CDS 110b: Lecture 3-1
Receding Horizon Control

Richard M. Murray

18 January 2006

Goals:

- Introduce receding horizon control (RHC) for constrained systems
- Describe how to use “differential flatness” to implement RHC
- Give examples of implementation on the Caltech ducted fan

Reading:

- Notes: “Online Control Customization via Optimization-Based Control”

Homework #3

- Design some simple RHC controllers (using MATLAB)
- Due Wed, 25 Jan by 5 pm, in box outside 109 Steele

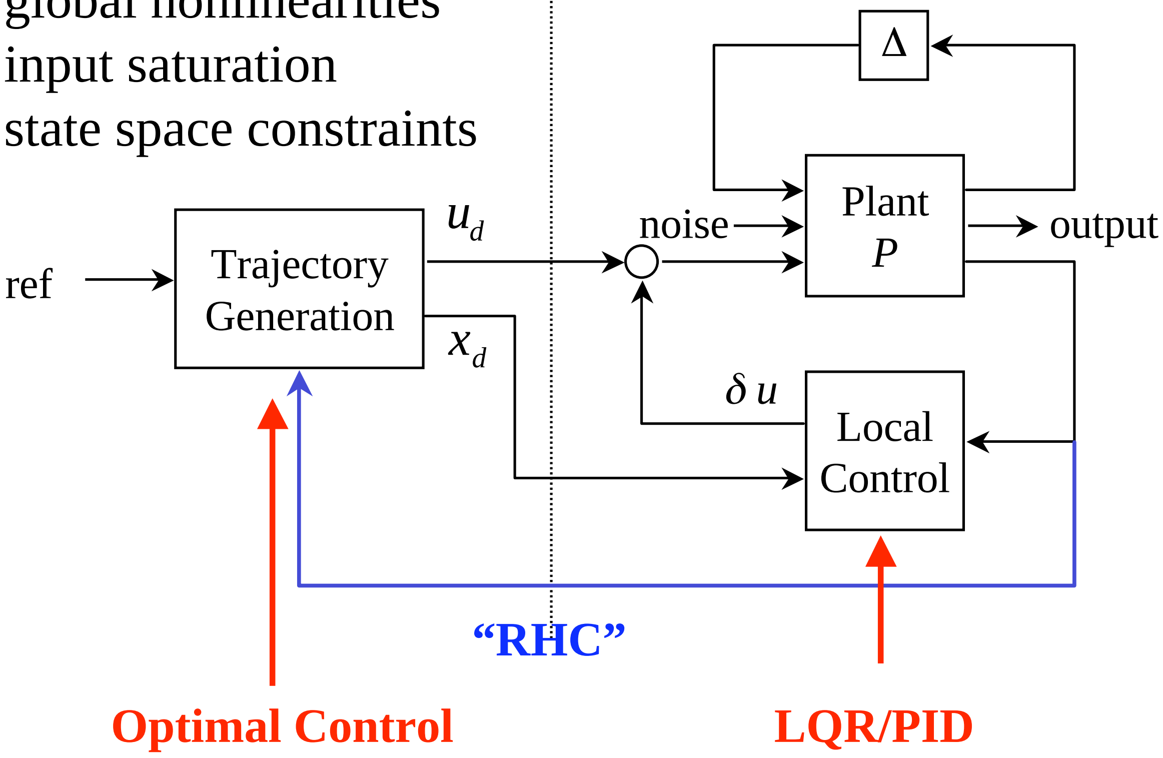
Control Architecture: Two DOF Design

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

Nonlinear design

- global nonlinearities
- input saturation
- state space constraints

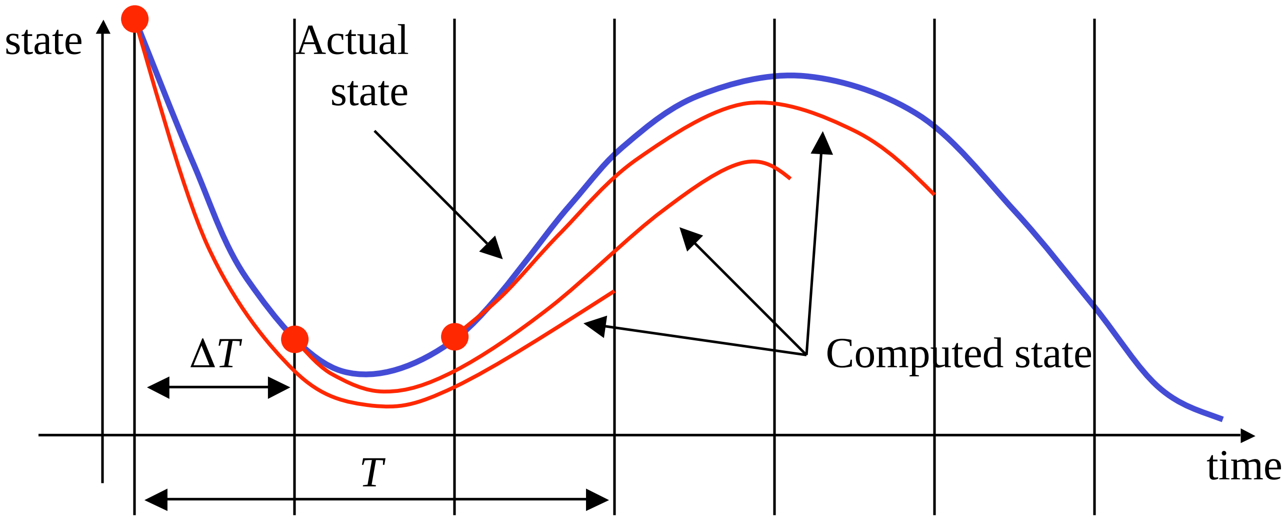
Local design



- 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 during operation

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

Receding Horizon Control



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

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

for each $x \in \Omega_r = \{x\colon V(x) < r^2\}$, for some $r > 0$. Then, for every $T > 0$ and $\delta \in (0; T]$, 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

RHC Design: Choice of Cost Function

Q: How do we choose RHC cost to get desired performance

- RHC deals w/ constraints, but shifts design freedom into choice of weights

Thm (Kalman, 1964) Given any state feedback law $u = Kx$, there exists a cost function such that the optimal controller for that cost generates the given feedback law

