AA/EE/ME 548: Linear Multivariable Control

Lecture 14

5/14/2025

Announcements

- May 19th lecture: Mega OH with Oliver!
  - Homework and project!
- **Quick poll:**
  - Interest in a guest lecture by PhD student on trajectory optimization + new code package
  - Format of final report
- Neuro-dynamic programming by Dimiti P. Bertsekas and John N. Tsitsiklis (Free online PDF)
  - https://web.mit.edu/dimitrib/www/NDP.pdf
  - Also has nice DP textbooks (but not free)

Trajectory optimization (direct methods)

MIT Underactuated Robotics (Chapter 10)

Matthew Kelly Trajectory Optimization notes

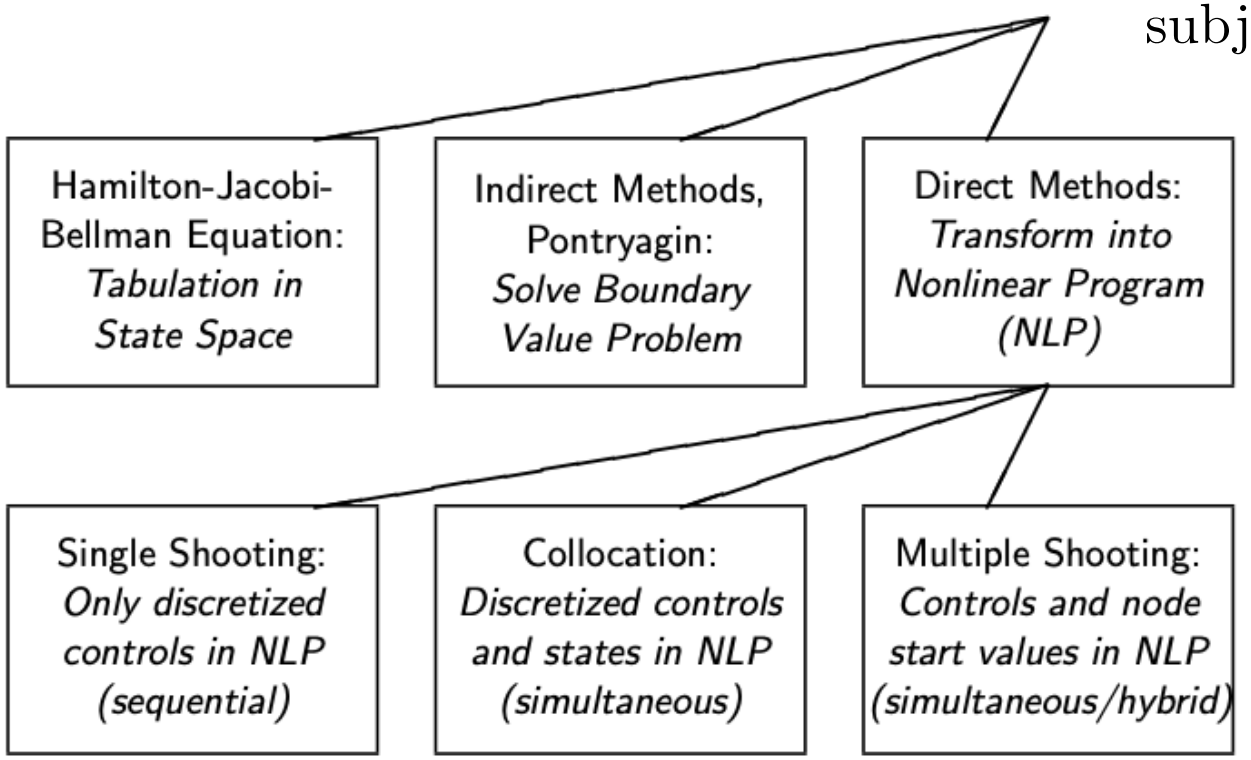
General trajectory optimization problem

$$\min_{\mathbf{u}} \quad \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T)$$

$$\text{subject to} \quad x_{t+1} = f(x_t, u_t, t), t = 0, ..., T-1$$

$$g(\mathbf{x}, \mathbf{u}) \leq 0$$

$$h(\mathbf{x}, \mathbf{u}) = 0$$



Moritz Diehl notes

The "art" of trajectory optimization (direct methods)

