CDS 270-2: Lecture 3-1
Real-Time Trajectory Generation

Richard M. Murray

10 April 2006

Goals:

- Introduce two degree of freedom design for motion control systems
- Describe how to use flatness for real-time motion planning using NTG
- Give examples of implementation on Caltech ducted fan, satellite formations

Reading:

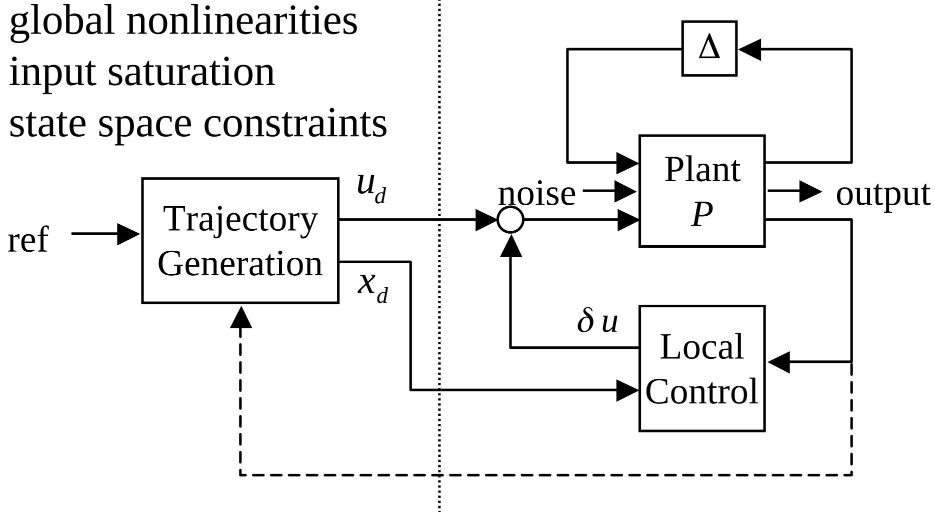
- "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems", M. B. Milam, K. Mushambi and R. M. Murray. Conference on Decision and Control, 2000.
- "Inversion Based Constrained Trajectory Optimization", N. Petit, M. B. Milam and R. M. Murray. IFAC Symposium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

Real-Time Trajectory Generation Using Flatness

Nonlinear design

- global nonlinearities
- input saturation
- state space constraints

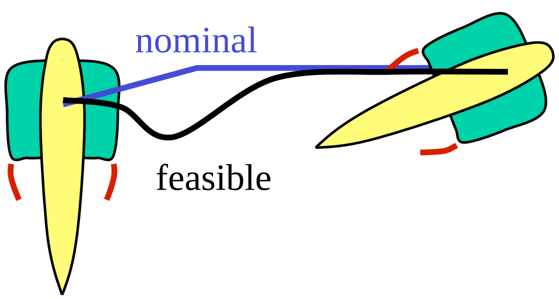
Linear design



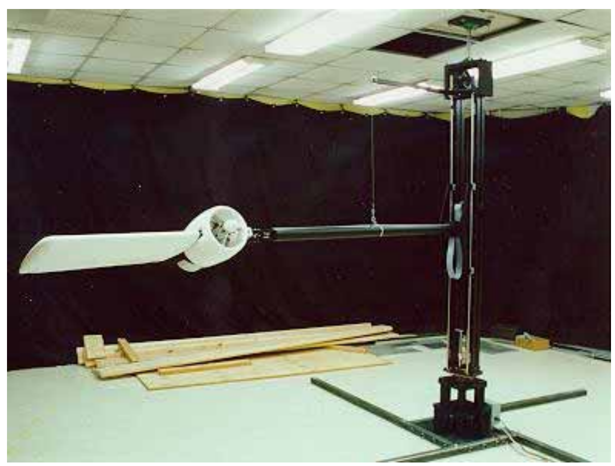
Approach: Two Degree of Freedom Design

- 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

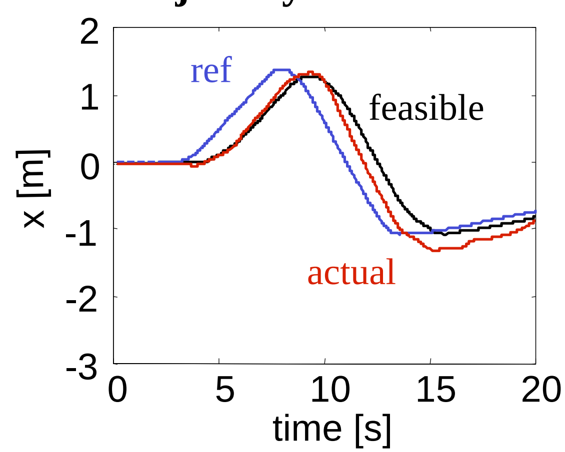
Rapid Transition from Hover to Forward Flight