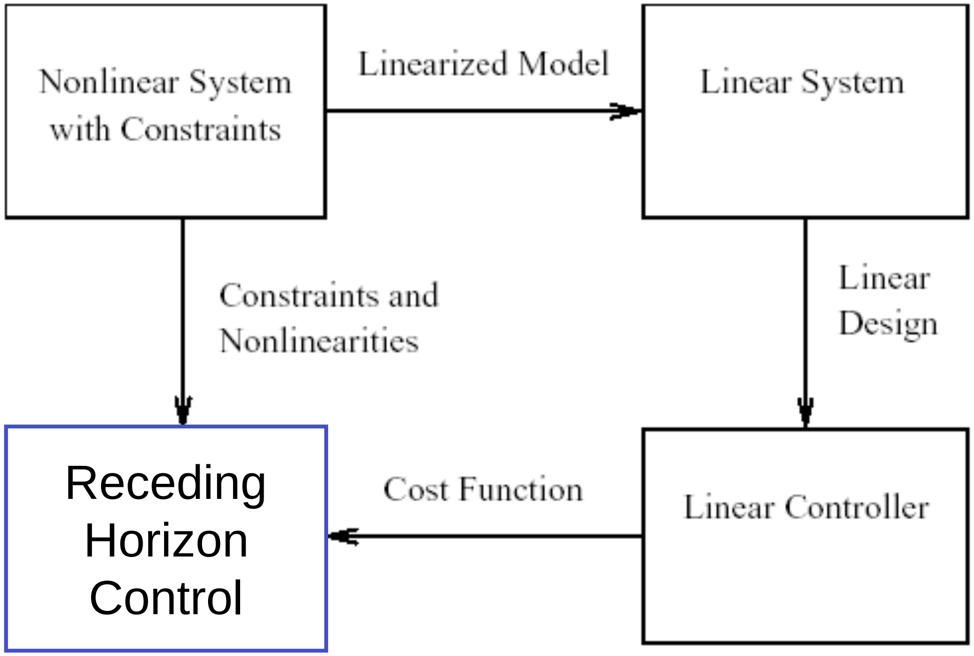
- Theorem can be used to show that finite time horizon cost function also exists
- Basic idea: solve the algebraic Riccati equation for P, Q, R given K

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$
$$-R^{-1}B^T P = K.$$

- Kalman showed you can always find positive definite solution to these eqns
- “Extension” to finite horizon problem: set $P_T = P$ and use

$$J = \int_0^T x^T Q x + u^T R u \, dt + x^T(T) P_T x(T)$$

RHC Design Philosophy



Use linear design as *specification* for RHC-based control

- Linearize system around representative operation point
- Design controller using linear tools (H_∞, loopshaping, etc)
- Compute finite horizon cost function with terminal constraint that yields controller
- Plug in to RHC computation to handle nonlinearities, constraints

Remarks

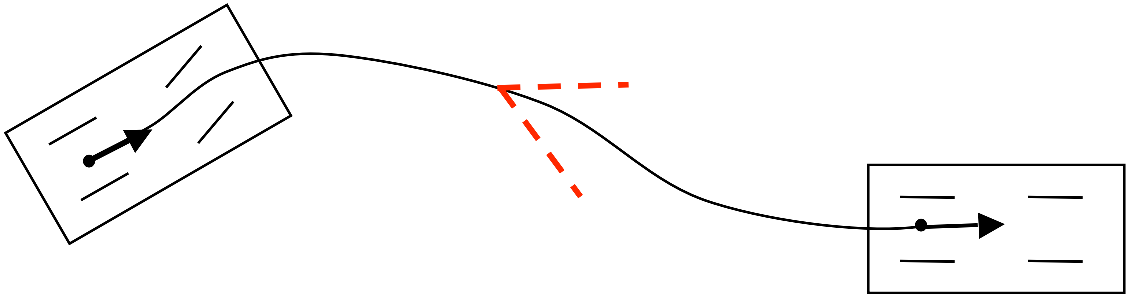
- Can extend linear state space results to NL systems with CLF-based control
- General theory of dynamic compensators (eg, loopshaping) still open

- Challenge: must be able to generate (optimal) trajectories *fast*…

Trajectory Generation Using Differential Flatness

$$\begin{aligned} \dot{x} &= f(x, u) \\ z &= h(x, u, \dot{u}, \ldots, u^{(p)}) \\ |u| &< L \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} x &= x(z, \dot{z}, \ldots, z^{(q)}) \\ u &= u(z, \dot{z}, \ldots, z^{(q)}) \end{aligned}$$

Complicated (algebraic) constraints



$$\bar{z}_0 = \begin{bmatrix} z(0) \\ \dot{z}(0) \\ \ddot{z}(0) \\ \vdots \\ z^{(q)}(0) \end{bmatrix} \quad \overset{z}{\rightsquigarrow} \quad \bar{z}_f = \begin{bmatrix} z(T) \\ \dot{z}(T) \\ \ddot{z}(T) \\ \vdots \\ z^{(q)}(T) \end{bmatrix}$$

$$z = \sum \alpha_i \psi^i(t)$$

$$M\alpha = \begin{bmatrix} \bar{z}_0 \\ \bar{z}_f \end{bmatrix}$$

- Use basis functions to parameterize output ⇒ linear problem in terms of coefficients

Optimal Control Using Differential Flatness

Can also solve constrained optimization problem via flatness

$$\min J = \int_{t_0}^{T} q(x,u)\,dt + V(x(T),u(T))$$

subject to

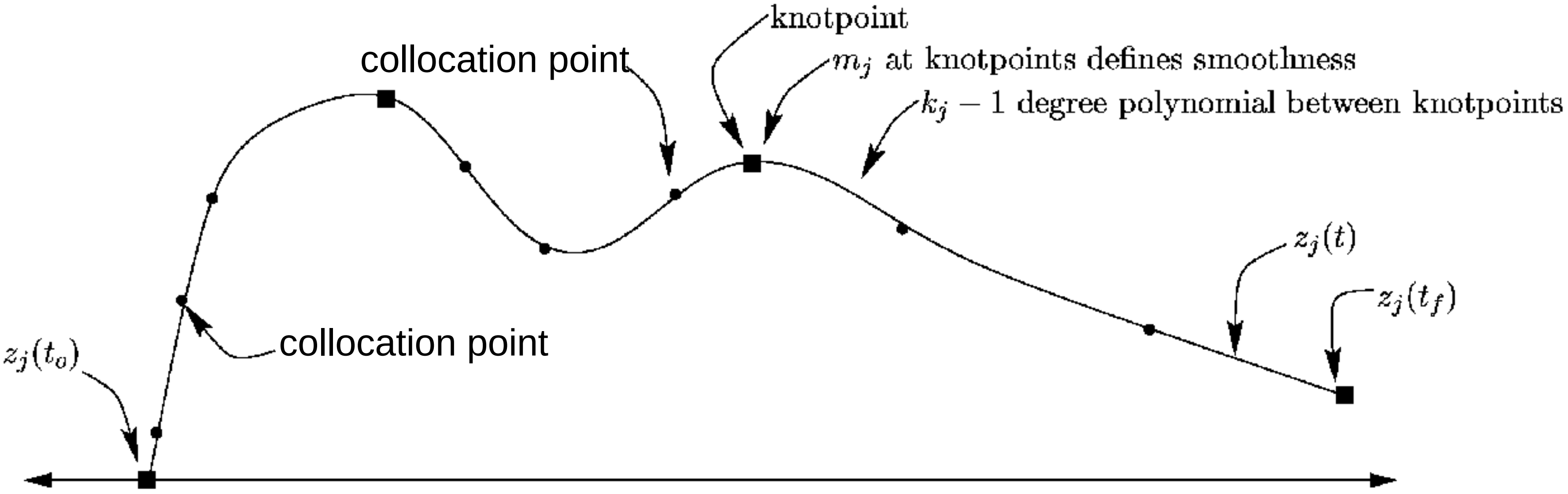
$$\dot{x} = f(x,u) \qquad g(x,u) \leq 0 \longleftarrow \left\{ \begin{array}{l} \bullet \text{ Input constraints} \\ \bullet \text{ State constraints} \end{array} \right.$$

