Stochastic Differential Dynamic Programming (SDDP)

Adel Abdelsamed

Chair of Intelligent Control Systems
RWTH Aachen University

Supervisor:
Dennis Gramlich M.Sc

Seminar Presentation
July 20, 2023

Control Theory Roadmap

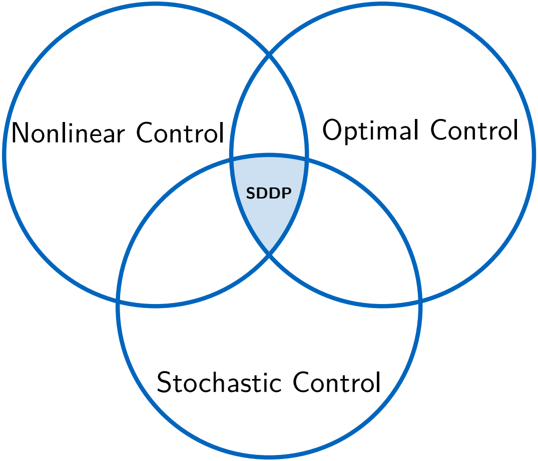

Figure: Classification within the control theory disciplines.

Table of Contents

Introduction

Problem statement

Relaxation of nonlinearities and discretization

Derivation of the SDDP framework

Handling the challenges of nonlinear optimization

Examples

Open challenges and limitations

Table of Contents

Introduction

Problem statement

Relaxation of nonlinearities and discretization

Derivation of the SDDP framework

Handling the challenges of nonlinear optimization

Examples

Open challenges and limitations

Introduction

We consider the optimal control problem

$$\begin{aligned} \underset{u(\cdot,\cdot)}{\text{minimize}} \quad & \int_{t_0}^{T} f_0(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau, x(\tau)))d\tau + \Phi_N(x(t_0+T)) \\ \text{subject to} \quad & \frac{dx(t)}{dt} = f(\boldsymbol{x}(t), \boldsymbol{u}(t, x(t))), \quad t \in [t_0, t_0+T] \\ & \boldsymbol{x}(t_0) = x_0 \end{aligned}$$

- Dynamic Programming (DP) provides a framework to solve optimal control problems.
  $\hookrightarrow$ Curse of Dimensionality
- Differential DP as one of the most efficient optimal control solvers.
- Stochastic DDP[1] combining DDP methods with stochastic control.

[1] Evangelos Theodorou, Yuval Tassa, and Emo Todorov. "Stochastic Differential Dynamic Programming". In: *Proceedings of the 2010 American Control Conference*. 2010, pp. 1125–1132. DOI: 10.1109/ACC.2010.5530971.

We consider the optimal control problem

$$\begin{aligned} \underset{u(\cdot,\cdot)}{\text{minimize}} \quad & \int_{t_0}^{T} f_0(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau, x(\tau)))d\tau + \Phi_N(x(t_0 + T)) \\ \text{subject to} \quad & \textbf{Stochastic System ??} \\ & \boldsymbol{x}(t_0) = x_0 \end{aligned}$$

- Dynamic Programming (DP) provides a framework to solve optimal control problems.
  $\hookrightarrow$ Curse of Dimensionality
- Differential DP as one of the most efficient optimal control solvers.
- Stochastic DDP[2] combining DDP methods with stochastic control.

[2] Theodorou, Tassa, and Todorov, "Stochastic Differential Dynamic Programming".

Introduction: Review of existing works

Comparison of existing works aiming to incorporate stochastic disturbances into DDP.

	iLQG[3]	SDDP[4]	PDDP[5]
Approximation cost-to-go function	Second order	Second order	Second order
Approximation system dynamics	First order $(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)$	Second order $(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)$	First order $(\varrho_{\bar{\boldsymbol{x}}_k}, \bar{\boldsymbol{u}}_k)$
System model	Known: WDSDE Space: $\boldsymbol{x}_k$	Known: WDSDE Space: $\boldsymbol{x}_k$	Unknown: GPR Space: μ_k, Σ_k

iLQG: Control-multiplicative noise

$$d\boldsymbol{x} = f(\boldsymbol{x}, \boldsymbol{u})dt + F(\boldsymbol{u})dw \tag{1}$$

SDDP: State- and control-multiplicative noise

$$d\boldsymbol{x} = f(\boldsymbol{x}, \boldsymbol{u})dt + F(\boldsymbol{x}, \boldsymbol{u})dw \tag{2}$$

[3] E. Todorov and Weiwei Li. "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems". In: *Proceedings of the 2005, American Control Conference, 2005.* 2005, 300–306 vol. 1. DOI: 10.1109/ACC.2005.1469949.

[4] Theodorou, Tassa, and Todorov, "Stochastic Differential Dynamic Programming".

[5] Yunpeng Pan and Evangelos Theodorou. "Probabilistic Differential Dynamic Programming". In: *Advances in Neural Information Processing Systems.* Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/7fec306d1e665bc9c748b5d2b99a6e97-Paper.pdf.