Turning

$$\begin{aligned} \min_{\mathbf{u}} \quad & \text{cost} \\ \text{subject to} \quad & \text{dynamics} \\ & \text{inequality constraints} \\ & \text{equality constraints} \end{aligned}$$

Into something we can tractably solve

1. *Solving* the optimization problem (convexification)
2. *Formulating* the optimization problem

Direct methods

- **Direct transcription:** "First discretize, then optimize"
  - Turn infinite dimensional problem (i.e., continuous time problem) into a finite dimensional problem (i.e., discrete time)
  - Solve finite dimension problem as NLP

- Leverage state-of-the-art methods for solving NLP
- Approximate solution to continuous time problem

Shooting method: Cannon example

Find: $\mathbf{z}(t) = [x(t), y(t), \dot{x}(t), \dot{y}(t)]$

Boundry Conditions:

$$x(t_0) = 0$$
$$y(t_0) = 0$$
$$x(t_f) = x_T$$
$$y(t_f) = y_T$$

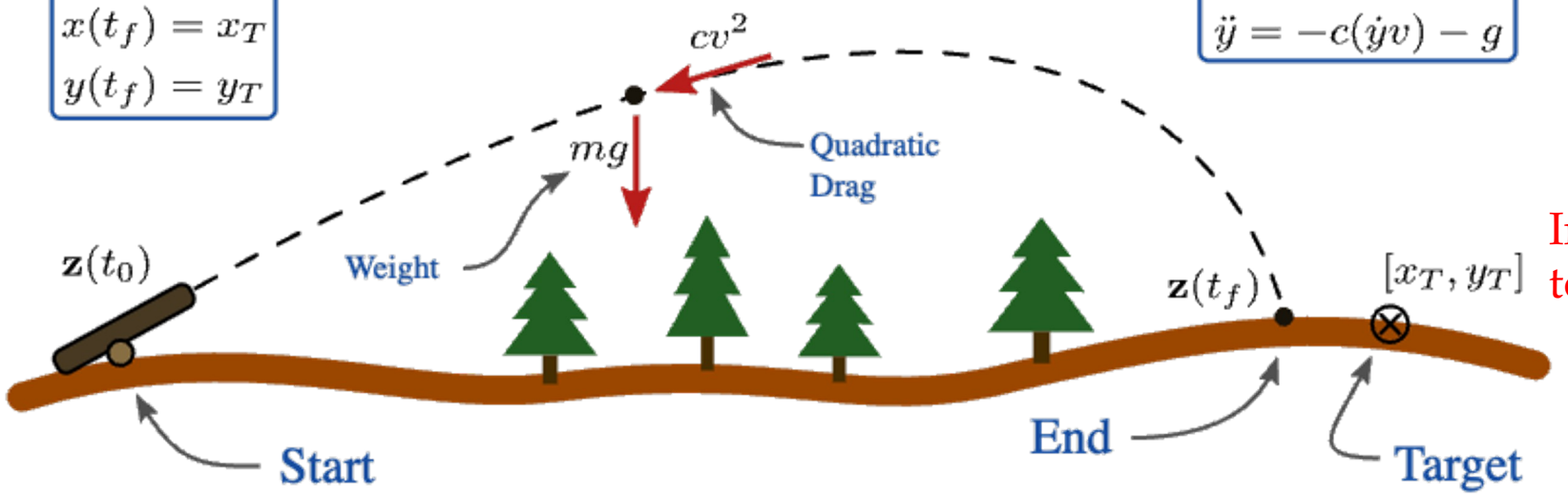
State constraints

Cost Function:

$$J = \dot{x}(t_0)^2 + \dot{y}(t_0)^2$$

Dynamics:

$$v = \sqrt{\dot{x}^2 + \dot{y}^2}$$
$$\ddot{x} = -c(\dot{x}v)$$
$$\ddot{y} = -c(\dot{y}v) - g$$

