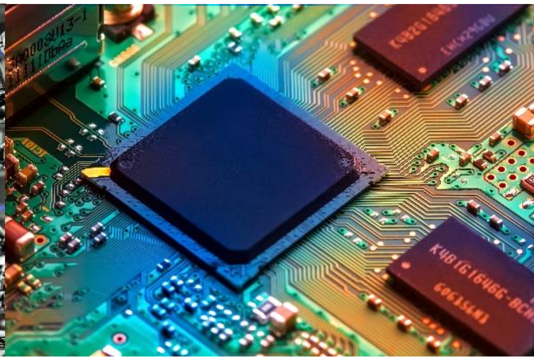
DIRECT DESIGN OPTIMIZATION FOR TRAJECTORY PERFORMANCE

Mathew Halm
University of Illinois at Urbana-Champaign



Design Minimality as Optimization

$$\underset{\ll design\ params \gg}{\text{minimize}} \quad \ll resource\ usage \gg$$

$$\text{subject to} \quad \ll design\ feasibility \gg$$

$$\ll laws\ of\ physics \gg$$

Direct Numerical Design Optimization

- Computers are great at solving optimization problems!
- Meeting input requirements of numerical solvers can be intractable
  - Requires real/discrete valued cost functions, decision variables, and constraints.
  - **Robot functionality and task structure would need to be specified implicitly or explicitly as constraints on the design parameters**

Minimality for Task Performance

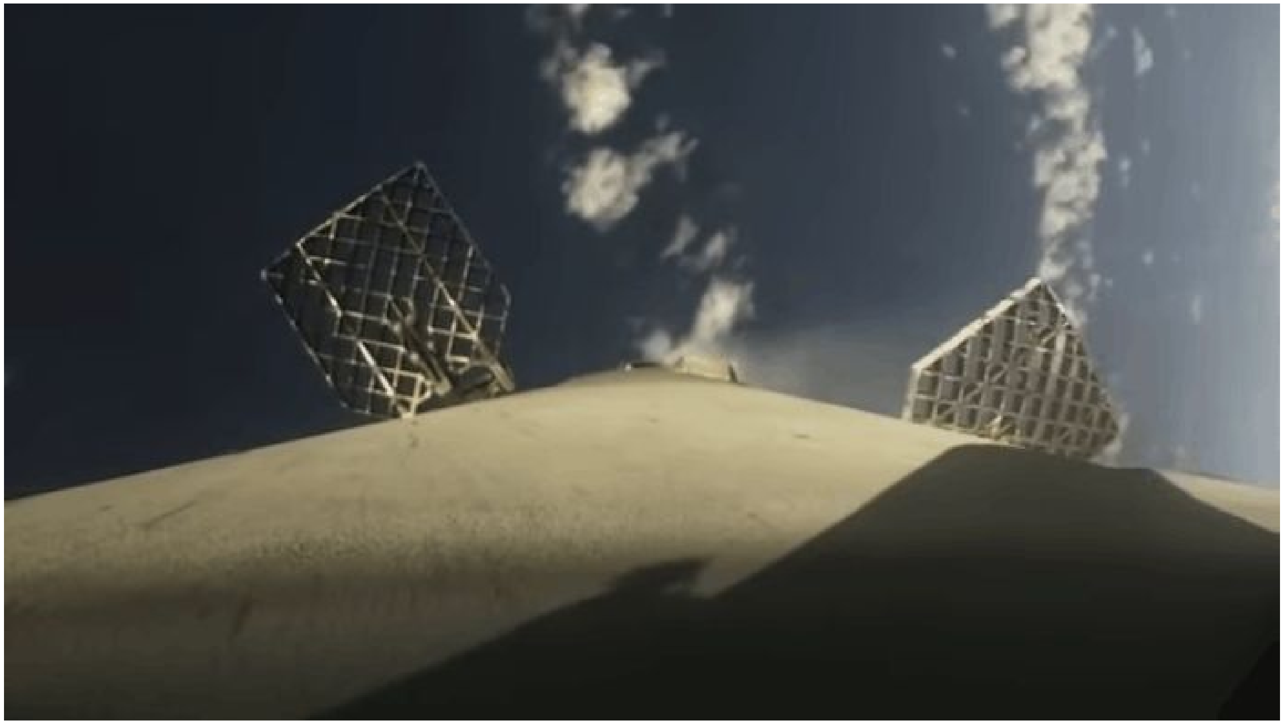
- Consider subset of design optimization primarily concerned with task kinematics and dynamics

$$\underset{\ll \textit{design params} \gg}{\text{minimize}} \quad \ll \textit{task cost} + \textit{design cost} \gg$$