Problem statement: Basic Ingredients

- System Dynamics:

$$d\boldsymbol{x} = f(\boldsymbol{x}, \boldsymbol{u})dt + F(\boldsymbol{x}, \boldsymbol{u})dw, \tag{3}$$

$\boldsymbol{x} \in \mathbb{R}^{n\times 1}$, $\boldsymbol{u} \in \mathbb{R}^{p\times 1}$, $dw \in \mathbb{R}^{m\times 1}$, $f \colon \mathbb{R}^{n\times 1} \times \mathbb{R}^{p\times 1} \mapsto \mathbb{R}^{n\times 1}$,
$F \colon \mathbb{R}^{n\times 1} \times \mathbb{R}^{p\times 1} \mapsto \mathbb{R}^{n\times m}$.

- Control Policy:

$$\boldsymbol{u} = \boldsymbol{\pi}(t, \boldsymbol{x}(t)) \tag{4}$$

- Expected cost starting at $\boldsymbol{x}(t_0)$ (Cost-to-go):

$$J^{\pi}(t_0, \boldsymbol{x}(t)) = \mathbb{E}\left[\int_{t_0}^{T} \ell(\tau, \boldsymbol{x}(\tau), \pi(\tau, \boldsymbol{x}(\tau)))\, d\tau + \phi_N(\boldsymbol{x}(T))\right] \tag{5}$$

Running Cost: $\boldsymbol{\ell} \colon \mathbb{R}^{1\times 1} \times \mathbb{R}^{n\times 1} \times \mathbb{R}^{p\times 1} \mapsto \mathbb{R}^{1\times 1}$
Terminal Cost: $\boldsymbol{\phi_N} \colon \mathbb{R}^{n\times 1} \mapsto \mathbb{R}^{1\times 1}$

Problem statement

- Objective of SDDP: Find the optimal policy $\boldsymbol{\pi}^*$ to steer the system from $\boldsymbol{x}(t_0)$ to $\boldsymbol{x}(T)$ while minimizing the cost function $J^\pi(t_0, \boldsymbol{x}(t))$.
  ... or more formally:

$$
\begin{aligned}
\underset{\pi(t,\boldsymbol{x}(t))}{\text{minimize}} \quad & \mathbb{E}\left[\int_{t_0}^{T} \ell(\tau, \boldsymbol{x}(\tau), \pi(\tau, \boldsymbol{x}(\tau)))\, d\tau + \phi_N(\boldsymbol{x}(T))\right] \\
\text{subject to} \quad & d\boldsymbol{x} = f(\boldsymbol{x}, \boldsymbol{u})dt + F(\boldsymbol{x}, \boldsymbol{u})dw \\
& \boldsymbol{x}(t_0) = x_0
\end{aligned}
$$

- Continuous-time, finite-horizon stochastic optimal control problem

Bigger Picture

Solving the continuous-time SOCP is complex, therefore some simplifications are required:

1. Replace nonlinear system by a local approximation of the system.
2. Discretize the approximated system.
   $\hookrightarrow$ Discretize the SOCP.

Bigger Picture

Solving the continuous-time SOCP is complex, therefore some simplifications are required:

1. Replace nonlinear system by a local approximation of the system.
2. Discretize the approximated system.
   $\hookrightarrow$ Discretize the SOCP.

Relaxation of nonlinearties (1/3)

- One particular challenge that remains, is the system nonlinearity.
  $\hookrightarrow$ Compute second-order approximation of the system dynamics around a local state and control trajectory $(\boldsymbol{x}_k, \boldsymbol{u}_k)$

- Notation used for the immininent expansion to avoid tensorial terms:

  - $F(\boldsymbol{x},\boldsymbol{u}) = \begin{pmatrix} {F_r}^1(\boldsymbol{x},\boldsymbol{u}) \\ \vdots \\ {F_r}^n(\boldsymbol{x},\boldsymbol{u}) \end{pmatrix} = \big({F_c}^1(\boldsymbol{x},\boldsymbol{u}), \ldots, {F_c}^m(\boldsymbol{x},\boldsymbol{u})\big)$

  - Dynamics function $\Phi(\boldsymbol{x},\boldsymbol{u},dw)\colon \mathbb{R}^{n\times 1} \times \mathbb{R}^{p\times 1} \times \mathbb{R}^{m\times 1} \mapsto \mathbb{R}^{n\times 1}$
    $\Phi(\boldsymbol{x},\boldsymbol{u},dw) \equiv f(\boldsymbol{x},\boldsymbol{u})dt + F(\boldsymbol{x},\boldsymbol{u})dw$

  - The j-th element of $\Phi(\boldsymbol{x},\boldsymbol{u},dw)$:
    $\Phi^{(j)}(\boldsymbol{x},\boldsymbol{u},dw) = f^{(j)}(\boldsymbol{x},\boldsymbol{u})dt + {F_r}^{(j)}(\boldsymbol{x},\boldsymbol{u})dw$

Relaxation of nonlinearties (2/3)

