NCS Lecture 5
Optimization-Based Control

Richard M. Murray

18 March 2008

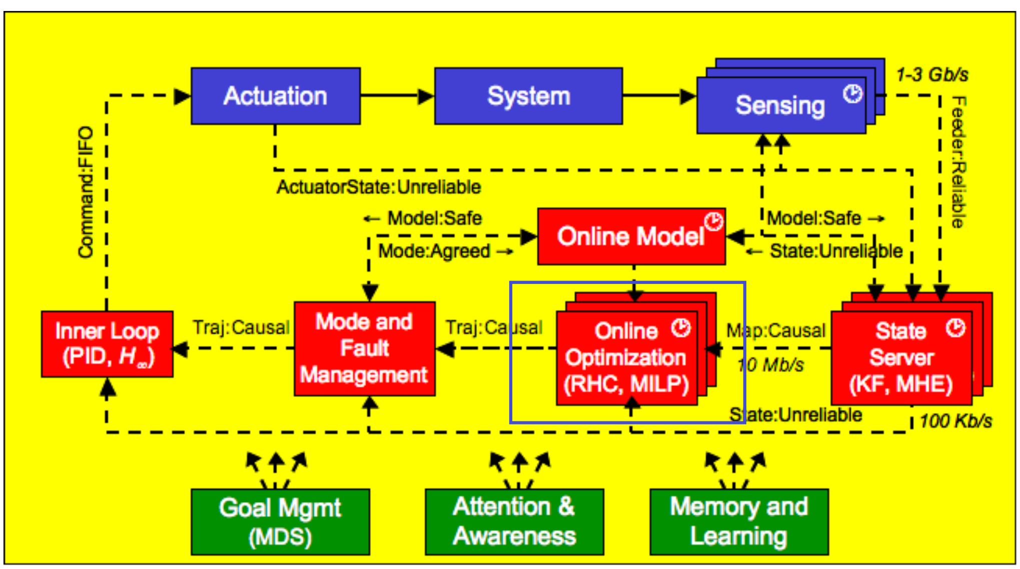
Goals:

- Introduce two degree of freedom design for motion control systems
- Describe how to use flatness for real-time motion planning using NTG
- Describe the use of receding horizon control for online customization
- Give examples of implementation on Caltech ducted fan

Reading:

- "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems", M. B. Milam, K. Mushambi and R. M. Murray. Conference on Decision and Control, 2000.
- "Inversion Based Constrained Trajectory Optimization", N. Petit, M. B. Milam and R. M. Murray. IFAC Symposium on Nonlinear Control Systems Design (NOLCOS), 2001.

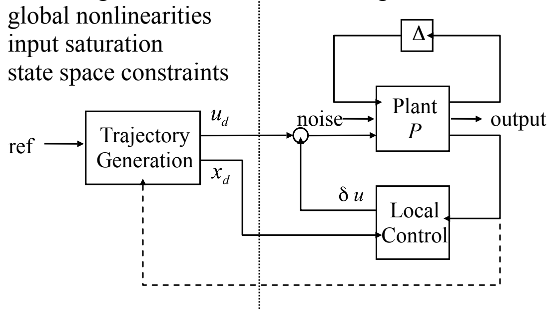
Networked Control Systems



Real-Time Trajectory Generation Using Flatness

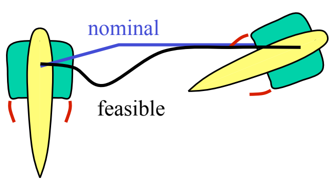
Nonlinear design
- global nonlinearities
- input saturation
- state space constraints

Linear design

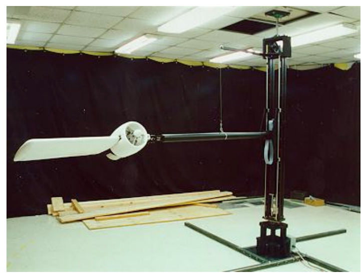


Approach: Two Degree of Freedom Design
- Use online trajectory generation to construct feasible trajectories
- Use linear control for local performance
- For many systems, dynamics are differentially flat ⇒ reduce dynamic system to algebraic equivalent and generate feasible trajectories in real time

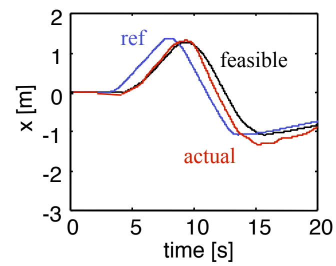
Rapid Transition from Hover to Forward Flight



