Stochastic Differential Dynamic Programming

Adel Abdelsamed*

ABSTRACT– Extending classical Differential Dynamic Programming (DDP) towards stochastic systems, where the maturity of such extensions is yet to be established, is highly sought-after. This report analyzes the proposed Stochastic Differential Dynamic Programming (SDDP) by Theodorou et al. [8] as a means to handle general stochastic systems. Relevant work related to stochastic systems is reviewed concisely. After deriving the classical DDP, we proceed to extend it to stochastic systems. Thorough computation of correction terms arising from the stochastic assumption is performed. Given the proposed quadratic approximation of the nonlinear dynamics, the implementation of SDDP heavily relies on line-search techniques and regularization. A simple backtracking line-search algorithm is elaborated. In light of the minimal attention to regularization in the literature, a brief review of present regularization techniques is also provided. The algorithm is demonstrated on three models with varying degrees of complexity, including a 7-DOF parafoil homing trajectory optimization problem. Lastly, a theoretical analysis is conducted to examine the main limitations of SDDP and suggests possible improvements.

I. INTRODUCTION

Having its origin back in the 1970's [1], Differential Dynamic Programming (DDP) remains one of the go-to methods in trajectory optimization, making it a fruitful research topic to this day. DDP attempts to solve general nonlinear programs, while utilizing the dynamic programming framework. It generates locally optimal linear feedback and feedforward controllers in the vicinity of a nominal trajectory. Owing to its quadratic convergence [2],[5] as well as numerical stability [4] the DDP is not affected by the curse of dimensionality.

Hence variations to the classical DDP are desired to extend the capabilities of DDP beyond deterministic systems. In this report we investigate a modification of the classical DDP to incorporate general state and control multiplicative stochastic disturbances. Hence, we provide here a brief comparative analysis of noteworthy work related to the integration of stochastic disturbances into the DDP framework.

The general DDP algorithm expands the cost-to-go function locally around a nominal trajectory to second order. Based on the linearization of the system dynamics different variations exist. An iterative Linear-Quadratic-Gaussian (iLQG) regulator [9] exploits linearized first order approximations of the system dynamics to obtain locally optimal controllers. The Stochastic DDP (SDDP) method [8] analyzed in this report extends the approximation of the system dynamics to second order while retaining the quadratic structure of the classical DDP algorithm. Both approaches assume perfect knowledge of the system dynamics in form of a Wiener-driven stochastic differential equation (WDSDE). Accounting for model uncertainties, a data-driven approach was proposed in the works by Pan et al. [6]. The presented Probabilistic DDP (PDDP) learns the system dynamics online using a Gaussian Process Regression (GPR), thereby transitioning from the state space to the Gaussian belief space. Similarly, first order approximations are generated around a nominal state distribution and control pair. The comparison is summarized in Table 1.

This report adheres to the following outline. The second section provides an overview of the optimal control problem under consideration. The third section presents the linearization and discretization of the system dynamics. In the fourth section, we introduce a modification of the deterministic DDP to incorporate stochastic disturbances and derive the optimal control law. In section five, implementational aspects of the algorithm are discussed and the SDDP is showcased on three different applications with varying complexities. In section six, limitations of the SDDP algorithm are analyzed followed by a conclusion in section seven.

II. PROBLEM STATEMENT

We consider the general class of nonlinear stochastic systems modeled using the following Wiener-driven stochastic

*Email: adel.abdelsamed@rwth-aachen.de

	iLQG [9]	SDDP [8]	PDDP [6]
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

Table 1: Comparison of existing works aiming to incorporate stochastic disturbances into DDP.

differential equation

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

where $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ denotes the state vector, $\boldsymbol{u} \in \mathbb{R}^{p\times 1}$ the control input vector and $dw \in \mathbb{R}^{m\times 1}$ the Brownian noise. $f(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{n\times 1}$, models the deterministic dynamics, while $F(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{n\times m}$ captures the stochastic nature of the system.
Furthermore, we define the cost-to-go $J^\pi(\boldsymbol{x}, t)$ as the accumulated expectation of the running cost $\ell(t, \boldsymbol{x}(t), \pi(t, \boldsymbol{x}(t)))$ starting from initial state $\boldsymbol{x}(t)$ subject to the control policy $\pi(t, \boldsymbol{x}(t))$ over the time horizon $(t, ..., T)$

$$J^\pi(\boldsymbol{x}, t) = \mathbb{E}\left[\int_t^T \ell(\tau, \boldsymbol{x}(\tau), \pi(\tau, \boldsymbol{x}(\tau)))\, d\tau + \phi_N(\boldsymbol{x}(T))\right]. \tag{2}$$

The objective of the finite time-horizon optimal control problem is to find the optimal control policy $\pi^*(\cdot, \cdot)$ that minimizes the following objective function $J^\pi(\boldsymbol{x}, t_0)$ subject to the system dynamics in Eqn.(1).

III. Linearization and Discretization of System Dynamics

In order to avoid tensor contraction terms in our derivations we introduce the following notation:

- The matrix $F(\boldsymbol{x}, \boldsymbol{u})$ representing stochastic dynamics can either be written as a collection of row or column vectors

$$F(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} {F_r}^1(\boldsymbol{x}, \boldsymbol{u}) \\ \vdots \\ {F_r}^n(\boldsymbol{x}, \boldsymbol{u}) \end{pmatrix} = \left({F_c}^1(\boldsymbol{x}, \boldsymbol{u}), \ldots, {F_r}^n(\boldsymbol{x}, \boldsymbol{u})\right). \tag{3}$$

- A new function $\Phi(\boldsymbol{x}, \boldsymbol{u}, dw) \in \mathbb{R}^{n\times 1}$ is defined, which is equivalent to the RHS of Eqn.(1)

$$\Phi(\boldsymbol{x}, \boldsymbol{u}, dw) \equiv f(\boldsymbol{x}, \boldsymbol{u})dt + F(\boldsymbol{x}, \boldsymbol{u})dw. \tag{4}$$

Using the newly defined function, the j-th element of the $\Phi(\boldsymbol{x}, \boldsymbol{u}, dw)$-vector is expressed as

$$\Phi^{(j)}(\boldsymbol{x}, \boldsymbol{u}, dw) = f^{(j)}(\boldsymbol{x}, \boldsymbol{u})dt + {F_r}^{(j)}(\boldsymbol{x}, \boldsymbol{u})dw. \tag{5}$$

A. Linearization

For the imminent linearization we introduce state $\delta\boldsymbol{x} = \boldsymbol{x} - \bar{\boldsymbol{x}}$ and control deviations $\delta\boldsymbol{u} = \boldsymbol{u} - \bar{\boldsymbol{u}}$ about a nominal trajectory $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ and expand the dynamics up to second order

$$\Phi(\bar{\boldsymbol{x}} + \boldsymbol{x}, \bar{\boldsymbol{u}} + \boldsymbol{u}, dw) \approx \Phi(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, dw) + \nabla_{\boldsymbol{x}}\Phi \cdot \delta\boldsymbol{x} + \nabla_{\boldsymbol{u}}\Phi \cdot \delta\boldsymbol{u} + \boldsymbol{O}(\delta\boldsymbol{x}, \delta\boldsymbol{u}, dw), \tag{6}$$

where $\boldsymbol{O}(\delta\boldsymbol{x}, \delta\boldsymbol{u}, dw) \in \mathbb{R}^{n\times 1}$ contains the second-order dynamics and each element can be expressed as

$$O^{(j)}(\delta\boldsymbol{x}, \delta\boldsymbol{u}, dw) = \frac{1}{2}\begin{pmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{pmatrix}^\mathsf{T}\begin{pmatrix}\nabla_{\boldsymbol{xx}}\Phi^j & \nabla_{\boldsymbol{xu}}\Phi^j \\ \nabla_{\boldsymbol{ux}}\Phi^j & \nabla_{\boldsymbol{uu}}\Phi^j\end{pmatrix}\begin{pmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{pmatrix}. \tag{7}$$

To conclude the linearization we specify the first-order derivatives appearing in Eqn.(6)

