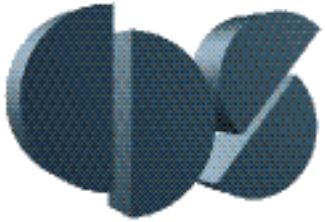
CDS 110/ChE 105: Lecture 6-3
Model Predictive Control



Richard M. Murray

10 May 2024

Goals:

- Introduce the key ideas behind receding horizon control (RHC; also more commonly known as model predictive control [MPC])
- Show how to use python-control explore the use/tradeoffs of RHC/MPC

Reading:

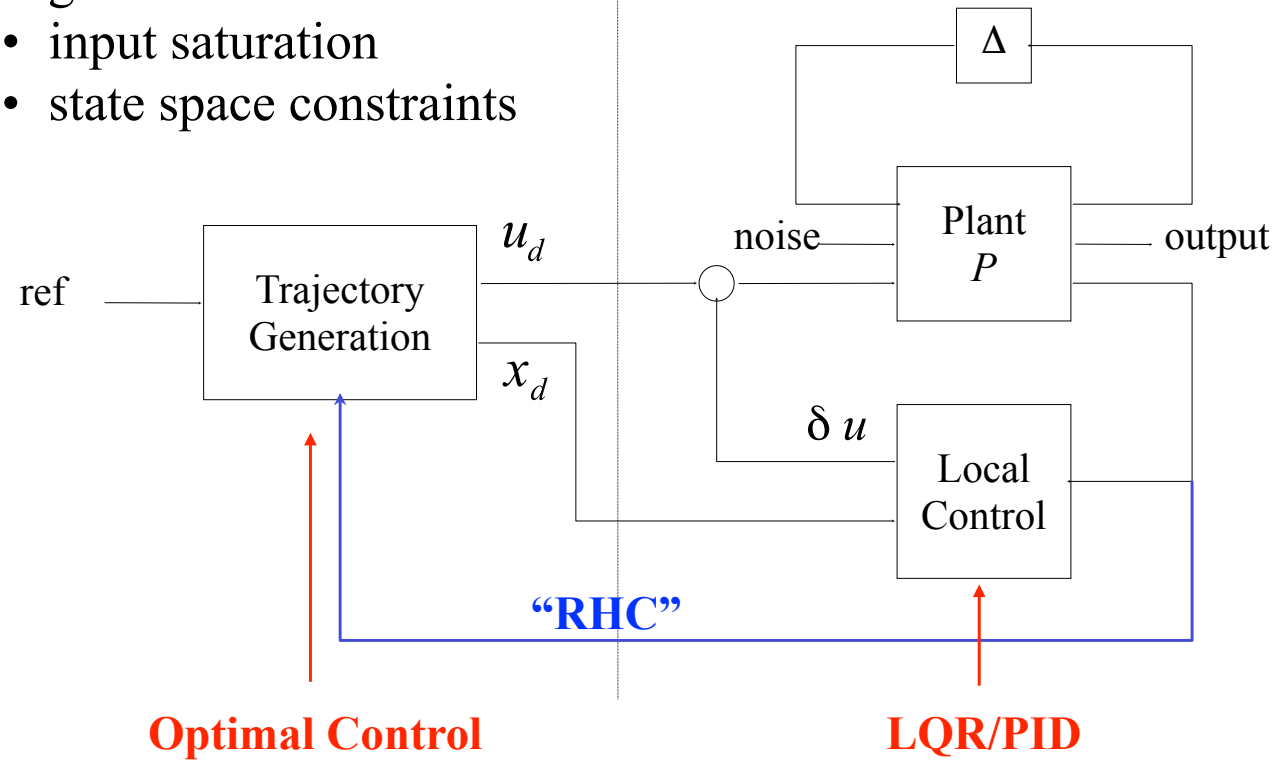
- Åström and Murray, *Feedback Systems,* Sections 8.4
- Murray, *Optimization-Based Control,* Chapter 3 (skim)
- Google Colab: L6-3_doubleint-rhc.ipynb