- A nominal state and control trajectory $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$
- State $\delta\boldsymbol{x} = \boldsymbol{x} - \bar{\boldsymbol{x}}$ and control deviations $\delta\boldsymbol{u} = \boldsymbol{u} - \bar{\boldsymbol{u}}$
- Expand dynamics function up to second order about the nominal trajectories

$$\begin{aligned}\Phi(\bar{\boldsymbol{x}} + \boldsymbol{x}, \bar{\boldsymbol{u}} + \boldsymbol{u}, dw) \approx \Phi(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, dw) + \nabla_{\boldsymbol{x}}\Phi \cdot \delta\boldsymbol{x} \\ + \nabla_{\boldsymbol{u}}\Phi \cdot \delta\boldsymbol{u} + \boldsymbol{O}(\delta\boldsymbol{x}, \delta\boldsymbol{u}, dw),\end{aligned}$$

with the expanded second order dynamics vector $\boldsymbol{O}(\delta\boldsymbol{x}, \delta\boldsymbol{u}, dw) \in \mathbb{R}^{n\times 1}$ specified element-wise as:

$$O^{(j)}(\delta\boldsymbol{x}, \delta\boldsymbol{u}, dw) = \frac{1}{2}\begin{pmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{pmatrix}^{\mathsf{T}} \begin{pmatrix}\nabla_{\boldsymbol{xx}}\Phi^{(j)} & \nabla_{\boldsymbol{xu}}\Phi^{(j)} \\ \nabla_{\boldsymbol{ux}}\Phi^{(j)} & \nabla_{\boldsymbol{uu}}\Phi^{(j)}\end{pmatrix} \begin{pmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{pmatrix}.$$

Relaxation of nonlinearties (3/3)

- First-order derivatives:

$$\nabla_{\boldsymbol{x}}\Phi = \nabla_{\boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{u})dt + \nabla_{\boldsymbol{x}}\left(\sum_{i=1}^{m} {F_c}^{(i)}(\boldsymbol{x},\boldsymbol{u})dw^{(i)}\right)$$

$$\nabla_{\boldsymbol{u}}\Phi = \nabla_{\boldsymbol{u}} f(\boldsymbol{x},\boldsymbol{u})dt + \nabla_{\boldsymbol{u}}\left(\sum_{i=1}^{m} {F_c}^{(i)}(\boldsymbol{x},\boldsymbol{u})dw^{(i)}\right)$$

- Second-order derivatives:

$$\nabla_{\boldsymbol{xx}}\Phi^{(j)} = \nabla_{\boldsymbol{xx}} f^{(j)}(\boldsymbol{x},\boldsymbol{u})dt + \nabla_{\boldsymbol{xx}}\Big({F_r}^{(j)}(\boldsymbol{x},\boldsymbol{u})dw\Big)$$

$$\nabla_{\boldsymbol{uu}}\Phi^{(j)} = \nabla_{\boldsymbol{uu}} f^{(j)}(\boldsymbol{x},\boldsymbol{u})dt + \nabla_{\boldsymbol{uu}}\Big({F_r}^{(j)}(\boldsymbol{x},\boldsymbol{u})dw\Big)$$

$$\nabla_{\boldsymbol{xu}}\Phi^{(j)} = \nabla_{\boldsymbol{xu}} f^{(j)}(\boldsymbol{x},\boldsymbol{u})dt + \nabla_{\boldsymbol{xu}}\Big({F_r}^{(j)}(\boldsymbol{x},\boldsymbol{u})dw\Big)$$

$$\nabla_{\boldsymbol{ux}}\Phi^{(j)} = \nabla_{\boldsymbol{ux}} f^{(j)}(\boldsymbol{x},\boldsymbol{u})dt + \nabla_{\boldsymbol{ux}}\Big({F_r}^{(j)}(\boldsymbol{x},\boldsymbol{u})dw\Big).$$

Bigger Picture

Solving the continuous-time SOCP is complex, therefore some simplifications are required:

1. Replace nonlinear system by a local approximation of the system.
2. Discretize the approximated system.
   $\hookrightarrow$ Discretize the SOCP.

Discretization (1/3)

- Discretization of the Wiener-driven stochastic DE is done using Euler-Maruyama discretization scheme
  $\hookrightarrow$ Discretization of deterministic dynamics corresponds to forward euler scheme
  $$\delta\dot{\boldsymbol{x}} \approx \frac{\delta\boldsymbol{x}_{t+\delta t} - \delta\boldsymbol{x}_t}{\delta t}$$
  with a sufficiently small discretization interval $\delta t = t_{k+1} - t_k$.
  $\hookrightarrow$ Discretization of stochastic dynamics must ensure the linear dependence of the variance of the Brownian motion noise on time
  $$dw \approx \sqrt{\delta t}\xi$$
  with $\xi \sim \mathcal{N}(0, \sigma^2 I_{m\times m})$.

