Efficient Methods for Control of Dynamical Systems

Interior Point Differential Dynamic Programming

Andrei Pavlov

The University of Melbourne

22 October 2020

PhD completion seminar

Structure of the seminar

First 5 minutes of the talk:

- General problem and the previous contributions

The rest of the talk:

- Interior Point Differential Dynamic programming

Motivation: Control problem

For a given system subject to constraints find a control law, that

1. guarantees performance,
2. satisfies constraints,
3. can be computed in real-time.

Modern approach: Model Predictive Control

Advantages of MPC

- Performance is related to the objective function,
- Explicit constraints handling,
- Closed-loop control (by iterative resolving).

Disadvantages of MPC

- Computationally expensive $\rightarrow$ need efficient optimisation algorithms
- Stability and recursive feasibility analysis is required.

The MPC problem

Setup:

- Dynamical system $x^+ = f(x,u)$
- State and input constraints $c(x,u) \leq 0$
- Stage and terminal costs $q(x,u)$ and $p(x)$

where $f(x,u)$, $c(x,u)$, $c(x,u)$ and $p(x)$ are twice continuously differentiable (possibly nonlinear) functions

Research focus:

Solution of the FTOC problem at real-time

Research directions:

1. Precompute the solution?
2. Optimise the problem's complexity?
3. Efficient optimisation algorithms?

FTOC problem:

$$\begin{aligned}
\min_{\boldsymbol{z},\boldsymbol{u}} \quad & \sum_{t=0}^{N-1} q(z_t,u_t) + p(z_N) \\
\text{s.t.} \quad & z_0 = x, \\
& \text{for } t \in \{0,\ldots,N-1\}: \\
& z_{t+1} = f(z_t,u_t), \\
& c(z_t,u_t) \leq 0.
\end{aligned}$$

Previous work: Minimax Approximate EMPC

Consider convex MPC problem as a multi-parametric problem in x:

$$
\begin{aligned}
J^\star(x) = \min_{\boldsymbol{z},\boldsymbol{u}} \quad & \sum_{t=0}^{N-1} q(z_t, u_t) + p(z_N) \\
s.t. \quad & z_0 = x,\ z_{k+1} = Az_t + Bu_t \\
& c(z_t, u_t) \leq 0,\ c_f(z_N) \leq 0 \leftarrow \textbf{terminal constraints}
\end{aligned}
$$

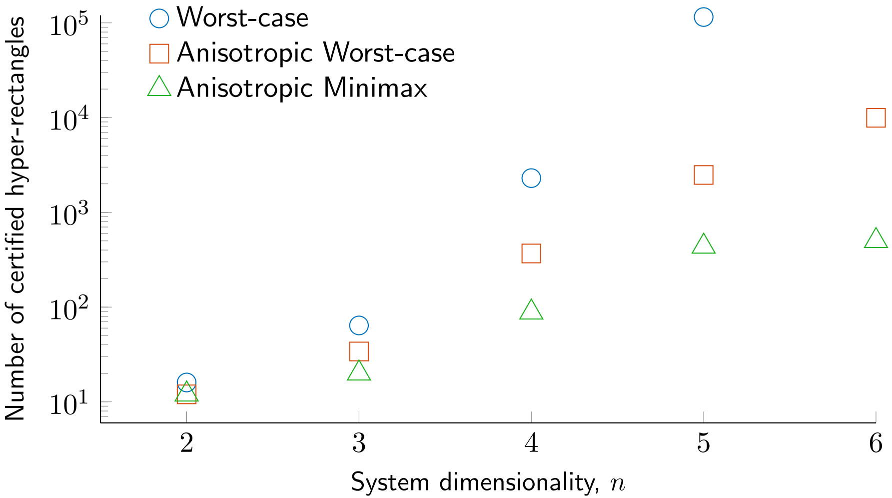
Explicit MPC: compute the optimal control law $u^\star(x) = F_j x + h_j$ (where j is a region's label)

Approximate Explicit MPC: approximate $u^\star(x)$ with $\hat{u}(x)$ such that stability and recursive feasibility properties are preserved

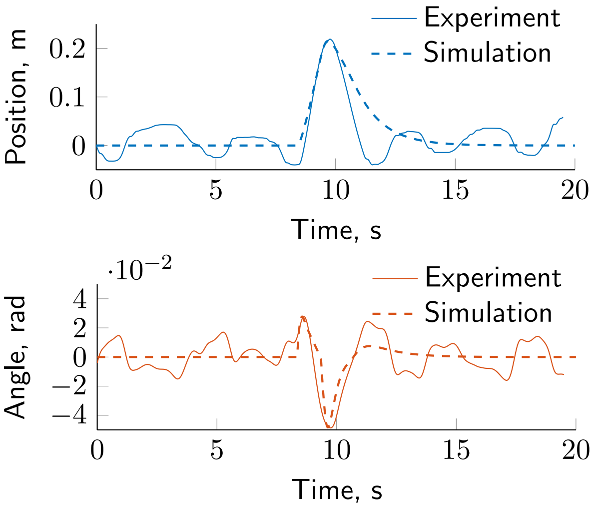
Proposed method: $\hat{u}(x) = \sum \lambda_i^\star u^\star(x_i)$ + minimax stability certificate, where $\lambda^\star = \arg\min \sum \lambda_i J^\star(x_i)$ s.t. *barycentric interpolation conditions*

