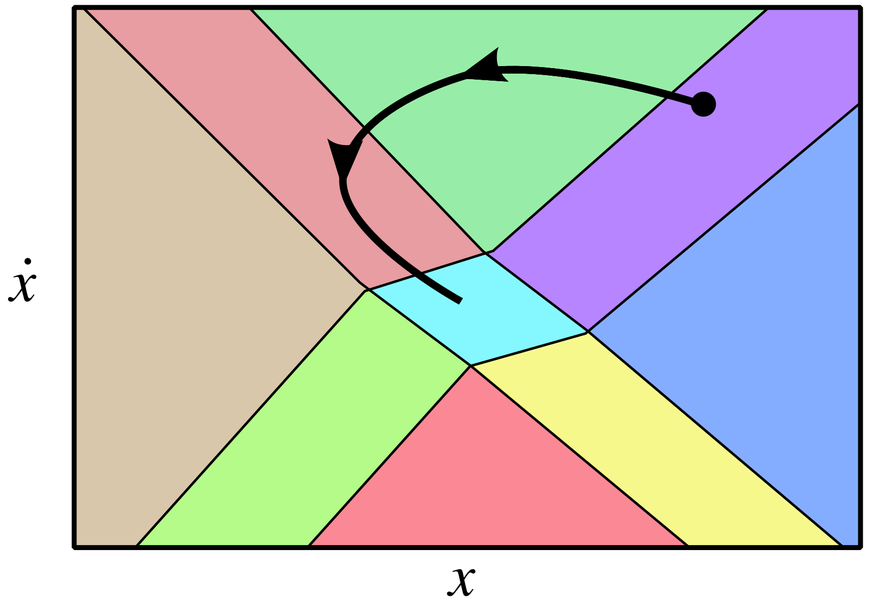
[Adapted from Bemporad et al., 2002]

Explicit Model Predictive Control

Example:

- Reasonable for small problems but scales poorly.





[Adapted from Bemporad et al., 2002]

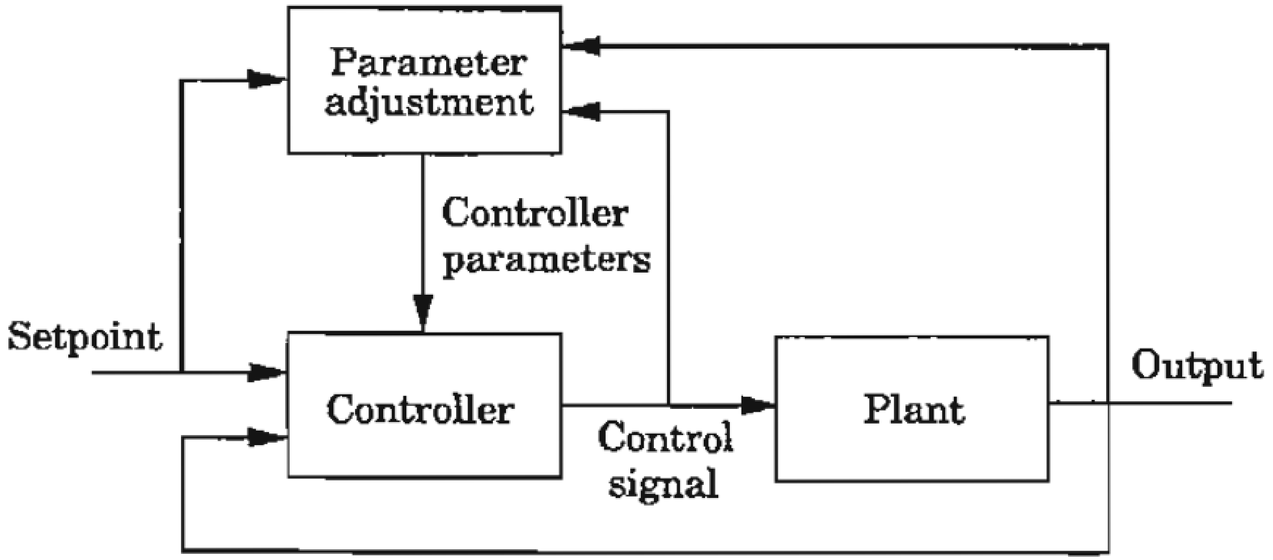
Partial Enumeration

- Given the complexity of Explicit MPC (and the need to compute everything), it can be beneficial to compute controllers online (as the system evolves), storing controllers to be used in the future.
  - Known as Partial Enumeration:
    - Build the database as you go by occasionally solving for a controller.