If system is flat, once again we get an *algebraic* problem:

$$\left. \begin{array}{l} x = x(z, \dot{z}, K, z^{(q)}) \\ u = u(z, \dot{z}, K, z^{(q)}) \\ \\ z = \sum \alpha_i \psi^i(t) \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} \min J = \int_{t_0}^{T} q(\alpha,t)\,dt + V(\alpha) \\ \\ g(\alpha,t) \leq 0 \\ \text{Finite parameter } \textit{optimization} \text{ problem} \end{array} \right.$$

- Constraints hold at all times ⇒ potentially over-constrained optimization
- Numerically solve by discretizing time (collocation)

Trajectory Generation Using Splines for Flat Outputs


knotpoint
collocation point
m_j at knotpoints defines smoothness
$k_j - 1$ degree polynomial between knotpoints
$z_j(t)$
$z_j(t_f)$
$z_j(t_o)$
collocation point

Rewrite flat outputs in terms of splines

$$z_j = \sum_{i=1}^{p_j} B_{i,k_j}(t) C_i^j \quad \text{for the knot sequence } t_j$$

$$p_j = l_j(k_j - m_j) + m_j$$

Evaluate constrained optimization at collocation points:

$$\min_{y \in \mathbb{R}^M} \quad \text{subject to} \quad lb \le c(y) \le ub$$

$B_{i,kj}$ = basis functions

C_i^j = coefficients

z_i = flat outputs

Application: Caltech Ducted Fan

Flight Dynamics

$$m\ddot{x} = -D\cos\gamma - L\sin\gamma + F_{X_b}\cos\theta + F_{Z_b}\sin\theta$$

$$m\ddot{z} = D\sin\gamma - L\cos\gamma - mg_{eff} + F_{X_b}\sin\theta + F_{Z_b}\cos\theta$$

$$J\ddot{\theta} = M_a - \frac{1}{r_s} I_p \Omega \dot{x}\cos\theta + M_T$$

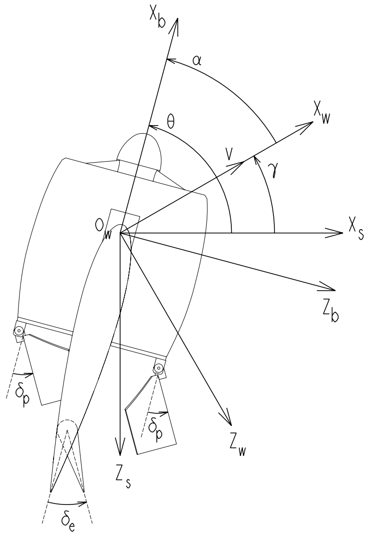
$\alpha = \theta - \gamma,$ angle of attack

$\gamma = \tan^{-1}\frac{-\dot{z}}{\dot{x}},$ flight path angle

$$L = \frac{1}{2}\rho V^2 S C_L(\alpha)$$

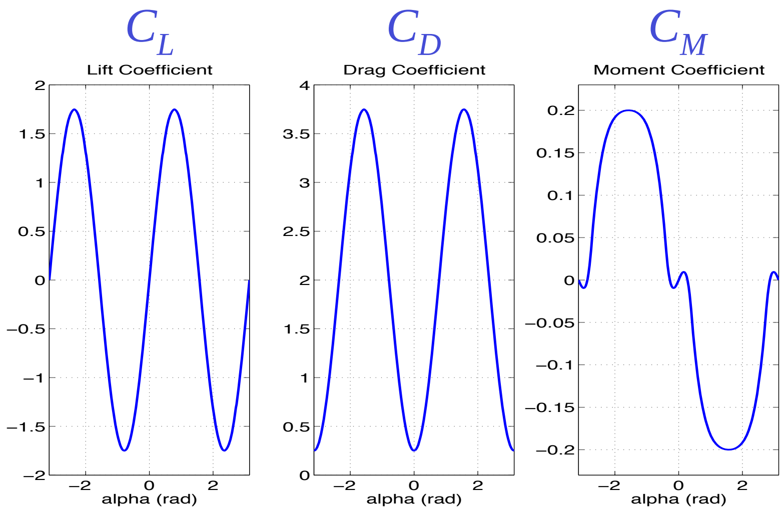
$$D = \frac{1}{2}\rho V^2 S C_D(\alpha)$$

$$M_a = \frac{1}{2}\bar{c}\rho V^2 S C_M(\alpha)$$



RHC Implementation

- System is approximately flat, even with aerodynamic forces
- More efficient to over-parameterize the outputs; use *z* = (*x, y*, θ)
- Input constraints: max thrust, flap limits, flap rates