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

as well as the second-order derivatives in Eqn.(7)

$$\nabla_{\boldsymbol{xx}}\Phi^{(j)} = \nabla_{\boldsymbol{xx}} f^{(j)}(\boldsymbol{x}, \boldsymbol{u})dt + \nabla_{\boldsymbol{xx}}\left({F_r}^{(j)}(\boldsymbol{x}, \boldsymbol{u})dw\right)$$
$$\nabla_{\boldsymbol{uu}}\Phi^{(j)} = \nabla_{\boldsymbol{uu}} f^{(j)}(\boldsymbol{x}, \boldsymbol{u})dt + \nabla_{\boldsymbol{uu}}\left({F_r}^{(j)}(\boldsymbol{x}, \boldsymbol{u})dw\right)$$
$$\nabla_{\boldsymbol{xu}}\Phi^{(j)} = \nabla_{\boldsymbol{xu}} f^{(j)}(\boldsymbol{x}, \boldsymbol{u})dt + \nabla_{\boldsymbol{xu}}\left({F_r}^{(j)}(\boldsymbol{x}, \boldsymbol{u})dw\right)$$
$$\nabla_{\boldsymbol{ux}}\Phi^{(j)} = \nabla_{\boldsymbol{ux}} f^{(j)}(\boldsymbol{x}, \boldsymbol{u})dt + \nabla_{\boldsymbol{ux}}\left({F_r}^{(j)}(\boldsymbol{x}, \boldsymbol{u})dw\right).$$

B. Discretization

After linearizing the dynamics about a nominal trajectory, the Euler-Maruyama discretization scheme is made use of to obtain a discrete-time representation of the system. Thus, the discretization of the deterministic dynamics correspond to the forward Euler scheme shown in

$$\delta\dot{\boldsymbol{x}} = \frac{\delta\boldsymbol{x}_{t+\delta t} - \delta\boldsymbol{x}_t}{\delta t} \tag{8}$$

with the discretization interval $\delta t = t_{k+1} - t_k$ chosen to be sufficiently small. This discretization scheme results in the discrete-time dynamics

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

where $\boldsymbol{\xi}_t \in \mathbb{R}^{m\times 1}$ is sampled from a normal distribution with mean zero and covariance matrix $\Sigma = \sigma^2 I_{m\times m}$. It is noted that the term $F(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\sqrt{\delta t}$ constitutes the first derivative of $\Phi(\boldsymbol{x},\boldsymbol{u},dw)$ w.r.t the Brownian motion noise dw. The $\sqrt{\delta t}$ term appears due to the fact that the variance of the Brownian motion noise grows linearly with time. Additionally, the term $\boldsymbol{O}_d(\delta \boldsymbol{x}_{\mathrm{t}}, \delta \boldsymbol{u}_{\mathrm{t}}, \boldsymbol{\xi}_t, \delta t)$ is considered the vector of discretized second-order dynamics. Each element of this vector is given as in Eqn.(7) but with discretized second-order derivatives

$$\begin{aligned}\nabla_{\boldsymbol{xx}}\Phi_d^{(j)} &= \nabla_{\boldsymbol{xx}} f^j(\boldsymbol{x},\boldsymbol{u})\delta t + \nabla_{\boldsymbol{xx}}\left({F_r}^j(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\right)\sqrt{\delta t} \\ \nabla_{\boldsymbol{uu}}\Phi_d^{(j)} &= \nabla_{\boldsymbol{uu}} f^j(\boldsymbol{x},\boldsymbol{u})\delta t + \nabla_{\boldsymbol{uu}}\left({F_r}^j(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\right)\sqrt{\delta t} \\ \nabla_{\boldsymbol{xu}}\Phi_d^{(j)} &= \nabla_{\boldsymbol{xu}} f^j(\boldsymbol{x},\boldsymbol{u})\delta t + \nabla_{\boldsymbol{xu}}\left({F_r}^j(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\right)\sqrt{\delta t} \\ \nabla_{\boldsymbol{ux}}\Phi_d^{(j)} &= \nabla_{\boldsymbol{ux}} f^j(\boldsymbol{x},\boldsymbol{u})\delta t + \nabla_{\boldsymbol{ux}}\left({F_r}^j(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\right)\sqrt{\delta t}.\end{aligned}$$

It is now possible to express the discrete-time linearized dynamics in condensed form

$$\delta \boldsymbol{x}_{t+\delta t} = A_t\delta \boldsymbol{x}_{\mathrm{t}} + B_t\delta \boldsymbol{u}_{\mathrm{t}} + \sqrt{\delta t}\Gamma_t\boldsymbol{\xi}_t + \boldsymbol{O}_d(\delta \boldsymbol{x}_{\mathrm{t}}, \delta \boldsymbol{u}_{\mathrm{t}}, \boldsymbol{\xi}_t, \delta t) \tag{9}$$

by defining new matrices $A_t \in \mathbb{R}^{n\times n}$, $B_t \in \mathbb{R}^{n\times p}$ and $\Gamma_t \in \mathbb{R}^{n\times m}$

$$\begin{aligned}A_t &= I_{n\times n} + \nabla_{\boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{u})\delta t \\ B_t &= \nabla_{\boldsymbol{u}} f(\boldsymbol{x},\boldsymbol{u})\delta t \\ \Gamma_t &= \left[\Gamma^{(1)}\ \Gamma^{(2)}\ \dots\ \Gamma^{(m)}\right]\end{aligned}$$

, where $\Gamma^{(i)} = \nabla_{\boldsymbol{x}}{F_c}^{(i)}\delta \boldsymbol{x}_{\mathrm{t}} + \nabla_{\boldsymbol{u}}{F_c}^{(i)}\delta \boldsymbol{u}_{\mathrm{t}} + F_c^{(i)}$. For subsequent derivations we dissect the matrix capturing the stochastic dynamics Γ_t into a term

$$\Gamma_t = \Delta_t(\delta \boldsymbol{x}_{\mathrm{t}}, \delta \boldsymbol{u}_{\mathrm{t}}) + F(\boldsymbol{x},\boldsymbol{u}) \tag{10}$$

that is dependent on variations of control and state and a second term independent thereof.

IV. Stochastic Differential Dynamic Programming

A. Optimal Control Policy

First we define the value function $V(\boldsymbol{x}_k)$ as the cost-to-go computed from a fixed state $\boldsymbol{x}_k$ and under the optimal control sequence $\pi^*(\cdot,\cdot)$. The value function constitutes a theoretical minimum over all possible control sequences

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

Furthermore, we have the state-action value function Q for stochastic systems as

$$\begin{aligned}Q(\boldsymbol{x}_k,\boldsymbol{u}_k) &= \textit{immediate cost + expected value of next state} \\ &= l(\boldsymbol{x}_k,\boldsymbol{u}_k) + \mathbb{E}[V(\boldsymbol{x}_{k+1})].\end{aligned}$$

Now we can express the discrete-time Bellman equation for stochastic systems as

$$\begin{aligned}V(\boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k} Q(\boldsymbol{x}_k,\boldsymbol{u}_k) \\ &= \min_{\boldsymbol{u}_k}\{l(\boldsymbol{x}_k,\boldsymbol{u}_k) + \mathbb{E}[V(\boldsymbol{x}_{k+1})]\}.\end{aligned} \tag{11}$$

It is noted here that in the DDP framework we minimize over variations in control. As DDP is a second-order method we compute the Taylor expansion of the Q-function $\tilde{Q}$ about a nominal trajectory $(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}})$ up to second order

$$\begin{aligned}Q(\bar{\boldsymbol{x}}+\delta \boldsymbol{x}, \bar{\boldsymbol{u}}+\delta \boldsymbol{u}) \approx \tilde{Q}(\bar{\boldsymbol{x}}+\delta \boldsymbol{x}, \bar{\boldsymbol{u}}+\delta \boldsymbol{u}) = \\ \frac{1}{2}\begin{pmatrix}1\\ \delta \boldsymbol{x}\\ \delta \boldsymbol{u}\end{pmatrix}^{\mathsf{T}}\begin{pmatrix}\bar{Q} & Q_{\boldsymbol{x}}^{\mathsf{T}} & Q_{\boldsymbol{u}}^{\mathsf{T}}\\ Q_{\boldsymbol{x}} & Q_{\boldsymbol{xx}} & Q_{\boldsymbol{xu}}\\ Q_{\boldsymbol{u}} & Q_{\boldsymbol{ux}} & Q_{\boldsymbol{uu}}\end{pmatrix}\begin{pmatrix}1\\ \delta \boldsymbol{x}\\ \delta \boldsymbol{u}\end{pmatrix},\end{aligned} \tag{12}$$

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

The quadratic approximation of the Q-function reduces the solution of the Bellman equation to the minimization of a quadratic program. Thus, the optimal control law can be obtained by setting the gradient of Eqn.(12) w.r.t $\delta \boldsymbol{u}$ to zero

$$\nabla_{\delta \boldsymbol{u}}\tilde{Q}|_{\delta \boldsymbol{u}=\delta \boldsymbol{u}^*} = Q_{\boldsymbol{u}}^{\mathsf{T}} + \delta \boldsymbol{x}^{\mathsf{T}}Q_{\boldsymbol{xu}}^{\mathsf{T}} + \delta \boldsymbol{u}^{*\mathsf{T}}Q_{\boldsymbol{uu}}^{\mathsf{T}} \stackrel{!}{=} 0 \tag{13}$$

Solving Eqn.(13) for $\delta \boldsymbol{u}^*$ results in the optimal update policy

$$\boxed{\delta \boldsymbol{u}^* = -Q_{\boldsymbol{uu}}^{-1}(Q_{\boldsymbol{u}} + Q_{\boldsymbol{ux}}\delta \boldsymbol{x})}. \tag{14}$$

The optimal update policy is hence linear w.r.t state deviations and is composed of a feed-forward as well as a feed-back term. It is notable that for the stochastic DDP we retain the exact structure of the optimal control policy as in the deterministic DDP.
We leave the derivation of the backward-pass equations to Appendix A and provide a pseudocode of the stochastic DDP algorithm in Algorithm 1.