$$\text{subject to} \quad \ll \boldsymbol{task} \ \textit{feasibility} \gg$$

$$\ll \textit{laws of physics} \gg$$

Example: Falcon 9



$$\underset{\ll \textit{design params} \gg}{\text{minimize}} \quad \ll \textit{price per launch} \gg$$

$$\text{subject to} \quad \ll \textit{orbit reachability} \gg$$

$$\ll \textit{rigid body dynamics, aerodynamics, etc.} \gg$$

Image: SpaceX

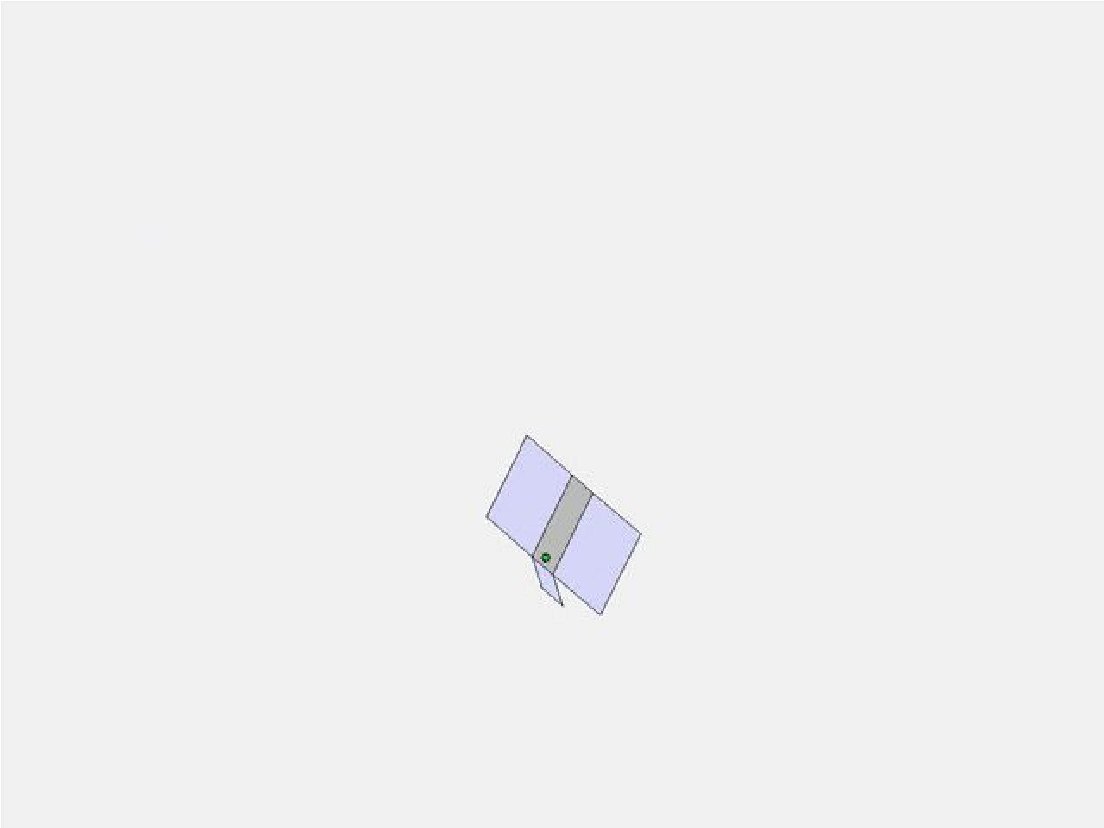
Trajectory Optimization

- Idea: Create process for design optimization by leveraging computational tools used for solving similar problems
- *Trajectory Optimization* problems search for input ***u*** that optimizes some *cost-to-go* ***J*** and satisfies constraints ***d*** for a dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$

$$\begin{aligned}
\underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \quad & \int_{t_0}^{t_f} l(\boldsymbol{x}(t), \boldsymbol{u}(t))dt + J_f(\boldsymbol{x}(t_f)) && //\textit{resource usage / cost} \\
\text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \forall t \in [t_0, t_f] && //\textit{laws of physics} \\
& \boldsymbol{d}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq \boldsymbol{0}, \forall t \in [t_0, t_f] && //\textit{trajectory feasibility}
\end{aligned}$$

Trajectory Optimization Examples

- Multi-contact traversal of uneven terrain (Dai, 2016)
- Dexterous hand manipulation (Mordatch, 2012)
- Perching of flapping-wing systems (Halm, 2017)