If interested see more details here for example:
https://doi.org/10.1109/ICRA.2016.7487255

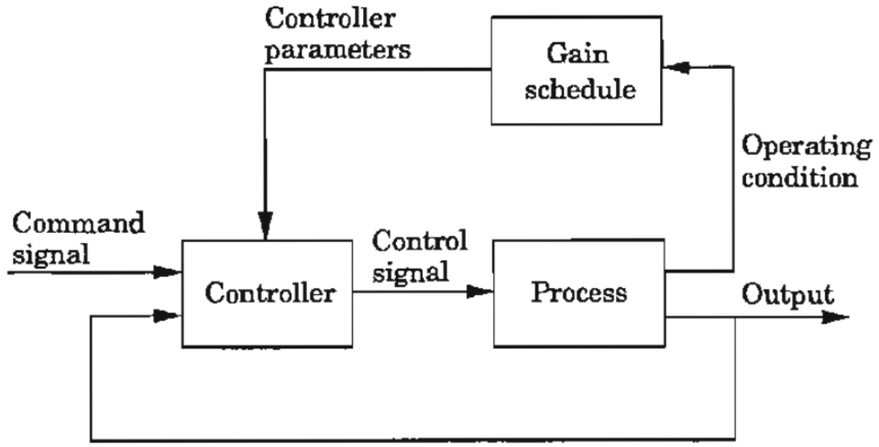
Adaptive Control

- What is adaptive control?
  - An adaptive controller is a controller with adjustable parameters and a mechanism for adjusting the parameters.