Caltech Ducted Fan



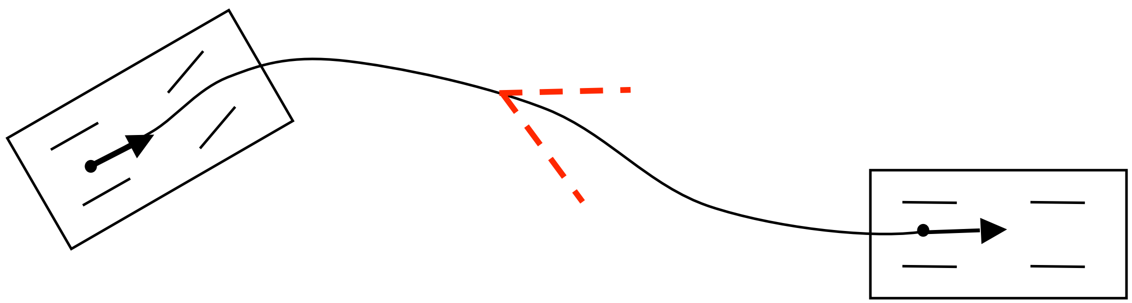
Real-Time Trajectory Generation



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

$$\dot x = f(x,u) \qquad g(x,u) \le 0$$

- Input constraints
- State constraints

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

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

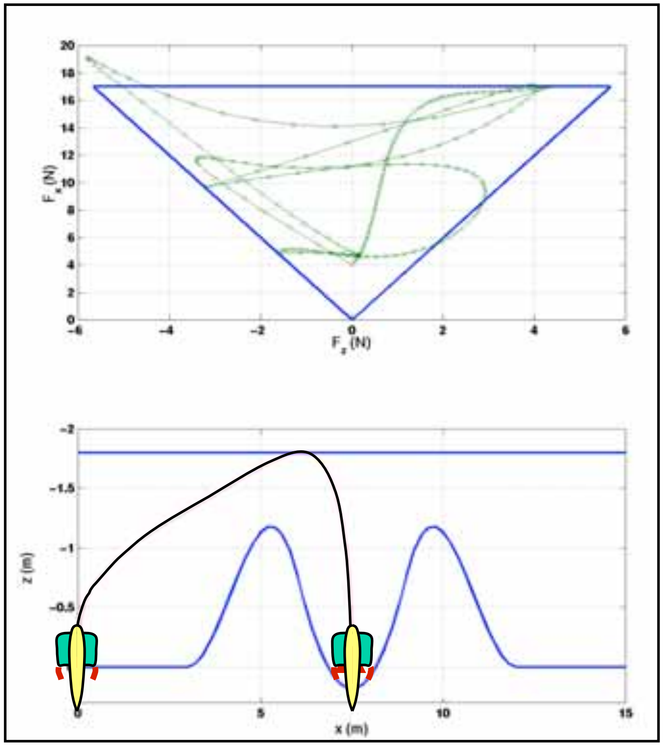
- Constraints hold at all times ⇒ potentially over-constrained optimization
- Numerically solve by discretizing time (collocation)

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 licens
- Warm start capability for receding horizon control