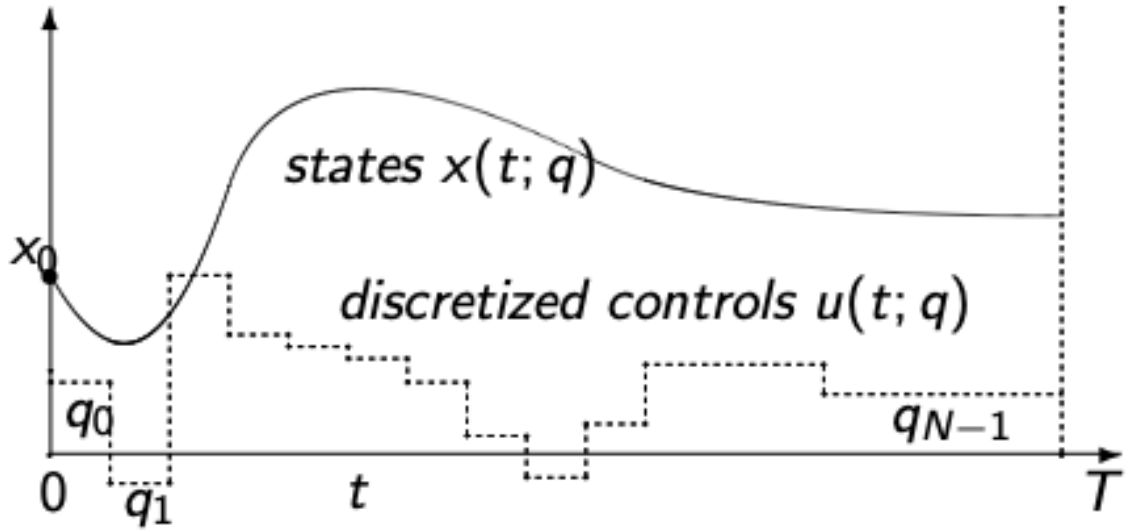


Integrate dynamics to get final state

Control: Initial angle and speed

Matthew Kelly Trajectory Optimization notes

Shooting method (general)



Pick controls, and your states will be determined by them.

Decision variables: Controls $u_0, u_1, \ldots, u_{T-1}$

Get states by integrating dynamics using $(x_0, u_0, u_1, \ldots, u_{T-1})$

Moritz Diehl notes

Shooting method (Discussion)

- Optimize only for control sequence. Need to *simulate dynamics* to get states
- Less variables
- Can be effective on short horizon, well-behaved dynamics
- Solution will be *dynamically feasible*
- Sensitive to initial guess
- Numerical conditioning for long horizons (especially for unstable or stiff systems)
  - Vanishing gradients
  - Earlier controls have less impact than later controls
- Difficult to enforce state constraints directly. Need to ”unroll” states first
- Inherently a serial operation

Shooting method

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T) \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t, t), t = 0, ..., T-1 \\ & g(\mathbf{x}, \mathbf{u}) \leq 0 \\ & h(\mathbf{x}, \mathbf{u}) = 0 \end{aligned}$$

$$x_{t+1} = f(...f(f(x_0, u_0), u_1), ..., u_t)$$
$$x_{t+1}(x_0, u_0, \ldots, u_t)$$

- State is a function of controls and initial state
- If dynamics are nonlinear, states later in horizon becomes more nonlinear
- Also makes state constraints more complex
- Results in a nonlinear optimization problem (plug into NLP solver)

Collocation/simultaneous method

- Optimize both state and controls simultaneously

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t, t), t = 0, ..., T-1 \\
& g(\mathbf{x}, \mathbf{u}) \leq 0 \\
& h(\mathbf{x}, \mathbf{u}) = 0
\end{aligned}
$$

Decision variables: Controls $(u_0, u_1, \ldots, u_{T-1})$ and states $(x_0, x_1, \ldots, x_T)$

Collocation/simultaneous method (Discussion)

- Optimize state **and** control sequence.
- More variables, more constraints, larger problem
- States and controls have "equal" contributions
- More direct constraint handling
- Less sensitive to initial guess
- Dynamics is treated as a constraint
  - Depending on the solver, it may not be met *exactly*

Some more details (collocation, CT)