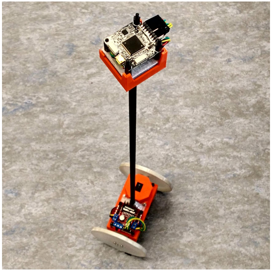
Minimax AEMPC: Numerical tests and comparisons

Here we generate random marginally stable systems for $n = 2, 3, 4, 5, 6$ with one constrained input $u \in [-1, 1]$ and partition the set $\|x\|_\infty \leq 1$.



Minimax AEMPC: Practical implementation





- In total $\approx 24k$ hyper-cubes
- Linear programs are solved at 100Hz on a micro-controller.

Publication: Pavlov et. al. "Minimax strategy in approximate model predictive control." Automatica 111 (2020): 108649. `https://youtu.be/233ZM8I6WBM`

The MPC problem

Setup:

- Dynamical system $x^+ = f(x, u)$
- State and input constraints $c(x, u) \leq 0$
- Stage and terminal costs $q(x, u)$ and $p(x)$

where $f(x, u)$, $c(x, u)$, $c(x, u)$ and $p(x)$ are twice continuously differentiable (possibly nonlinear) functions

Research focus:

Solution of the FTOC problem at real-time

Research directions:

1. Precompute the solution?
2. Optimise the problem's complexity?
3. Efficient optimisation algorithms?

FTOC problem:

$$
\begin{aligned}
\min_{\boldsymbol{z}, \boldsymbol{u}} \quad & \sum_{t=0}^{N-1} q(z_t, u_t) + p(z_N) \\
\text{s.t.} \quad & z_0 = x, \\
& \text{for } t \in \{0, \ldots, N-1\} : \\
& z_{t+1} = f(z_t, u_t), \\
& c(z_t, u_t) \leq 0.
\end{aligned}
$$

Previous work: MPC complexity minimisation

MPC without terminal constraints (i.e., $p(x) \equiv 0$)

$$
\begin{aligned}
J^\star(x) = \min_{\boldsymbol{x},\boldsymbol{u}} \quad & \sum_{t=0}^{N-1} q(z_t, u_t) \quad \Big[=: J(x, \boldsymbol{u}) \Big] \\
\text{s.t.} \quad & z_0 = x, \\
& \text{for } t \in \{0, \ldots, N-1\} : \\
& z_{t+1} = f(z_t, u_t), \\
& c(z_t, u_t) \leq 0.
\end{aligned}
$$

Property: Provable closed-loop stability when N is sufficiently big

How to make the problem simpler to solve?

1. Choose an optimisation algorithm
2. MPC complexity $=$ #iterations $\times$ per-iteration complexity
3. #iterations $=$ iterations for a "suitable" suboptimal solution
4. $\min$ (MPC complexity)
   subject to (stability and feasibility guarantees)

Previous work: MPC complexity minimisation

Definition: a γ-suboptimal solution if $J(x, \boldsymbol{u}) - J^\star(x) \leq \gamma q(x, u_0)$ for $\gamma \in [0, 1)$
Certificate: Duality gap $G(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{s}; x) \leq \gamma q(x, u_0)$ ($\boldsymbol{s}$ is a solution to the dual problem)
Stability conditions: closed-loop stability and performance bound under γ-suboptimality conditions

$$1 - \gamma - \frac{(\gamma + \nu_N - 1) \prod_{i=2}^{N} (\nu_i - 1)}{\prod_{i=2}^{N} \nu_i - \prod_{i=2}^{N} (\nu_i - 1)} \geq \alpha_{min}$$

Publications:

1. Pavlov et. al. "Early Termination of NMPC Interior Point Solvers: Relating the Duality Gap to Stability." 2019 18th European Control Conference (ECC), 2019.
2. Pavlov et. al. "Complexity minimisation of suboptimal MPC without terminal constraints", in the Proceedings of IFAC World Congress, Berlin, July 2020.
3. Pavlov et. al. "Algorithmic complexity minimisation of suboptimal MPC without terminal conditions." *Journal version, in progress.*