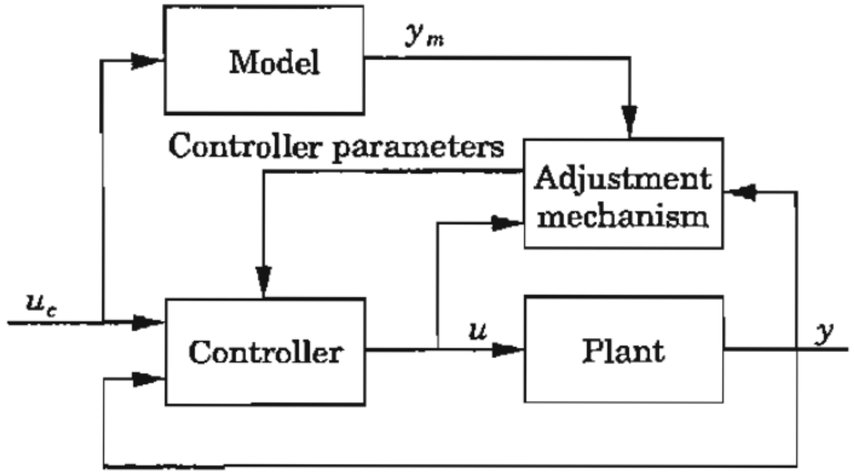


There are two loops
- Regular feedback loop
- Parameter adjustment loop

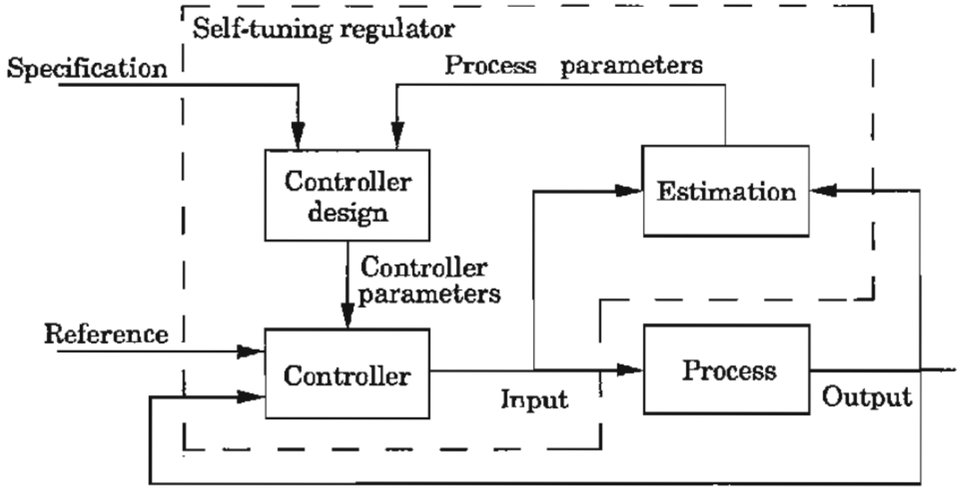
Adaptive Schemes



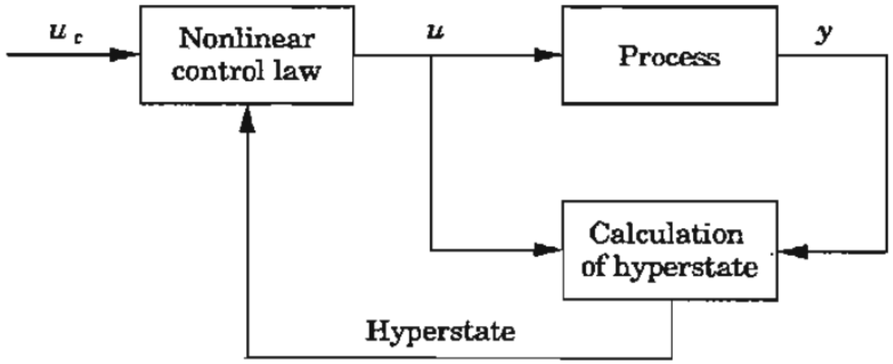
Gain scheduling



Model-reference adaptive control (MRAC)

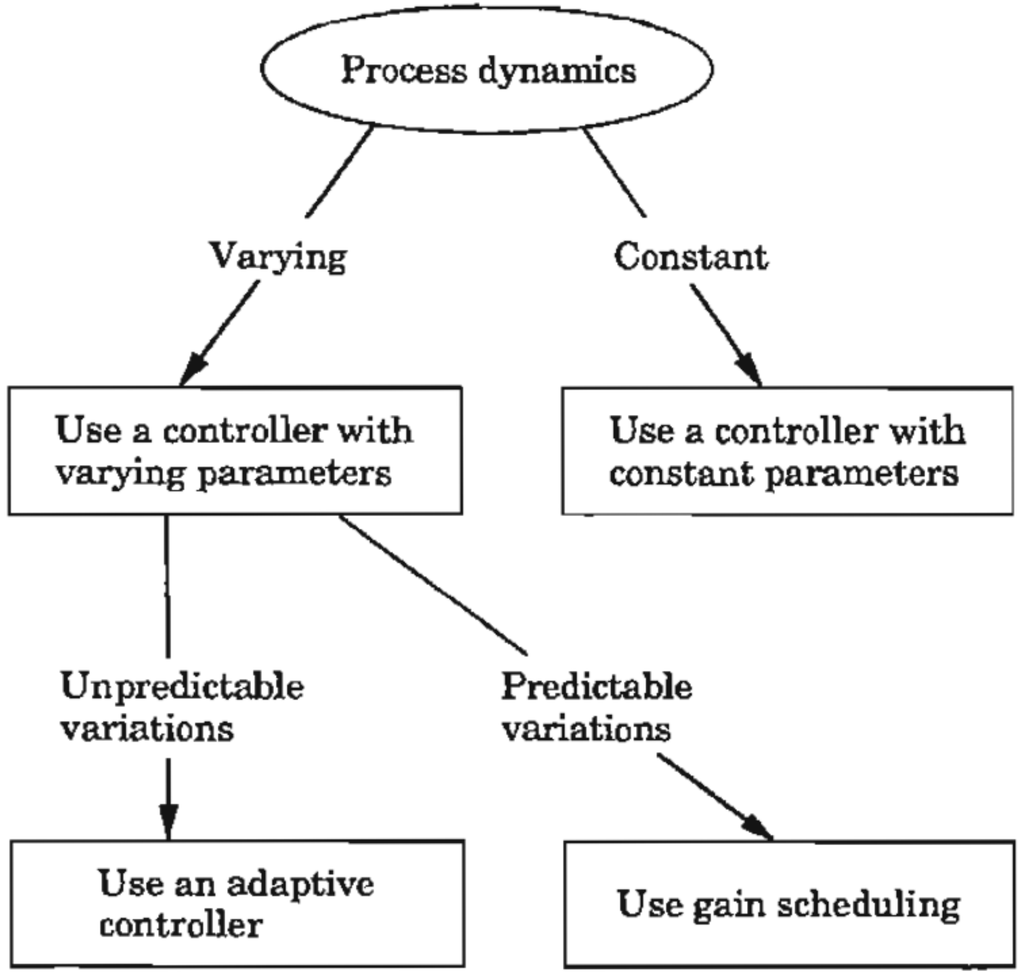


Self-tuning regulator (STR)