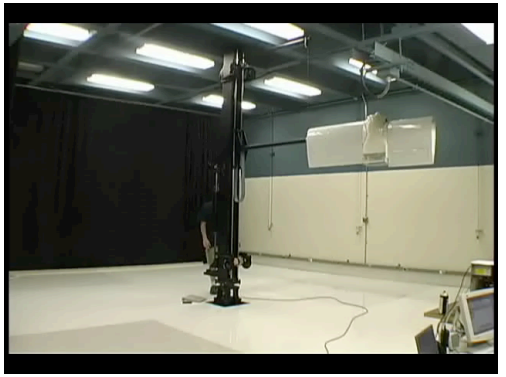
Murray, Hauser et al
SEC chapter (IEEE, 2002)

Control Architecture: Two DOF Design

Nonlinear design
- global nonlinearities
- input saturation
- state space constraints

Local design

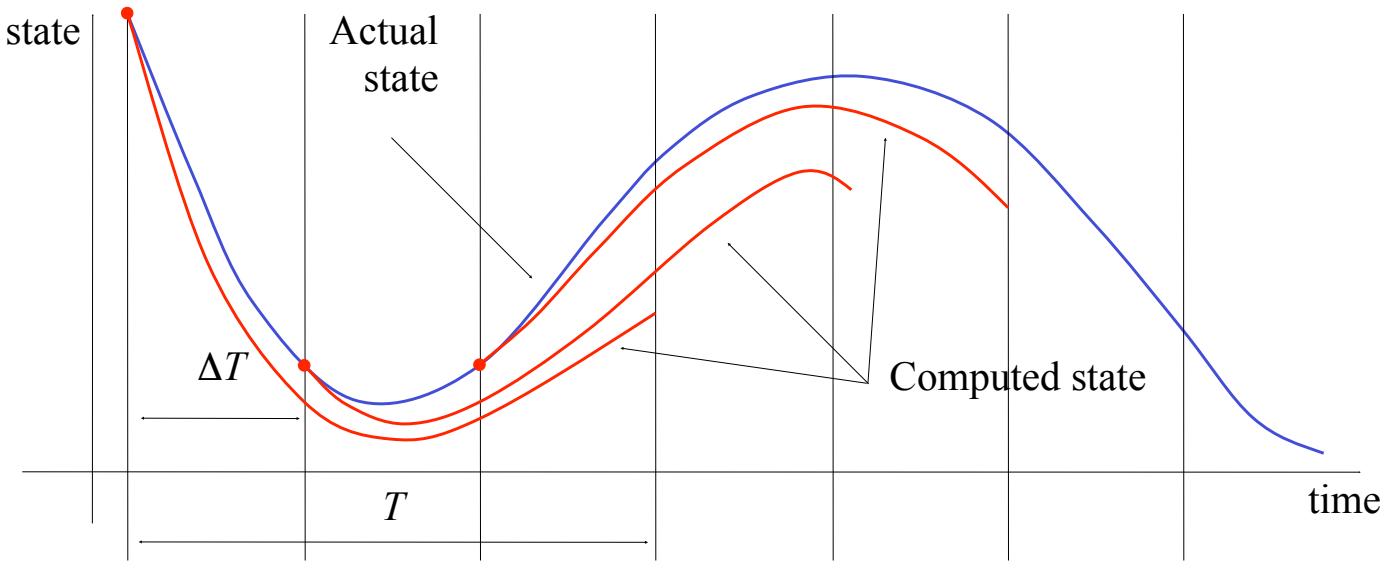




- Use nonlinear trajectory generation to construct (optimal) feasible trajectories
- Use local control to handle uncertainty and small scale (fast) disturbances
- Receding horizon control: iterate trajectory generation

Murray, Hauser et al
SEC chapter (IEEE, 2002)