Milam, Mushambi, M
2000 CDC

Implementation using NTG Software Library

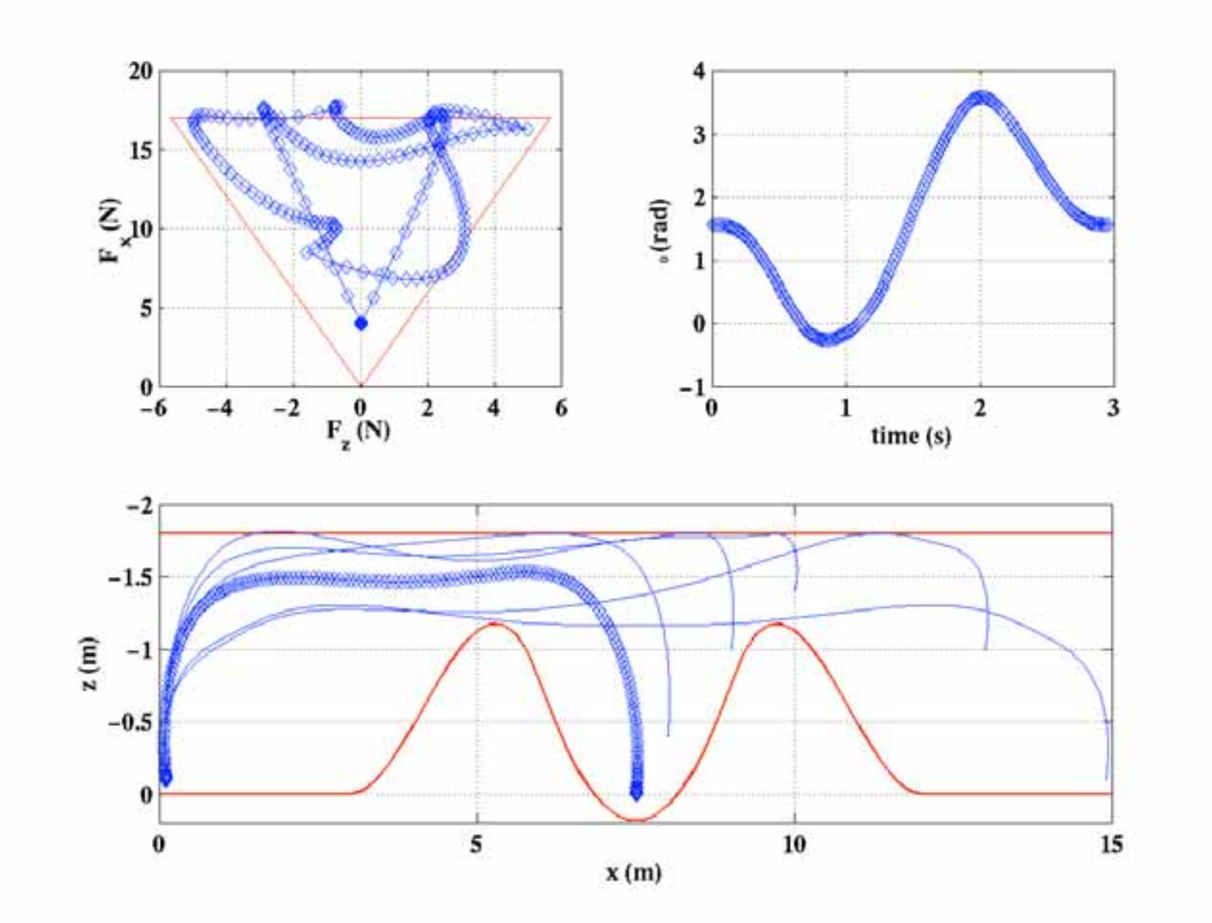
Features

- 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

Example: Trajectory Generation for the Ducted Fan

Caltech Ducted Fan

- Ducted fan engine with vectored thrust
- Airfoil to provide lift in forward flight mode
- Design to emulate longitudinal flight dynamics
- Control via dSPace-based real-time controller

Trajectory Generation Task: point to point motion avoiding obstacles

- Use differential flatness to represent trajectories satisfying dynamics
- Use B-splines to parameterize trajectories
- Solve *constrained* optimization to avoid obstacles, satisfy thrust limits

From Real-Time Trajectory Generation to RHC

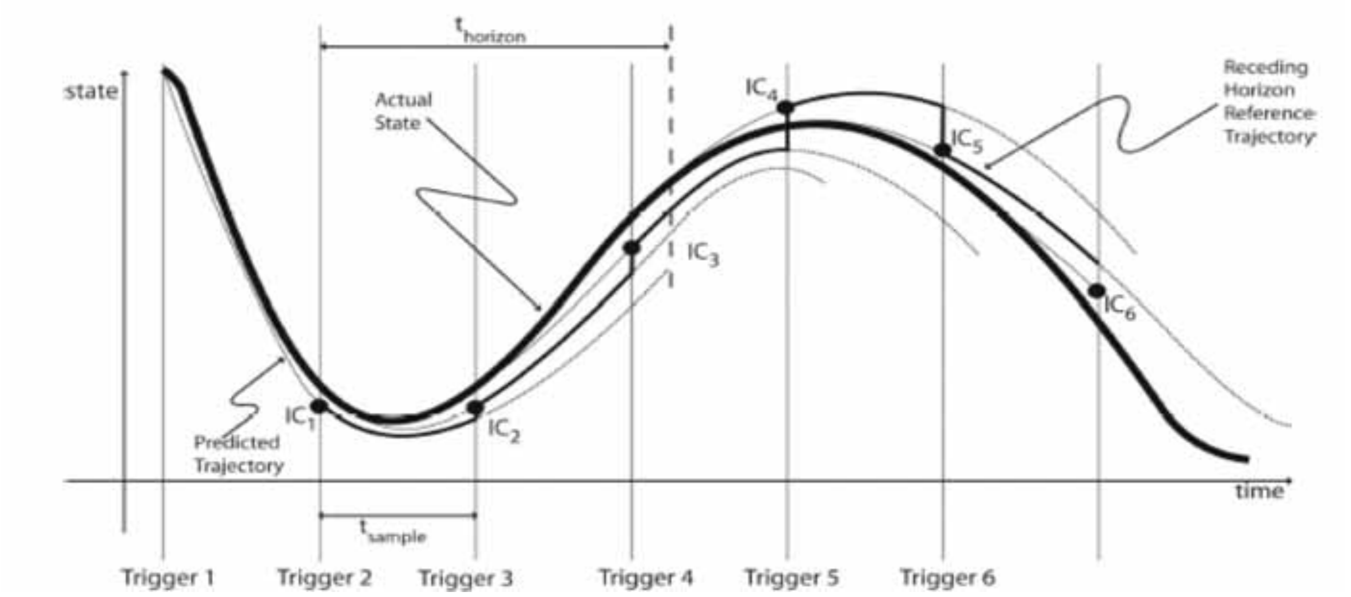
Three key elements for making RHC fast enough for motion control applications

- *Fast computation* to optimize over many variables quickly
- *Differential flatness* to minimize the number of dynamic constraints
- *Optimized algorithms* including B splines, colocation, and SQP solvers