B. Derivation of Q-function derivatives

The optimal control policy in the previous subsection is given in terms of derivatives of the Q-function, which remain unknown. The derivatives of the Q-function are provided at this stage, with the proof for their derivation to follow

$$Q_{\boldsymbol{x}} = \ell_{\boldsymbol{x}} + A_t V_{\boldsymbol{x}} + \tilde{\boldsymbol{\mathcal{S}}} \tag{15}$$

$$Q_{\boldsymbol{u}} = \ell_{\boldsymbol{u}} + B_t V_{\boldsymbol{x}} + \tilde{\boldsymbol{\mathcal{U}}} \tag{16}$$

$$Q_{\boldsymbol{xx}} = \ell_{\boldsymbol{xx}} + A_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t + \kappa \boldsymbol{\mathcal{F}} + \tilde{\boldsymbol{\mathcal{F}}} + \kappa \tilde{\boldsymbol{\mathcal{M}}} \tag{17}$$

$$Q_{\boldsymbol{xu}} = \ell_{\boldsymbol{xu}} + A_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t + \kappa \boldsymbol{\mathcal{L}} + \tilde{\boldsymbol{\mathcal{L}}} + \kappa \tilde{\boldsymbol{\mathcal{N}}} \tag{18}$$

$$Q_{\boldsymbol{uu}} = \ell_{\boldsymbol{uu}} + B_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t + \kappa \boldsymbol{\mathcal{Z}} + \tilde{\boldsymbol{\mathcal{Z}}} + \kappa \tilde{\boldsymbol{\mathcal{G}}}. \tag{19}$$

For $\kappa = 1$ we have the second-order accurate SDDP algorithm, for $\kappa = 0$ we have the iLQG. Insofar this parameter unifies both results. At this point, we utilize the Bellman equation once again and expand the running cost as well as the expected value of the next state to second order. By equating the coefficients between the expanded Q-function and the expanded RHS of the discrete Bellman equation, an expression for the derivatives of the Q-function can be obtained. Expanding the running cost to second order results in

$$\ell(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}, \bar{\boldsymbol{u}} + \delta\boldsymbol{u}) \approx \tilde{\ell}(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}, \bar{\boldsymbol{u}} + \delta\boldsymbol{u}) = \frac{1}{2} \begin{pmatrix} 1 \\ \delta\boldsymbol{x} \\ \delta\boldsymbol{u} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \bar{\ell} & \ell_{\boldsymbol{x}}^{\mathsf{T}} & \ell_{\boldsymbol{u}}^{\mathsf{T}} \\ \ell_{\boldsymbol{x}} & \ell_{\boldsymbol{xx}} & \ell_{\boldsymbol{xu}} \\ \ell_{\boldsymbol{u}} & \ell_{\boldsymbol{ux}} & \ell_{\boldsymbol{uu}} \end{pmatrix} \begin{pmatrix} 1 \\ \delta\boldsymbol{x} \\ \delta\boldsymbol{u} \end{pmatrix} \tag{20}$$

, where again $\bar{\ell} = 2\ell(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ is the zero order term of the expansion.
Similarly, the value function is expanded to second order around $\bar{\boldsymbol{x}}$ yielding

$$V(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}) \approx \tilde{V}(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}) = V(\bar{\boldsymbol{x}}) + V_{\boldsymbol{x}}^{\mathsf{T}} \delta\boldsymbol{x} + \frac{1}{2} \delta\boldsymbol{x}^{\mathsf{T}} V_{\boldsymbol{xx}} \delta\boldsymbol{x}. \tag{21}$$

As the system dynamics are stochastic, it is required to compute the expectation of Eqn.(21) for the state $\boldsymbol{x}_{t+\delta t}$ in order to apply the Bellman Eqn.(11). Therefore, the linearized system dynamics from Eqn.(**??**) are substituted for $\boldsymbol{x}_{t+\delta t}$ and we get

$$\begin{aligned} &\mathbb{E}[\tilde{V}(\bar{\boldsymbol{x}}_{t+\delta t} + \delta\boldsymbol{x}_{t+\delta t})] = \mathbb{E}[V(\bar{\boldsymbol{x}}_{t+\delta t})] + \\ &\mathbb{E}[V_{\boldsymbol{x}}^{\mathsf{T}}(A_t \delta\boldsymbol{x}_{\mathfrak{t}} + B_t \delta\boldsymbol{u}_{\mathfrak{t}} + \Gamma_t \boldsymbol{\xi}_t + \boldsymbol{O}_d))] + \\ &\frac{1}{2}\mathbb{E}[(A_t \delta\boldsymbol{x}_{\mathfrak{t}} + B_t \delta\boldsymbol{u}_{\mathfrak{t}} + \Gamma_t \boldsymbol{\xi}_t + \boldsymbol{O}_d)^{\mathsf{T}} V_{\boldsymbol{xx}} \\ &\times (A_t \delta\boldsymbol{x}_{\mathfrak{t}} + B_t \delta\boldsymbol{u}_{\mathfrak{t}} + \Gamma_t \boldsymbol{\xi}_t + \boldsymbol{O}_d)]. \end{aligned} \tag{22}$$

The ensuing section is thus focused on computing the expectation of the first $\mathbb{E}[V_{\boldsymbol{x}}^{\mathsf{T}} \delta\boldsymbol{x}_{t+\delta t}]$ and second order $\mathbb{E}[\delta\boldsymbol{x}_{t+\delta t}^{\mathsf{T}} V_{\boldsymbol{xx}} \delta\boldsymbol{x}_{t+\delta t}]$ approximation of the cost-to-go.

Algorithm 1 Pseudocode of the SDDP Algorithm

repeat

- Compute matrices of the linearized dynamics A_t, B_t, Γ_t around nominal trajectory $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$
- Compute the approximated running cost function $\bar{\ell}, \ell_{\boldsymbol{x}}, \ell_{\boldsymbol{u}}, \ell_{\boldsymbol{xx}}, \ell_{\boldsymbol{uu}}, \ell_{\boldsymbol{xu}}$
- Compute the approximated Q-function $Q_{\boldsymbol{x}}, Q_{\boldsymbol{u}}, Q_{\boldsymbol{xx}}, Q_{\boldsymbol{uu}}, Q_{\boldsymbol{xu}}$
- **Backward-Pass**: Compute the approximation of the value function in a back-propagation fashion

$$\begin{aligned} V_0^{(k+1)} &= V_0^{(k+1)} - \frac{1}{2} Q_{\boldsymbol{u}} Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{u}} \\ V_{\boldsymbol{x}}^{(k+1)} &= Q_{\boldsymbol{x}} - Q_{\boldsymbol{xu}} Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{u}} \\ V_{\boldsymbol{xx}}^{(k+1)} &= Q_{\boldsymbol{xx}} - Q_{\boldsymbol{xu}} Q_{\boldsymbol{uu}}^{-1} Q_{\boldsymbol{ux}} \end{aligned}$$

- Compute optimal control policy $\delta\boldsymbol{u}^* = -Q_{\boldsymbol{uu}}^{-1}(Q_{\boldsymbol{u}} + Q_{\boldsymbol{ux}} \delta\boldsymbol{x})$
- 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 in Eqn.(1) 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*;

C. Expectation of the Cost-to-go Function Approximation

This subsection is divided into the computation of the first and second order expectation arising in Eqn.(22).