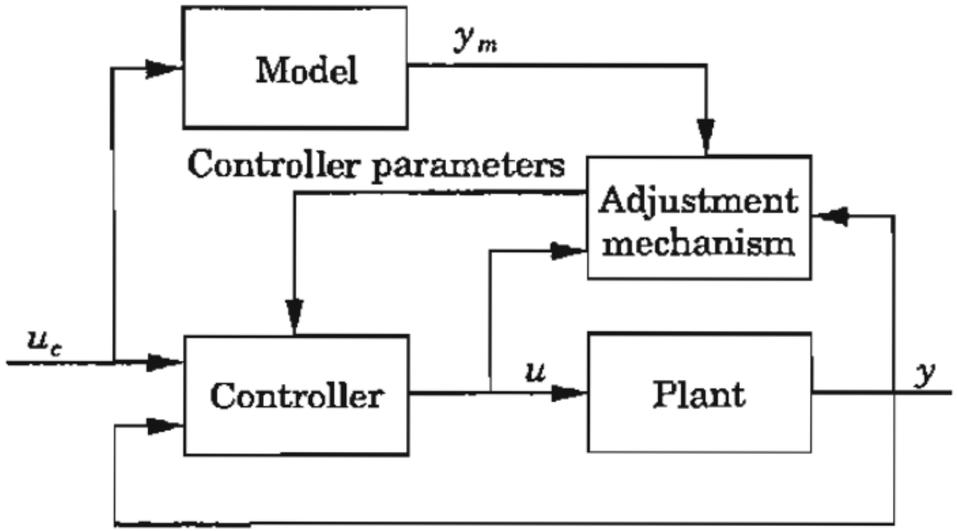


Dual controller

How should a controller adapt?



Model-reference Adaptive Control (MRAC)



The mechanism for adjusting the parameters can be obtained in two ways:

- Gradient method
- Stability theory (Lyapunov theory)

The MIT Rule

- Step 1: Define the error between the plant and reference model output as

$$e(t,\theta) = y(t,\theta) - y_m(t)$$

Observe that the output of the reference model does not depend on the adjustable controller parameter θ

- Step 2: Introduce a loss function for quantifying close-loop performance

$$J(t,\theta) = \frac{1}{2}e^2(t,\theta)$$

- Step 3: Minimize J be moving θ in the direction of the negative gradient of J

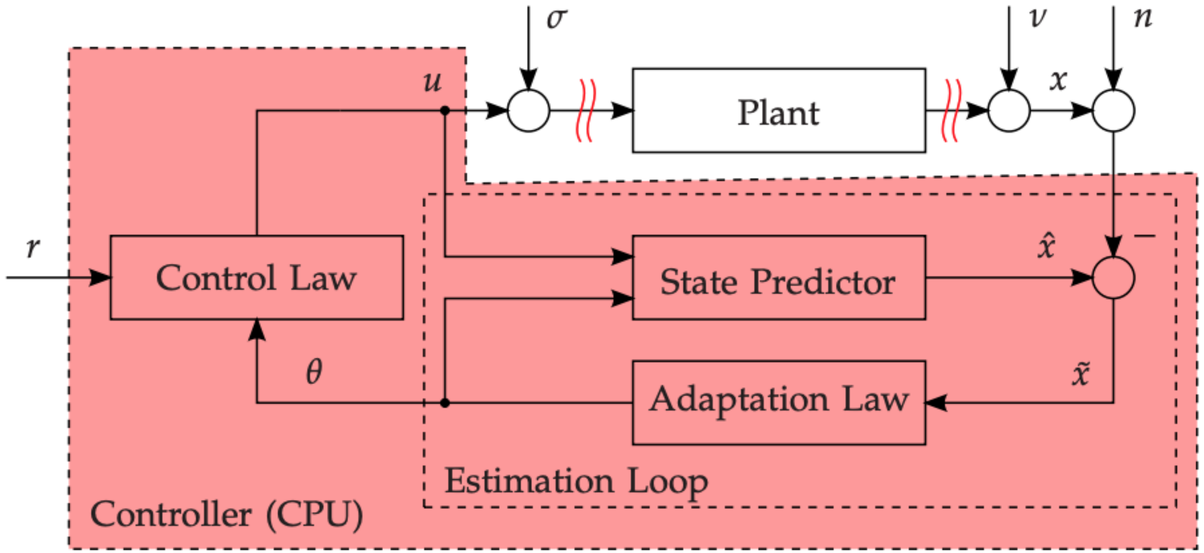
$$\frac{d\theta}{dt} = -\gamma\frac{\partial J}{\partial \theta} = -\gamma e\frac{\partial e}{\partial \theta}$$