Direct Trajectory Optimization

- Strategy: solve direct approximation of continuous time problem by optimizing over a sampling of the inputs and state
- *Direct Transcription* explicitly enumerates samples of state and input trajectory as decision variables

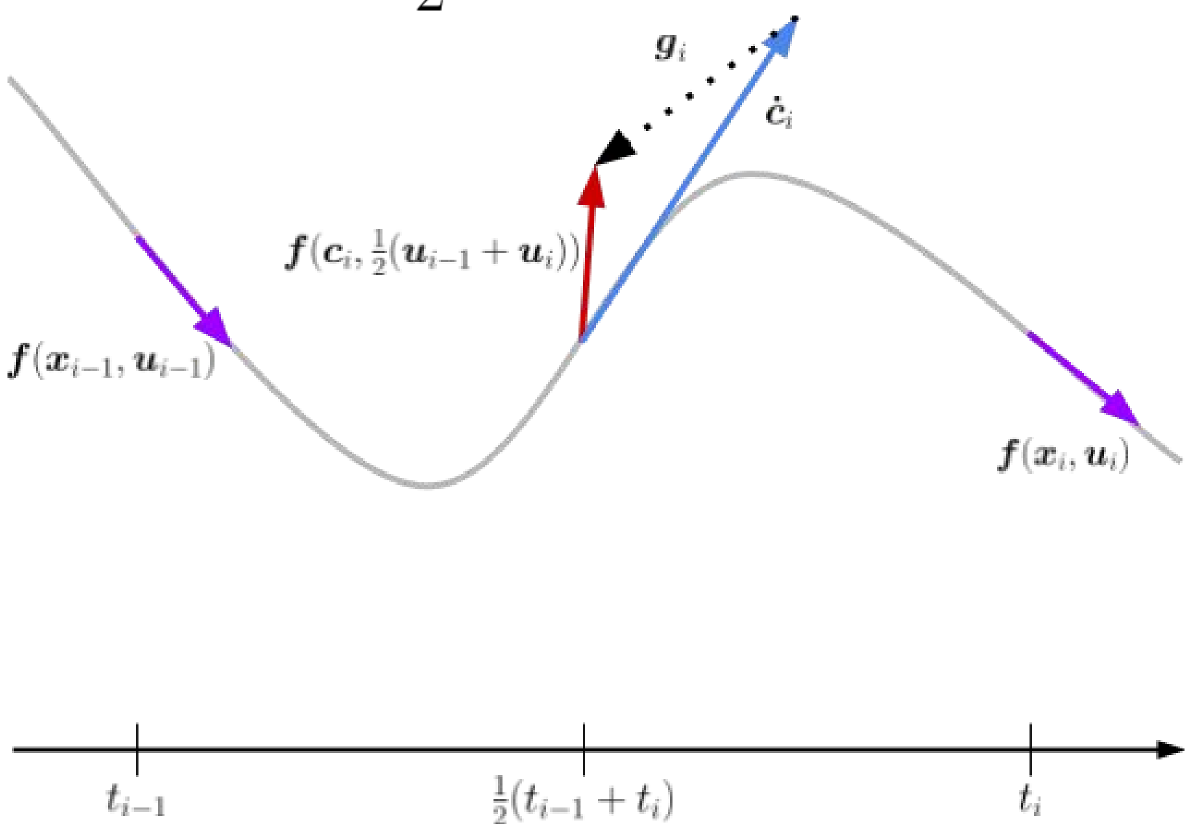
$$\begin{aligned}
\underset{(\boldsymbol{x}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{x}_N,\boldsymbol{u}_N)}{\text{minimize}} \quad & \sum_{i=1}^{N} \frac{\Delta t_i}{2}\left(l(\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1}) + l(\boldsymbol{x}_i,\boldsymbol{u}_i)\right) + J_f(\boldsymbol{x}_N) && \textit{//resource usage / cost} \\
\text{subject to} \quad & \Delta t_i > 0, \forall i \in 1,\ldots,N \\
& \boldsymbol{g}(\Delta t_i, \boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}, \boldsymbol{x}_i, \boldsymbol{u}_i) = \boldsymbol{0}, \forall i \in 1,\ldots,N && \textit{//laws of physics} \\
& \boldsymbol{d}(\boldsymbol{x}_i,\boldsymbol{u}_i) \geq \boldsymbol{0}, \forall i \in 0,\ldots,N && \textit{//trajectory feasibility}
\end{aligned}$$