Solves general nonlinear optimization problem

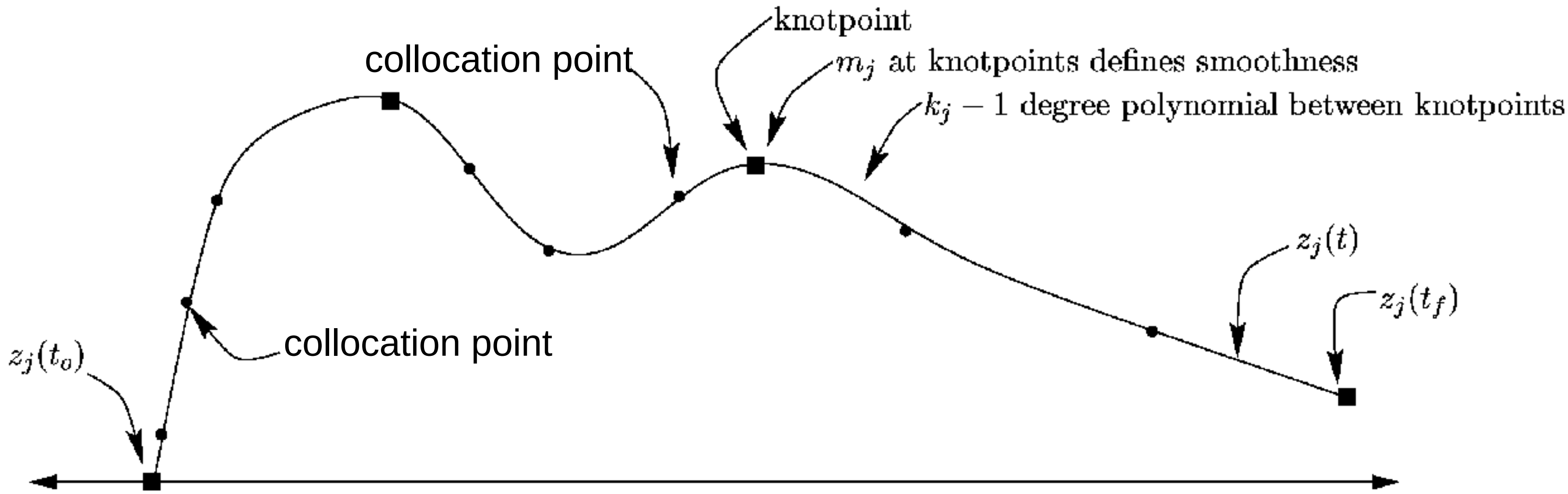
$$\min J = \int_{t_0}^{T} q(x,u)\,dt + V(x(T), u(T))$$

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

- 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

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

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

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

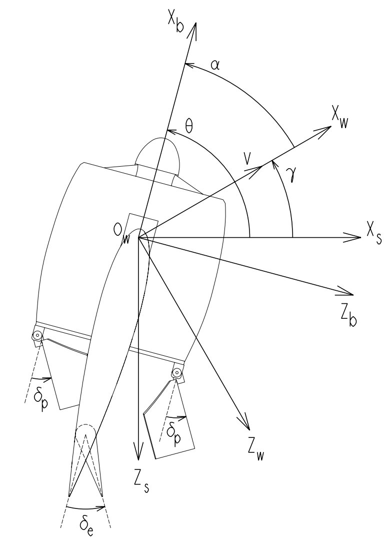
$$\alpha = \theta - \gamma, \qquad \text{angle of attack}$$

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

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

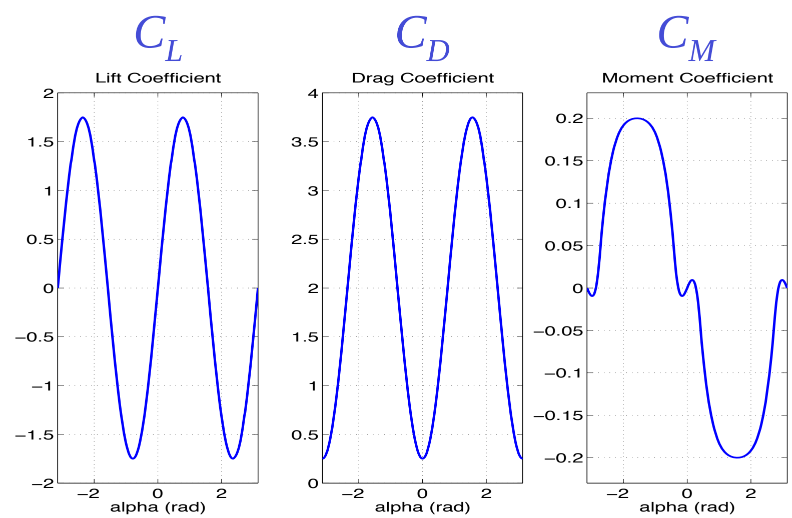
$$D = \frac{1}{2}\rho V^2 S C_D(\alpha)$$