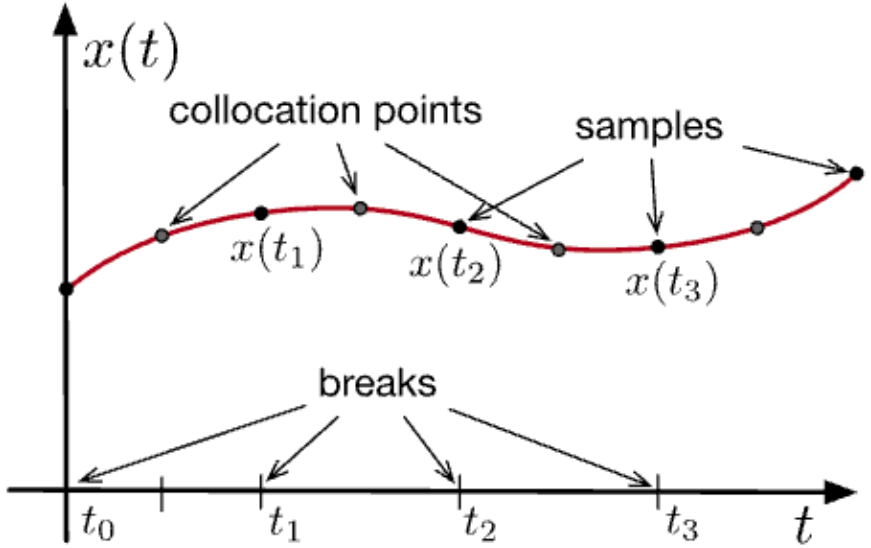


Figure 10.1 - Cubic spline parameters used in the direct collocation method.

$$\begin{aligned}
\min_{\mathbf{x}[\cdot],\mathbf{u}[\cdot]} \quad & \ell_f(\mathbf{x}[N]) + \sum_{n=0}^{N-1} h_n \ell(\mathbf{x}[n], \mathbf{u}[n]) \\
\text{subject to} \quad & \dot{\mathbf{x}}(t_{c,n}) = f(\mathbf{x}(t_{c,n}), \mathbf{u}(t_{c,n})), \qquad \forall n \in [0, N-1] \\
& \mathbf{x}[0] = \mathbf{x}_0 \\
& + \text{additional constraints.}
\end{aligned}$$

Discretize state and controls at times $t_0, t_1, \ldots, t_T$

Evaluate derivatives at midpoints

- Use spline to represent curve

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

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

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

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

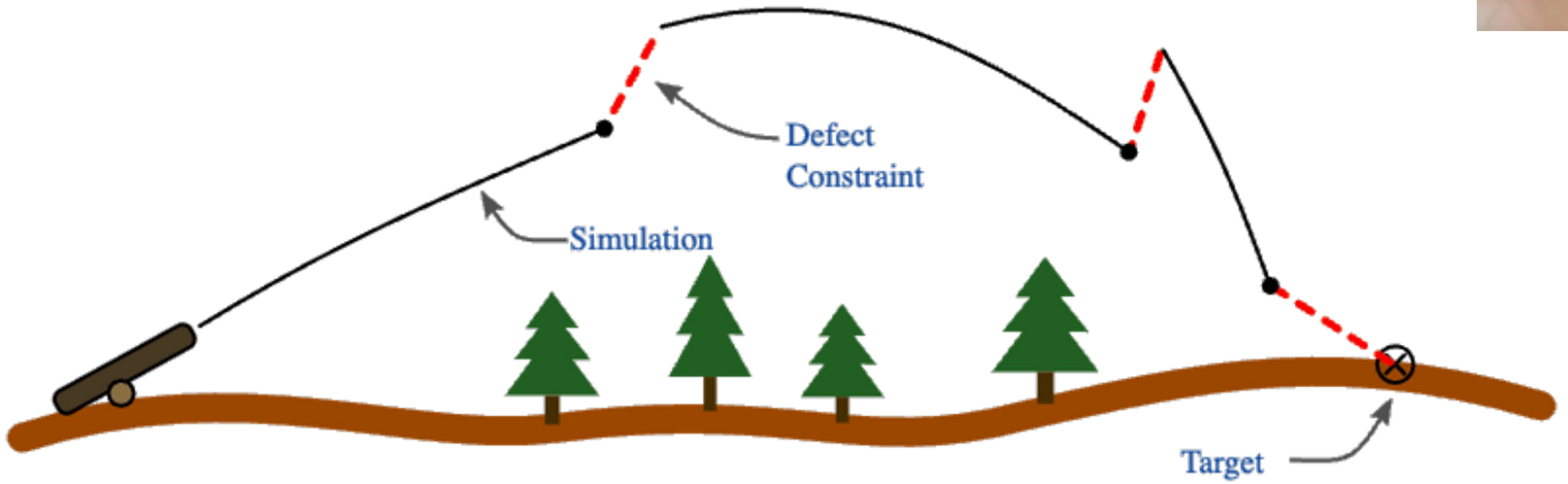
MIT Underactuated Robotics (Chapter 10)