Example: NMPC + interior-point methods

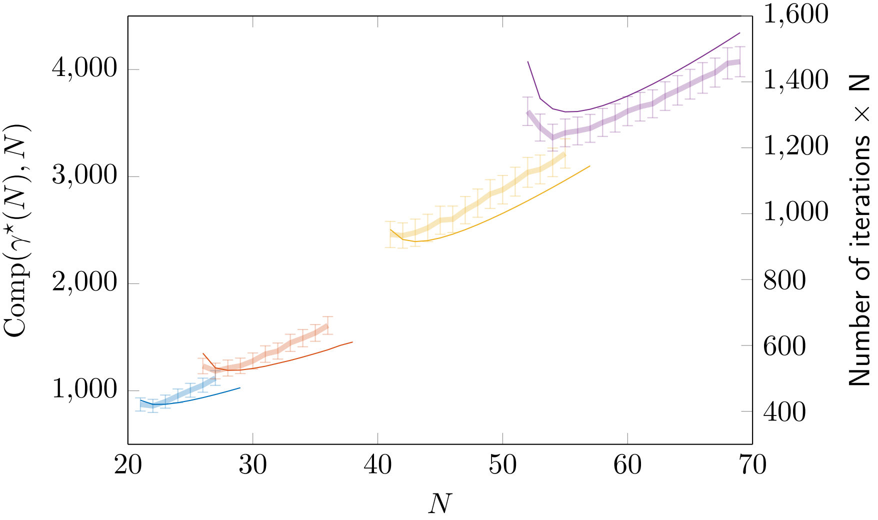


Figure: Computational complexity (thin lines) and experimentally measured computational efforts with its standard deviation (thick semi-transparent lines).

The MPC problem

Setup:

- Dynamical system $x^+ = f(x,u)$
- State and input constraints $c(x,u) \leq 0$
- Stage and terminal costs $q(x,u)$ and $p(x)$

where $f(x,u)$, $c(x,u)$, $c(x,u)$ and $p(x)$ are twice continuously differentiable (possibly nonlinear) functions

Research focus:

Solution of the FTOC problem at real-time

Research directions:

1. Precompute the solution?
2. Optimise the problem's complexity?
3. Efficient optimisation algorithms?

FTOC problem:

$$
\begin{aligned}
\min_{\boldsymbol{z},\boldsymbol{u}} \quad & \sum_{t=0}^{N-1} q(z_t,u_t) + p(z_N) \\
\text{s.t.} \quad & z_0 = x, \\
& \text{for } t \in \{0,\dots,N-1\}: \\
& z_{t+1} = f(z_t,u_t), \\
& c(z_t,u_t) \leq 0.
\end{aligned}
$$

Outline

1. Motivation
   - Feedback laws
   - Dynamical feasibility
   - Inequality constraints
2. Contribution
   - Roadmap
   - DDP recursion
   - Properties
3. Verification
   - Practical implementation
   - Numerical comparisons
   - Conclusions

Problem formulation

Develop an efficient method for solving the finite-time optimal control problem (FTOC):

$$
\begin{aligned}
\min_{\boldsymbol{z},\boldsymbol{u}} \quad & \sum_{t=0}^{N-1} q(z_t, u_t) + p(z_N) \\
\text{s.t.} \quad & z_0 = x, \\
& \text{for } t \in \{0, \dots, N-1\}: \\
& z_{t+1} = f(z_t, u_t), \\
& c(z_t, u_t) \leq 0.
\end{aligned}
$$

- **Inequality constrained problem:** Sequential quadratic programming (SQP), Augmented Lagrangian (AL), Interior-Point (IP) methods, etc.
- **Feasible (suboptimal) solutions:** Condensed problem, Differential Dynamic Programming (DDP) method
- **Closed-loop control:** Linear Quadratic Regulator (LQR), Model predictive Control (MPC)

Research direction: combination of the methods and their properties

Motivation: The optimal closed-loop control

Techniques behind the optimal closed-loop control:

- **Model Predictive Control**
  Idea: Resolving the problem for each new state
  Limitation: "Solving from scratch" is hard
  Take-away: Need fast convergence for the real-time capabilities

- **Bellman's Optimality Principle**
  Idea: $\min\limits_{u_0,\ldots,u_{N-1}} [\ldots] = \min\limits_{u_0} \Big[\ldots + \min\limits_{u_1} \big[\ldots + \min\limits_{u_2} [\ldots] \big] \Big]$
  Limitation: Analytical solution is rarely available (e.g., LQR)
  Take-away: Compute locally optimal control law

Motivation: Dynamically feasible solutions

Property: Feasible solutions can be checked for suboptimality
Issue: general-purpose optimisation methods satisfy $z_{t+1} = f(z_t, u_t)$ only in the limit

Solution #1: Propagate the dynamics for the obtained control solution
Disadvantage: Potential violation of the inequality constraints

Solution #2: Eliminate $z_{t+1} = f(z_t, u_t)$ by substitution, i.e., $\min_{u_0,\ldots,u_{N-1}} J(x_0, \boldsymbol{u})$ – optimisation in control inputs only
Disadvantage: Dense Hessian – Newton method has $O(N^3)$ complexity

Solution #3: Stage-wise Newton method with $O(N)$ complexity
Disadvantage: Hard to implement

Solution #4: Differential Dynamic Programming (DDP)
Disadvantage: Inequality constraints? $\leftarrow$ let's go for it

Motivation: SQP vs AL vs IP for constrained problem

Sequential quadratic programming method

- **Idea:** Use constrained quadratic models (QP) for solution updates
- **Properties:** Well-established and popular method, but significant computational cost for each iteration

