Goals:

• Review 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, satellites, etc
Control Architecture: Two DOF Design

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

Local design

Plant $P$

Δ

noise

output

Trajectory Generation

ref

$u_d$

$x_d$

δ $u$

Local Control

“RHC”

Optimal Control

LQR/PID

• 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)
Solve finite time optimization over $T$ seconds and implement first $\Delta T$ seconds

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

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

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

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
Additional Concepts for Receding Horizon Control

Discrete time systems
- "Clocked" transition between states
- New state is function of current state + inputs
- State is represented as a continuous variable

Optimization-based control for discrete time systems
- Same approach: parameterize inputs and/or states
- Cost functions: integrals become sums
  \[ J(x[·], u[·]) = \sum_{k=0}^{N-1} L(x[k], u[k]) + V(x[N], u[N]) \]
- Python: `solve_ocp()` and `create_mpc_iosystem()`

RHC versus layered
- Pure RHC/MPC: close the entire loop via RHC
- Layered: use RHC for trajectory, track via "inner loop" (local control)
  - Allows for slower RHC update with good performance
  - Use inner loop to handle disturbances/noise
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

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

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

$$\begin{align*}
x &= x(z, \dot{z}, \ldots, z^{(q)}) \\
u &= u(z, \dot{z}, \ldots, z^{(q)}) \\
z &= \sum \alpha_i \psi^i(t)
\end{align*}$$

$$\begin{align*}
\min J &= \int_{t_0}^{T} L(\alpha, t) \, dt + V(\alpha) \\
g(\alpha, t) &\leq 0
\end{align*}$$

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

\[ \min_{C \in \mathbb{R}^M} J(\tilde{z}(t_i)) \quad \text{subject to } \quad lb \leq c(\tilde{z}(t_i)) \leq ub \]

\[ B_{i,k_j} = \text{basis functions} \]
\[ C_i^j = \text{coefficients} \]
\[ z_i = \text{flat outputs} \]
Application: Caltech Ducted Fan

Flight Dynamics

\[ m \ddot{x} = -D \cos \gamma - L \sin \gamma + F_{X_b} \cos \theta + F_{Z_b} \sin \theta \]
\[ m \ddot{z} = D \sin \gamma - L \cos \gamma - m g_{eff} + F_{X_b} \sin \theta + F_{Z_b} \cos \theta \]
\[ J \ddot{\theta} = M_a - \frac{1}{r_s} I_p \Omega \dot{x} \cos \theta + M_T \]
\[ \alpha = \theta - \gamma, \quad \text{angle of attack} \]
\[ \gamma = \tan^{-1} \frac{-\dot{z}}{\dot{x}}, \quad \text{flight path angle} \]

Trajectory Generation

• 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

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

https://github.com/murrayrm/ntg
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
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**
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

\[ \dot{x} = f(x, u) \]
\[ z = z(x, u, \dot{u}, \ldots, u^{(q)}) \]
\[ y = h(x, u) \]
\[ x = x(z, \dot{z}, \ldots, z^{(q)}) \]
\[ u = u(z, \dot{z}, \ldots, z^{(q)}) \]
\[ (x, u) = \Gamma(y, \dot{y}, \ldots, y^{(q)}) \]
\[ 0 = \Phi(y, \dot{y}, \ldots, y^{(p)}) \]

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

\[ \begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= u \\
y_1 &= x_1 \\
y_2 &= x_2
\end{align*} \]
\[ \begin{align*}
x_1 &= y_1 \\
x_2 &= y_2 \\
u &= \dot{y}_2
\end{align*} \]
\[ \begin{align*}
\dot{x}(t_i) &= f(x(t_i), u(t_i)) \\
(x, u) &= \sum \alpha_i \psi^i(t)
\end{align*} \]

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

Each equation gives constraints at collocation points \( \Rightarrow \) highly constrained optimization
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 cubically ⇒ minimize the number of free variables

\[
\begin{align*}
\dot{x}_1 &= 5x_2 \\
\dot{x}_2 &= \sin x_1 + x_2^2 + 5x_3 \\
\dot{x}_3 &= -x_1x_2 + x_3 + 5x_4 \\
\dot{x}_4 &= x_1x_2x_3 + x_2x_3 + x_4 + 5x_5 \\
\dot{x}_5 &= -x_5 + u
\end{align*}
\]

Dramatic speedup through reduction of differential constraints
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