C.1. Computing the expectation of the first order term of the cost-to-go $\mathbb{E}[V_{\boldsymbol{x}}^{\mathsf{T}} \delta\boldsymbol{x}_{t+\delta t}]$

After substituting the linearized dynamics for $\delta\boldsymbol{x}_{t+\delta t}$ in Eqn.(21) and taking the expectation thereof, the first order term can be written as

$$\begin{aligned} &\mathbb{E}[V_{\boldsymbol{x}}^{\mathsf{T}}(A_t \delta\boldsymbol{x}_{\mathfrak{t}} + B_t \delta\boldsymbol{u}_{\mathfrak{t}} + \Gamma_t \boldsymbol{\xi}_t + \boldsymbol{O}_d)] = \\ &V_{\boldsymbol{x}}^{\mathsf{T}}(A_t \delta\boldsymbol{x}_{\mathfrak{t}} + B_t \delta\boldsymbol{u}_{\mathfrak{t}} + \mathbb{E}[\boldsymbol{O}_d]), \end{aligned} \tag{23}$$

where the term $V_{\boldsymbol{x}}^{\mathsf{T}} \mathbb{E}[\Gamma_t \boldsymbol{\xi}_t]$ vanishes due to the fact that the noise is defined to have zero mean $\mathbb{E}[\boldsymbol{\xi}_t] = 0$. Therefore, it remains to compute the expectation of the discretized second order dynamics $\mathbb{E}[\boldsymbol{O}_d]$. The expectation of the j-th element of the column vector $\boldsymbol{O}_d$ corresponds to

$$\begin{aligned} &\mathbb{E}[\boldsymbol{O}_d^{(j)}(\delta\boldsymbol{x}_t, \delta\boldsymbol{u}_t, \xi_t, \delta t)] = \\ &\mathbb{E}\left[\frac{1}{2} \begin{pmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \nabla_{\boldsymbol{xx}} \Phi_d^{(j)} & \nabla_{\boldsymbol{xu}} \Phi_d^{(j)} \\ \nabla_{\boldsymbol{ux}} \Phi_d^{(j)} & \nabla_{\boldsymbol{uu}} \Phi_d^{(j)} \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{pmatrix}\right] = \\ &\frac{\delta t}{2} \begin{pmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \nabla_{\boldsymbol{xx}} f^{(j)} & \nabla_{\boldsymbol{xu}} f^{(j)} \\ \nabla_{\boldsymbol{ux}} f^{(j)} & \nabla_{\boldsymbol{uu}} f^{(j)} \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{pmatrix} = \tilde{O}_d^{(j)}. \end{aligned} \tag{24}$$

The second-order gradients of the dynamics function Φ are reduced to the gradients of the deterministic dynamics due to the fact that the noise enters linearly in the stochastic dynamics, rendering their contribution zero when applying the expectation operator. We proceed to define a new column vector $\tilde{\boldsymbol{O}}_d \in \mathbb{R}^{n\times 1}$ whose elements are defined in Eqn.(24).
To conclude the computation of the first order term it is desired to reformulate $V_{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{O}}_d$ in terms of its dependence on variations in control and state. Accordingly, we define the symmetric matrices $\boldsymbol{\mathcal{F}} \in \mathbb{R}^{n\times n}$, $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{p\times p}$ and $\boldsymbol{\mathcal{L}} \in \mathbb{R}^{p\times n}$ such that

$$V_{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{O}}_d = \frac{1}{2}\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}}\boldsymbol{\mathcal{F}}\delta\boldsymbol{x}_{\mathsf{t}} + \frac{1}{2}\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}}\boldsymbol{\mathcal{Z}}\delta\boldsymbol{u}_{\mathsf{t}} + \delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}}\boldsymbol{\mathcal{Z}}\delta\boldsymbol{x}_{\mathsf{t}}, \quad (25)$$

where

$$\boldsymbol{\mathcal{F}} = \left(\sum_{j=1}^{n} \nabla_{\boldsymbol{xx}} f^{(j)} V_{x_j}\right) \quad (26)$$

$$\boldsymbol{\mathcal{Z}} = \left(\sum_{j=1}^{n} \nabla_{\boldsymbol{uu}} f^{(j)} V_{x_j}\right) \quad (27)$$

$$\boldsymbol{\mathcal{L}} = \left(\sum_{j=1}^{n} \nabla_{\boldsymbol{ux}} f^{(j)} V_{x_j}\right). \quad (28)$$

Finally it is observed that the first order term $\mathbb{E}[V_{\boldsymbol{x}}^{\mathsf{T}}\delta\boldsymbol{x}_{t+\delta t}]$ depends quadratically on variations in control and state.

C.2. Computing the expectation of the second order term of the cost-to-go $\mathbb{E}[\delta\boldsymbol{x}_{t+\delta t}^{\mathsf{T}} V_{\boldsymbol{xx}}\delta\boldsymbol{x}_{t+\delta t}]$

In a similar fashion we substitute the linearized dynamics into the expectation of the second order term and obtain 16 terms. These terms are categorized into five classes

$$\mathbb{E}[\delta\boldsymbol{x}_{t+\delta t}^{\mathsf{T}} V_{\boldsymbol{xx}}\delta\boldsymbol{x}_{t+\delta+t}] = \boldsymbol{\Pi_1}+\boldsymbol{\Pi_2}+\boldsymbol{\Pi_3}+\boldsymbol{\Pi_4}+\boldsymbol{\Pi_5} \quad (29)$$

, where a membership to a class is dictated by the dependency on the noise term $\boldsymbol{\xi}_t$ and the linearized second order dynamics $\boldsymbol{O}_d$.

The five classes are defined as

$$\begin{aligned}
\boldsymbol{\Pi_1} &= \mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}] + \mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] + \\
&\quad \mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] + \mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}], \\
\boldsymbol{\Pi_2} &= \mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}] + \mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] + \\
&\quad \mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] + \mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] + \\
&\quad \mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t], \\
\boldsymbol{\Pi_3} &= \mathbb{E}[\boldsymbol{O}_d^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] + \mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\boldsymbol{O}_d], \\
\boldsymbol{\Pi_4} &= \mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}}\boldsymbol{O}_d] + \mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}}\boldsymbol{O}_d] + \\
&\quad \mathbb{E}[\boldsymbol{O}_d^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] + \mathbb{E}[\boldsymbol{O}_d^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}], \\
\boldsymbol{\Pi_5} &= \mathbb{E}[\boldsymbol{O}_d^{\mathsf{T}} V_{\boldsymbol{xx}}\boldsymbol{O}_d].
\end{aligned}$$

The terms in class one $\boldsymbol{\Pi}_1$ are independent of $\boldsymbol{\xi}_t$ and $\boldsymbol{O}_d$. In the second class $\boldsymbol{\Pi}_2$, the terms that depend on $\boldsymbol{\xi}_t$ and not on $\boldsymbol{O}_d$ are grouped. The third class $\boldsymbol{\Pi}_3$ contains all the terms which depend on both $\boldsymbol{\xi}_t$ and $\boldsymbol{O}_d$. The fourth class $\boldsymbol{\Pi}_4$ features the terms that depend linearly on $\boldsymbol{O}_d$. The last class $\boldsymbol{\Pi}_5$ consists of the term depending quadratically on $\boldsymbol{O}_d$.
In the computation of these terms a recurrent scheme is followed, where the expectation operator will nullify the contributions of terms linearly depending on the noise $\boldsymbol{\xi}_t$. In addition, terms that depend quadratically depend on $\boldsymbol{O}_d$ will be manipulated to obtain an expression that is dependent on the covariance matrix. Lastly, terms containing variations in control or state higher than second order are disregarded.
As all the terms in the first class $\boldsymbol{\Pi}_1$ are deterministic, the expectation operator is dropped and we obtain

$$\begin{aligned}
\mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}] &= \delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}} \\
\mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] &= \delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}} \\
\mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] &= \delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}} \\
\mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}] &= \delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}.
\end{aligned} \quad (30)$$

Having computed all term in the first class $\boldsymbol{\Pi_1}$, we proceed by computing the terms of the second class $\boldsymbol{\Pi_2}$. Terms that linearly depend on the noise $\boldsymbol{\xi}_t$ are equal zero as the noise term has zero mean $\mathbb{E}[\boldsymbol{\xi}_t] = 0$

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}} A_t\delta\boldsymbol{x}_{\mathsf{t}}] &= 0 \\
\mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}} B_t\delta\boldsymbol{u}_{\mathsf{t}}] &= 0 \\
\mathbb{E}[\delta\boldsymbol{x}_{\mathsf{t}}{}^{\mathsf{T}} A_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] &= 0 \\
\mathbb{E}[\delta\boldsymbol{u}_{\mathsf{t}}{}^{\mathsf{T}} B_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] &= 0.
\end{aligned} \quad (31)$$

The last term in the second class $\boldsymbol{\Pi_2}$ depends quadratically on the noise random variable, hence we utilize the covariance matrix Σ_t to eliminate the expectation operator

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] &= \mathbb{E}[\boldsymbol{trace}(\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t)] \\
&= \boldsymbol{trace}(\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\mathbb{E}[\boldsymbol{\xi}_t\boldsymbol{\xi}_t^{\mathsf{T}}]) \\
&= \boldsymbol{trace}(\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\Sigma_t).
\end{aligned} \quad (32)$$