Discretization (2/3)

- The resulting discrete-time dynamics is

$$\delta \boldsymbol{x}_{t+\delta t} = \left(I_{n\times n} + \nabla_{\boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{u})\delta t + \nabla_{\boldsymbol{x}} \left(\sum_{i=1}^{m} {F_c}^{(i)} {\xi_t}^{(i)} \sqrt{\delta t} \right) \right) \delta \boldsymbol{x}_{\mathsf{t}}$$
$$+ \left(\nabla_{\boldsymbol{u}} f(\boldsymbol{x},\boldsymbol{u})\delta t + \nabla_{\boldsymbol{u}} \left(\sum_{i=1}^{m} {F_c}^{(i)} {\xi_t}^{(i)} \sqrt{\delta t} \right) \right) \delta \boldsymbol{u}_{\mathsf{t}} + F(\boldsymbol{x},\boldsymbol{u}) \boldsymbol{\xi}_t \sqrt{\delta t}$$
$$+ \boldsymbol{O}_d(\delta \boldsymbol{x}_{\mathsf{t}}, \delta \boldsymbol{u}_{\mathsf{t}}, \boldsymbol{\xi}_t, \delta t).$$

- Or more compactly, we define the new matrices
System matrix $A_t = I_{n\times n} + \nabla_{\boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{u})\delta t \in \mathbb{R}^{n\times n}$
Input matrix $B_t = \nabla_{\boldsymbol{u}} f(\boldsymbol{x},\boldsymbol{u})\delta t \in \mathbb{R}^{n\times p}$
Noise matrix $\Gamma_t = \left[\Gamma^{(1)} \; \Gamma^{(2)} \; \ldots \; \Gamma^{(m)} \right] \in \mathbb{R}^{n\times m}$ with
$\Gamma^{(i)} = \nabla_{\boldsymbol{x}} {F_c}^{(i)} \delta \boldsymbol{x}_{\mathsf{t}} + \nabla_{\boldsymbol{u}} {F_c}^{(i)} \delta \boldsymbol{u}_{\mathsf{t}} + F_c^{(i)}$.

Discretization (3/3)

The discrete-time dynamics can then be compactly expressed as

$$\delta \boldsymbol{x}_{t+\delta t} = A_t \delta \boldsymbol{x}_{\mathsf{t}} + B_t \delta \boldsymbol{u}_{\mathsf{t}} + \sqrt{\delta t}\Gamma_t \boldsymbol{\xi}_t + \boldsymbol{O}_d(\delta \boldsymbol{x}_{\mathsf{t}}, \delta \boldsymbol{u}_{\mathsf{t}}, \boldsymbol{\xi}_t, \delta t).$$

Now the discrete-time, simplified SOCP can be given as

$$\begin{aligned}
\underset{\substack{\delta \boldsymbol{u}_k \\ k=k_0,\dots k_0+N-1}}{\text{minimize}} \quad & \mathbb{E}\left[\sum_{k=k_0}^{k_0+N-1} \ell(k, \boldsymbol{x_k}, \boldsymbol{u_k}) \cdot \delta t + \phi_N(\boldsymbol{x_{k_0+N}})\right] \\
\text{subject to} \quad & \delta \boldsymbol{x_{k+1}} = A_k \delta \boldsymbol{x}_k + B_k \delta \boldsymbol{u}_k + \sqrt{\delta t}\Gamma_k \boldsymbol{\xi}_k + \boldsymbol{O}_d(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k, \boldsymbol{\xi}_k, \delta t) \\
& \boldsymbol{\xi_k} \sim \mathcal{N}(0, \sigma^2 I_{m\times m}) \\
& \boldsymbol{x_{k_0}} = \bar{\boldsymbol{x}}_{k_0}.
\end{aligned}$$

Bigger Picture

Solving the continuous-time SOCP is complex, therefore some simplifications are required:

1. Replace nonlinear system by a local approximation of the system.
2. Discretize the approximated system.
   $\hookrightarrow$ Discretize the SOCP.

Derivation of the SDDP scheme (1/6)

- Value function

$$V(\boldsymbol{x}_k) = \min_{\pi(\cdot,\cdot)} J^{\pi}(\boldsymbol{x}_k, k).$$

- State-action value function Q

$$Q(\boldsymbol{x}_k, \boldsymbol{u}_k) = l(\boldsymbol{x}_k, \boldsymbol{u}_k) + \mathbb{E}[V(\boldsymbol{x}_{k+1})]$$

- Discrete-time Bellman Equation for stochastic systems

$$\begin{aligned} V(\boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k} Q(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= \min_{\boldsymbol{u}_k} \{ l(\boldsymbol{x}_k, \boldsymbol{u}_k) + \mathbb{E}[V(\boldsymbol{x}_{k+1})] \} \end{aligned}$$

for $k = k_0, ..., k_0 + N - 1$ and $V(x_N) = \mathbb{E}\left[\Phi_N(x_N)\right]$.