Receding Horizon Control



Solve finite time optimization over *T* seconds and implement first ΔT seconds

$$u_{[t,t+\Delta T]} = \arg\min \int_t^{t+T} L(x(\tau),u(\tau))\,d\tau + V(x(t+T))$$

$$x_0 = x(t) \quad x_f = x_d(t+T)$$

$$\dot{x} = f(x,u) \quad g(x,u) \le 0$$

Finite horizon optimization

Terminal cost

Requires that computation time be small relative to time horizons

- Initial implementation in process control, where time scales are fairly slow
- Real-time trajectory generation enables implementation on faster systems

Stability of Receding Horizon Control

RHC can destabilize systems if not done properly

- For properly chosen cost functions, get stability with T sufficiently large
- For shorter horizons, counter examples show that stability is trickier

Thm (Jadbabaie & Hauser, 2002). Suppose that the terminal cost $V(x)$ is a control Lyapunov function such that

$$\min_u (\dot{V} + L)(x, u) < 0$$

for each $x \in \Omega_r = \{x: V(x) < r^2\}$, for some $r > 0$. Then, for every $T > 0$ and $\Delta T \in (0; T]$, the resulting receding horizon trajectories go to zero exponentially fast.

Remarks

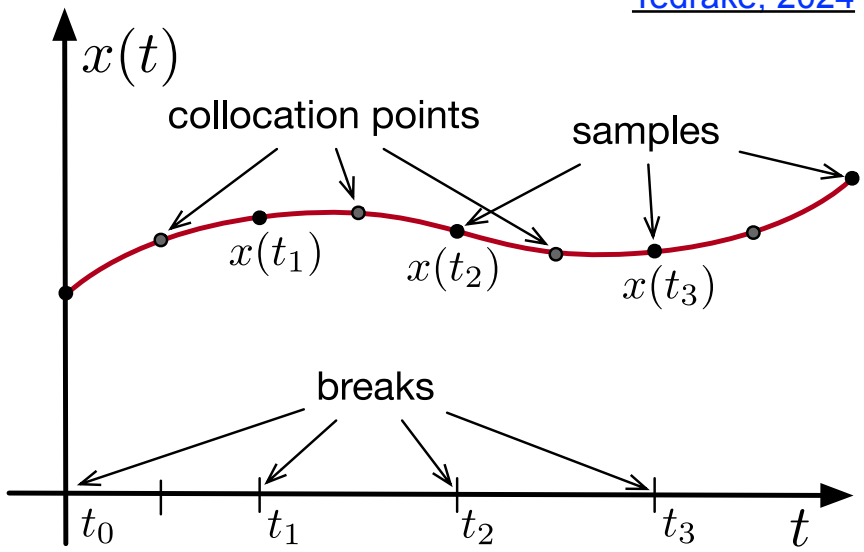
- Earlier approach used terminal trajectory constraints; hard to implement in real-time
- CLF terminal cost is difficult to find in general, but LQR-based solution at equilibrium point often works well - choose $V = x^T P x$ where $P =$ Riccati soln

Trajectory Generation Via Collocation

Modulate inputs and states at "break points"

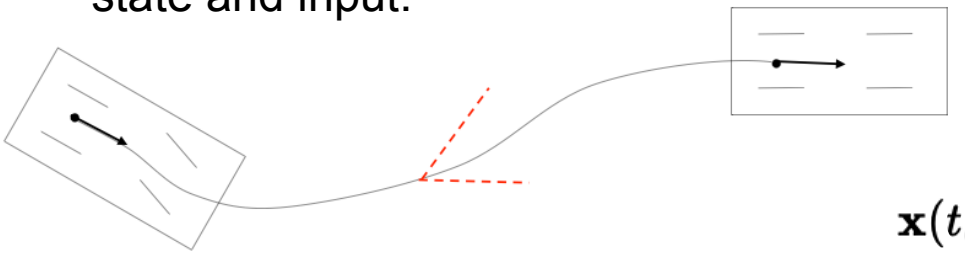
- Choose a set of points at which we choose the inputs and states (=> (n + m) * k free variables)
- Constrain the curves to be smooth at breakpoints (eg, by using cubic splines)
- Constrain the curve to match the dynamics [f(x,u)] at mid-points (again, using cubic splines)
- Can implement in efficient way (python-control)