The term Γ_t is a first order function in $\delta\boldsymbol{x}_{\mathsf{t}}$ and $\delta\boldsymbol{u}_{\mathsf{t}}$, thus it is desired to simplify the above expression to yield an expression depending on variations of control and state

$$\begin{aligned}
&\boldsymbol{trace}(\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\Sigma_t) = \sigma^2\delta t \times \\
&\boldsymbol{trace}\left(\begin{pmatrix}\Gamma^{(1)\,\mathsf{T}} \\ \dots \\ \Gamma^{(m)\,\mathsf{T}}\end{pmatrix} V_{\boldsymbol{xx}}\left(\Gamma^{(1)\,\mathsf{T}},\dots,\Gamma^{(n)\,\mathsf{T}}\right)\right) \\
&= \sigma^2\delta t\sum_{i=1}^{m}\Gamma^{(i)\,\mathsf{T}} V_{\boldsymbol{xx}}\Gamma^{(i)}.
\end{aligned} \quad (33)$$

Substituting the vectors $\Gamma^{(i)}$ in the above expression and simplifying yields

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\xi}_t^{\mathsf{T}}\Gamma_t^{\mathsf{T}} V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] &= \delta\boldsymbol{x}^{\mathsf{T}}\tilde{\boldsymbol{\mathcal{F}}}\delta\boldsymbol{x} + \delta\boldsymbol{u}^{\mathsf{T}}\tilde{\boldsymbol{\mathcal{Z}}}\delta\boldsymbol{u} + 2\delta\boldsymbol{x}^{\mathsf{T}}\tilde{\boldsymbol{\mathcal{L}}}\delta\boldsymbol{u} \\
&\quad + 2\delta\boldsymbol{u}^{\mathsf{T}}\tilde{\boldsymbol{\mathcal{U}}} + 2\delta\boldsymbol{u}^{\mathsf{T}}\tilde{\boldsymbol{\mathcal{S}}} + \gamma
\end{aligned} \quad (34)$$

, where the quantities $\tilde{\boldsymbol{\mathcal{F}}} \in \mathcal{R}^{n\times n}$, $\tilde{\boldsymbol{\mathcal{Z}}} \in \mathcal{R}^{p\times p}$, $\tilde{\boldsymbol{\mathcal{L}}} \in \mathcal{R}^{n\times p}$, $\tilde{\boldsymbol{\mathcal{S}}} \in \mathcal{R}^{n\times 1}$, $\tilde{\boldsymbol{\mathcal{U}}} \in \mathcal{R}^{p\times 1}$ and $\gamma \in \mathcal{R}$ are specified as

$$\tilde{\boldsymbol{\mathcal{F}}} = \sigma^2\delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{x}} F_c^{(i)\,T} V_{\boldsymbol{xx}} \nabla_{\boldsymbol{x}} F_c^{(i)} \tag{35}$$

$$\tilde{\boldsymbol{\mathcal{Z}}} = \sigma^2\delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{u}} F_c^{(i)\,T} V_{\boldsymbol{xx}} \nabla_{\boldsymbol{u}} F_c^{(i)} \tag{36}$$

$$\tilde{\boldsymbol{\mathcal{L}}} = \sigma^2\delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{x}} F_c^{(i)\,T} V_{\boldsymbol{xx}} \nabla_{\boldsymbol{u}} F_c^{(i)} \tag{37}$$

$$\tilde{\boldsymbol{\mathcal{S}}} = \sigma^2\delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{x}} F_c^{(i)\,T} V_{\boldsymbol{xx}} F_c^{(i)} \tag{38}$$

$$\tilde{\boldsymbol{\mathcal{U}}} = \sigma^2\delta t \sum_{i=1}^{m} \nabla_{\boldsymbol{u}} F_c^{(i)\,T} V_{\boldsymbol{xx}} F_c^{(i)} \tag{39}$$

$$\gamma = \sigma^2\delta t \sum_{i=1}^{m} F_c^{(i)\,T} V_{\boldsymbol{xx}} F_c^{(i)}. \tag{40}$$

This concludes the terms in class $\mathbf{\Pi_2}$. The third class encompasses terms depending both on the noise and the linearized second order dynamics. We compute

$$\begin{aligned}\mathbb{E}[\boldsymbol{O}_d^{\mathsf{T}} V_{\boldsymbol{xx}} \Gamma_t \boldsymbol{\xi}_t] &= \mathbb{E}[\boldsymbol{trace}(V_{\boldsymbol{xx}} \Gamma_t \boldsymbol{\xi}_t \boldsymbol{O}_d^{\mathsf{T}})] \\ &= \boldsymbol{trace}(V_{\boldsymbol{xx}} \Gamma_t \mathbb{E}[\boldsymbol{\xi}_t \boldsymbol{O}_d^{\mathsf{T}}]) \\ &= \boldsymbol{trace}(V_{\boldsymbol{xx}} \Gamma_t \mathbb{E}[\boldsymbol{\xi}_t O_d^{(1)} \; ... \; \boldsymbol{\xi}_t O_d^{(n)}]).\end{aligned} \tag{41}$$

In order to compute the above expectation term inside the trace operator we compute the j-th column vector of the expectation term $\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t O_d^{(j)}]$

$$\begin{aligned}\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t O_d^{(j)}] &= \frac{1}{2}\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \Phi_{d,\boldsymbol{xx}}^{(j)} \delta\boldsymbol{x}] \\ &+ \frac{1}{2}\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{u}^{\mathsf{T}} \Phi_{d,\boldsymbol{uu}}^{(j)} \delta\boldsymbol{u}] + \mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{u}^{\mathsf{T}} \Phi_{d,\boldsymbol{ux}}^{(j)} \delta\boldsymbol{x}].\end{aligned} \tag{42}$$

The analysis precedes by computing all terms arising above. Starting with the first term we obtain

$$\begin{aligned}&\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \Phi_{d,\boldsymbol{xx}}^{(j)} \delta\boldsymbol{x}] = \\ &\mathbb{E}\left[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \left(\nabla_{\boldsymbol{xx}} f^{(j)}(\boldsymbol{x},\boldsymbol{u})\delta t\right)\delta\boldsymbol{x}\right] + \\ &\mathbb{E}\left[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \left(\nabla_{\boldsymbol{xx}} F_r{}^{(j)}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\sqrt{\delta t}\right)\delta\boldsymbol{x}\right] = \\ &\mathbb{E}\left[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \left(\nabla_{\boldsymbol{xx}} F_r{}^{(j)}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\sqrt{\delta t}\right)\delta\boldsymbol{x}\right].\end{aligned} \tag{43}$$

The first term equals zero due to its linear dependence on the noise term. Obtaining the second term requires further computation

$$\begin{aligned}&\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \Phi_{d,\boldsymbol{xx}}^{(j)} \delta\boldsymbol{x}] = \\ &\mathbb{E}\left[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \nabla_{\boldsymbol{xx}} \left(F_r{}^{(j)}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\sqrt{\delta t}\right)\delta\boldsymbol{x}\right].\end{aligned} \tag{44}$$

The scalar product inside the derivative can be written as the summation of the element-wise components, where the j-th row vector is denoted as $F_r^{(j)} = (F_r^{(j1)}, ..., F_r^{(jm)})$

$$\begin{aligned}&\mathbb{E}\left[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \nabla_{\boldsymbol{xx}} \left(F_r{}^{(j)}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi}_t\sqrt{\delta t}\right)\delta\boldsymbol{x}\right] = \\ &\mathbb{E}\left[\delta t\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \nabla_{\boldsymbol{xx}} \left(\sum_{k=1}^{m} F_r{}^{(jk)}\xi_t^{(k)}\right)\delta\boldsymbol{x}\right] = \\ &\mathbb{E}\left[\delta t\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \left(\sum_{k=1}^{m} \nabla_{\boldsymbol{xx}} F_r{}^{(jk)}\xi_t^{(k)}\right)\delta\boldsymbol{x}\right] = \\ &\mathbb{E}\left[\delta t\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \left(\sum_{k=1}^{m} \xi_t^{(k)} \nabla_{\boldsymbol{xx}} (F_r{}^{(jk)})\right)\delta\boldsymbol{x}\right].\end{aligned} \tag{45}$$