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



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



Milam, Mushambi, M
2000 CDC

Implementation using NTG Software Library

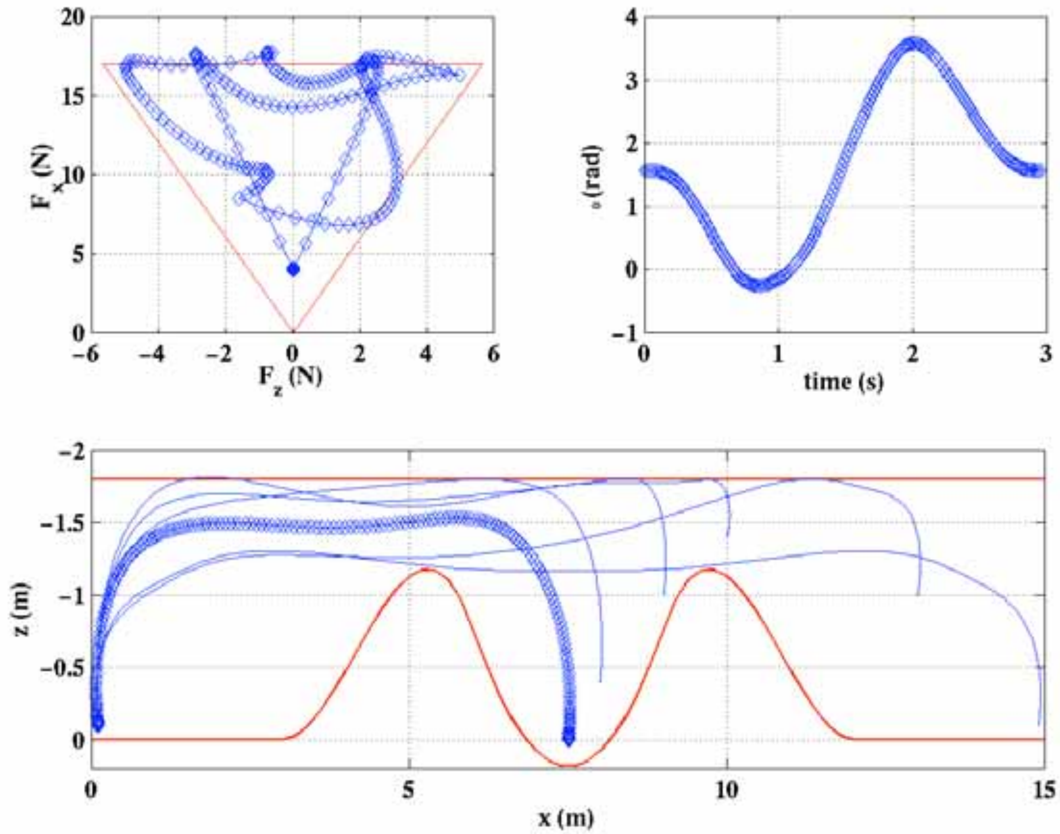
Features

- 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 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