Use of *feedback* allows substantial approximation

- OK to approximate computations since result will be recomputed using actual state
- NTG exploits this principle through the use of collocation

Can further optimize to take into account finite computation times

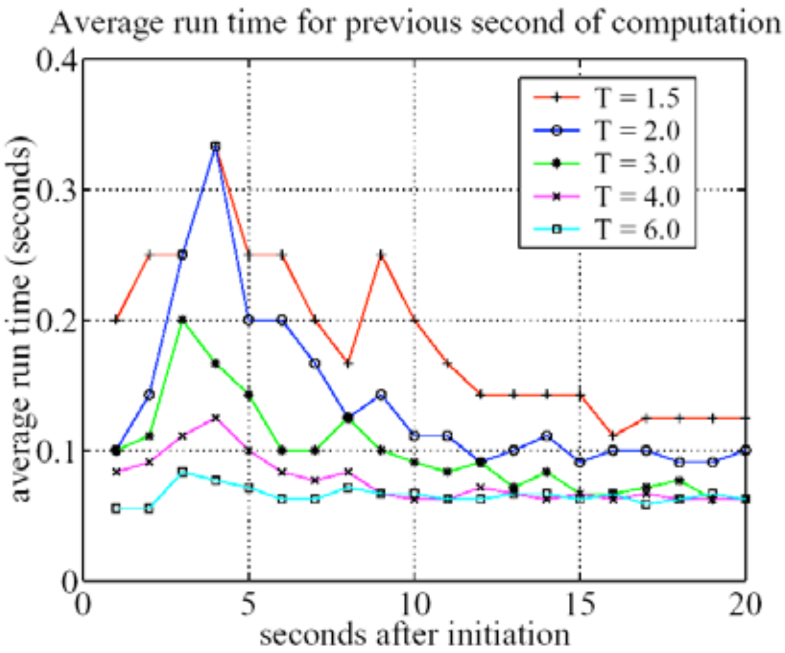


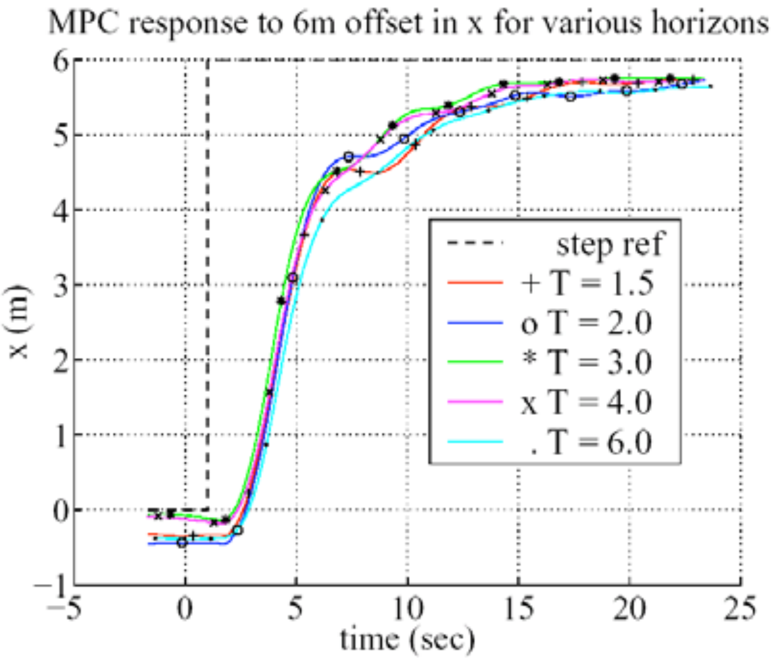
Tuning tricks

- Compute predicted state to account for computation times
- Optimize collocation times and optimization horizon
- Choose sufficiently smooth spline basis