Derivation of the SDDP scheme (2/6)

- Approximate the Q-function by computing second order Taylor expansion about a nominal trajectory pair $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$

$$Q(\bar{\boldsymbol{x}} + \delta \boldsymbol{x}, \bar{\boldsymbol{u}} + \delta \boldsymbol{u}) \approx \tilde{Q}(\bar{\boldsymbol{x}} + \delta \boldsymbol{x}, \bar{\boldsymbol{u}} + \delta \boldsymbol{u}) = \frac{1}{2} \begin{pmatrix} 1 \\ \delta \boldsymbol{x} \\ \delta \boldsymbol{u} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \bar{Q} & Q_{\boldsymbol{x}}^{\mathsf{T}} & Q_{\boldsymbol{u}}^{\mathsf{T}} \\ Q_{\boldsymbol{x}} & Q_{\boldsymbol{xx}} & Q_{\boldsymbol{xu}} \\ Q_{\boldsymbol{u}} & Q_{\boldsymbol{ux}} & Q_{\boldsymbol{uu}} \end{pmatrix} \begin{pmatrix} 1 \\ \delta \boldsymbol{x} \\ \delta \boldsymbol{u} \end{pmatrix}$$

  where $\bar{Q} = 2Q(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ is the zero order term of the expansion.

- Quadratic expansion of the Q-function reduces the solution of the Bellman equation at any step k to the minimization of a quadratic program.
  $\hookrightarrow$ Optimal control law can be obtained by setting the gradient w.r.t $\delta \boldsymbol{u}$ to zero

Assume $Q_{\boldsymbol{uu}}$ is positive definite, then the first-order necessary condition becomes sufficient

$$\nabla_{\delta \boldsymbol{u}} \tilde{Q}|_{\delta \boldsymbol{u}=\delta \boldsymbol{u}^*} = Q_{\boldsymbol{u}}^\mathsf{T} + \delta \boldsymbol{x}^\mathsf{T} Q_{\boldsymbol{xu}}^\mathsf{T} + \delta \boldsymbol{u}^{*\mathsf{T}} Q_{\boldsymbol{uu}}^\mathsf{T} \stackrel{!}{=} 0$$

Solving for the optimal update policy $\delta \boldsymbol{u}^*$ the optimal control variation at step k becomes

$$\boxed{\delta \boldsymbol{u}^* = -Q_{\boldsymbol{uu}}^{-1}(Q_{\boldsymbol{u}} + Q_{\boldsymbol{ux}}\delta \boldsymbol{x})}$$

Note: Optimal update policy in SDDP is identical to the optimal update policy in classical DDP.

How to compute the derivatives of the Q-function?

- Recall the definition of the Q-function

$$Q(\boldsymbol{x}_k, \boldsymbol{u}_k) = \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) + \mathbb{E}[V(\boldsymbol{x}_{k+1})].$$

- Idea: Compute second order Taylor expansion of $\ell(\boldsymbol{x}_k, \boldsymbol{u}_k)$ and $\mathbb{E}[V(\boldsymbol{x}_{k+1})]$ about a nominal trajectory pair $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ and equate the coefficients

$\hookrightarrow$ The expectation of the value function expanded to second order

$$\begin{aligned}\mathbb{E}\left[V(\bar{\boldsymbol{x}}_{t+\delta t} + \delta\boldsymbol{x}_{t+\delta t})\right] \approx \mathbb{E}\left[\tilde{V}(\bar{\boldsymbol{x}}_{t+\delta t} + \delta\boldsymbol{x}_{t+\delta t})\right] &= \mathbb{E}\left[V(\bar{\boldsymbol{x}}_{t+\delta t})\right] \\ &+ \mathbb{E}\left[V_{\boldsymbol{x}}^{\mathsf{T}}\delta\boldsymbol{x}_{t+\delta t}\right] + \mathbb{E}\left[\frac{1}{2}\delta\boldsymbol{x}_{t+\delta t}^{\mathsf{T}} V_{\boldsymbol{x}\boldsymbol{x}}\delta\boldsymbol{x}_{t+\delta t}\right].\end{aligned}$$

Derivation of the SDDP scheme (5/6)

Note: The expectation terms require a lengthy derivation and their proof is covered in detail in the report.

$\hookrightarrow$ The running cost is also expanded to second order resulting in

$$\ell(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}, \bar{\boldsymbol{u}}+\delta\boldsymbol{u}) \approx \tilde{\ell}(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}, \bar{\boldsymbol{u}} + \delta\boldsymbol{u}) = \frac{1}{2}\begin{pmatrix} 1 \\ \delta\boldsymbol{x} \\ \delta\boldsymbol{u} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \bar{\ell} & \ell_{\boldsymbol{x}}^{\mathsf{T}} & \ell_{\boldsymbol{u}}^{\mathsf{T}} \\ \ell_{\boldsymbol{x}} & \ell_{\boldsymbol{xx}} & \ell_{\boldsymbol{xu}} \\ \ell_{\boldsymbol{u}} & \ell_{\boldsymbol{ux}} & \ell_{\boldsymbol{uu}} \end{pmatrix} \begin{pmatrix} 1 \\ \delta\boldsymbol{x} \\ \delta\boldsymbol{u} \end{pmatrix}$$