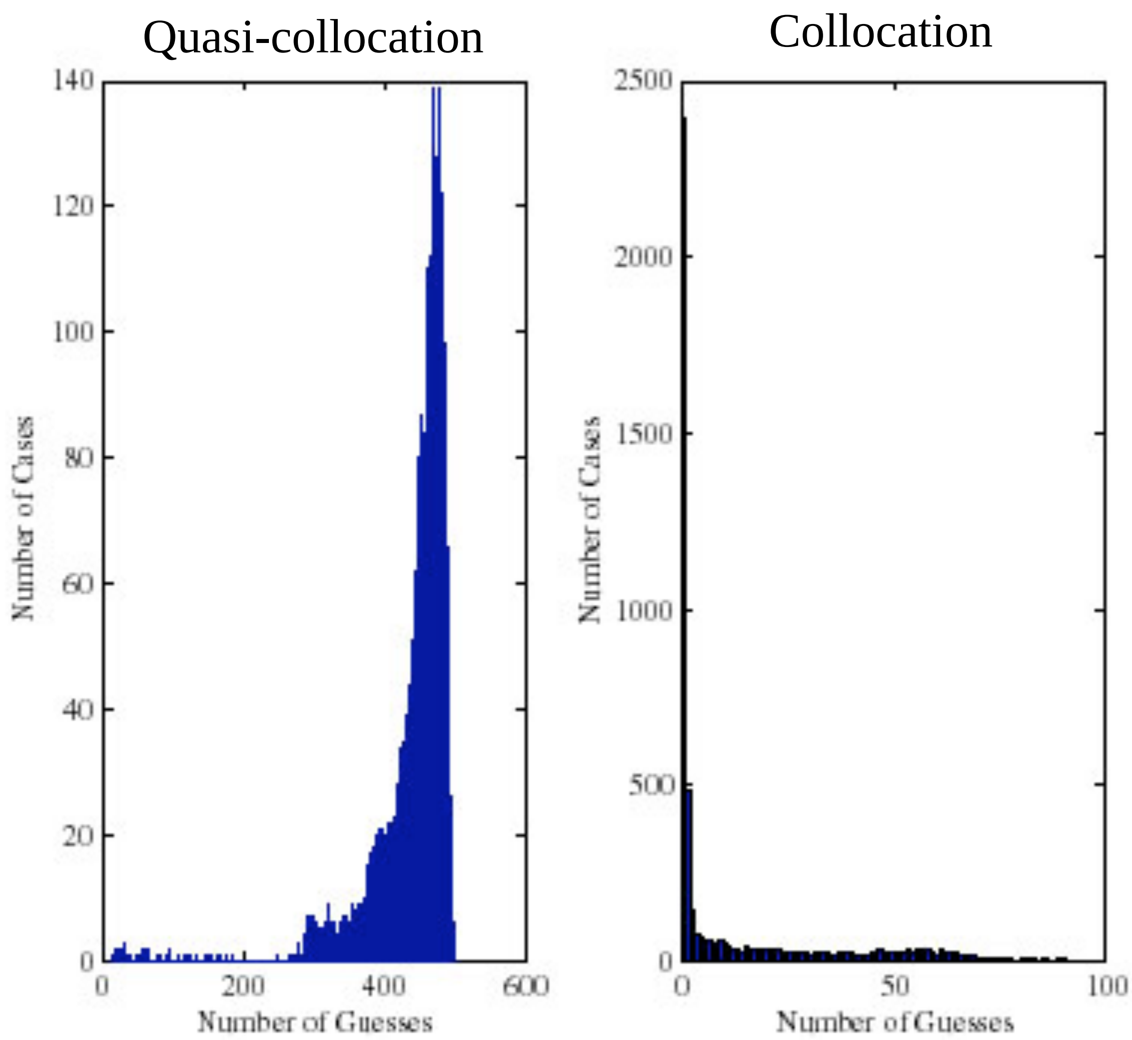
NTG Convergence Properties

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

$$\dot{x} = f(x,u) \quad \begin{cases} z = z(x,u,\dot{u},\ldots,u^{(q)}) \longrightarrow & x = x(z,\dot{z},\ldots,z^{(q)}) \\ & u = u(z,\dot{z},\ldots,z^{(q)}) \\ y = h(x,u) \longrightarrow & (x,u) = \Gamma(y,\dot{y},\ldots,y^{(q)}) \\ & 0 = \Phi(y,\dot{y},\ldots,y^{(p)}) \end{cases}$$