Tedrake, 2024



Another trick to play: differential flatness

- Can show that in some cases you can parameterize "flat outputs" and retrieve state and input:



- Main idea: rewrite trajectories in terms of flat outputs => don't have to solve ODEs => 100X faster (at least)

$$t_{c,k} = \frac{1}{2}(t_k + t_{k+1}), \qquad h_k = t_{k+1} - t_k,$$

$$\mathbf{u}(t_{c,k}) = \frac{1}{2}(\mathbf{u}(t_k) + \mathbf{u}(t_{k+1})),$$

$$\mathbf{x}(t_{c,k}) = \frac{1}{2}(\mathbf{x}(t_k) + \mathbf{x}(t_{k+1})) + \frac{h}{8}(\dot{\mathbf{x}}(t_k) - \dot{\mathbf{x}}(t_{k+1})),$$

$$\dot{\mathbf{x}}(t_{c,k}) = -\frac{3}{2h}(\mathbf{x}(t_k) - \mathbf{x}(t_{k+1})) - \frac{1}{4}(\dot{\mathbf{x}}(t_k) + \dot{\mathbf{x}}(t_{k+1})).$$

Implementation using NTG Software Library

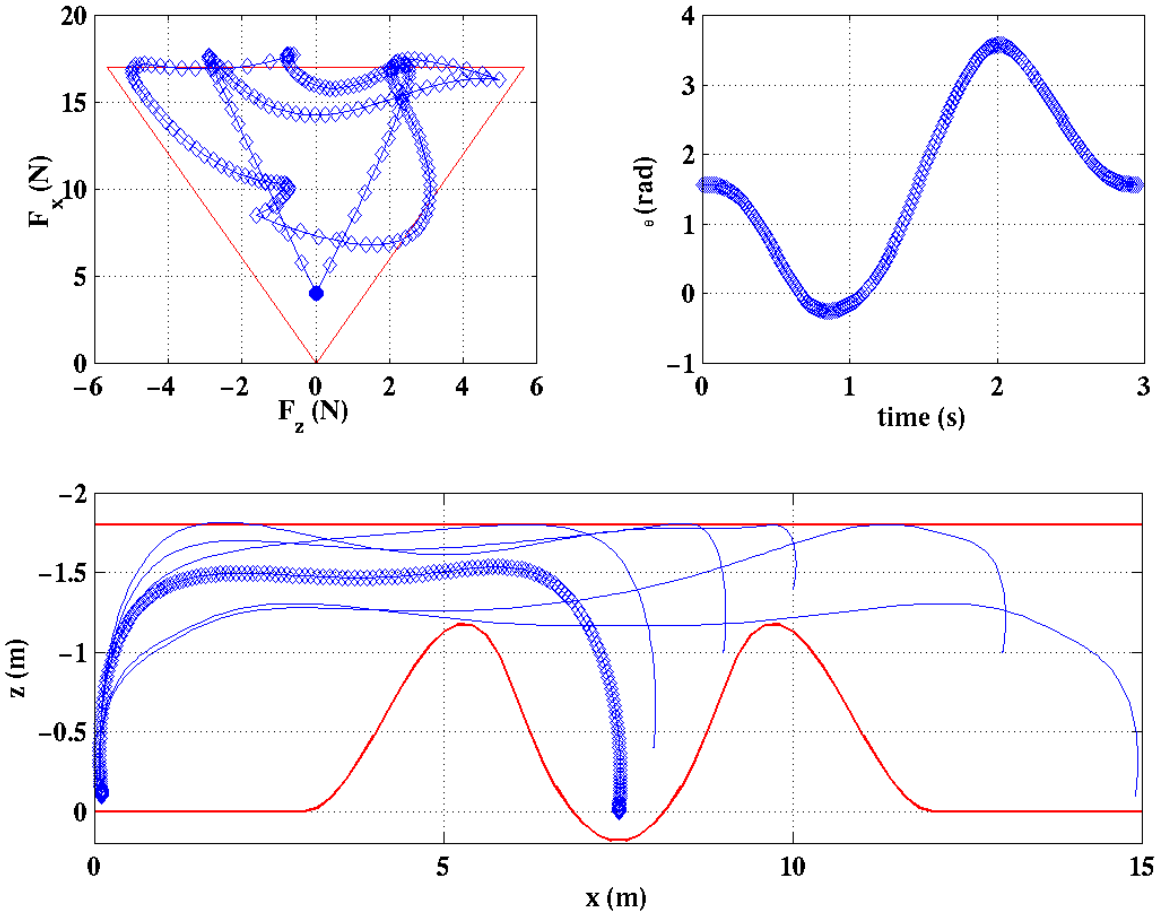
Features

- Handles constraints
- *Very* fast (real-time), especially from warm start
- Good convergence

Weaknesses

- No convergence proofs
- Misses constraints between collocation points
- Doesn't exploit mechanical structure (except through flatness)