Caltech Ducted Fan



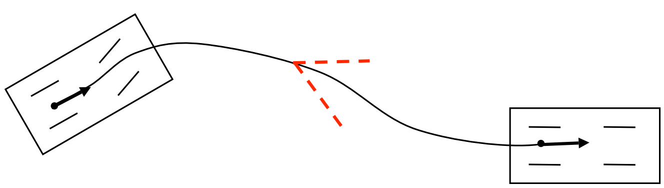
Real-Time Trajectory Generation



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

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

subject to

$$\dot{x} = f(x,u) \qquad g(x,u) \leq 0$$

- Input constraints
- State constraints

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

$$\left.\begin{aligned} x &= x(z, \dot{z}, \dots, z^{(q)}) \\ u &= u(z, \dot{z}, \dots, z^{(q)}) \\ z &= \sum \alpha_i \psi^i(t) \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} &\min J = \int_{t_0}^{T} L(\alpha,t)\,dt + V(\alpha) \\ &g(\alpha,t) \leq 0 \end{aligned}\right.$$

Finite parameter *optimization* problem

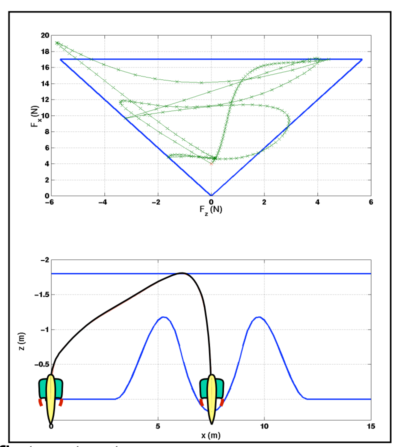
- Constraints hold at all times ⇒ potentially over-constrained optimization
- Numerically solve by discretizing time (collocation)

NTG: Nonlinear Trajectory Generation

Petit, Milam, Murray
NOLCOS, 2001

Flatness-based optimal control package

- B-spline representation of (partially) flat outputs
- Collocation based optimization approach
- Built on NPSOL optimization pkg (requires license)
- Warm start capability for receding horizon control



Solves general nonlinear optimization problem

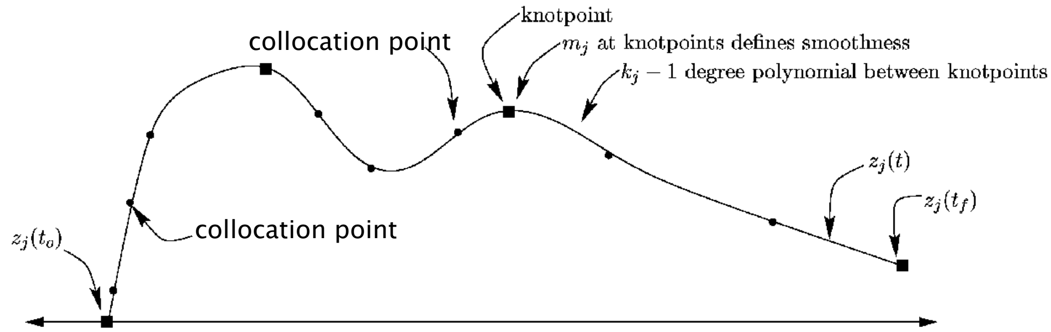
$$\min J = \int_{t_0}^{T} q(x,u)\,dt + V(x(T),u(T))$$

$$\dot{x} = f(x,u) \qquad lb \leq g(x,u) \leq ub$$

- Assumes x and u are given in terms of (partially) flat outputs
- Constraints are enforced at a user-specified set of collocation points
- Gives *approximate* solution; need to use w/ feedback to ensure robustness (2 DOF)

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

Trajectory Generation Using Splines for Flat Outputs



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:

$$\min_{\vec{C} \in \mathbb{R}^M} J(\bar{z}(t_i)) \quad \text{subject to} \quad lb \le c(\bar{z}(t_i)) \le ub$$

$B_{i,kj}$ = basis functions

C_i^j = coefficients

z_i = flat outputs

Application: Caltech Ducted Fan

Flight Dynamics

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

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

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

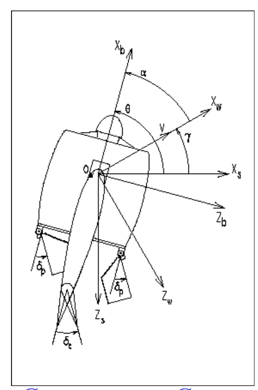
$$L = \frac{1}{2}\rho V^2 S C_L(\alpha)$$