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

Experiments: Caltech Ducted Fan

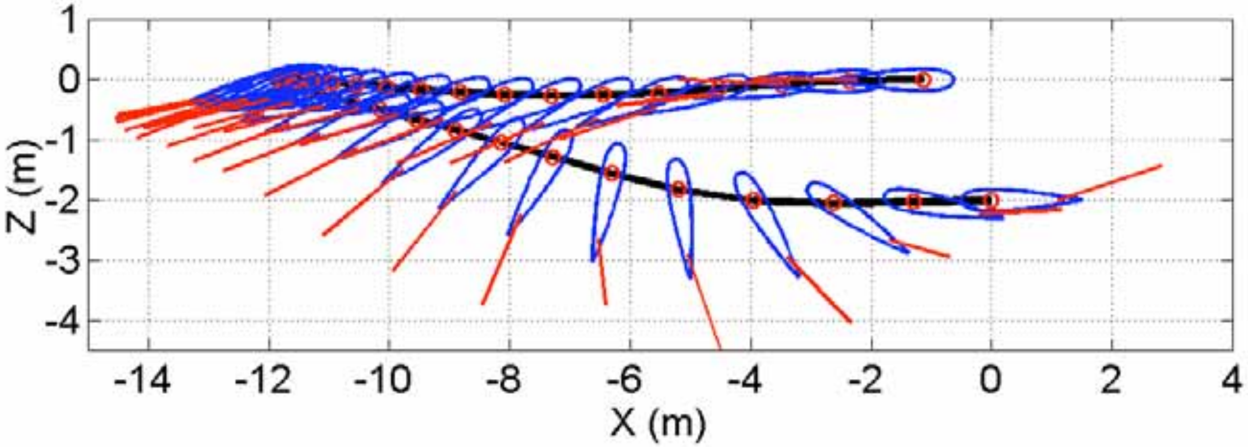




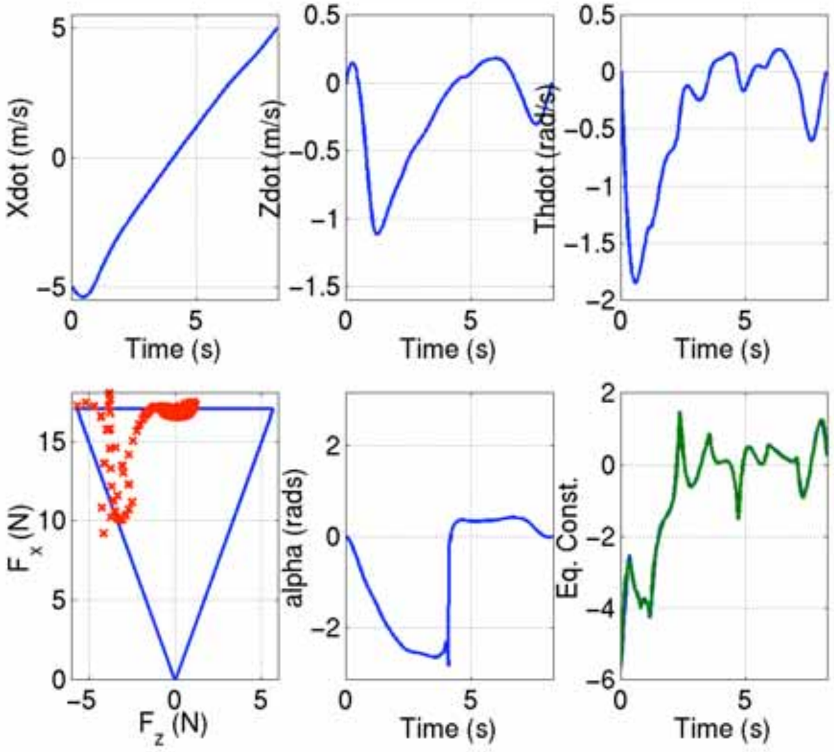
Real-Time RHC on Caltech Ducted Fan (Aug 01)

- 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

Highly Aggressive Constrained Turnaround



- Goal: -5 to 5 m/s. Final x position arbitrary, z within state constraint, Thrust vectoring within constraints
- Initial guess: Random
- Computation Time: 1.12 sec Sparc Ultra 10 83.3% CPU usage
- 6th order B-splines, seven intervals for each output, 30 equally spaced collocation points
- Full aerodynamic model



Franz, Milam et al
ACC 2002

Example: Flight Control

dSPACE-based control system

- Two C30 DSPs + two 500 MHz DEC/Compaq/HP Alpha processors
- Effective servo rates of 20 Hz (guidance loop)

Trajectory Generation for Non-Flat Systems

If system is not fully flat, can *still* apply NTG

$$\begin{tikzcd}
& z = z(x,u,\dot{u},\ldots,u^{(q)}) \arrow[r] & \begin{array}{l} x = x(z,\dot{z},\ldots,z^{(q)}) \\ u = u(z,\dot{z},\ldots,z^{(q)}) \end{array} \\
\dot{x} = f(x,u) \arrow[ur] \arrow[dr] & & \\
& y = h(x,u) \arrow[r] & \begin{array}{r} (x,u) = \Gamma(y,\dot{y},\ldots,y^{(q)}) \\ 0 = \Phi(y,\dot{y},\ldots,y^{(p)}) \end{array}
\end{tikzcd}$$

When system is not flat, use *quasi-collocation*

- Choose output *y*=*h*(*x,u*) that can be used to compute the full state and input
- Remaining dynamics are treated as *constraints* for trajectory generation
- Example: chain of integrators

$$\begin{tikzcd}
\begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = u \end{array} \arrow[r] & \begin{array}{l} y_1 = x_1 \\ y_2 = x_2 \end{array} \arrow[r] & \left.\begin{array}{l} x_1 = y_1 \\ x_2 = y_2 \\ u = \dot{y}_2 \end{array} + \quad \dot{y}_1 = y_2 \right\}
\end{tikzcd}$$

Solve using NTG with $lb = ub$

Can also do full collocation (treat all dynamics as constraints)

$$\left.\begin{array}{l} (x,u) = \sum \alpha_i \psi^i(t) \\ \dot{x}(t_i) = f(x(t_i),u(t_i)) \end{array}\right\}$$

Each equation gives constraints at collocation points $\Rightarrow$ highly constrained optimization

Petit, Milam, Murray
NOLCOS, 2001

Effect of Defect on Computation Time

Defect as a measure of flatness

- Defect = number of remaining differential equations
- Defect 0 ⇒ differentially flat

Sample problem: 5 states, 1 input

- x_1 is possible flat output
- Can choose other outputs to get systems with nonzero *defect*
- 200 runs per case, with random initial guess