The expression $\delta\boldsymbol{x}^{\mathsf{T}} \left(\sum_{k=1}^{m} \xi_t^{(k)} \nabla_{\boldsymbol{xx}} (F_r{}^{(jk)})\right)\delta\boldsymbol{x}$ is a scalar quantity, therefore by writing $\boldsymbol{\xi}_t$ in vector form the above expression can be written as

$$\begin{aligned}&\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \Phi_{d,\boldsymbol{xx}}^{(j)} \delta\boldsymbol{x}] = \\ &\begin{pmatrix} \delta t\mathbb{E}\left[\xi_t^{(1)} \delta\boldsymbol{x}^{\mathsf{T}} \left(\sum_{k=1}^{m} \xi_t^{(k)} \nabla_{\boldsymbol{xx}} (F_r{}^{(jk)})\right)\delta\boldsymbol{x}\right] \\ \cdots \\ \delta t\mathbb{E}\left[\xi_t^{(m)} \delta\boldsymbol{x}^{\mathsf{T}} \left(\sum_{k=1}^{m} \xi_t^{(k)} \nabla_{\boldsymbol{xx}} (F_r{}^{(jk)})\right)\delta\boldsymbol{x}\right] \end{pmatrix}.\end{aligned} \tag{46}$$

Utilizing the statistical properties of the noise

$$\mathbb{E}[\xi^{(i)}\xi^{(j)}] = \begin{cases} 0 & ,\text{i} \neq \text{j} \\ \sigma^2 & ,\text{i} = \text{j} \end{cases} \tag{47}$$

, we derive a compact expression for Eqn.(46)

$$\begin{aligned}&\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{x}^{\mathsf{T}} \Phi_{d,\boldsymbol{xx}}^{(j)} \delta\boldsymbol{x}] = \\ &\sigma^2\delta t \begin{pmatrix} \delta\boldsymbol{x}^{\mathsf{T}} \nabla_{\boldsymbol{xx}} F_r{}^{(j1)} \delta\boldsymbol{x} \\ \cdots \\ \sigma^2\delta t \delta\boldsymbol{x}^{\mathsf{T}} \nabla_{\boldsymbol{xx}} F_r{}^{(jm)} \delta\boldsymbol{x} \end{pmatrix}.\end{aligned} \tag{48}$$

The same approach can be taken to compute all terms in Eqn.(42), yielding

$$\begin{aligned}&\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{u}^{\mathsf{T}} \Phi_{d,\boldsymbol{uu}}^{(j)} \delta\boldsymbol{u}] = \\ &\sigma^2\delta t \begin{pmatrix} \delta\boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{uu}} F_r{}^{(j1)} \delta\boldsymbol{u} \\ \cdots \\ \sigma^2\delta t \delta\boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{uu}} F_r{}^{(jm)} \delta\boldsymbol{u} \end{pmatrix},\end{aligned} \tag{49}$$

$$\begin{aligned}&\mathbb{E}[\sqrt{\delta t}\boldsymbol{\xi}_t \delta\boldsymbol{u}^{\mathsf{T}} \Phi_{d,\boldsymbol{ux}}^{(j)} \delta\boldsymbol{x}] = \\ &\sigma^2\delta t \begin{pmatrix} \delta\boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{ux}} F_r{}^{(j1)} \delta\boldsymbol{x} \\ \cdots \\ \sigma^2\delta t \delta\boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{ux}} F_r{}^{(jm)} \delta\boldsymbol{x} \end{pmatrix}.\end{aligned} \tag{50}$$

Based on Eqns.(48)-(50) the term in Eqn.(41) can be formulated as

$$\mathbb{E}[\boldsymbol{O}_d^{\mathsf{T}} V_{\boldsymbol{xx}} \Gamma_t \boldsymbol{\xi}_t] = \boldsymbol{trace}(V_{\boldsymbol{xx}} \Gamma_t (\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{N}})), \tag{51}$$

where the matrices $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{m\times n}$, $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{\mathcal{N}} \in \mathbb{R}^{m\times n}$ are defined as

$$\boldsymbol{\mathcal{M}} = \sigma^2\delta t \begin{pmatrix} \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xx}}F_r{}^{(11)}\delta\boldsymbol{x} & \dots & \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xx}}F_r{}^{(1n)}\delta\boldsymbol{x} \\ \dots & \dots & \dots \\ \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xx}}F_r{}^{(m1)}\delta\boldsymbol{x} & \dots & \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xx}}F_r{}^{(mn)}\delta\boldsymbol{x} \end{pmatrix}, \tag{52}$$

$$\boldsymbol{\mathcal{G}} = \sigma^2\delta t \begin{pmatrix} \delta\boldsymbol{u}^\mathsf{T}\nabla_{\boldsymbol{uu}}F_r{}^{(11)}\delta\boldsymbol{u} & \dots & \delta\boldsymbol{u}^\mathsf{T}\nabla_{\boldsymbol{uu}}F_r{}^{(1n)}\delta\boldsymbol{u} \\ \dots & \dots & \dots \\ \delta\boldsymbol{u}^\mathsf{T}\nabla_{\boldsymbol{uu}}F_r{}^{(m1)}\delta\boldsymbol{u} & \dots & \delta\boldsymbol{u}^\mathsf{T}\nabla_{\boldsymbol{uu}}F_r{}^{(mn)}\delta\boldsymbol{u} \end{pmatrix}, \tag{53}$$

$$\boldsymbol{\mathcal{N}} = \sigma^2\delta t \begin{pmatrix} \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xu}}F_r{}^{(11)}\delta\boldsymbol{u} & \dots & \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xu}}F_r{}^{(1n)}\delta\boldsymbol{u} \\ \dots & \dots & \dots \\ \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xu}}F_r{}^{(m1)}\delta\boldsymbol{u} & \dots & \delta\boldsymbol{x}^\mathsf{T}\nabla_{\boldsymbol{xu}}F_r{}^{(mn)}\delta\boldsymbol{u} \end{pmatrix}. \tag{54}$$

Γ_t in Eqn.(51) can decomposed using Eqn.(10) into

$$\begin{aligned}\mathbb{E}[\boldsymbol{O}_d^\mathsf{T}V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] &= \boldsymbol{trace}(V_{\boldsymbol{xx}}(\Delta+F)(\boldsymbol{\mathcal{M}}+\boldsymbol{\mathcal{G}}+\boldsymbol{\mathcal{N}})) \\ &= \boldsymbol{trace}(V_{\boldsymbol{xx}}F(\boldsymbol{\mathcal{M}}+\boldsymbol{\mathcal{G}}+\boldsymbol{\mathcal{N}})).\end{aligned} \tag{55}$$

Since Δ is a first order function in variations of control and state and the matrices $\boldsymbol{\mathcal{M}}$, $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{N}}$ are second order functions thereof, the product $\Delta(\boldsymbol{\mathcal{M}}+\boldsymbol{\mathcal{G}}+\boldsymbol{\mathcal{N}})$ results in cubic terms in variations of control and state, rendering the product term negligible.

Once more the analysis of the third class is concluded by reformulating the results as functions of $\delta\boldsymbol{x}$ and $\delta\boldsymbol{u}$. Therefore we decompose the term in Eqn.(55) into

$$\begin{aligned}&\boldsymbol{trace}(V_{\boldsymbol{xx}}F(\boldsymbol{\mathcal{M}}+\boldsymbol{\mathcal{G}}+\boldsymbol{\mathcal{N}})) = \boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{M}})+ \\ &\boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{G}}) + \boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{N}}).\end{aligned} \tag{56}$$

To simplify the first term in Eqn.(56), it is required to express the (i,j)-th element of the product $\boldsymbol{\mathcal{C}} = V_{\boldsymbol{xx}}F$ as $\boldsymbol{\mathcal{C}}^{(i,j)} = \sum_{r=1}^n V_{\boldsymbol{xx}}^{(i,r)}F^{(r,j)}$, where $\boldsymbol{\mathcal{C}}^{(i,j)} \in \mathbb{R}^{(n\times m)}$. Now the (τ,υ)-th element of the product $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}$ can be expressed as $\boldsymbol{\mathcal{H}}^{(\tau,\upsilon)} = \sum_{k=1}^n \boldsymbol{\mathcal{C}}^{(\tau,k)}\boldsymbol{\mathcal{M}}^{(k,\upsilon)}$ with $\boldsymbol{\mathcal{H}}^{(\tau,\upsilon)} \in \mathbb{R}^{(n\times m)}$. Making use of the element-wise notation we can formulate the term $\boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{M}})$ as

$$\begin{aligned}\boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{M}}) &= \sum_{\lambda=1}^n \boldsymbol{\mathcal{H}}^{(\lambda,\lambda)} \\ &= \sum_{\lambda=1}^n\sum_{k=1}^m \boldsymbol{\mathcal{C}}^{(\lambda,k)}\boldsymbol{\mathcal{M}}^{(k,\lambda)} \\ &= \sum_{\lambda=1}^n\sum_{k=1}^m \left(\sum_{r=1}^n V_{\boldsymbol{xx}}^{(k,r)}F^{(r,\lambda)}\right)\boldsymbol{\mathcal{M}}^{(k,\lambda)}.\end{aligned} \tag{57}$$