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

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

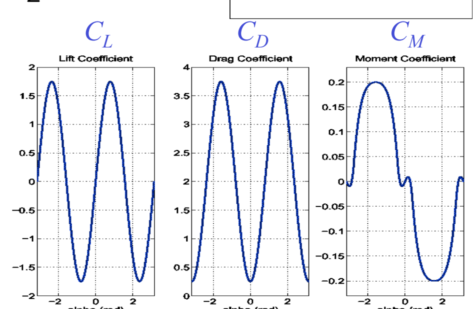
$\alpha = \theta - \gamma,$ angle of attack

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



Trajectory Generation Implementation

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



Milan, Mushambi, M
2000 CDC

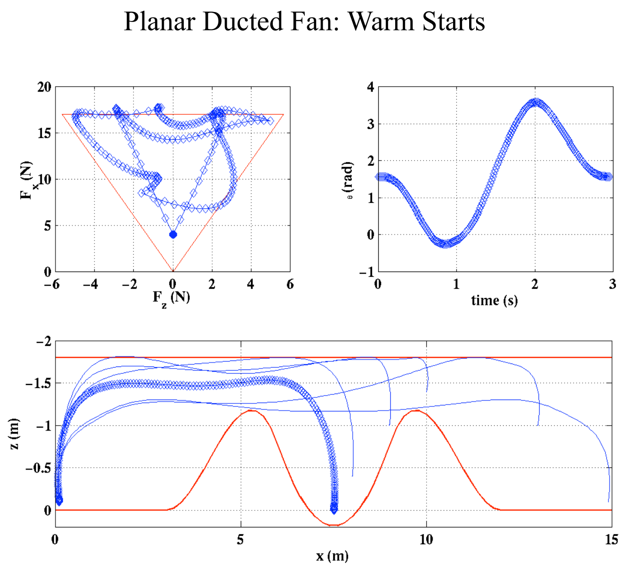
Implementation using NTG Software Library

Features

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

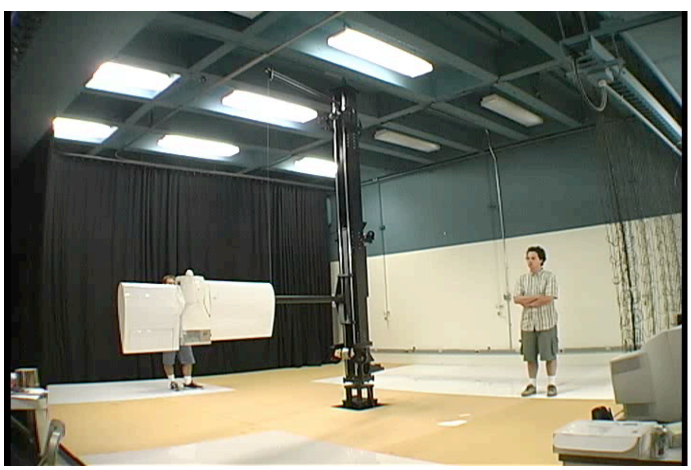
Weaknesses

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



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

Example 1: 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

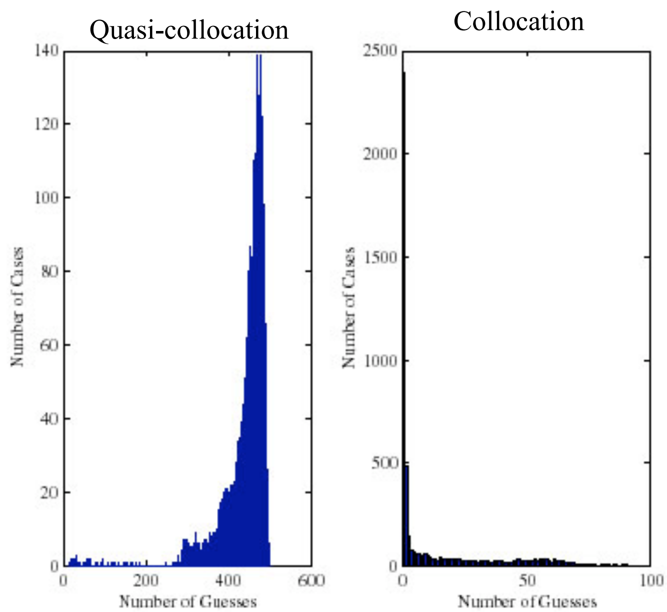
NTG Convergence Properties

Numerical Studies using Caltech Ducted Fan

- 6461 test cases
- 500 initial guess for spline coefficients
- Total of > 3M runs