Computation time related to defect through power law

- SQP scales cublicly ⇒ minimize the number of free variables

Dramatic speedup through reduction of differential constraints

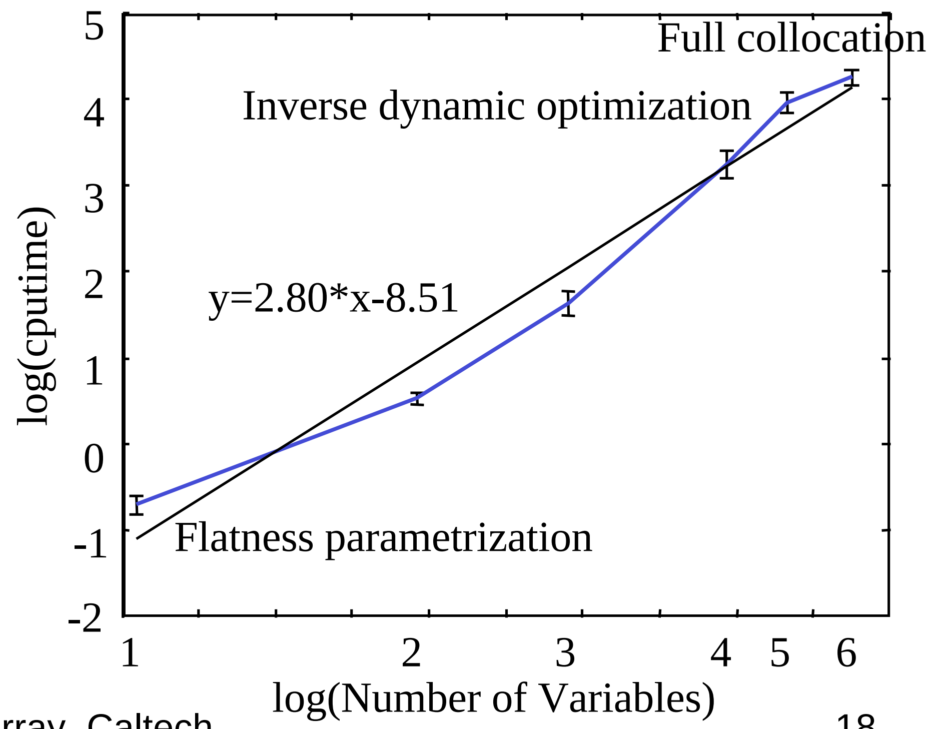
$$\dot{x}_1 = 5x_2$$

$$\dot{x}_2 = \sin x_1 + x_2^2 + 5x_3$$

$$\dot{x}_3 = -x_1 x_2 + x_3 + 5x_4$$

$$\dot{x}_4 = x_1 x_2 x_3 + x_2 x_3 + x_4 + 5x_5$$

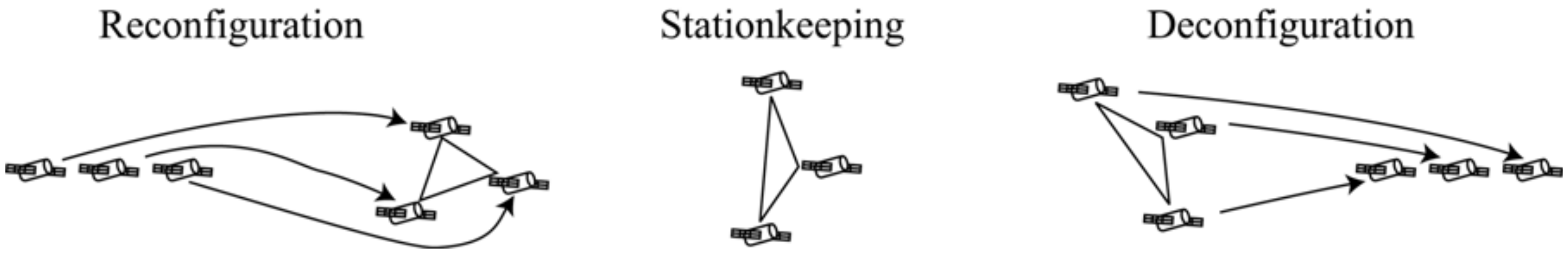
$$\dot{x}_5 = -x_5 + u$$



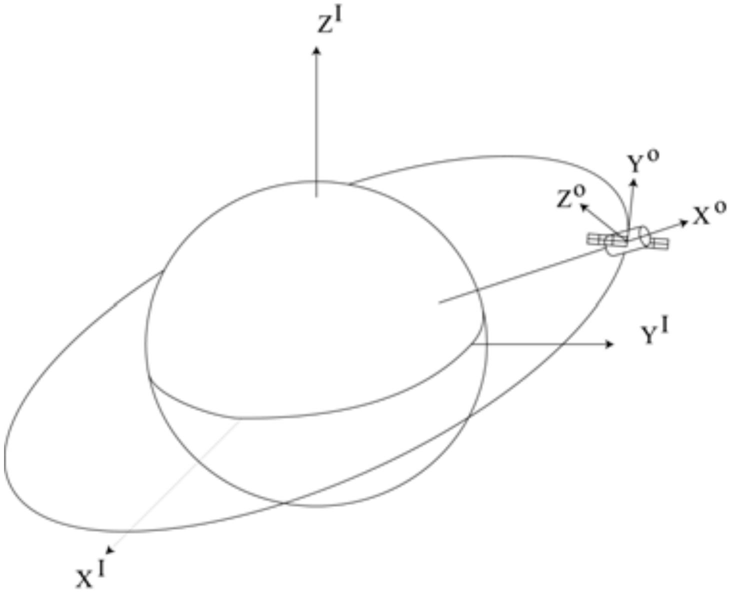
Milam, Petit, Murray
AIAA GNC, 2001

Example 2: Satellite Formation Control

Goal: reconfigure cluster of satellites using minimum fuel



Dynamics given by Hill's equations (fully actuated ⇒ flat)



$$\ddot{q}_1 = \frac{\mu q_1}{|\vec{q}|^3} - \frac{3J_2\mu R_e^2 q_1\left(q_1^2+q_2^2-4q_3^2\right)}{2|\vec{q}|^7} + u_1^I$$

$$\ddot{q}_2 = \frac{\mu q_2}{|\vec{q}|^3} - \frac{3J_2\mu R_e^2 q_2\left(q_1^2+q_2^2-4q_3^2\right)}{2|\vec{q}|^7} + u_2^I$$

$$\ddot{q}_3 = \frac{\mu q_3}{|\vec{q}|^3} - \frac{3J_2\mu R_e^2 q_3\left(3q_1^2+3q_2^2-2q_3^2\right)}{2|\vec{q}|^7} + u_3^I$$

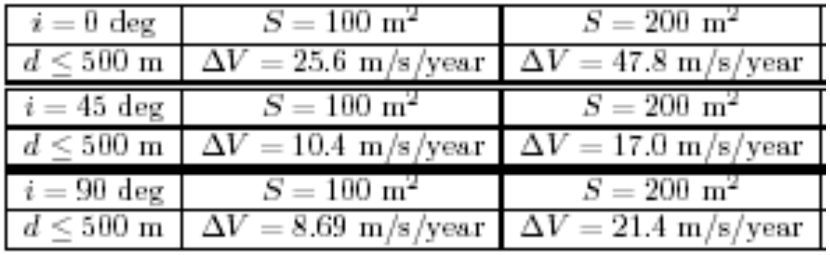
Satellite Formation Results

Station-keeping optimization

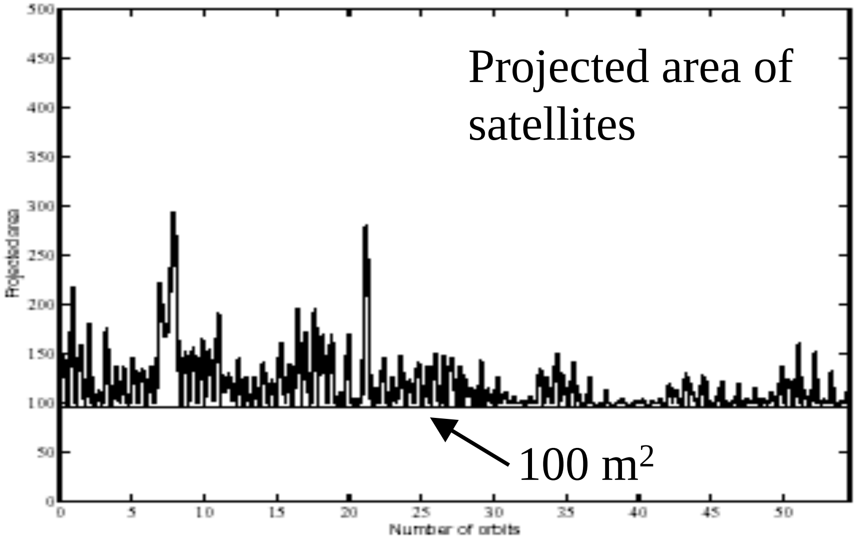
- Maintain a given area between the satellites (for good imaging) while minimizing the amount of fuel
- Idea: exploit natural dynamics of orbital equations as much as possible
- Input constraints: $\Delta V < 20$ m/s/year