$$\frac{d\theta}{dt} = -\gamma\nabla_\theta J = -\gamma e\nabla_\theta e$$

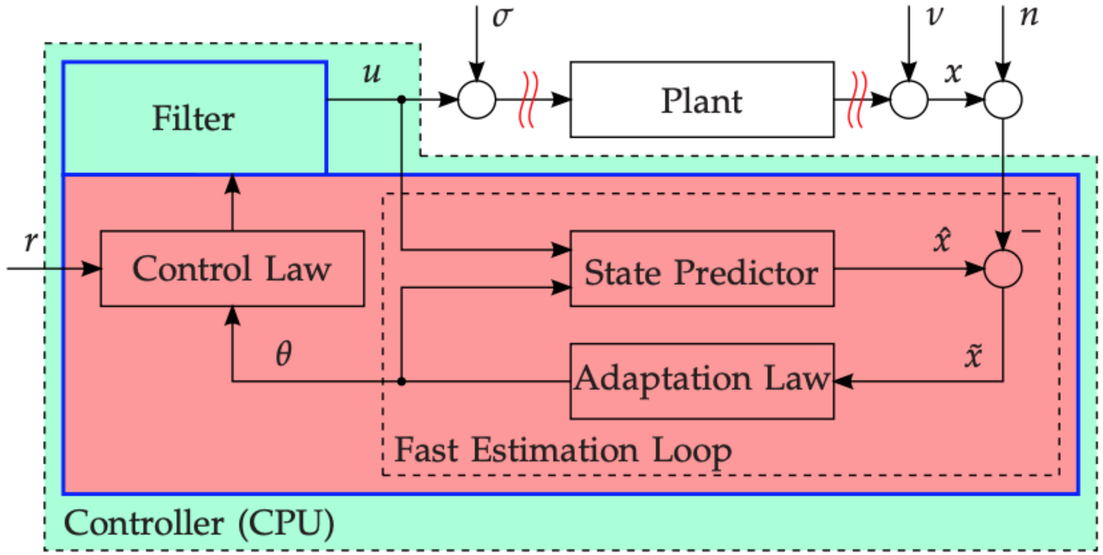
$\frac{\partial e}{\partial \theta}$: sensitivity derivative

γ : adaptation gain

L1-Adaptive Control



(a) MRAC



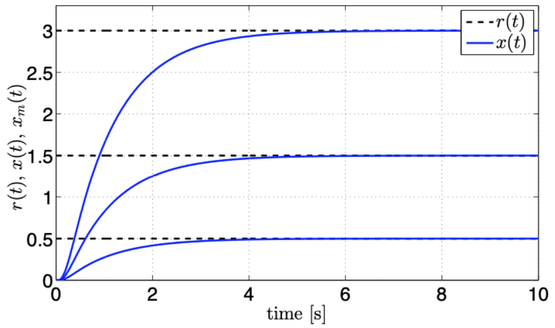
(b) $\mathcal{L}_1$ adaptive controller

Issue:

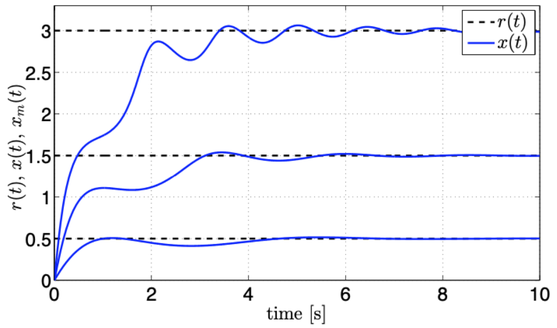
- In real application, the unknown parameter vector may have frequencies that lie outside the control channel bandwidth.
- Time delay

- Estimation loop is decoupled from the control loop
- The filter protect the control signal from exceeding the available control system bandwidth to avoid high gain oscillation.
- The filter position ensures that the fast estimation rate will not hurt system robustness.

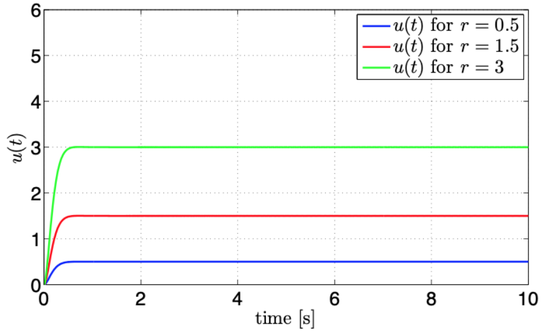
L1 Controller vs MRAC



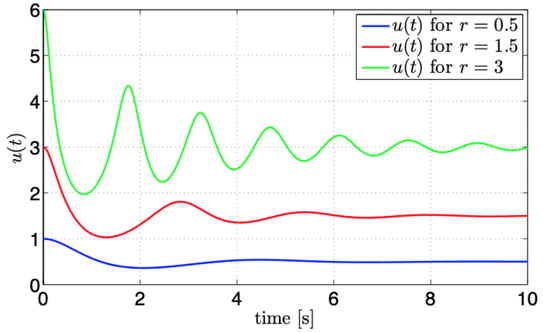
(a) $\mathcal{L}_1$ Controller system state