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

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

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

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

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

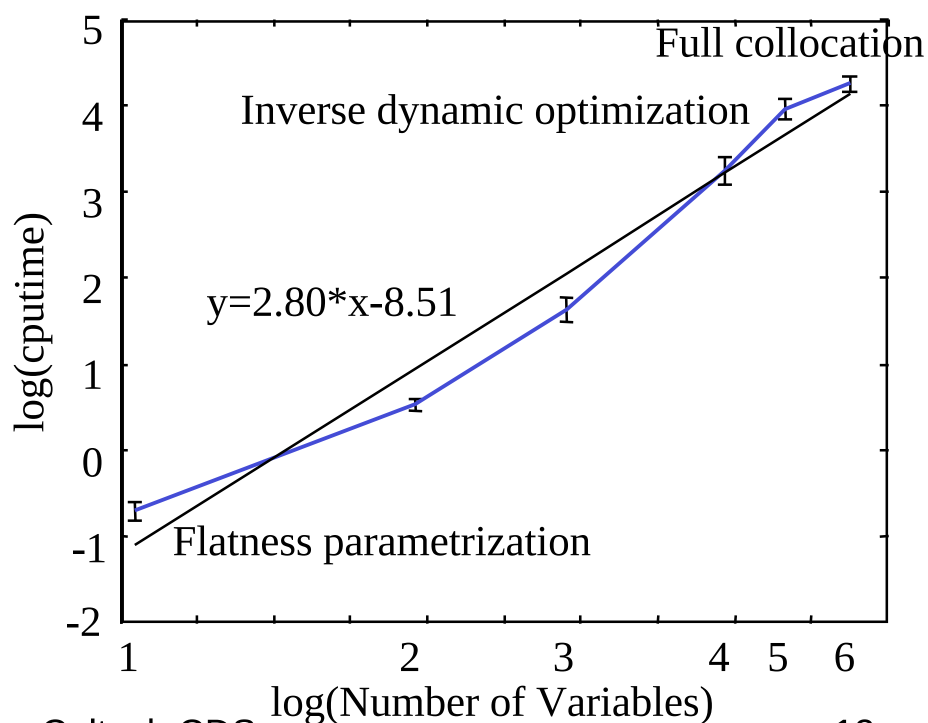
- x_1 is possible flat output
- Can choose other outputs to get systems with nonzero *defect*
- 200 runs per case, with random initial guess