Direct Trajectory Optimization

$$\boldsymbol{g}_{euler} = \boldsymbol{x}_i - (\boldsymbol{x}_{i-1} + \Delta t_i \boldsymbol{f}(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})) = \boldsymbol{0}$$

$$\boldsymbol{g}_{back} = \boldsymbol{x}_i - (\boldsymbol{x}_{i-1} + \Delta t_i \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)) = \boldsymbol{0}$$

$$\boldsymbol{g}_{hermite} = \boldsymbol{f}(\boldsymbol{c}_i, \frac{1}{2}(\boldsymbol{u}_{i-1} + \boldsymbol{u}_i)) - \dot{\boldsymbol{c}}_i = \boldsymbol{0}$$



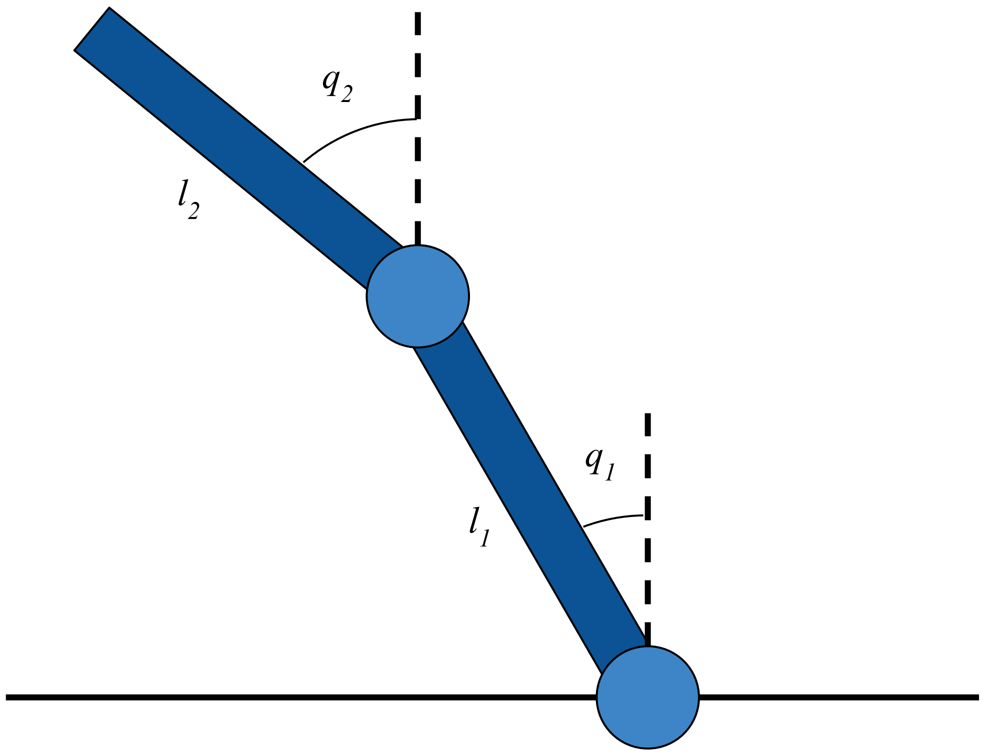
Direct Design Optimization

- Add design parameters ***α*** explicitly as decision variable to optimization problem
- Handle variations in equations of motion ***f***, trajectory costs *l* and J_f, and trajectory constraints ***d*** by adding dependence on ***α***
- Add additional, independent design cost J_d, and augment ***d*** with any necessary design feasibility constraints.
- Solution of resulting problem should provide optimal task trajectory and optimal system parameters.

$$\begin{aligned}
\underset{(\boldsymbol{x}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{x}_N,\boldsymbol{u}_N),\boldsymbol{\alpha}}{\text{minimize}} \quad & \sum_{i=1}^{N} \frac{\Delta t_i}{2}\left(l(\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1},\boldsymbol{\alpha}) + l(\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{\alpha})\right) + J_f(\boldsymbol{x}_N,\boldsymbol{\alpha}) + J_d(\boldsymbol{\alpha}) \\
\text{subject to} \quad & \Delta t_i > 0, \forall i \in 1,\ldots,N \\
& \boldsymbol{g}(\Delta t_i, \boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}, \boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\alpha}) = \boldsymbol{0}, \forall i \in 1,\ldots,N \\
& \boldsymbol{d}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\alpha}) \geq \boldsymbol{0}, \forall i \in 0,\ldots,N
\end{aligned}$$