The (k,λ)-th element of the matrix $\boldsymbol{\mathcal{M}} = \delta t\sigma^2\delta\boldsymbol{x}^\mathsf{T}F_{\boldsymbol{xx}}^{(k\lambda)}\delta\boldsymbol{x}$ is substituted in the expression above, resulting in

$$\begin{aligned}&\boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{M}}) \\ &= \sum_{\lambda=1}^n\sum_{k=1}^m \left(\left(\sum_{r=1}^n V_{\boldsymbol{xx}}^{(k,r)}F^{(r,\lambda)}\right)\delta t\sigma^2\delta\boldsymbol{x}^\mathsf{T}F_{\boldsymbol{xx}}^{(k\lambda)}\delta\boldsymbol{x}\right) \\ &= \delta\boldsymbol{x}^\mathsf{T}\delta t\sigma^2\sum_{\lambda=1}^n\sum_{k=1}^m \left(\left(\sum_{r=1}^n V_{\boldsymbol{xx}}^{(k,r)}F^{(r,\lambda)}\right)F_{\boldsymbol{xx}}^{(k\lambda)}\right)\delta\boldsymbol{x} \\ &= \delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{M}}}\delta\boldsymbol{x},\end{aligned} \tag{58}$$

where we define the matrix $\tilde{\boldsymbol{\mathcal{M}}} \in \mathbb{R}^{n\times n}$ as follows

$$\tilde{\boldsymbol{\mathcal{M}}} = \delta t\sigma^2\sum_{\lambda=1}^n\sum_{k=1}^m \left(\left(\sum_{r=1}^n V_{\boldsymbol{xx}}^{(k,r)}F^{(r,\lambda)}\right)F_{\boldsymbol{xx}}^{(k\lambda)}\right). \tag{59}$$

Proceeding in a similar fashion we can derive comparable expressions for the second and third term in Eqn.(55) generating

$$\boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{G}}) = \delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{G}}}\delta\boldsymbol{x} \tag{60}$$

with the matrix $\tilde{\boldsymbol{\mathcal{G}}} \in \mathbf{R}^{p\times p}$ defined as

$$\tilde{\boldsymbol{\mathcal{G}}} = \delta t\sigma^2\sum_{\lambda=1}^n\sum_{k=1}^m \left(\left(\sum_{r=1}^n V_{\boldsymbol{xx}}^{(k,r)}F^{(r,\lambda)}\right)F_{\boldsymbol{uu}}^{(k\lambda)}\right) \tag{61}$$

and

$$\boldsymbol{trace}(V_{\boldsymbol{xx}}F\boldsymbol{\mathcal{N}}) = \delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{N}}}\delta\boldsymbol{x} \tag{62}$$

with the matrix $\tilde{\boldsymbol{\mathcal{N}}} \in \mathbf{R}^{n\times p}$ defined as

$$\tilde{\boldsymbol{\mathcal{N}}} = \delta t\sigma^2\sum_{\lambda=1}^n\sum_{k=1}^m \left(\left(\sum_{r=1}^n V_{\boldsymbol{xx}}^{(k,r)}F^{(r,\lambda)}\right)F_{\boldsymbol{xu}}^{(k\lambda)}\right). \tag{63}$$

Finally, the analyzed term yields

$$\mathbb{E}[\boldsymbol{O}_d^\mathsf{T}V_{\boldsymbol{xx}}\Gamma_t\boldsymbol{\xi}_t] = \frac{1}{2}\delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{M}}}\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{u}^\mathsf{T}\tilde{\boldsymbol{\mathcal{G}}}\delta\boldsymbol{u} + \delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{N}}}\delta\boldsymbol{u}. \tag{64}$$

It can be proven that the above matrices are symmetric, thus yielding

$$\mathbb{E}[\boldsymbol{\xi}_t^\mathsf{T}\Gamma_t^\mathsf{T}V_{\boldsymbol{xx}}\boldsymbol{O}_d] = \frac{1}{2}\delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{M}}}\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{u}^\mathsf{T}\tilde{\boldsymbol{\mathcal{G}}}\delta\boldsymbol{u} + \delta\boldsymbol{x}^\mathsf{T}\tilde{\boldsymbol{\mathcal{N}}}\delta\boldsymbol{u}. \tag{65}$$

Class $\boldsymbol{\Pi}_4$ includes the term that depend linearly on the linearized second order dynamics $\boldsymbol{O}_d(\delta\boldsymbol{x},\delta\boldsymbol{u},\boldsymbol{\xi},\delta t)$ and are independent of the noise $\boldsymbol{\xi}_t$. Similar to the calculation performed in Eqn. (24), the terms in class $\boldsymbol{\Pi}_4$ yield

$$\begin{aligned}&\mathbb{E}[\delta\boldsymbol{x}_\mathfrak{t}{}^\mathsf{T}A_t^\mathsf{T}V_{\boldsymbol{xx}}\boldsymbol{O}_d] = \delta\boldsymbol{x}_\mathfrak{t}{}^\mathsf{T}A_t^\mathsf{T}V_{\boldsymbol{xx}}\tilde{\boldsymbol{O}}_d = 0 \\ &\mathbb{E}[\delta\boldsymbol{u}_\mathfrak{t}{}^\mathsf{T}B_t^\mathsf{T}V_{\boldsymbol{xx}}\tilde{\boldsymbol{O}}_d] = \delta\boldsymbol{u}_\mathfrak{t}{}^\mathsf{T}B_t^\mathsf{T}V_{\boldsymbol{xx}}\tilde{\boldsymbol{O}}_d = 0 \\ &\mathbb{E}[\boldsymbol{O}_d^\mathsf{T}V_{\boldsymbol{xx}}B_t\delta\boldsymbol{u}_\mathfrak{t}] = \tilde{\boldsymbol{O}}_d^\mathsf{T}V_{\boldsymbol{xx}}B_t\delta\boldsymbol{u}_\mathfrak{t} = 0 \\ &\mathbb{E}[\boldsymbol{O}_d^\mathsf{T}V_{\boldsymbol{xx}}A_t\delta\boldsymbol{x}_\mathfrak{t}] = \tilde{\boldsymbol{O}}_d^\mathsf{T}V_{\boldsymbol{xx}}A_t\delta\boldsymbol{x}_\mathfrak{t} = 0.\end{aligned} \tag{66}$$

Due to the fact that $\tilde{\boldsymbol{O}}_d$ is a quadratic function in control and state variations, all terms in class $\boldsymbol{\Pi}_4$ lead to variations

higher than second order in $\delta \boldsymbol{x}_\mathrm{t}$ and $\delta \boldsymbol{u}_\mathrm{t}$. Due to the fact that the DDP is a second order method these terms are set to zero. The last class $\boldsymbol{\Pi}_5$ comprises a single term exhibiting quadratic depends on $\tilde{\boldsymbol{O}}_d$

$$\begin{aligned}\mathbb{E}[\boldsymbol{O}_d^\mathsf{T} V_{\boldsymbol{xx}} \boldsymbol{O}_d] &= E[\boldsymbol{trace}(V_{\boldsymbol{xx}} \boldsymbol{O}_d \boldsymbol{O}_d^\mathsf{T})] \\ &= \boldsymbol{trace}(V_{\boldsymbol{xx}} \mathbb{E}[\boldsymbol{O}_d \boldsymbol{O}_d^\mathsf{T}]).\end{aligned} \tag{67}$$

It is apparent that the term $\mathbb{E}[\boldsymbol{O}_d \boldsymbol{O}_d^\mathsf{T}]$ induces fourth order variations in control and state,which again renders their contribution negligible.