(b) MRAC system state

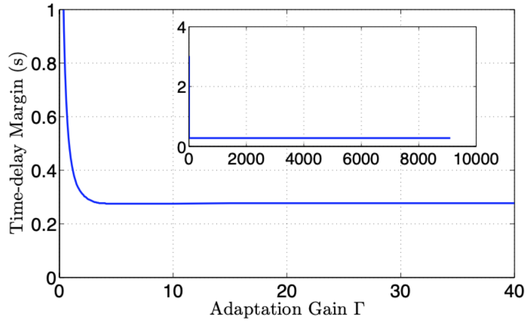


(c) $\mathcal{L}_1$ Controller control signal

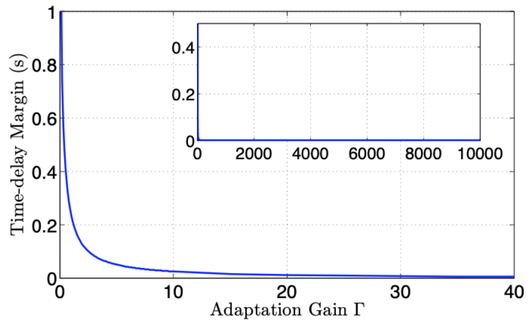


(d) MRAC control signal

Fig. 8: Comparison of transient responses of $\mathcal{L}_1$ adaptive controller and MRAC in the presence of unmodeled dynamics. The state response of the $\mathcal{L}_1$ adaptive controller scales with the magnitude of the step command, whereas the transient response of MRAC is highly nonlinear and changes in an unpredictable manner.



(a) $\mathcal{L}_1$ adaptive controller



(b) MRAC

Fig. 6: Time-delay margin of $\mathcal{L}_1$ adaptive controller and MRAC as a function of adaptation gain. The time-delay margin of MRAC vanishes as the adaptation gain is increased, while the time-delay margin of $\mathcal{L}_1$ adaptive controller remain bounded away from zero as the adaptation gain increases. For large adaptation gains it approaches the value $\mathcal{T} = 0.28$ s.

[1] Hovakimyan, Naira. "L1 adaptive control." *Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Tech. Rep* (2014): 1-15.

Preview: Flatness, Feed-Forward, and Geometric Control

Next lecture: *Trajectory Generation*

- Geometric control
  - Dynamics and control with *"natural"* representations of state-geometry & error
  - No singularities, valid&stable for any reference + large error
- Flatness
  - Direct correspondence between trajectory (time-parametrized polynomials&position) and full state (including rotation)
  - Simple optimization problems without MPC
  - Feasibility: Sufficient smoothness of trajectory → continuous state
- Feedforward control
  - Compute nominal state & thrusts via *flatness*
  - Feedforward anticipates via derivatives / MPC anticipates explicitly

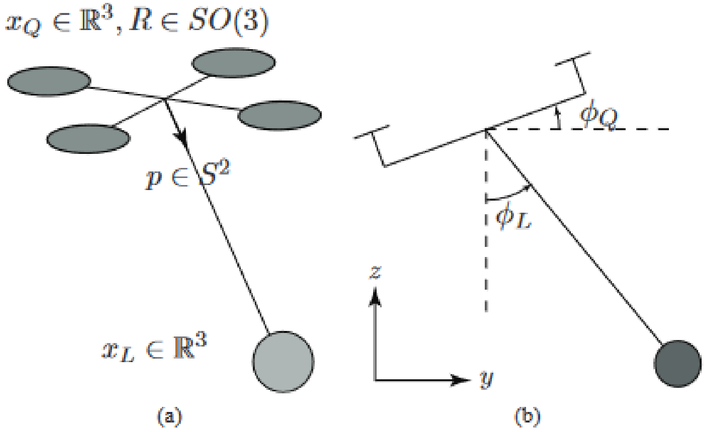


Fig. 1: (a) A 3D quadrotor with a cable suspended load. When the cable is taut, the system evolves on $SE(3) \times S^2$, and has 8 degrees of freedom with 4 degrees of underactuation. (b) A planar quadrotor with a cable suspended load evolving on $SE(2) \times S^1$

[1] Sreenath, K., Michael, N., & Kumar, V. (2013, May). Trajectory generation and control of a quadrotor with a cable-suspended load-a differentially-flat hybrid system. In 2013 IEEE international conference on robotics and automation.
[2] Lee, T., Leok, M., & McClamroch, N. H. (2010, December). Geometric tracking control of a quadrotor UAV on SE (3). In 49th IEEE conference on decision and control (CDC).

Next Lecture

Trajectory Generation

To prepare please:

- Come with your laptop
- Install Matlab
- Install the matlab planning tutorial:
  https://bitbucket.org/castacks/matlab_planning_toolbox/src/master/

- PDF slides will be posted.