Simultaneous method (DT)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t, t), t = 0, ..., T-1 \\
& g(\mathbf{x}, \mathbf{u}) \leq 0 \\
& h(\mathbf{x}, \mathbf{u}) = 0
\end{aligned}
$$

- No more of nested dynamics function: $x_{t+1} = f(...f(f(x_0, u_0), u_1), ..., u_t)$
  - Less complex cost and constraints
- Results in a (less) nonlinear optimization problem but more variables

Multiple shooting



- Combine shooting and collocation
- Have multiple shooting pieces

Pseudospectral method

- Parameterize state and control as *polynomials*
  - $x(t) = \sum_{i=0}^{N} a_i p_i(t)$, $u(t) = \sum_{i=0}^{N} b_i p_i(t)$, a_i, b_i are coefficients
- Need to choose polynomial basis
  - Standard (monomial) basis leads to poor conditioning, numerical instability
  - Other nice choices: Lagrange, Legendre, Chebyshev.
    - Better numerical properties
    - Nice properties (e.g., differentiation, integration, roots, collocation points)

Pseudospectral method

$$x(t) = \sum_{i=0}^{N} a_i p_i(t) \qquad u(t) = \sum_{i=0}^{N} b_i p_i(t)$$

$$\min_{\mathbf{a},\mathbf{b}} \sum_{t=0}^{T-1} J(x_t, u_t, t) + J_T(x_T)$$

$$\text{subject to } \dot{x}_t = f(x_t, u_t)$$

$$x(0) = x_0$$

$$\text{other constraints}$$

- Represents state and controls as smooth function
- Reduces # of nodes needed
- Can derive convergence properties
- Continuous-time problems

Differential flatness

- A system is **differentially flat** if all of its states and inputs can be expressed as algebraic functions of a set of outputs called **flat outputs** and a finite number of their derivatives.
  - $x(t) = \alpha\left(z(t), \dot{z}(t), \ldots, z^{(r)}(t)\right), u(t) = \beta\left(z(t), \dot{z}(t), \ldots, z^{(r)}(t), z^{(r+1)}(t)\right)$
- Solve for trajectory in flat output space
  - Easy, use polynomial or splines
  - Lower dimensional
- Not all systems are differentially flat! Determining DF is non-trivial
- Not straightforward to enforce constraints

Differential flatness (Example)

- Unicycle model $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}$

- Flat output: $z_1 = x,\ z_2 = y$

- Let $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3$

$$x(0) = x_0, y(0) = y_0, \theta(0) = \theta_0, v(0) = v_0$$

$$x(t_f) = x_f, y(t_f) = y_f, \theta(t_f) = \theta_f, v(t_f) = v_f$$

https://homes.cs.washington.edu/~bboots/files/InformationTheoreticMPC.pdf

Sampling-based: Model Predictive Path Integral

What if the cost and dynamics are complicated (e.g., neural network), or there's stochasticity?

Information Theoretic MPC for Model-Based Reinforcement Learning

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews,
James M. Rehg, Byron Boots, and Evangelos A. Theodorou

***Abstract*— We introduce an information theoretic model predictive control (MPC) algorithm capable of handling complex cost criteria and general nonlinear dynamics. The generality of the approach makes it possible to use multi-layer neural networks as dynamics models, which we incorporate into our MPC algorithm in order to solve model-based reinforcement learning tasks. We test the algorithm in simulation on a cart-pole swing up and quadrotor navigation task, as well as on actual hardware in an aggressive driving task. Empirical results demonstrate that the algorithm is capable of achieving a high level of performance and does so only utilizing data collected from the system.**

Fig. 1. Aggressive driving with MPPI and neural network dynamics.