Example: 2-Link Planar Arm

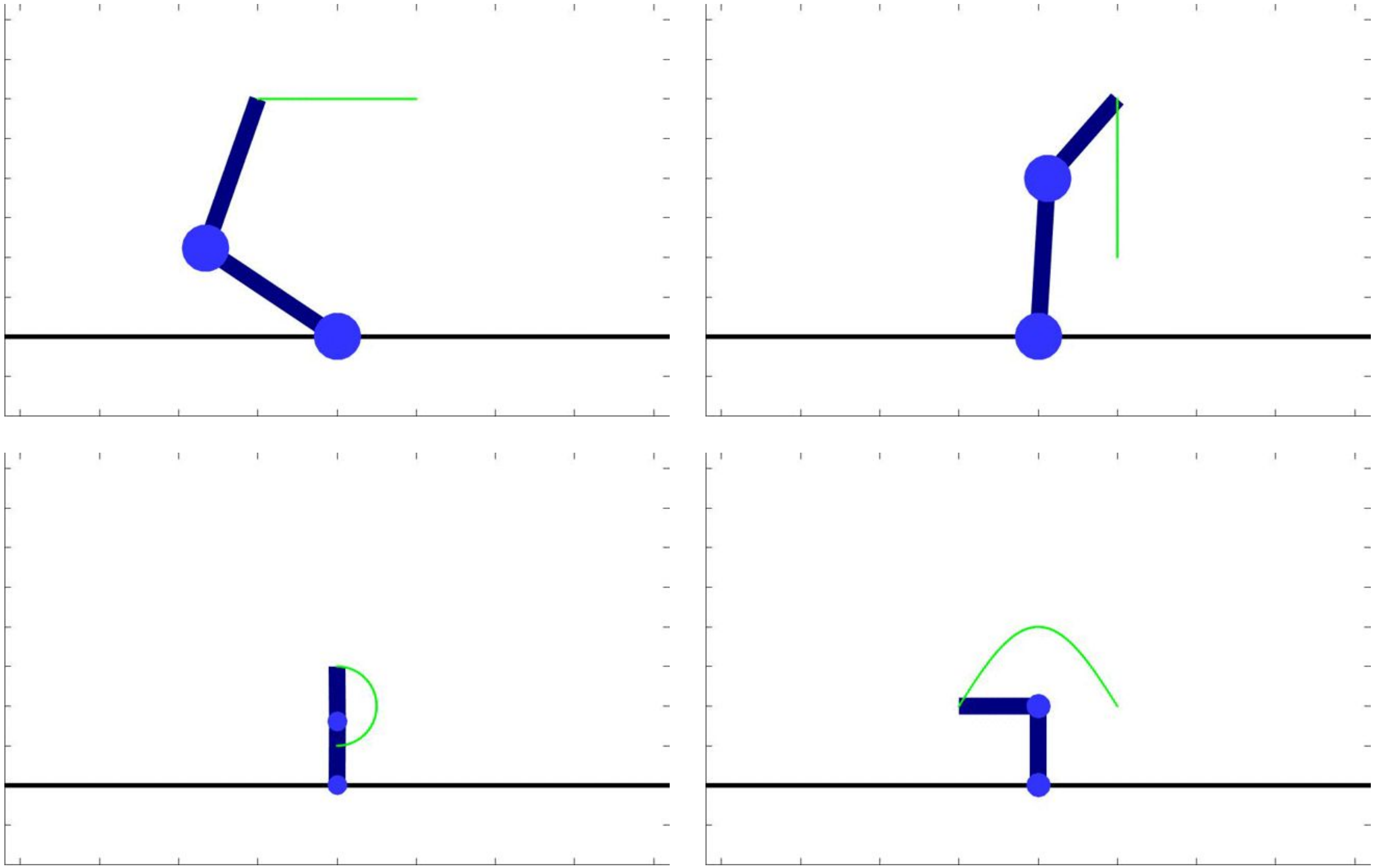
$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}$$



$$\boldsymbol{\alpha} = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix}$$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}(\boldsymbol{q}, \boldsymbol{\alpha})^{-1}(\boldsymbol{B}\boldsymbol{u} - \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\alpha})) \end{bmatrix}$$

2-Link Planar Arm Results



Discussion and Future Work

- Base formulation is incredibly flexible
  - Arbitrary nonlinear dynamics allow for application in a very wide range of scenarios, from UAVs to legged locomotion
  - Arbitrary constraints on design parameters allow for discrete decision variables (such as number of linkages) to be handled with Mixed Integer Programming
- Exact solutions to arbitrary problems is NP-Hard
  - High flexibility limits computational efficiency by ignoring structure
  - Only very small numbers of discrete variables can be handles for nonconvex problems
- Consider permuted formulations for large classes of robots, e.g. rigid body systems with frictional contact

Q&A