Geometric Adaptive Controller

Quadrotor model

$$\begin{cases} m\ddot{\boldsymbol{p}} = m\boldsymbol{g} + f\boldsymbol{b}_z + \boldsymbol{\theta}_f \\ \dot{\boldsymbol{R}} = \boldsymbol{R}\hat{\boldsymbol{\Omega}} \\ \boldsymbol{J}\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega}\times\boldsymbol{J}\boldsymbol{\Omega} + \boldsymbol{\tau} + \boldsymbol{\theta}_\tau \end{cases}$$

Geometric adaptive controller

$$\begin{aligned} f &= -\boldsymbol{b}_z^{\mathsf{T}}(\overbrace{k_p\boldsymbol{e}_p + k_v\boldsymbol{e}_v + m\boldsymbol{g} - m\ddot{\boldsymbol{p}}_d}^{geometric} + \overbrace{\hat{\boldsymbol{\theta}}_f}^{adaptive}) \\ \boldsymbol{\tau} &= -k_r\boldsymbol{e}_r - k_\Omega\boldsymbol{e}_\Omega + \boldsymbol{\Omega}\times\boldsymbol{J}\boldsymbol{\Omega} \\ &\quad \underbrace{-\boldsymbol{J}(\hat{\boldsymbol{\Omega}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}_d\boldsymbol{\Omega}_d - \boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}_d\dot{\boldsymbol{\Omega}}_d)}_{geometric} - \underbrace{\hat{\boldsymbol{\theta}}_\tau}_{adaptive} \end{aligned}$$

Error

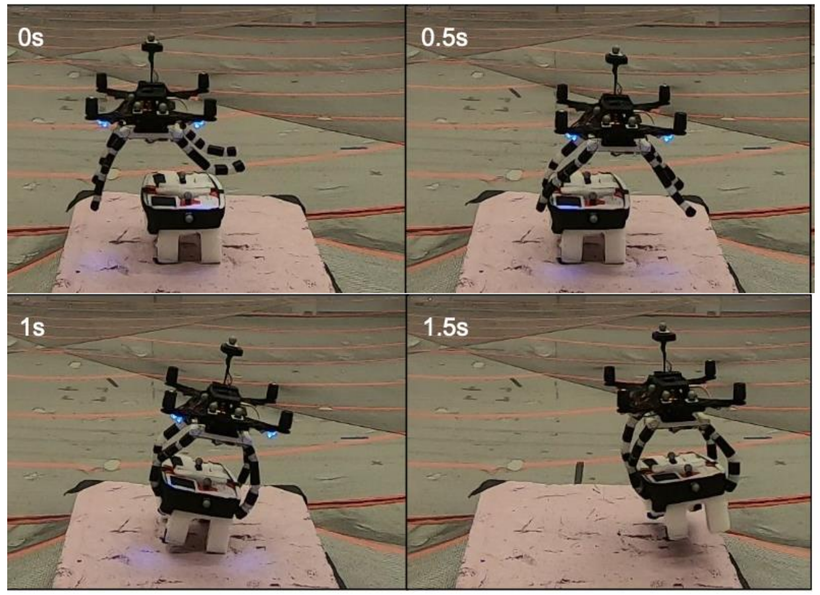
$$\begin{aligned} \mathbf{e}_p &= \mathbf{x} - \mathbf{x}_d \\ \mathbf{e}_v &= \mathbf{v} - \mathbf{v}_d \\ \mathbf{e}_r &= \tfrac{1}{2}(\mathbf{R}^T\mathbf{R} - \mathbf{R}^T\mathbf{R}_d)^\vee \\ \mathbf{e}_\Omega &= \boldsymbol{\Omega} - \mathbf{R}^T\mathbf{R}_d\boldsymbol{\Omega}_d \end{aligned}$$

Adaptation law

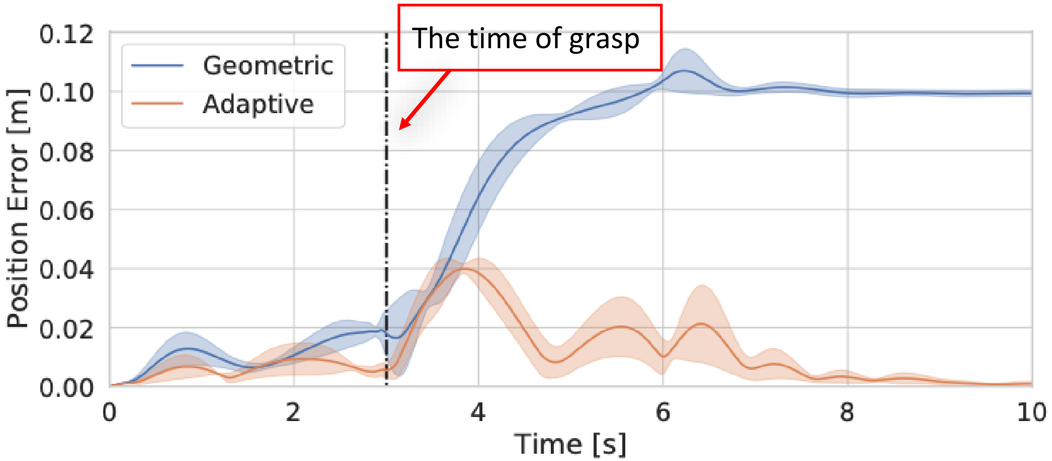
$$\frac{d\hat{\boldsymbol{\theta}}_f}{dt} = \Pi\left(\gamma_f(\boldsymbol{e}_v + k_{af}\boldsymbol{e}_p)\right), \qquad \frac{d\hat{\boldsymbol{\theta}}_\tau}{dt} = \gamma_\tau(\boldsymbol{e}_\Omega + k_{a\tau}\boldsymbol{e}_r)$$