- Count # of cases that converge for given # of initial guesses

- Comparison between quasi-collocation (x, y, θ) and full collocation (states and inputs)



Trajectory Generation for Non-Flat Systems

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

$$\dot{x} = f(x,u)$$

$$z = z(x,u,\dot{u},\ldots,u^{(q)}) \longrightarrow \begin{array}{l} x = x(z,\dot{z},\ldots,z^{(q)}) \\ u = u(z,\dot{z},\ldots,z^{(q)}) \end{array}$$

$$y = h(x,u) \longrightarrow \begin{array}{l} (x,u) = \Gamma(y,\dot{y},\ldots,y^{(q)}) \\ 0 = \Phi(y,\dot{y},\ldots,y^{(p)}) \end{array}$$

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

$$\begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = u \end{array} \longrightarrow \begin{array}{l} y_1 = x_1 \\ y_2 = x_2 \end{array} \longrightarrow \left.\begin{array}{l} x_1 = y_1 \\ x_2 = y_2 \quad + \quad \dot{y}_1 = y_2 \\ u = \dot{y}_2 \end{array}\right\} \text{Solve using NTG with } lb = ub$$

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

$$\left.\begin{array}{l} (x,u) = \sum \alpha_i \psi^i(t) \\ \dot{x}(t_i) = f(x(t_i),u(t_i)) \end{array}\right\} \text{Each equation gives constraints at collocation points} \Rightarrow \text{highly constrained optimization}$$

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 $\Rightarrow$ differentially flat

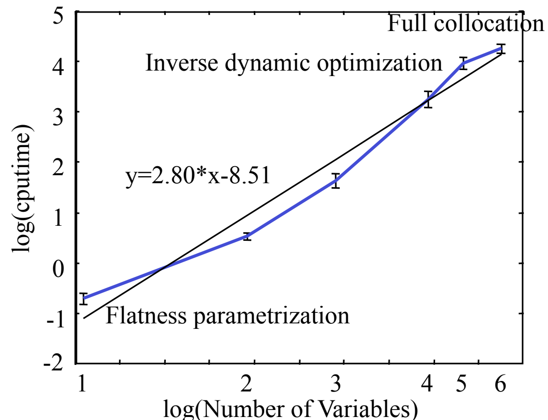
Sample problem: 5 states, 1 input

- 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 $\Rightarrow$ minimize the number of free variables

$$\dot{x}_1 = 5x_2$$
$$\dot{x}_2 = \sin x_1 + x_2^2 + 5x_3$$
$$\dot{x}_3 = -x_1 x_2 + x_3 + 5x_4$$
$$\dot{x}_4 = x_1 x_2 x_3 + x_2 x_3 + x_4 + 5x_5$$
$$\dot{x}_5 = -x_5 + u$$



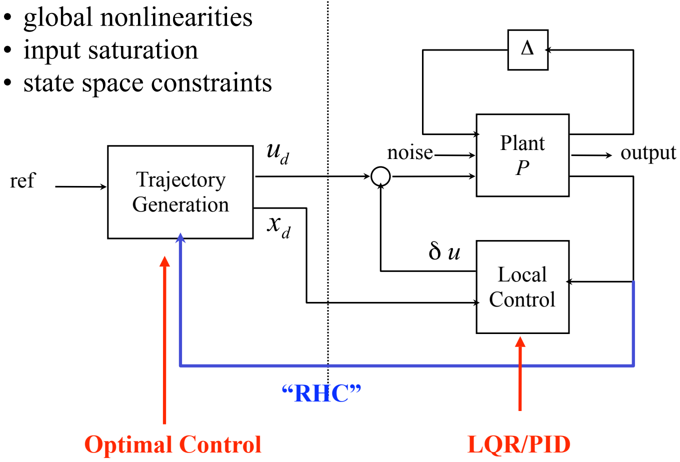
Dramatic speedup through reduction of differential constraints

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

Control Architecture: Two DOF Design

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

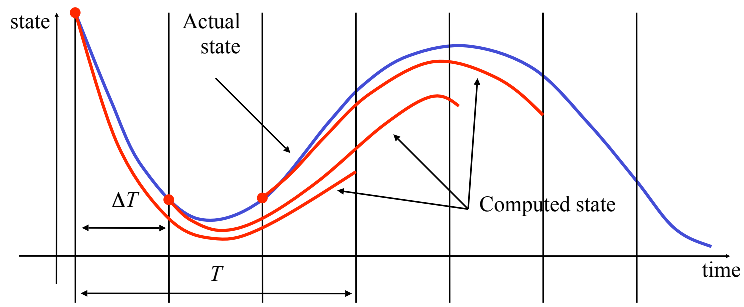
Local design



- 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 during operation

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 $x \in \Omega_r = \{x\colon V(x) < r^2\}$, 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