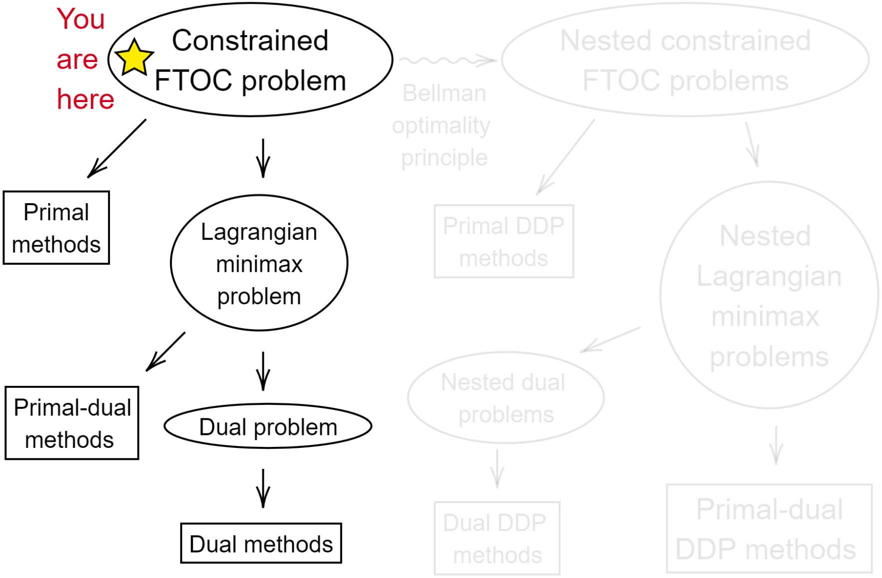
Penalty and Augmented Lagrangian methods

- **Idea:** Add "penalties" in the objective or Lagrangian functions, solve the resulting unconstrained problem
- **Properties:** Sometimes useful, usually slower convergence

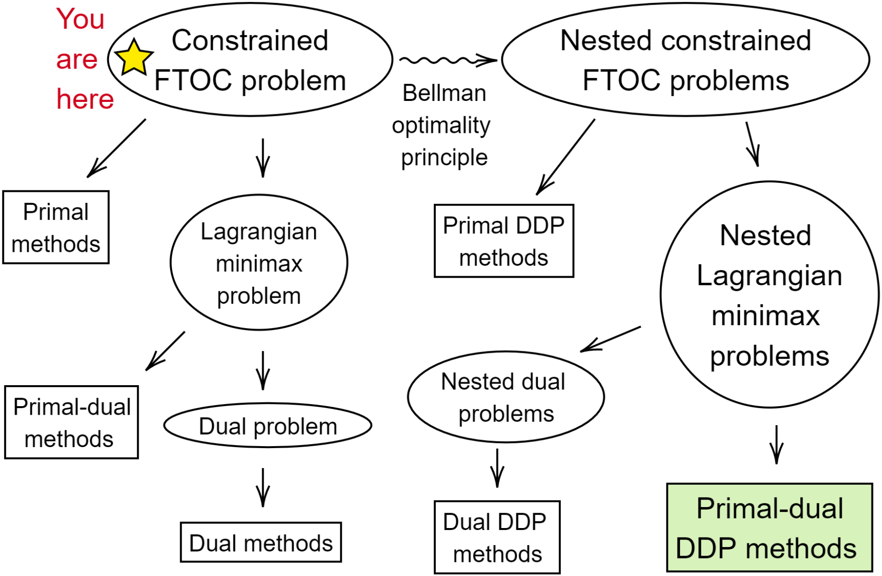
Interior-point method

- **Idea:** Perturb the KKT system and solve it for decaying perturbation
- **Properties:** Theoretically appealing and remarkably successful in practice

Roadmap: Optimisation algorithms



Roadmap: Optimisation algorithms



Three steps to the nested minimax problems

1. Apply **Bellman's principle of optimality** to the FTOC problem

$$J_N^\star(x_0) = \min_{\boldsymbol{z},\boldsymbol{u}} \quad \sum_{t=0}^{N-1} q(z_t, u_t) + p(z_N)$$
$$\text{s.t.} \quad \text{constraints}$$

2. Obtain the **nested formulation** of the OC problem

$$\min_{\substack{u_0 \ s.t. \\ c(x_0,u_0)\leq 0}} \left[q(x_0, u_0) + \min_{\substack{u_1 \ s.t. \\ c(f(x_0,u_0),u_1)\leq 0}} \left[q(f(x_0, u_0), u_1) + \dots \right] \right]$$

3. Represent the above problem as the **minimax problem**

$$\min_{u_0} \max_{s_0 \geq 0} \left[\ell(x_0, u_0, s_0) + \min_{u_1} \max_{s_1 \geq 0} \left[\ell(f(x_0, u_0), u_1, s_1) + \dots \right] \right],$$

where $\ell(x, u, s) = q(x, u) + s^T c(x, u) \leftarrow$ stage-wise Lagrangian.