Sampling-based: Model Predictive Path Integral

1. Start with nominal trajectory
2. Add noise to it to generate many trajectories
3. Evaluate cost of each trajectory
4. Compute weight for each trajectory
5. Compute weighted sum over controls to compute control

Idea: Take a noisy estimate of the gradient and move along that direction

Model Predictive Control

Middle ground between open-loop and closed-loop?

Open-loop control

Planning a trajectory/control sequence at time zero.

Plan becomes invalid very quickly

Over the entire, potentially very long, horizon

"Easier" to compute

Model Predictive Control (MPC)

Continually recompute an open-loop trajectory at each time step.

Solve an OCP problem over a shorter horizon

Update problem parameters as new information is received

"Open-loop control with feedback"

Closed-loop control

Very difficult to solve for stochastic, high dimensional, nonlinear problems.

Computed offline. Minimal computation online.

Invalid if there's model mismatch

Give optimal policy

Researchers program an autonomous vehicle to drift around obstacles using Nonlinear Model Predictive Control (NMPC)

Bram Geenen
19 Jul, 2022

FOLLOW

image: Toyota Research Institute

Obstacle appearing, tire friction changing, road conditions changing

https://www.wevolver.com/article/researchers-program-an-autonomous-vehicle-to-drift-around-obstacles-using-nonlinear-model-predictive-control-nmpc

PAMPC: Perception-Aware Model Predictive Control for Quadrotors

Unknown perception quality beforehand. Only known during deployment.

Falanga et al, IROS 2018
https://arxiv.org/pdf/1804.04811.pdf

Quadruped trajectory optimization

Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control

Donghyun Kim[1], Jared Di Carlo[2], Benjamin Katz[1], Gerardo Bledt[1], and Sangbae Kim[1]

***Abstract*— Dynamic legged locomotion is a challenging topic because of the lack of established control schemes which can handle aerial phases, short stance times, and high-speed leg swings. In this paper, we propose a controller combining whole-body control (WBC) and model predictive control (MPC). In our framework, MPC finds an optimal reaction force profile over a longer time horizon with a simple model, and WBC computes joint torque, position, and velocity commands based on the reaction forces computed from MPC. Unlike existing WBCs, which attempt to track commanded body trajectories, our controller is focused more on the reaction force command, which allows it to accomplish high speed dynamic locomotion with aerial phases. The newly devised WBC is integrated with MPC and tested on the Mini-Cheetah quadruped robot. To demonstrate the robustness and versatility, the controller is tested on six different gaits in a number of different environments, including outdoors and on a treadmill, reaching a top speed of 3.7 m/s.**

I. Introduction

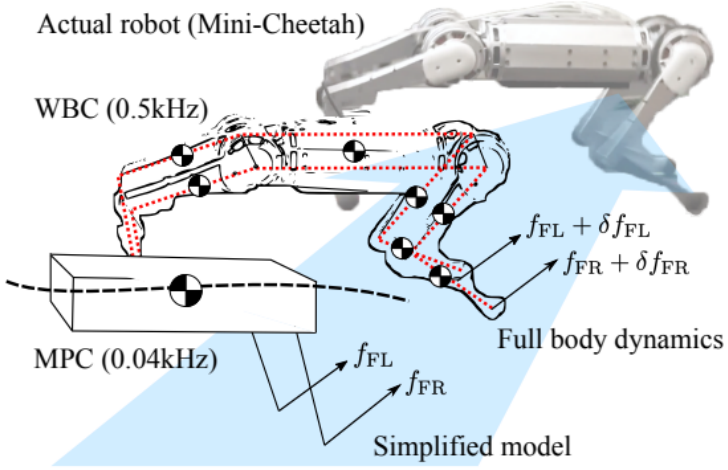


Fig. 1. **Control Architecture.** The proposed control architecture consists of two parts: Model predictive control and whole-body control. The reaction forces computed by MPC are modified by WBC to incorporate body stabilization and swing leg control. The final commands found in WBC are sent to the robot to perform dynamic locomotion.