After equating the coefficients, the Q-function derivatives can be obtained

$$\begin{aligned}
Q_{\boldsymbol{x}} &= \ell_{\boldsymbol{x}} + A_t V_{\boldsymbol{x}} + \tilde{\boldsymbol{\mathcal{S}}} \\
Q_{\boldsymbol{u}} &= \ell_{\boldsymbol{u}} + B_t V_{\boldsymbol{x}} + \tilde{\boldsymbol{\mathcal{U}}} \\
Q_{\boldsymbol{xx}} &= \ell_{\boldsymbol{xx}} + A_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t + \kappa\boldsymbol{\mathcal{F}} + \tilde{\boldsymbol{\mathcal{F}}} + \kappa\tilde{\boldsymbol{\mathcal{M}}} \\
Q_{\boldsymbol{xu}} &= \ell_{\boldsymbol{xu}} + A_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t + \kappa\boldsymbol{\mathcal{L}} + \tilde{\boldsymbol{\mathcal{L}}} + \kappa\tilde{\boldsymbol{\mathcal{N}}} \\
Q_{\boldsymbol{uu}} &= \ell_{\boldsymbol{uu}} + B_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t + \kappa\boldsymbol{\mathcal{Z}} + \tilde{\boldsymbol{\mathcal{Z}}} + \kappa\tilde{\boldsymbol{\mathcal{G}}}.
\end{aligned}$$

Derivation of the SDDP scheme (6/6)

The matrices are given here without proof

$$\boldsymbol{\mathcal{F}} = \left(\sum_{j=1}^{n} \nabla_{\boldsymbol{xx}} f^{(j)} V_{x_j}\right), \boldsymbol{\mathcal{Z}} = \left(\sum_{j=1}^{n} \nabla_{\boldsymbol{uu}} f^{(j)} V_{x_j}\right), \boldsymbol{\mathcal{L}} = \left(\sum_{j=1}^{n} \nabla_{\boldsymbol{ux}} f^{(j)} V_{x_j}\right)$$

$$\tilde{\boldsymbol{\mathcal{F}}} = \sigma^2 \delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{x}} F_c^{(i)\ T} V_{\boldsymbol{xx}} \nabla_{\boldsymbol{x}} F_c^{(i)}, \tilde{\boldsymbol{\mathcal{Z}}} = \sigma^2 \delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{u}} F_c^{(i)\ T} V_{\boldsymbol{xx}} \nabla_{\boldsymbol{u}} F_c^{(i)}$$

$$\tilde{\boldsymbol{\mathcal{L}}} = \sigma^2 \delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{x}} F_c^{(i)\ T} V_{\boldsymbol{xx}} \nabla_{\boldsymbol{u}} F_c^{(i)}, \tilde{\boldsymbol{\mathcal{S}}} = \sigma^2 \delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{x}} F_c^{(i)\ T} V_{\boldsymbol{xx}} F_c^{(i)}$$

$$\tilde{\boldsymbol{\mathcal{U}}} = \sigma^2 \delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{u}} F_c^{(i)\ T} V_{\boldsymbol{xx}} F_c^{(i)}, \tilde{\boldsymbol{\mathcal{M}}} = \delta t \sigma^2 \sum_{\lambda=1}^{n} \sum_{k=1}^{m} \left(\left(\sum_{r=1}^{n} V_{\boldsymbol{xx}}^{(k,r)} F^{(r,\lambda)}\right) F_{\boldsymbol{xx}}^{(k\lambda)}\right)$$

$$\tilde{\boldsymbol{\mathcal{G}}} = \delta t \sigma^2 \sum_{\lambda=1}^{n} \sum_{k=1}^{m} \left(\left(\sum_{r=1}^{n} V_{\boldsymbol{xx}}^{(k,r)} F^{(r,\lambda)}\right) F_{\boldsymbol{uu}}^{(k\lambda)}\right)$$

$$\tilde{\boldsymbol{\mathcal{N}}} = \delta t \sigma^2 \sum_{\lambda=1}^{n} \sum_{k=1}^{m} \left(\left(\sum_{r=1}^{n} V_{\boldsymbol{xx}}^{(k,r)} F^{(r,\lambda)}\right) F_{\boldsymbol{xu}}^{(k\lambda)}\right)$$

Algorithm Pseudocode of the SDDP Algorithm

Given: $\{\boldsymbol{u}_{\mathbf{0}}^{k}\}_{k=k_0}^{k_0+N-1}$

repeat

- **Backward-Pass**: Compute the approximation of the value function in a back-propagation fashion and the optimal control variation

$$\boxed{\delta \boldsymbol{u}_k^* = -Q_{\boldsymbol{uu}}^{-1}(Q_{\boldsymbol{u}} + Q_{\boldsymbol{ux}}\delta \boldsymbol{x}_k)}$$

- Update control policy (**Step-size control**) $\boldsymbol{u}^+ = \boldsymbol{u}^* - \alpha \cdot Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{u}} - Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{ux}} \delta \boldsymbol{x}$
- **Forward-Pass**: Roll out the system dynamics utilizing $\boldsymbol{u}^*$ to obtain a new trajectory $\boldsymbol{x}^+$.
- Update the trajectories $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}) = (\boldsymbol{x}^+, \boldsymbol{u}^+)$

until *Convergence*;

Figure: Pseudocode of the SDDP algorithm.