The nested minimax problems

Consider the following recursion:

- $\underbrace{\min_{u_0} \max_{s_0 \geq 0} \Big[\ell(x_0, u_0, s_0) + \min_{u_1} \max_{s_1 \geq 0} \big[\ell(f(x_0, u_0), u_1, s_1) + \ldots \big] \Big]}_{Solve?}$
- $\min_{u_0} \max_{s_0 \geq 0} \Big[\underbrace{\ell(x_0, u_0, s_0) + \min_{u_1} \max_{s_1 \geq 0} \big[\ell(f(x_0, u_0), u_1, s_1) + \ldots \big]}_{Approximate?} \Big]$
- $\min_{u_0} \max_{s_0 \geq 0} \Big[\ell(x_0, u_0, s_0) + \underbrace{\min_{u_1} \max_{s_1 \geq 0} \big[\ell(f(x_0, u_0), u_1, s_1) + \ldots \big]}_{Solve?} \Big]$
- ...

It's a finite recursion with the terminal cost $p(x)$ in the end:

$$\min_{u_0} \max_{s_0 \geq 0} \Big[\ldots + \min_{u_{N-1}} \max_{s_{N-1} \geq 0} \big[\ell(x_{N-1}, u_{N-1}, s_{N-1}) + p(f(x_{N-1}, u_{N-1})) \big] \Big]$$

Checkpoint: Overview

The idea behind any DDP method:

Start with an initial solution guess and repeat until convergence:

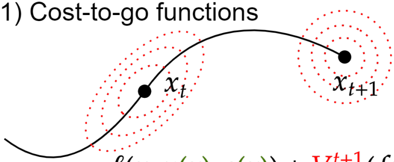
1. Backward pass: Resolve the recursion from **(end)** to **(start)**
2. Forward pass: Update the solution from **(start)** to **(end)**

Need to answer:

- How to resolve the recursion?
- How to update the solution?
- When (and why) the algorithm works?

Backward pass: Ingredients

1) Cost-to-go functions



$$\ell(x, u(x), s(x)) + V^{t+1}(f(x, u(x)))$$

3) Local behaviour

$$u(x) = u_t + \delta u(x - x_t)$$
$$s(x) = s_t + \delta s(x - x_t)$$

2) Minimax problems

$$\min_{\delta u} \max_{s_t + \delta s \geq 0} \Big[\ell(x_t + \delta x, u_t + \delta u, s_t + \delta s) + V^{t+1}(f(x_t + \delta x, u_t + \delta u)) \Big]$$

Cost-to-go function $\Rightarrow$ minimax problem

- Assume trajectory $\{x_t, u_t, s_t\}_{t=0}^{N-1}$ is given,
  where $x_{t+1} = f(x_t, u_t)$, $c(x_t, u_t) \leq 0$ and $s_t > 0$.
- Consider quadratic models of the cost-to-go functions

$$V^t(x) := V_0^t + \left(V_x^t\right)^T (x - x_t) + \frac{1}{2}(x - x_t)^T V_{xx}^t (x - x_t) + \ldots,$$

- Note that $V^N(x) \equiv p(x)$, thus

  $V_0^N = p(x_N)$, $V_x^N = p_x(x_N)$ and $V_{xx}^N = p_{xx}(x_N)$.

Problem at time t is

$$\min_u \max_{s \geq 0} \Big[\underbrace{\ell(x, u, s) + V^{t+1}\big(f(x, u)\big)}_{\text{call it } Q\text{-function}} \Big]$$

Minimax problem $\Rightarrow$ local behaviour

- Replace $\underbrace{\min_{u} \max_{s \geq 0} Q^t(x,u,s)}_{\substack{\text{exact problem} \\ \text{with } Q\text{-function}}}$ with $\underbrace{\min_{\delta u} \max_{s+\delta s \geq 0} \delta Q^t(\delta x, \delta u, \delta s)}_{\text{approximation}}$

where $\delta Q^t(\delta x, \delta u, \delta s) := \begin{bmatrix} Q_x^t \\ Q_u^t \\ Q_s^t \end{bmatrix}^T \begin{bmatrix} \delta x \\ \delta u \\ \delta s \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \\ \delta s \end{bmatrix}^T \begin{bmatrix} Q_{xx}^t & Q_{xu}^t & Q_{xs}^t \\ Q_{ux}^t & Q_{uu}^t & Q_{us}^t \\ Q_{sx}^t & Q_{su}^t & Q_{ss}^t \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta s \end{bmatrix}$

- Optimisation with respect to δu and δs yields

$$Q_u^t + Q_{ux}^t \delta x + Q_{uu}^t \delta u + Q_{us}^t \delta s = 0$$
$$(s_t + \delta s) \odot (Q_s^t + Q_{sx}^t \delta x + Q_{su}^t \delta u) = 0$$

Perturb the equations with $\mu > 0$ and drop the second-order terms

$$\begin{bmatrix} Q_{uu}^t & Q_{us}^t \\ S_t Q_{su}^t & C_t \end{bmatrix} \begin{bmatrix} \delta u \\ \delta s \end{bmatrix} = - \begin{bmatrix} Q_u^t \\ S_t c(x_t, u_t) + \boldsymbol{\mu} \end{bmatrix} - \begin{bmatrix} Q_{ux}^t \\ S_t Q_{sx} \end{bmatrix} \delta x,$$

where $C_t := \mathrm{diag}[c(x_t, u_t)]$ and $S_t := \mathrm{diag}[s_t]$.