Contact dynamics are hard!
Need to adapt to the terrain and rigid body dynamics.

https://arxiv.org/pdf/1909.06586.pdf

MPC Core Idea

Repeatedly solve an optimization problem over a finite horizon, execute first control, replan at next time step with updated state (and environment) information

General set-up

- Suppose we want to plan over a long time horizon (T very large, or potentially infinite)
  - E.g., chemical plant that needs to operate continuously
  - Not tractable to compute over such a long time horizon

- You can't compute the optimal policy (i.e., compute value function) offline
- Your model isn't perfect, and you don't trust your open-loop solution

Receding horizon planning / MPC

1. Plan a trajectory over a **finite horizon** (i.e., a sequence of control inputs)
2. Execute the **first control input** in the sequence
3. Go to step 1.

- Solve long horizon problem with a sequence of shorter horizon problem.
  - Just make T smaller…right?

- **Potential issue?**

Potential challenges

- Limited prediction horizon: MPC planning horizon is often not long enough to reach the end goal (if it exists).
- May lead to a situation where after a few steps, the MPC problem becomes infeasible
- Persistent feasibility?
  - Ensure that future problems will be feasible
- Closed-loop stability?
  - Ensure that in the future, the system will eventually converge to the goal state

Need some "foresight" to account for the future timesteps

- Let N be the shorter planning horizon
- Let t be the current time.
- We are planning from t to $t + N$

$$\min_{\boldsymbol{u}=(u_{t|t},\ldots,u_{t+N|t})} \sum_{k=0}^{N-1} g(x_{t+k|t}, u_{t+k|t}) + p\left(x_{t+N|t}\right)$$

$$\text{s.t.} \quad x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t})$$

$$x_{t+k|t} \in X, u_{t+k|t} \in U \quad \forall k$$

$$x(t) = x_{t|t}$$

$$x_{t+N|t} \in X_f$$

Cost-to-go

Something to ensure that the system will end up in a "reasonable" state

Terminal set constraint

Persistent feasibility theorem

- If X_f is a *control invariant set* for the system $x_{k+1} = f(x_k, u_k)$ then the MPC law is persistently feasible. i.e., the next planning step will have a feasible solution

$$\min_{\boldsymbol{u}=(u_{t|t},\ldots,u_{t+N|t})} \sum_{k=0}^{N-1} g(x_{t+k|t}, u_{t+k|t}) + p(x_{t+N|t})$$

$$\text{s.t.} \quad x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t})$$

$$x_{t+k|t} \in X, u_{t+k|t} \in U \quad \forall k$$

$$x(t) = x_{t|t}$$

$$x_{t+N|t} \in X_f \quad \text{Control invariant set}$$

Proof (sketch)

- Suppose at time t, you have a feasible solution to the current planning problem. $\boldsymbol{u_0^*} = (u_0, u_1, \ldots, u_{N-1})$ and $x_N \in X_f$

- At the next planning step (shift time step forward by one), note that $\boldsymbol{u_1} = (u_1, u_2, \ldots, u_{N-1}, \tilde{u})$ is a feasible solution to the next planning problem where $\tilde{x}_{N+1} = f(x_N, \tilde{u}) \in X_f$ ($\tilde{u}$ is a feasible control corresponding to the control invariant set definition)

- We have just found a feasible solution to the next planning problem. So we will do at least as good, if not better at the next time step.

Practical considerations

- The terminal set X_f is introduced artificially for the sole purpose of leading to a sufficient condition for persistent feasibility
- Want it to be large so it does not compromise on performance
- How do we pick X_f?
- These sets can be computed by using the MPT toolbox https://www.mpt3.org/

MPC stability theorem

Assume:

1. X, U contains the origin in their interior
2. $x = 0, u = 0$ is an equilibrium $f(0,0) = 0$
3. $\beta^- \|x\| \leq g(x,u) \leq \beta^+ \|x\| \quad \forall x \in X, u \in U$
4. $X_f \subseteq X$ is control invariant
5. The terminal cost p is a local control Lyapunov function satisfying
   - $p(x) \leq \beta \|x\|$
   - $\exists \mu \quad p(x) - p(f(x, \mu(x)) \geq g(x, \mu(x)). \quad \forall x \in X_f$