Handling the challenges of nonlinear optimization (1/2)

Problem: Decrease of the cost function in every iteration is not guaranteed!
Remedy 1: Line-search scheme

- General Idea: Choose the step-size α_k sufficiently small, until a decrease in the cost function is observed.
  $\hookrightarrow$ Best performing line-search scheme: Backtracking line-search

Algorithm Pseudocode of the Line-search Algorithm

$\alpha = 1;$
repeat
- Forward Pass: $\boldsymbol{u}^{+} = \boldsymbol{u}^{*} - \alpha \cdot Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{u}} - Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{ux}} \delta \boldsymbol{x}$
- Evaluate J
- Backtracking: $\alpha = \rho \cdot \alpha;$

until $\Delta J < 0;$

Is the line-search scheme sufficient to guarantee convergence?

Handling the challenges of nonlinear optimization (2/2)

No. When computing the minimum, we assume the Hessian $Q_{\boldsymbol{uu}}$ is positive definite. This condition is equivalent to guaranteeing a descent direction in Newton's method!
$\hookrightarrow$ Additional terms arising from second-order dynamics could render the Hessian $Q_{\boldsymbol{uu}}$ not positive definite.
Remedy 2: Regularization scheme

- General idea of Levenberg-Marquardt schemes: Add an identity matrix scaled with a sufficiently large parameter μ_k to the Hessian $Q_{\boldsymbol{uu}}$
- Best convergence was found using the proposed regularization scheme[6].
- Key difference: Penalize deviations from the states AND the control inputs.

[6] Yuval Tassa, Tom Erez, and Emanuel Todorov. "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 4906–4913. DOI: 10.1109/IROS.2012.6386025.

Examples: Simple Inverted Pendulum (1/7)

2 DOF model of an inverted pendulum

$$dx = \begin{pmatrix} x_2 \\ 4\sin(x_1) + u \end{pmatrix} dt + \begin{pmatrix} 0 \\ \beta u \end{pmatrix} dw,$$

where the first state denotes the angle $x_1 = \theta$, the second state denotes the turn-rate $x_2 = \dot{\theta}$, and β represents the measurement noise parameter.
Goal: Find a control input sequence to steer the inverted pendulum from the suspended state ($\theta = -\pi$ rad) to the swung-up state ($\theta = 0$ rad) in $T = 4$ s .

- Cost function: $J^{\pi}(\boldsymbol{x}, t) = \mathbb{E}\left[\int_{t=0}^{T} R \cdot u(\tau)^2 \, d\tau + \phi_N(\boldsymbol{x}(T))\right]$, where $R = 10^{-2}$.
- Terminal cost: $\phi_N(\boldsymbol{x}(T)) = (x(T) - x_{des,T})^T \cdot Q_f \cdot (x(T) - x_{des,T})$, where $Q_f = \begin{pmatrix} 10 & 0 \\ 0 & 0 \end{pmatrix}$.

Time step was chosen to be $\delta t = 20$ ms.

Examples: Simple Inverted Pendulum (2/7)

Optimal Trajectories

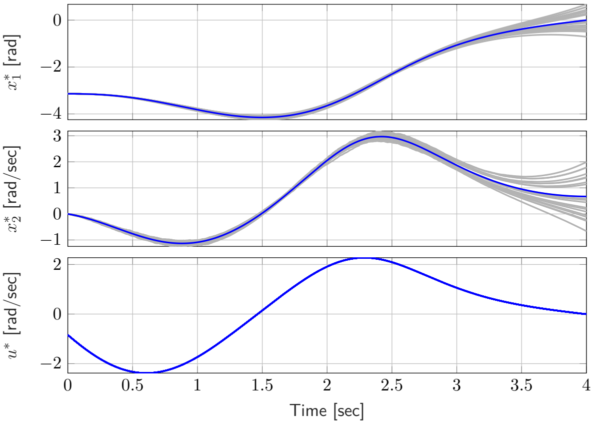

Figure: Multiple realizations of the optimal states trajectories (grey) and the optimal control trajectory for the simple inverted pendulum model. The state trajectories in blue is the mean over 500 samples.

Examples: Simple Inverted Pendulum (3/7)

Convergence plot

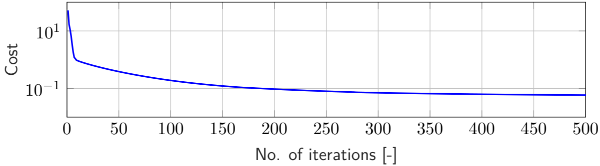

Figure: Convergence plot for the simple inverted pendulum.

Examples: Simple Parafoil (4/7)

Consider the simplified 4 DOF model representing the parafoil dynamics

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \frac{dx_3}{dt} \\ \frac{dx_4}{dt} \end{bmatrix} = \begin{bmatrix} v \cdot cos(x_3(t)) \\ v \cdot sin(x_3(t)) \\ u(t) \\ -r \end{bmatrix},$$

where $x_1(t) = x(t)$, $x_2(t) = y(t)$, $x_3(t) = \theta(t)$ and $x_4(t) = z(t)$.
v denotes the horizontal velocity. For realistic values, we assume $\mathsf{v} = 15\,\frac{\mathsf{m}}{\mathsf{s}}$.
Consider the reduced order model with wind gusts $w_x(t)$ and $w_y(t)$ as stochastic disturbances

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \frac{dx_3}{dt} \end{bmatrix} = \begin{bmatrix} v \cdot cos(x_3(t)) + w_x(t) \\ v \cdot sin(x_3(t)) + w_y(t) \\ u(t) \end{bmatrix}.$$

Wind gusts $w_x(t)$ and $w_y(t)$ are generated by second order Dryden wind turbulence model.