Perturbation parameter μ

The "perturbed" parametric system for δu and δs

$$\begin{bmatrix} Q^t_{uu} & Q^t_{us} \\ S_t Q^t_{su} & C_t \end{bmatrix} \begin{bmatrix} \delta u \\ \delta s \end{bmatrix} = - \begin{bmatrix} Q^t_u \\ S_t c(x_t, u_t) + \boldsymbol{\mu} \end{bmatrix} - \begin{bmatrix} Q^t_{ux} \\ S_t Q_{sx} \end{bmatrix} \delta x,$$

Theoretical aspects of using μ

- Keeps solutions away from the feasible set boundaries (think of log-barrier)
- Make problems "smoother" and easier to optimise (think of homotopy)

Practical aspects of using μ

- Controls the convergence (by following the interior-point central path)
- Allows for balance between optimality and algorithmic complexity

Local behaviour $\Rightarrow$ cost-to-go function

- Step t depends on step $t+1$, e.g.,

$$
\begin{aligned}
Q_x^t &= \ell_x + f_x^T V_x^{t+1} \\
Q_{xu}^t &= \ell_{xu} + f_x^T V_{xx}^{t+1} f_u + V_x^{t+1} \cdot f_{xu}
\end{aligned}
$$

- Can solve for $(\delta u, \delta s)$ at step t:

$$
\begin{bmatrix} \delta u \\ \delta s \end{bmatrix} = \begin{bmatrix} \alpha_t \\ \eta_t \end{bmatrix} + \begin{bmatrix} \beta_t \\ \theta_t \end{bmatrix} \delta x
$$

The cost-to-go model at time t is well-defined (given info from $t+1$)

$$
\begin{aligned}
V_x^t &= \hat{Q}_x^t + \hat{Q}_{xu}^t \alpha_t \\
V_{xx}^t &= \hat{Q}_{xx}^t + \hat{Q}_{xu}^t \beta_t
\end{aligned}
$$

Forward pass: updating the solution guess

1. Define the update functions

$$\phi_t(x) := u_t + \alpha_t + \beta_t(x - x_t),$$
$$\psi_t(x) := s_t + \eta_t + \theta_t(x - x_t),$$

2. Denote a new solution guess by

$$\boldsymbol{x}^+ = (x_0^+, \ldots, x_N^+)$$
$$\boldsymbol{u}^+ = (u_0^+, \ldots, u_{N-1}^+)$$
$$\boldsymbol{s}^+ = (s_0^+, \ldots, s_{N-1}^+)$$

3. Initialise $x_0^+ = x_0$ and compute for $t = 0, \ldots, N-1$:

$$u_t^+ = \phi_t(x_t^+),$$
$$s_t^+ = \psi_t(x_t^+),$$
$$x_{t+1}^+ = f(x_t^+, u_t^+)$$

When and why IPDDP works

Assume

1. Strict primal-dual feasibility, i.e., $c(x_t, u_t) < 0$ and $s_t > 0$
2. Matrices $\hat{Q}^t_{uu}$ (related to Q^t_{uu}) are positive definite

Then

1. Stationary points of IPDDP $\iff$ perturbed KKT points
2. IPDDP has a local quadratic convergence rate

Moreover

- Global convergence with line-search, regularisation and step filter
- Convergence to the locally optimal solutions[1]
- Constrained feedback control laws
- Can handle primal infeasible guess? Yes! (after modification)

[1] under regularity and 2nd order sufficient optimality conditions

Infeasible IPDDP

Backward pass:

1. Introduce slack variables: $c(x_t, u_t) + y_t = 0$, where $y_t \geq 0$
2. New parametric system of equations

$$\begin{bmatrix} Q^t_{uu} & Q^t_{us} & 0 \\ Q^t_{su} & 0 & I \\ 0 & Y_t & S_t \end{bmatrix} \begin{bmatrix} \delta u \\ \delta s \\ \delta y \end{bmatrix} = - \begin{bmatrix} Q^t_u \\ c(x_t, u_t) + y_t \\ S_t y_t - \boldsymbol{\mu} \end{bmatrix} - \begin{bmatrix} Q^t_{ux} \\ Q^t_{sx} \\ 0 \end{bmatrix} \delta x$$

Forward pass: slack updates $y_t^+ = y_t + \chi_t + \zeta_t(x - x_t)$.