Then the origin of the closed-loop system under MPC control is exponentially stable with region of attraction X_0 (N-step BRS of X_f)

Proof (sketch)

- Show that the cost is Lyapunov stable for the closed-loop system
  - i.e., the cost is shrinking over time
- Suppose you have an optimal trajectory $\boldsymbol{u_0^*} = (u_0, u_1, \ldots, u_{N-1})$
- At the next time step $\boldsymbol{u_1} = (u_1, u_2, \ldots, u_{N-1}, \tilde{u})$ ($\tilde{u}$ keep $x_{N+1} \in X_f$)
- But $\boldsymbol{u_1}$ is not necessarily optimal

Rough proof continued

$$J_1^*(x_1) \leq \sum_{k=1}^{N} g(x_k, u_k) + p(x_{N+1})$$

$$= \sum_{k=1}^{N-1} g(x_k, u_k) + g(x_N, \tilde{u}) + p(x_{N+1})$$

$$= \sum_{k=1}^{N-1} g(x_k, u_k) + g(x_N, \tilde{u}) + p(x_{N+1}) + p(x_N) - p(x_N) + g(x_0, u_0) - g(x_0, u_0)$$

$$J_1^*(x_1) \leq J_0^*(x_0) + g(x_N, \tilde{u}) + p(x_{N+1}) - p(x_N) - g(x_0, u_0)$$

$$J_0^*(x_0) - J_1^*(x_1) \geq g(x_0, u_0) \geq \beta^- \|x_0\|$$

MPC stability theorem takeaway

- If we choose the cost-to-go to be a control Lyapunov function, then we can ensure stability (with some assumptions about the problem structure)

$$\min_{\boldsymbol{u}=(u_{t|t},\ldots,u_{t+N|t})} \sum_{k=0}^{N-1} g(x_{t+k|t}, u_{t+k|t}) + p(x_{t+N|t})$$

$$\text{s.t.} \quad x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t})$$

$$x_{t+k|t} \in X, u_{t+k|t} \in U \quad \forall k$$

$$x(t) = x_{t|t}$$

$$x_{t+N|t} \in X_f \text{ Control invariant set}$$

Control Lyapunov function

What are reasonable choices of X_f and p?

Assuming linear dynamics & quadratic cost

Case 1: A is asymptotically stable

- $u = 0$ is a feasible control
- X_f: maximally positive invariant set for $x_{t+1} = Ax_t$
- p: Solution to the Lyapunov equation

$$-P + Q + A^T P A = 0$$

"Once you reach the end, just do nothing and you will asymptotically converge".

What are reasonable choices of X_f and p?

Assuming linear dynamics & quadratic cost

Case 2: A is open-loop unstable

- Let K_∞ be the optimal gain for infinite horizon LQR
- X_f: maximally positive invariant set for $x_{t+1} = (A + BK_\infty)x_t$
- p: Solution to the discrete Riccati equation

$$-P + Q + A^T PA - (A^T PB)(R + B^T PB)^{-1}(B^T PA) = 0$$

"Once you reach the end, apply the infinite horizon LQR controller".

Some comments

- Presented cases are suboptimal choices
- Need to make sure the control constraints are satisfied (increasing R)
- At present there is no other technique than MPC to design controllers for general large linear multivariable systems with input and output constraints with a stability guarantee
- Design approach (for squared 2-norm cost):
  - Choose horizon length N and the control invariant target set X_f
  - Control invariant target set X_f should be as large as possible for performance
  - Choose the parameters Q and R freely to affect the control performance
  - Adjust p as per the stability theorem
  - Useful toolbox (MATLAB): https://www.mpt3.org/
- In practice, sometimes choosing a good terminal cost is enough (i.e., don't need to enforce a terminal control invariant condition), though you may be sacrificing guarantees

So far…

- Assumed perfect dynamics model
- Assumed no stochasticity
- What about moving obstacles? (Prediction models, human-robot interactions)
- Theorems hold in general but challenging for non-LQR-like structure (i.e., not linear & not quadratic)
- But provide conceptual understanding to what heuristics we would like to design for nonlinear systems
- There is a whole research area on *robust* MPC and *tube* MPC (beyond the scope of this course)

Code demo

Planning a trajectory to avoid moving asteroids