Examples: Simple Parafoil (5/7)

Power Spectrum Density of the turbulence model

$$\Phi(\omega) = \frac{\sigma^2 L}{\pi v} \cdot \frac{1 + 3 \cdot \left(\frac{L}{v}\omega\right)^2}{\left(1 + \left(\frac{L}{v}\omega\right)^2\right)^2},$$

with σ^2 turbulence intensity and L turbulence length scale.

Instances of the wind profiles generated by the turbulence model

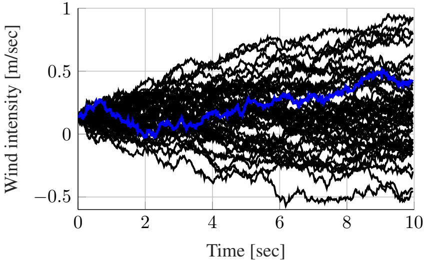

Figure: 100 instances of wind gusts generated by the Dryden filter.

Examples: Simple Parafoil (6/7)

After augmenting both filters in the overall model, a 7 DOF model is obtained

$$\begin{bmatrix} dx_1 \\ dx_2 \\ dx_3 \\ dx_4 \\ dx_5 \\ dx_6 \\ dx_7 \end{bmatrix} = \begin{bmatrix} v \cdot \cos(x_3(t)) + C_{\xi_{1,1}} x_4(t) + C_{\xi_{1,2}} x_5(t) \\ v \cdot \sin(x_3(t)) + C_{\xi_{1,1}} x_6(t) + C_{\xi_{1,2}} x_7(t) \\ u(t) \\ A_{\xi_{1,1}} x_4(t) + A_{\xi_{1,2}} x_5(t) \\ A_{\xi_{2,1}} x_4(t) + A_{\xi_{2,2}} x_5(t) \\ A_{\xi_{1,1}} x_6(t) + A_{\xi_{1,2}} x_7(t) \\ A_{\xi_{2,1}} x_6(t) + A_{\xi_{2,2}} x_7(t) \end{bmatrix} \delta t + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ B_{\xi,1} & 0 \\ B_{\xi,2} & 0 \\ 0 & B_{\xi,1} \\ 0 & B_{\xi,2} \end{bmatrix} dw.$$

Goal: Find a control input sequence to land the parafoil from initial position ($x_0 = 0$, $y_0 = 0$) at a desired position ($x_f = 0$, $y_f = 100$) in T $= 100\,$s.

- Cost function: $J^\pi(\boldsymbol{x}, t) = \mathbb{E}\left[\int_{t=0}^{T} R \cdot u(\tau)^2 \, d\tau + \phi_N(\boldsymbol{x}(T))\right]$, where $R = 10^{-2}$.
- Terminal cost: $\phi_N(\boldsymbol{x}(T)) = (x(T) - x_{des,T})^T \cdot Q_f \cdot (x(T) - x_{des,T})$.

Optimal Trajectories

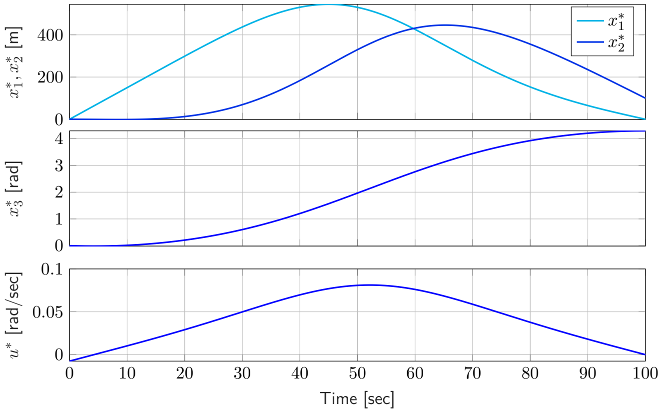

Figure: Upper plot presents the mean of the states x_1 and x_2 indicating the location in the x-y-plane. Middle plot shows the mean of the state x_3 representing the angle. Bottom plot exhibits the optimal turn rate.

Open challenges and Limitations

- Future disturbances are disregarded in SDDP framework.
  $\hookrightarrow$ SDDP optimizes only for the current disturbance.
- For a constant stochastic dynamics $F(x, u) = F$, SDDP gets reduced to classical DDP.
  $\hookrightarrow$ SDDP remains **"blind"** to additive noise no matter the magnitude.
- Author claims that the cubic and quartic terms cancel out. In fact, these higher order terms are neglected.
- Optimization framework and the approximation of system dynamics are treated as distinct components. This is particularly challenging as optimization framework does not allow for uncertainty in the system.
  $\hookrightarrow$ A unified approach towards uncertainty optimization

Questions

Thank You!