Checkpoint: Results overview

Feasible IPDDP

- Primal-dual feasibility: $c(x_t, u_y) < 0$ and $s_t > 0$
- Stationary points $\Leftrightarrow$ Pertb. KKT points
- Local quadratic convergence if $\hat{Q}^t_{uu} \succ 0$

Infeasible IPDDP

- Dual feasibility: $s_t > 0$ and $y_t > 0$
- Stationary points $\Leftrightarrow$ Pertb. KKT points
- Local quadratic convergence if $\hat{Q}^t_{uu} \succ 0$

Practical implementation:

1. Regularisation: use $\hat{Q}_{uu} + \sigma I$ when $\hat{Q}_{uu} \not\succ 0$
2. Line-search with a step-size $\gamma \in (0; 1]$:

$$u_t^+ = u_t + \gamma\alpha_t + \beta_t(x - x_t),$$
$$s_t^+ = s_t + \gamma\eta_t + \theta_t(x - x_t),\ etc$$

3. Step filter: strict reduction of the optimality error
4. Perturbation: start with $\mu > 0$ and reduce it in the process

Advantages of IPDDP: Perturbation $\Rightarrow$ smoothness

Inverted pendulum problem:

$$f(x,u) = \begin{bmatrix} \varphi + h\omega \\ \omega + h\sin(\varphi) + h \end{bmatrix} \text{ subject to } -0.25 \le u \le 0.25$$

Solutions to the perturbed problem:

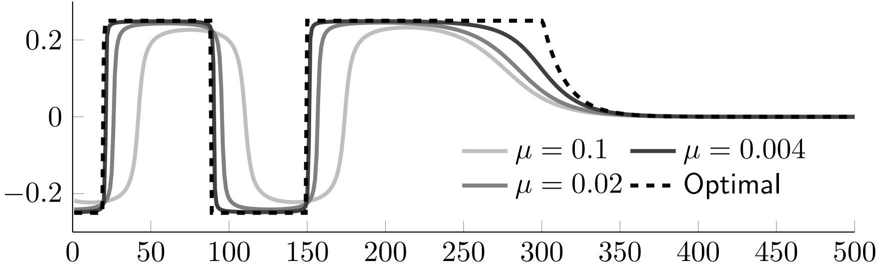
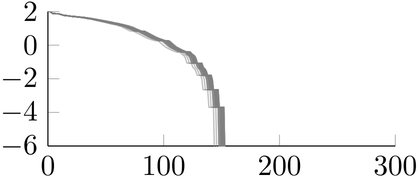


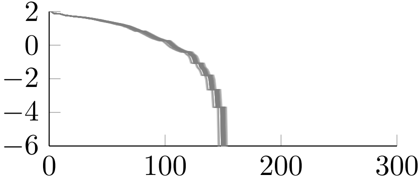
Figure: Control inputs u_t on y-axis vs time-index t on x-axis

Advantages of IPDDP: Numerical comparisons

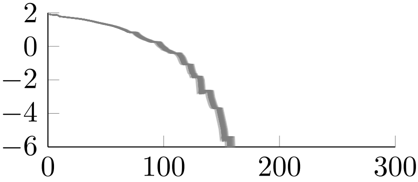
Inverted pendulum problem:



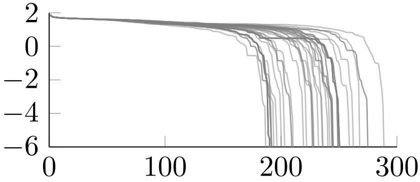
(a) Feasible-IPDDP



(b) Infeasible-IPDDP



(c) Log-barrier DDP



(d) SQP-type DDP

Figure: $\log[J(\boldsymbol{x},\boldsymbol{u}) - J(\boldsymbol{x}^\star,\boldsymbol{u}^\star)]$ on y-axis vs iteration number on x-axis

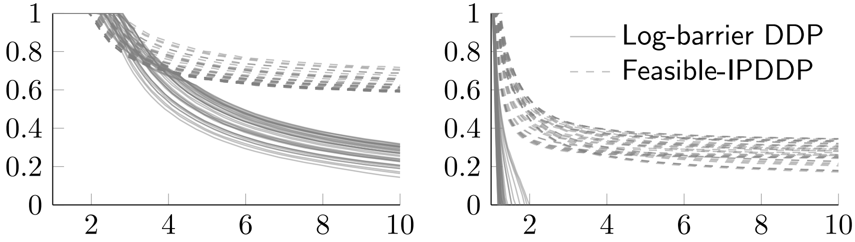
Advantages of IPDDP: IP Central path-following