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

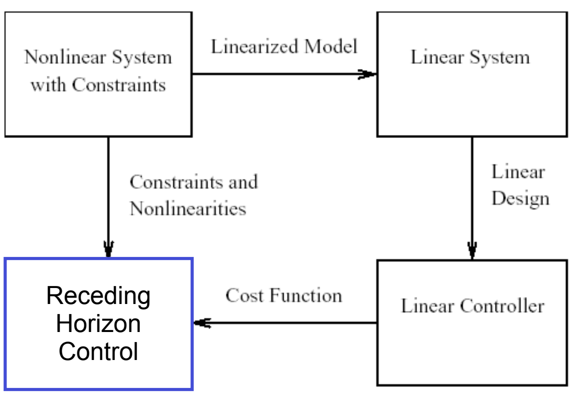
- Theorem can be used to show that finite time horizon cost function also exists
- Basic idea: solve the algebraic Riccati equation for P, Q, R given K

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

- 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*…

Application: Caltech Ducted Fan

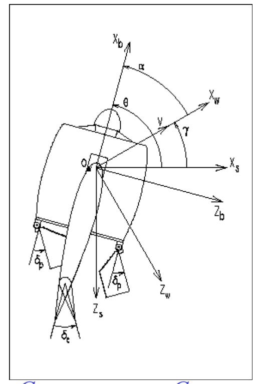
Flight Dynamics

$$m\ddot{x} = -D\cos\gamma - L\sin\gamma + F_{X_b}\cos\theta + F_{Z_b}\sin\theta$$
$$m\ddot{z} = D\sin\gamma - L\cos\gamma - mg_{eff} + F_{X_b}\sin\theta + F_{Z_b}\cos\theta$$
$$J\ddot{\theta} = M_a - \frac{1}{r_s} I_p \Omega \dot{x}\cos\theta + M_T$$

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

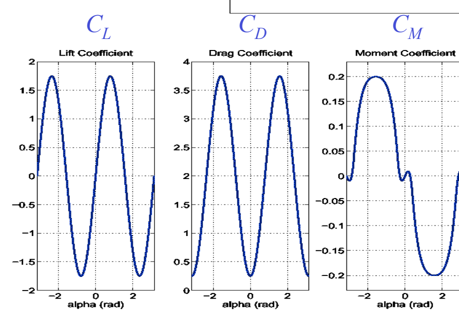
$\gamma = \tan^{-1}\frac{-\dot{z}}{\dot{x}},$ flight path angle

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



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



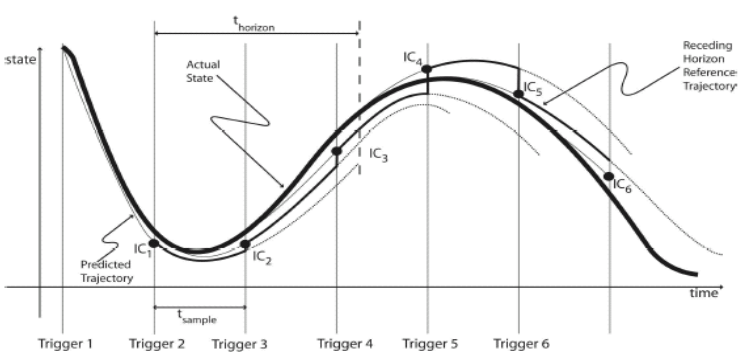
From Real-Time Trajectory Generation to RHC