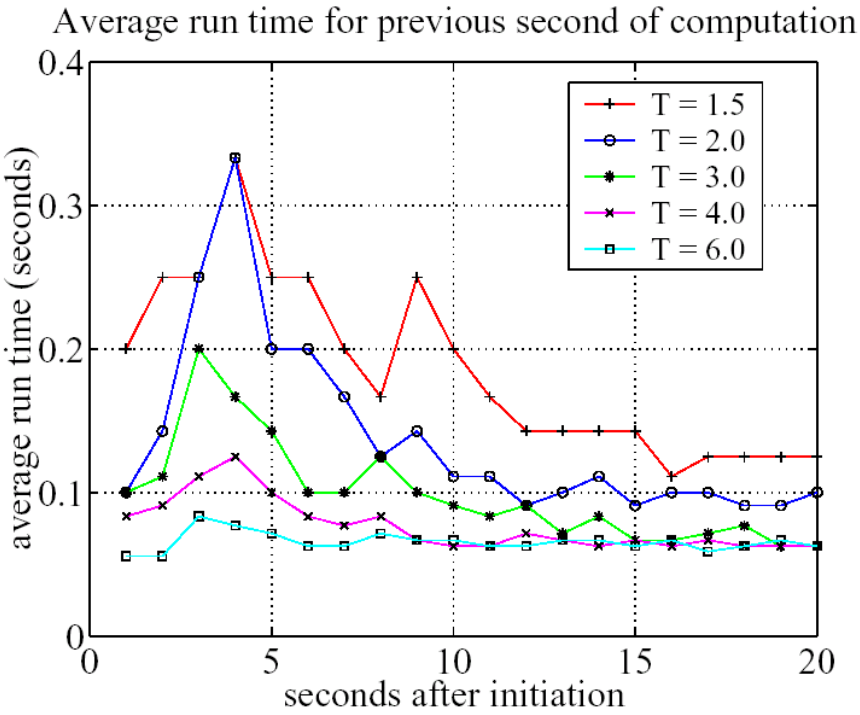
Planar Ducted Fan: Warm Starts

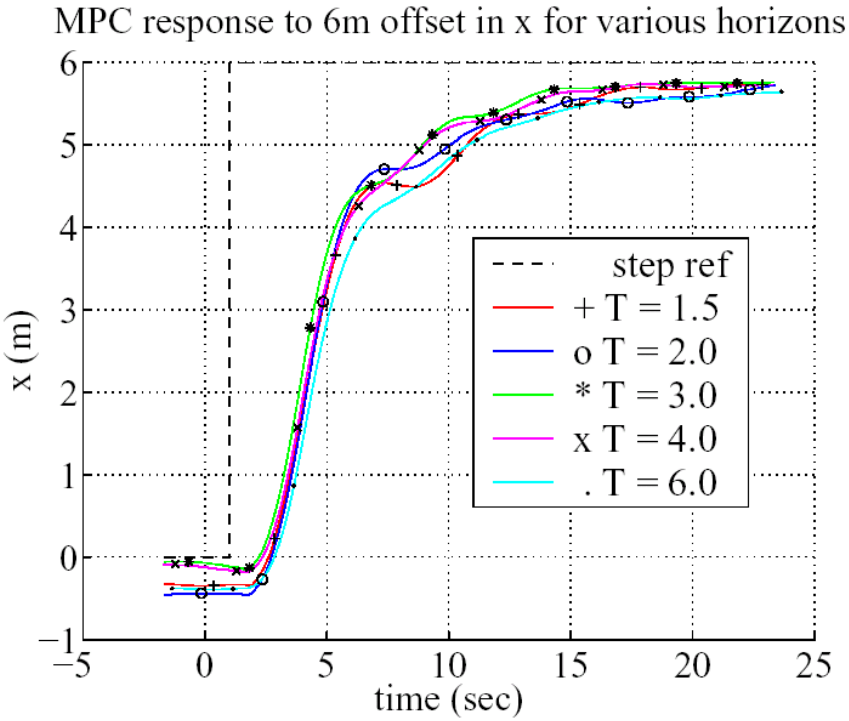


http://www.cds.caltech.edu/~murray/software/2002a_ntg.html

Milam, Franz, Hauser, 2002 ACC
Dunbar, Milam, Murray, 2002 IFAC

Experiments: Caltech Ducted Fan

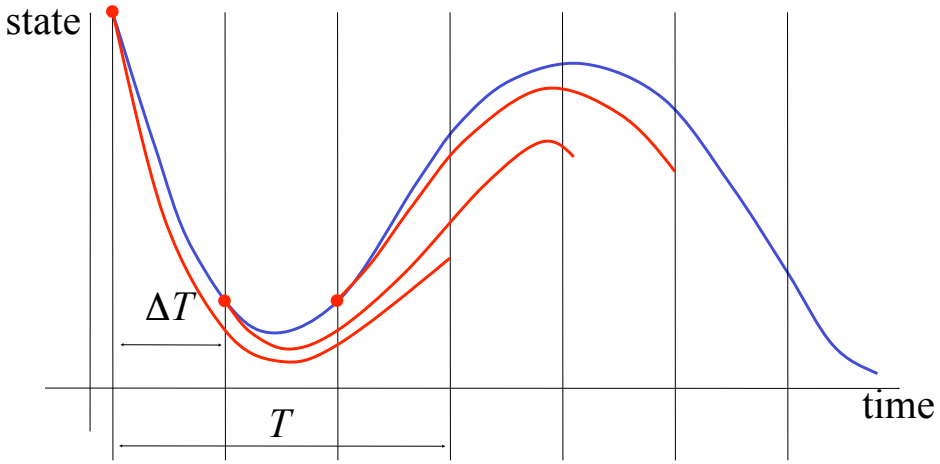




Real-Time RHC on Caltech Ducted Fan (Aug 01)

- NTG with quasi-flat outputs + *Lyapunov CLF*
- Average computation time of ~100 msec
- Inner (pitch) loop closed using local control law; RHC for position variables
- Inner/outer tradeoff: how much can be pushed into optimization

Summary: Optimization-Based Control



Receding horizon control (RHC) for constrained systems

- Allows nonlinear dynamics + input and state constraints
- Need to be careful with terminal conditions to insure stability

Differential flatness is an enabler for practical implementation of RHC [CDS 212?]

- Allows *fast* computation of (optimal) trajectories
- NTG can be used to implement RHC; works for (slightly) non-flat systems

Caltech ducted fan implementation illustrates applicability of results

- Real-time control on representative flight control platform with *no* inner loop
- Extensions to multi-vehicle testbed are being implemented