Car-parking problem:

$$f(x,u) = \begin{bmatrix} r_x + b(v,w)\cos(\varphi) \\ r_y + b(v,w)\sin(\varphi) \\ \varphi + \sin^{-1}\left(\frac{hv}{d}\sin(w)\right) \\ v + ha \end{bmatrix} \text{ subject to } -\begin{bmatrix} 0.5 \\ 2 \end{bmatrix} \le \begin{bmatrix} w \\ a \end{bmatrix} \le \begin{bmatrix} 0.5 \\ 2 \end{bmatrix},$$

where $b(v,w) = d + hv\cos(\omega) - \sqrt{d^2 - h^2v^2\sin^2(w)}$

Perturbation strategy: $\mu \leftarrow \min(\mu/\kappa, \mu^{1.2})$

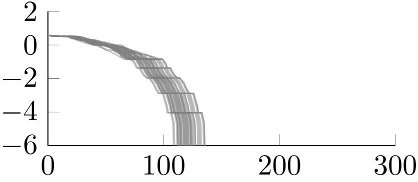


(a) $0.0001 \le \mu \le 0.0005$ (b) $0.0006 \le \mu \le 0.0010$

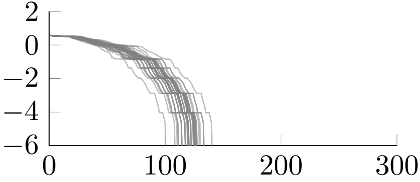
Figure: The maximum accepted stepsize on y-axis vs reduction factor κ on x-axis

Advantages of IPDDP: Numerical comparisons

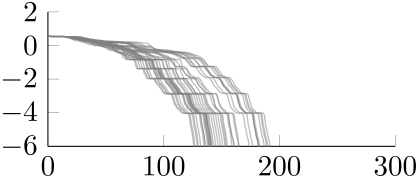
Car-parking problem:



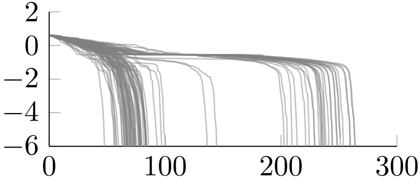
(a) Feasible-IPDDP



(b) Infeasible-IPDDP



(c) Log-barrier DDP



(d) SQP-type DDP

Figure: $\log[J(\boldsymbol{x}, \boldsymbol{u}) - J(\boldsymbol{x}^\star, \boldsymbol{u}^\star)]$ on y-axis vs iteration number on x-axis

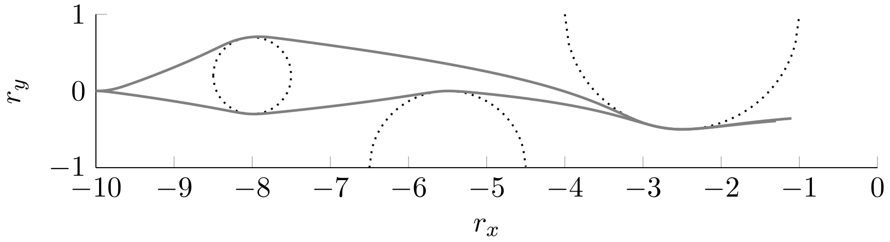
Advantages of IPDDP: Infeasible guesses

Unicycle motion problem:

$$f(x,u) = \begin{bmatrix} r_x + hv\cos(\varphi) \\ r_y + hv\sin(\varphi) \\ \varphi + hu \end{bmatrix}$$

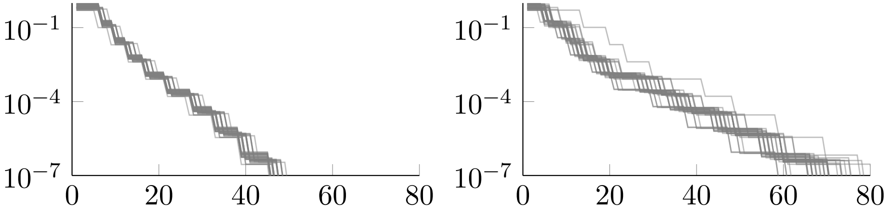
subject to the input and state constraints

$$-1.5 \leq u \leq 1.5,\ -1 \leq r_y \leq 1,\ \|r_x + 5.5, r_y + 1\|^2 \geq 1,$$
$$\|[r_x + 8,\ r_y - 0.2]\|^2 \geq 0.5^2,\ \|[r_x + 2.5,\ r_y - 1]\|^2 \geq 1.5^2.$$



Advantages of IPDDP: Numerical comparisons

Unicycle motion problem:



(a) Infeasible-IPDDP (b) Relaxed* log-barrier DDP

Figure: Unicycle motion control: optimality error on y-axis vs iteration number on x-axis.

$$^{*}\beta_\delta(z) = \begin{cases} -\log z & z > \delta, \\ \frac{1}{2}\left[\left(\frac{z-2\delta}{\delta}\right)^2 - 1\right] - \log\delta & z \leq \delta, \end{cases}$$

Conclusions and future work

Properties of IPDDP:

1. Locally (sub)optimal solutions
2. Local quadratic convergence (super-linear in practice)
3. Linear complexity in N
4. Dynamical feasibility
5. Comparable (or better) numerical performance
6. Infeasible solution guesses
7. Constrained feedback control laws (reinforcement learning?)

Further reaseach:

- Predictor-corrector steps
- Adaptive perturbation strategy
- Bilevel programs - optimisation problems with extra optimisation problems as constraints (aka MPEC)
- Optimisation on manifolds

Preprint: arXiv:2004.12710