Π=Projection function

$\|\boldsymbol{\theta}_f\| \le \beta,\ \|\boldsymbol{\theta}_\tau\| \le \beta$



Dynamic grasping with a “soft” drone [1]



[1] J. Fishman, S. Ubellacker, N. Hughes and L. Carlone, "Aggressive Grasping with a “Soft” Drone: From Theory to Practice" IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021

Robust MPC

- Consider the state estimation error as an additional, unknown but bounded uncertainty that can be accounted for in a suitably modified robust MPC rather than the unknown but bounded system state.
- The state estimation error is bounded by a simple, pre-computed, invariant set.
- The controller uses a tube, the center of which is obtained by solving a conventional – (disturbance free) MPC that yields the tube center.
- The tube is obtained by "adding" to its center another simple, pre-computed, invariant set.
- Solve a tighter constraints in the MCP online to ensure that all realizations of the state trajectory satisfy the state and control constraints.

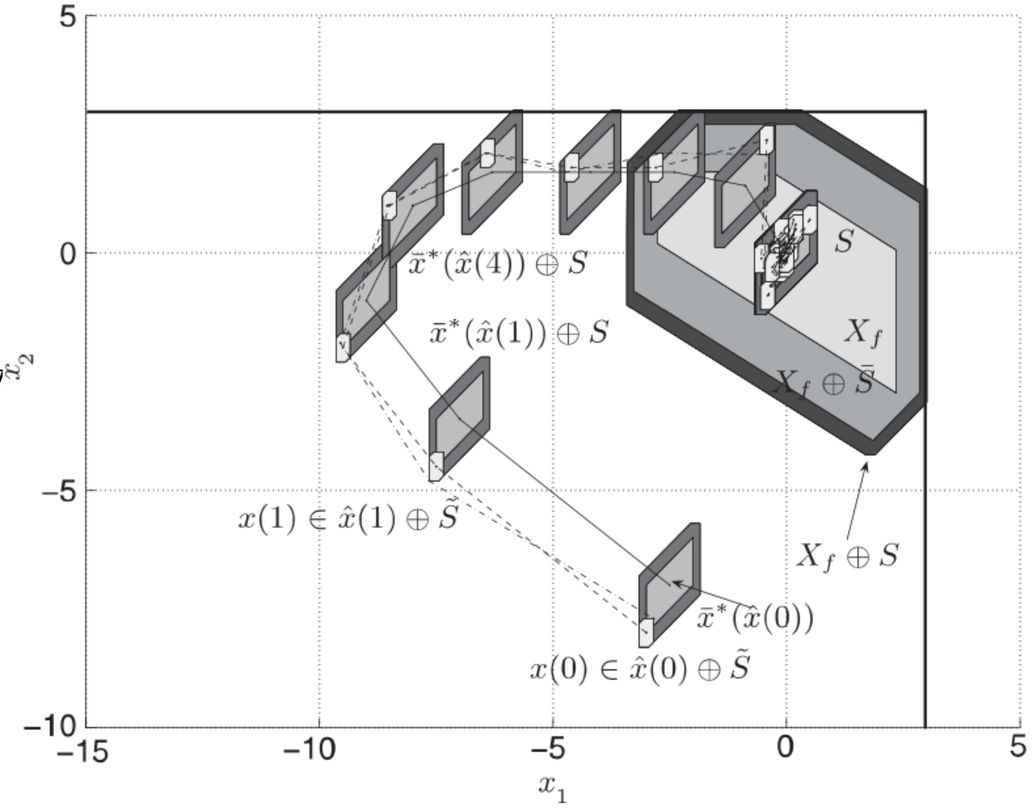


Fig. 1. The robust output-feedback MPC tube $\{\bar{x}^*(\hat{x}(i)) \oplus S\}$.

[1] Mayne, David Q., et al. "Robust output feedback model predictive control of constrained linear systems." *Automatica* 42.7 (2006): 1217-1222.