Results

- Use NTG to optimize over 60 orbits (~3 days), then repeat
- Results: at 45° inclination, obtain 10.4 m/s/year



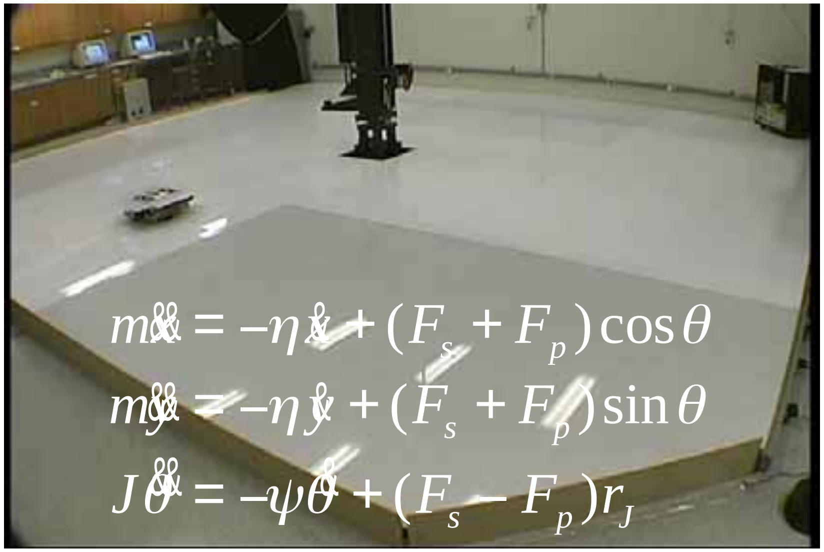
$i = 0$ deg	$S = 100$ m^2	$S = 200$ m^2
$d \leq 500$ m	$\Delta V = 25.6$ m/s/year	$\Delta V = 47.8$ m/s/year
$i = 45$ deg	$S = 100$ m^2	$S = 200$ m^2
$d \leq 500$ m	$\Delta V = 10.4$ m/s/year	$\Delta V = 17.0$ m/s/year
$i = 90$ deg	$S = 100$ m^2	$S = 200$ m^2
$d \leq 500$ m	$\Delta V = 8.69$ m/s/year	$\Delta V = 21.4$ m/s/year



Projected area of satellites

100 m^2

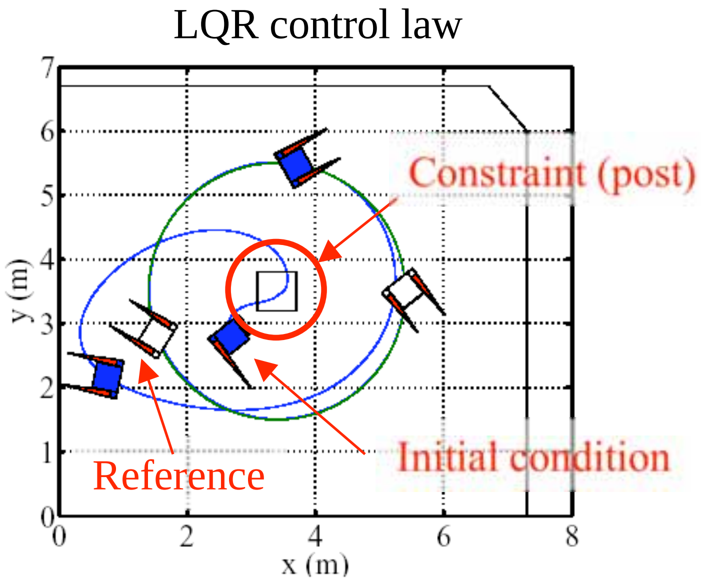
Example 3: MVWT Control Design

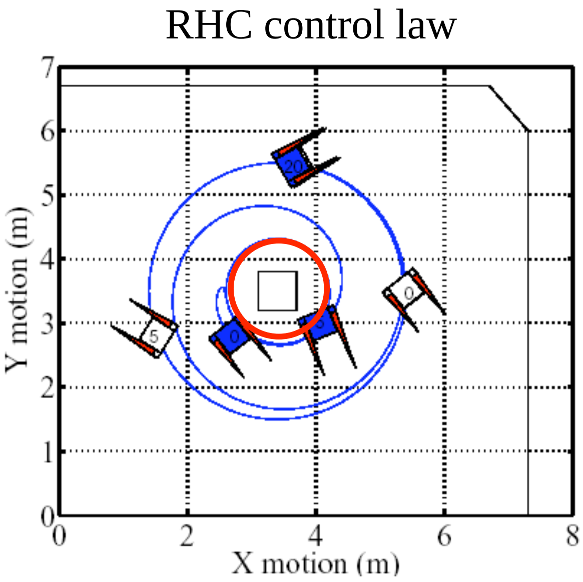


$$m\ddot{x} = -\eta\dot{x} + (F_s + F_p)\cos\theta$$

$$m\ddot{y} = -\eta\dot{y} + (F_s + F_p)\sin\theta$$

$$J\ddot{\theta} = -\psi\dot{\theta} + (F_s - F_p)r_J$$


LQR control law

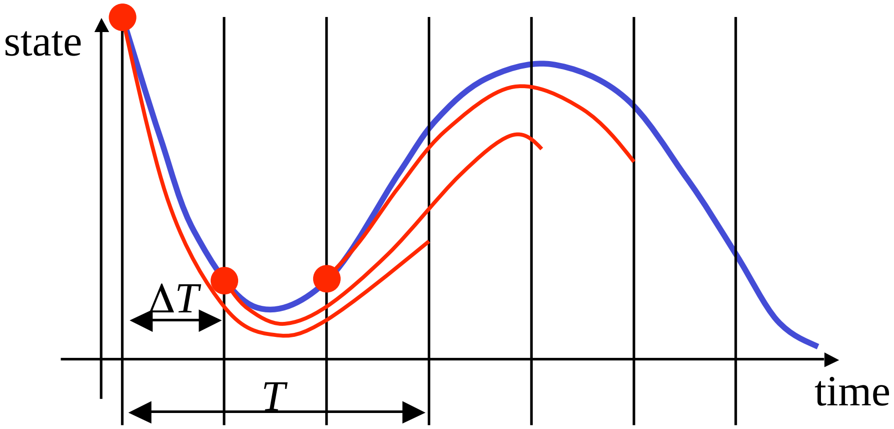

RHC control law

Control design technique

1. LQR design of state space controller K around reference velocity
2. Choose P, Q, R using Kalman’s formula
3. Implement as a receding horizon control with input and state space constraints

- RHC controller respects state space constraint

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

- 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