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

- 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

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

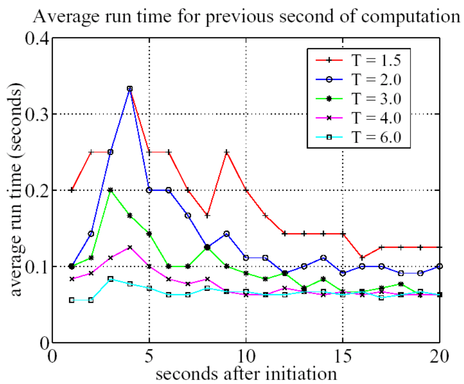


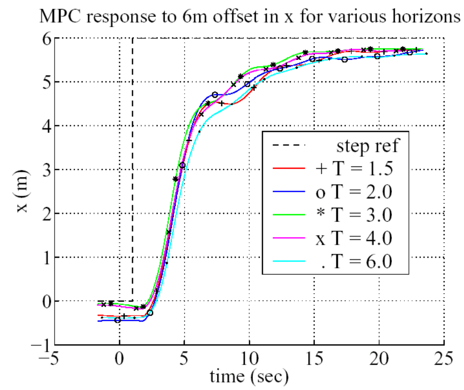
Tuning tricks

- 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

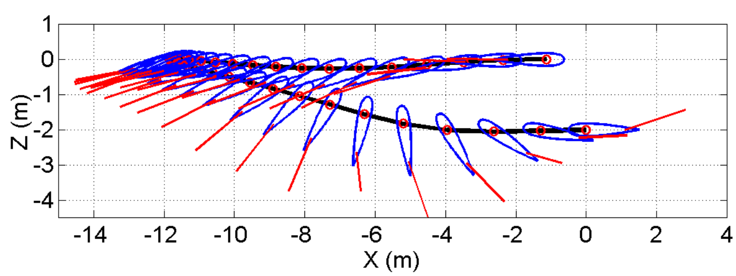




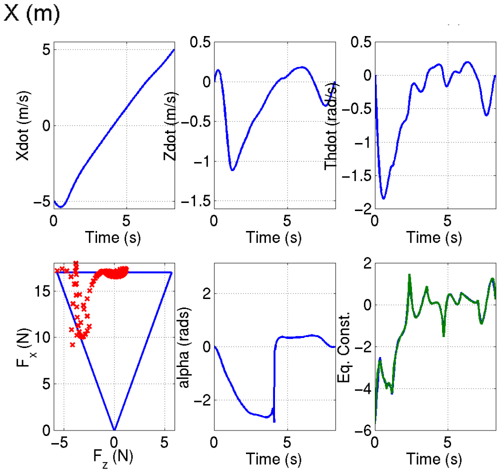
Real-Time RHC on Caltech Ducted Fan (Aug 01)

- 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

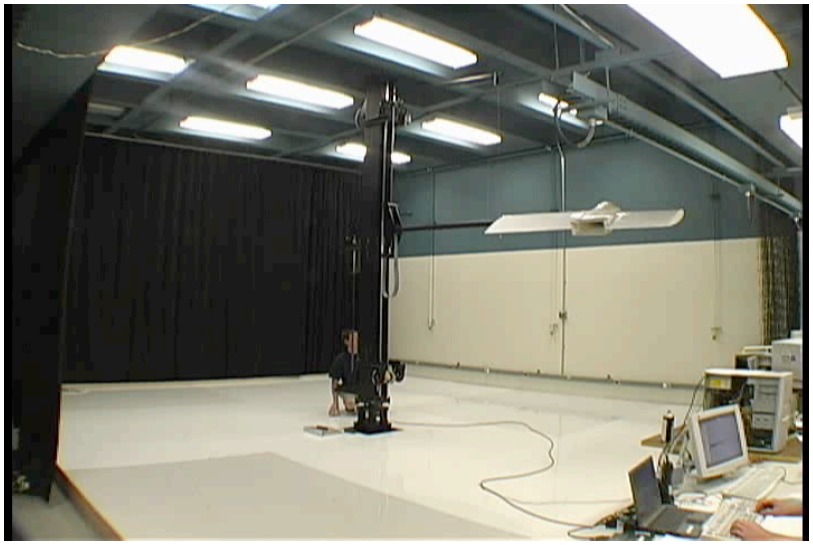


- Goal: -5 to 5 m/s. Final x position arbitrary, z within state constraint, Thrust vectoring within constraints
- Initial guess: Random
- Computation Time: 1.12 sec Sparc Ultra 10 83.3% CPU usage
- 6th order B-splines, seven intervals for each output, 30 equally spaced collocation points
- Full aerodynamic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