Computation time related to defect through power law

- SQP scales cublicly ⇒ minimize the number of free variables

Dramatic speedup through reduction of differential constraints

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

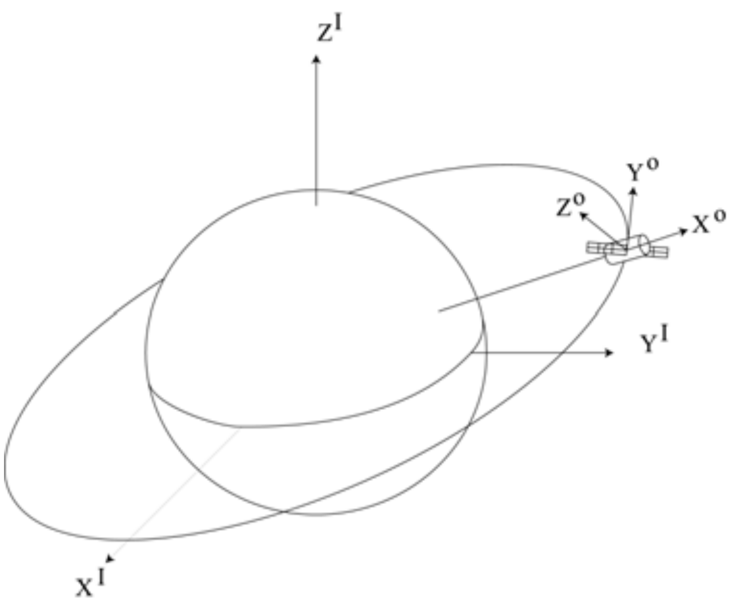


Example 2: Satellite Formation Control

Milam, Petit, Murray
AIAA GNC, 2001

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

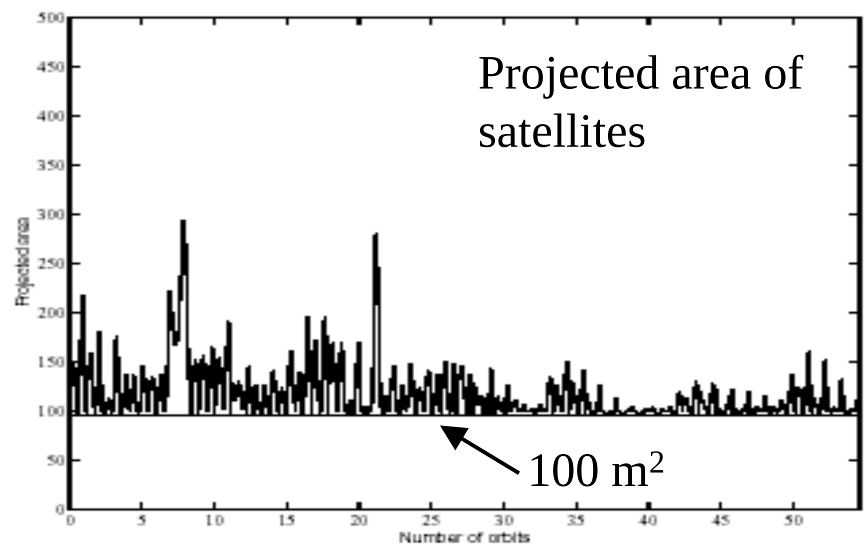
Station-keeping optimization

- 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: ΔV < 20 m/s/year

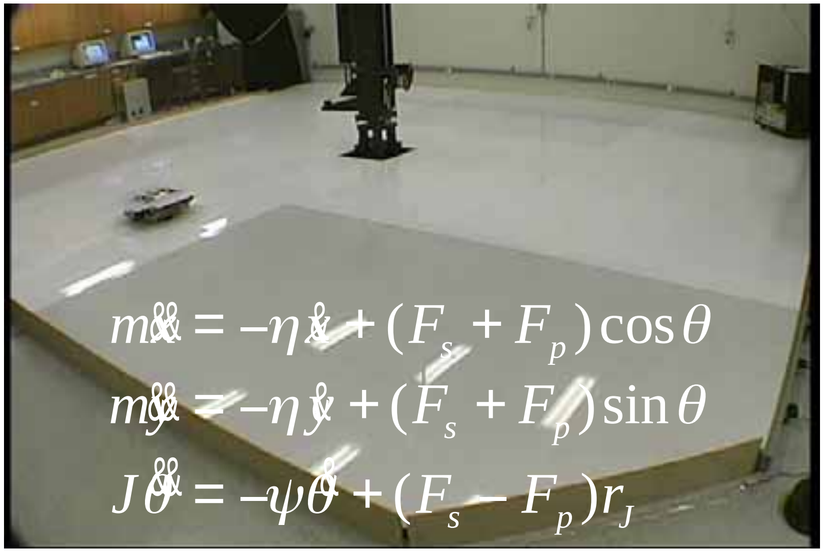
Results

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

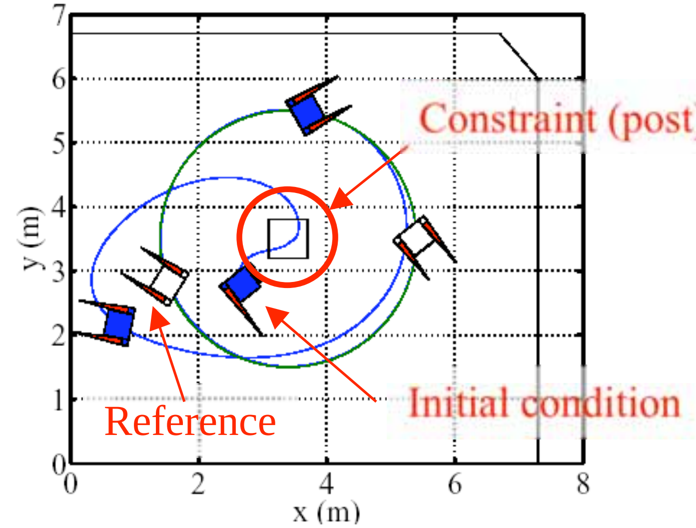
$i = 0$ deg	$S = 100\ \mathrm{m}^2$	$S = 200\ \mathrm{m}^2$
$d \le 500$ m	$\Delta V = 25.6$ m/s/year	$\Delta V = 47.8$ m/s/year
$i = 45$ deg	$S = 100\ \mathrm{m}^2$	$S = 200\ \mathrm{m}^2$
$d \le 500$ m	$\Delta V = 10.4$ m/s/year	$\Delta V = 17.0$ m/s/year
$i = 90$ deg	$S = 100\ \mathrm{m}^2$	$S = 200\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