Franz, Milam et al
ACC 2002

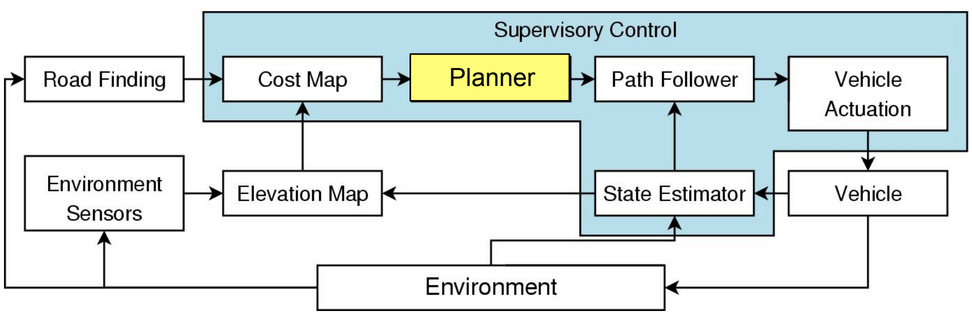
Example: Flight Control



dSPACE-based control system

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

Path Planner Specification



Planner Specification

- 15 cm high obstacles at ~7.5 m/s
- 30 cm high/deep obstacles at ~1 m/s
- Detect and avoid situations that are worse than this

Method:

- Two stage, optimization-based planner
- Cost function given by velocity maps

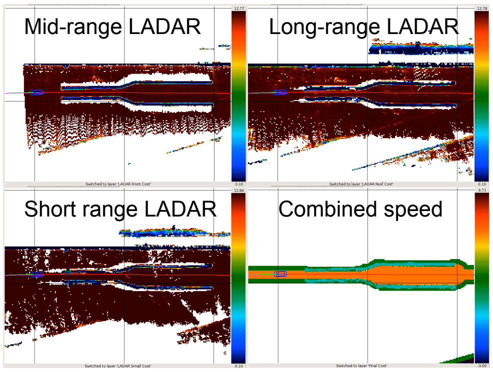
Inputs

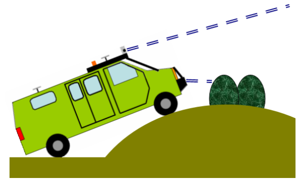
- Speed maps from fusionMapper (sent as deltas)
- Current position of vehicle (state)
- Supervisory commands

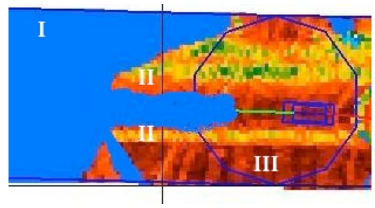
Outputs

- Planned trajectory for next 20-40 m

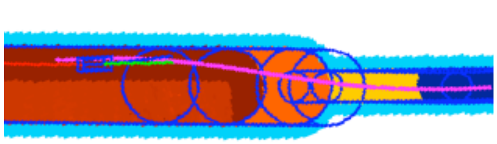
Sensor Fusion and Cost Map Processing



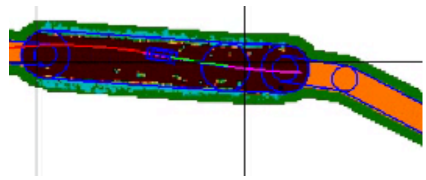




Simple cost map:



Realistic cost map:



Planner Approach

Underlying strategy

- Nonlinear programming optimization with SNOPT
- Traversal time optimized over space of trajectories
- Dynamic feasibility as constraints
- Obstacle avoidance as constraint/cost
- Receding horizon

Reparameterization

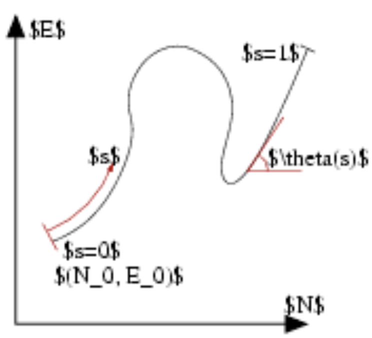
- Choose θ, *v* as indep variables; parameterize by quadratic splines
- Integrate from initial pos to get N, E
- Impose speed limit, accel limit, steering mag/rate limits, rollover constraints
- Minimize traversal time

$$T = S_f \int_0^1 \frac{1}{v(N(s), E(s))} ds$$

plus steering, acceleration "effort"

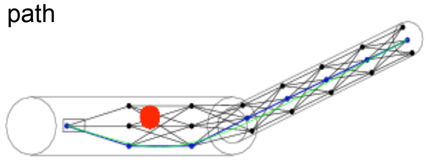
Vehicle dynamics

$$\begin{aligned} \dot{N} &= v\cos\theta & & s.t. \\ \dot{E} &= v\sin\theta & \phi &\in [\phi_{min}, \phi_{max}] \\ \dot{\theta} &= \frac{v}{L}\tan\phi & \omega &\in [\omega_{min}, \omega_{max}] \\ \dot{\phi} &= \omega = u_1 & v &\in (0, v_{max}] \\ \dot{v} &= a = u_2 & a &\in [a_{min}, a_{max}] \end{aligned}$$



Additional Modifications

Seed path (stage 1 planner)
- Provides initial conditions for optimization based planner
- Minimize time based on velocity along path



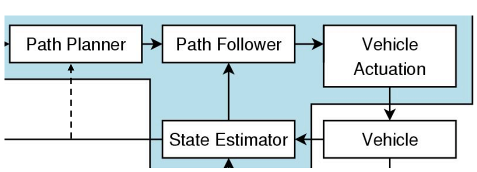
Obstacle convexification
- Flat obstacles create problems for optimization algorithm
- Solution: create gradient on obstacles to allow optimizer convergence

Growing obstacles
- Need to grow obstacles to take into account vehicle width
- Use orientation of obstacles in deciding how to grow

No data regions
- Encode no data regions as “negative” velocity in maps
- No data regions inside vehicle stopping distance: use low speed (3 m/s)
- No data regions further than stopping distance: use 2X current speed
- Allows vehicle to keep moving at high speed until slowing down is required

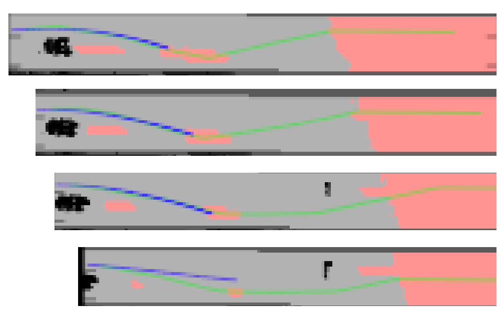
Receding Horizon Implementation

Problems with RHC implementation
- Original plan: replan trajectory from current path
- Issue: gave very oscillatory operation
- Cause: interaction between planner and trajectory tracker - update times too close



Solution: plan from *previous* path
- Use planner to mitigate uncertainty from new obstacles in obstacle map
- Use follower to mitigate uncertainty in vehicle dynamics, path changes

Example: NQE run
- Seed path in green; planner path in blue
- Dark areas = obstacles
- Pink area = no data regions



Notes
- Path changes as obstacles appear
- No data regions beyond stopping distance treated favorably

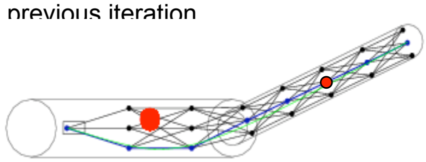
Interface Issues

Planning horizon

- Allow planning to increase as speed increases
- Min horizon = 25 m, max = 65 m
- Use lag to avoid rapid changes

Stage 1 "indecisiveness"

- If obstacle is directly in front of vehicle, stage 1 can switch sides each iteration
- Solution: penalize distance from path of previous iteration

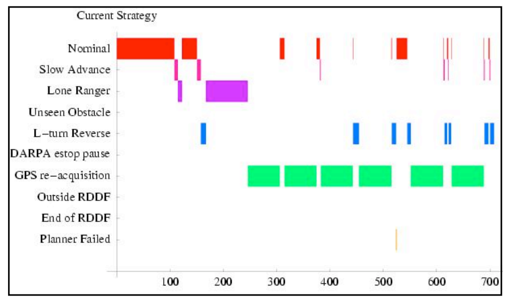


Vehicle width

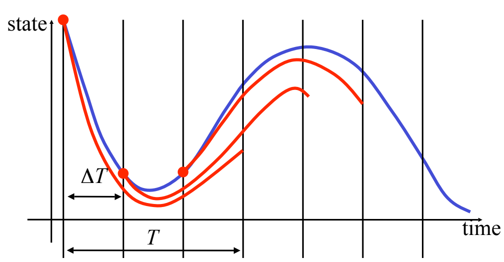
- Increase width of vehicle at high speed

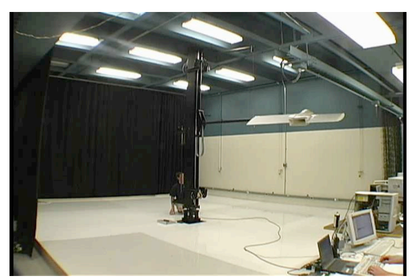
Supervisory control

- Trajectories can go through obstacles or outside "corridor"
  - Obstacles appear after plan is made
  - Optimizer fails to converge
- Supervisory control catches these cases and brings vehicle to stop
- Supervisory controller can then "force" vehicle along last planned path ("lone ranger")



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