Example 3: MVWT Control Design



LQR control law

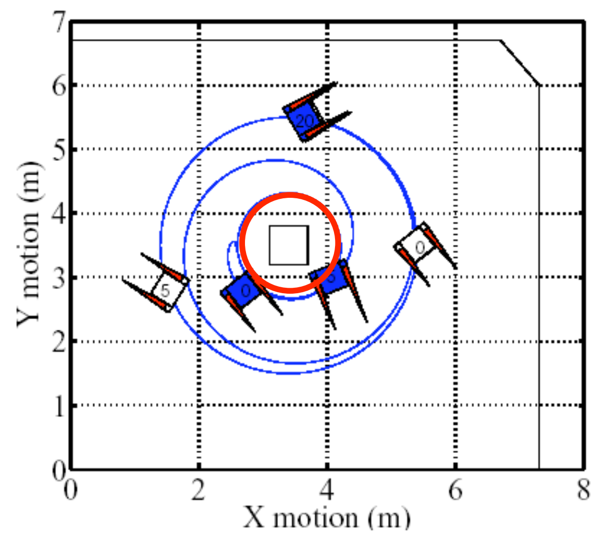


Control design technique

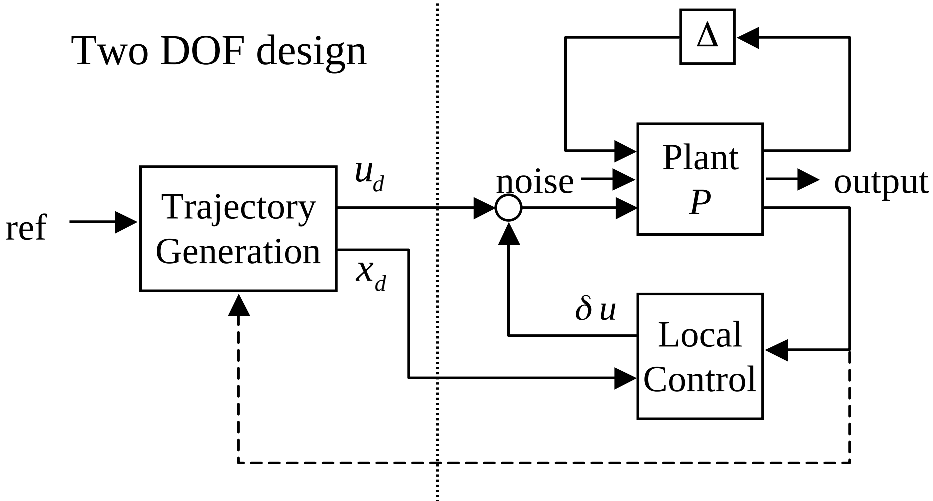
1. LQR design of state space controller K around reference velocity
2. Choose P, Q, R using Kalman's formula
3. Implement as a receding horizon control with input and state space constraints

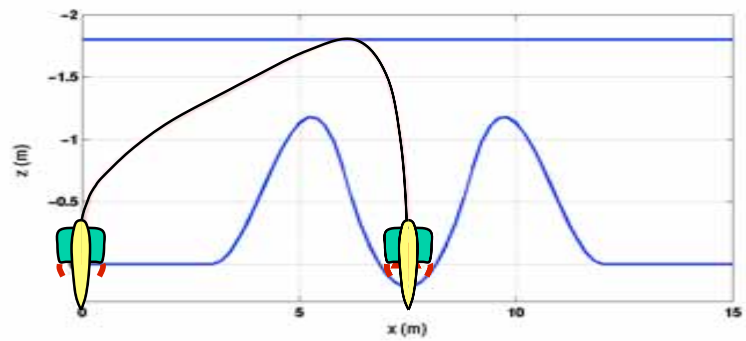
- RHC controller respects state space constraint

RHC control law



Summary: Real-Time Trajectory Generation





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

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

Flatness is a key property for efficient motion planning

- Allows conversion of dynamics into algebra ⇒ much faster algorithms

NTG software package implements required calculations

- Allows solution of general constrained optimization, w/ parameterized outputs
- Gives *approximate* results ⇒ need to use in feedback context (not open loop)

Growing collection of applications

- Caltech ducted fan, satellite formation control
- Underwater vehicles, wheeled mobile robots, RoboFlag, Alice, …

Homework and Project Ideas

Homework

- Download NTG and implement the point to point motion control problem for Alice or a RoboFlag vehicle.

Project ideas:

- For multi-vehicle applications, need to distribute the computation across multiple computers
- Use spread to implement a distributed trajectory generation capability