After computing all terms arising from the expectation of the cost-to-go function, the resulting expression for the expectation of the expanded cost-to-go function is presented

$$\begin{aligned}&\mathbb{E}[\tilde{V}(\bar{\boldsymbol{x}}_{t+\delta t} + \delta \boldsymbol{x}_{t+\delta t}] = V(\bar{\boldsymbol{x}}_{t+\delta t}) + V_{\boldsymbol{x}}^\mathsf{T} A_t \delta \boldsymbol{x}_\mathrm{t} \\ &+ V_{\boldsymbol{x}}^\mathsf{T} B_t \delta \boldsymbol{u}_\mathrm{t} + \frac{1}{2} \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} \boldsymbol{\mathcal{F}} \delta \boldsymbol{x}_\mathrm{t} + \frac{1}{2} \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} \boldsymbol{\mathcal{Z}} \delta \boldsymbol{u}_\mathrm{t} + \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} \boldsymbol{\mathcal{Z}} \delta \boldsymbol{x}_\mathrm{t} \\ &+ \frac{1}{2} \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} A_t^\mathsf{T} V_{\boldsymbol{xx}} A_t \delta \boldsymbol{x}_\mathrm{t} + \frac{1}{2} \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} B_t^\mathsf{T} V_{\boldsymbol{xx}} B_t \delta \boldsymbol{u}_\mathrm{t} \\ &+ \frac{1}{2} \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} A_t^\mathsf{T} V_{\boldsymbol{xx}} B_t \delta \boldsymbol{u}_\mathrm{t} + \frac{1}{2} \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} B_t^\mathsf{T} V_{\boldsymbol{xx}} A_t \delta \boldsymbol{x}_\mathrm{t} \\ &+ \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{F}}} \delta \boldsymbol{x}_\mathrm{t} + \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{Z}}} \delta \boldsymbol{u}_\mathrm{t} + 2 \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{L}}} \delta \boldsymbol{u}_\mathrm{t} + 2 \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{U}}} \\ &+ 2 \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{S}}} + \gamma + \frac{1}{2} \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{M}}} \delta \boldsymbol{x}_\mathrm{t} + \frac{1}{2} \delta \boldsymbol{u}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{G}}} \delta \boldsymbol{u}_\mathrm{t} \\ &+ \delta \boldsymbol{x}_\mathrm{t}^\mathsf{T} \tilde{\boldsymbol{\mathcal{N}}} \delta \boldsymbol{u}_\mathrm{t}.\end{aligned} \tag{68}$$

V. Implementational Issues and Simulation

In this section, we discuss implementational issues concerning the SDDP algorithm that are crucial for the algorithm's convergence. Furthermore, the section demonstrates the algorithm on different models with various orders.

A. Implementational Issues

A.1. iLQR and DDP

As SDDP ($\kappa = 1$) is a second-order accurate method, additional computationally expensive tensorial terms must be computed at each iteration. These terms arise from the second-order expansion of the dynamics. The inclusion of quadratic approximation improves the local fidelity of the approximation. However, if the quadratic approximation of the dynamics strays too much from the region of validity of the original model, these additional terms may render Q_{uu} negative definite requiring regularization. The required regularization would increase the computational load of second-order accurate methods. Furthermore, as DDP is second-order accurate convergence is fast near the local minimum, but can be slow far away from the minimum. In these cases, iLQG ($\kappa = 0$) can and should be considered. iLQG approximate the dynamics to first-order and thus convexity of the problem is always guaranteed, therefore regularization is not needed. Despite the superlinear convergence properties of iLQG, it can experience faster convergence due to the alleviation of the tensorial terms and is mostly favoured in the recent literature.

Nevertheless, there is no way to predict which method would perform better and their application thus remains problem-dependent.

A.2. Line Search

The presented SDDP algorithm approximates a quadratic expansion of the nonlinear system dynamics. Should the quadratic approximation deviate significantly from the region of validity of the nonlinear model, a decrease in the cost function is not guaranteed, which may ultimately result in divergence. This effect can be remedied by line search schemes. The general idea is to choose the step-size parameter α sufficiently small in order to observe a reduction the cost function. To reduce computational complexity inexact line search techniques were investigated. The best performance was found using backtracking line search, where the step-size parameter α is initially set to 1 in the forward pass and this parameter is iteratively reduced by a backtracking decay parameter ρ until a decrease in the cost function is observed.

Algorithm 2 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 $\Delta J < 0$;

Furthermore, a maximum number of backtracking iterations B is initially set. If a decrease in the cost function is not achieved within B iterations of the line search algorithm, no step is taken and the backward pass is repeated with increased regularization paremater

A.3. Regularization

For any line search scheme to converge, it is required to guarantee that the Hessian Q_{uu} is positive definite. In the case of iLQG, where a linear approximation of the general nonlinear dynamics is utilized, it can be proven that a sufficient condition for Q_{uu} to be positive definite is a convex cost function [3]. This is easliy obtained as the cost function is usually a designer choice. In the case of a quadratic approximation of the nonlinear system dynamics, the positive definiteness of Q_{uu} is not guaranteed anymore, thereby a descent direction is not necessarily obtained at every iteration. To ensure positive definiteness and hence a descent direction, regularization techniques are required. In addition, to the desired convergence guarantees, numerical stability of the algorithm is crucial. Therefore, the generation

of locally stabilizing feedback terms is preferred. Tassa et al. [7] thus proposed guaranteeing the positive definiteness of the matrix $\boldsymbol{P} = \boldsymbol{Q_{xx}} - \boldsymbol{Q_{xu}}\boldsymbol{Q_{uu}^{-1}}\boldsymbol{Q_{ux}}$, which generates locally stabilizing feedback terms. Using Schur's decomposition both conditions are combined, requiring the positive definiteness of the matrix H

$$\boldsymbol{H} = \begin{pmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{pmatrix} > 0 \tag{69}$$

to ensure both convergence of the algorithm and numerical stability. The application of regularization techniques is hence indispensable in the case of second-order accurate DDP methods. Despite its significance it is frequently either not explicitly addressed or given minimal attention in papers, more than often confined to a small paragraph. Due to this limited consideration, various methods were investigated with varying model order.

The first method constitutes adding an identity matrix scaled with a sufficiently large parameter to the matrix H

$$\tilde{\boldsymbol{H}} = \begin{pmatrix} Q_{xx} + \mu I_n & Q_{xu} \\ Q_{ux} & Q_{uu} + \mu I_p \end{pmatrix} > 0 \tag{70}$$

, where $\mu > 0$. This amounts to adding a quadratic cost around the current control and cost sequence. The regularization parameter μ is iteratively increased stage-wise during the backward pass after computing the derivatives of the state-action value function until the matrix H is positive definite. The selection of the regularization parameter μ significantly affects convergence. Choosing the parameter μ very large slows down convergence. Near the local minimum μ should be zero for fast convergence. Thus, if the Hessian $\boldsymbol{H}$ is not positive definite, μ should be chosen large enough, but not more than necessary. As μ is chosen stage-wise a drawback to this regularization scheme is that the same control perturbation produces different effects at different stages depending on B_t as explained in [7]. This regularization scheme was applied to several models with varying orders. In higher order models this scheme fails as μ blows to infinity before $\boldsymbol{H}$ becomes positive definite. This means convergence is not achieved and divergence may occur.

The second method is also related to Levenberg-Marquardt schemes and was introduced in [9]. In [9] this method was used to ensure the positive definiteness of Q_{uu}, however in our case we utilize the proposed scheme to ensure the positive definiteness of the $\boldsymbol{H}$-matrix. The proposed method entails computing the eigenvalue decomposition of the matrix $\boldsymbol{H} = VDV^T$. As $\boldsymbol{H}$ is a symmetric matrix, all eigenvalues obtained in the diagonal matrix D should be positive for positive definiteness $\boldsymbol{H}$. Therefore, all negative eigenvalues are replaced with 0 and a positive constant λ is added to the diagonal matrix D. The modified $\tilde{\boldsymbol{H}} = V\tilde{D}V^T$ is guaranteed to be positive definite. Obviously, the parameter λ assumes the role of the regularization parameter and influences the convergence rate. This parameter is hence adapted during the main iteration of DDP in accordance with the distance to the local minimum. If $\lambda = 0$, this means we have the true Hessian and we are near the minimum. On the other hand, if λ is large, this means we replace the Hessian with λI_{n+p}, and we take conservative steps towards the minimum. In order to reflect the progress of algorithm, we decrease λ if the forward pass generates a trajectory with a decreasing cost. Otherwise, we do not take a step, increase λ and recompute the backward pass equations before once again rolling out the dynamics. From experience, we have found that the initial λ_0 is a tuning parameter, which has a significant effect on convergence. Empirically, we have found the best performance, when $0.001 \leq \lambda \leq 1$. When applied to low order models, this method yielded the best convergence performance. However, when applied to higher order models, this method fails as the matrix D eventually has complex eigenvalues. Even when we only consider the real part of the eigenvalues and eigenvectors, the positive definiteness of $\boldsymbol{H}$ is not guaranteed resulting in divergence.

The third method is again related to Levenberg-Marquardt schemes [7] and similar to the first method. The key difference is that this method penalizes deviations from the states rather than deviations from the controls as in the first method, while taking into account the regularized terms to modify the backward pass equations. Again, this regularization occurs during the backward pass by utilizing the regularized matrices

$$\begin{aligned} \tilde{Q}_{xx} &= Q_{xx} + A_t^T(V_{\boldsymbol{xx}} + \mu I_n)A_t \\ \tilde{Q}_{ux} &= Q_{ux} + B_t^T(V_{\boldsymbol{xx}} + \mu I_n)A_t \\ \tilde{Q}_{uu} &= Q_{uu} + B_t^T(V_{\boldsymbol{xx}} + \mu I_p)B_t \\ \boldsymbol{k} &= -\tilde{Q}_{uu}^{-1}Q_u \textit{ Feedfoward Gain} \\ \boldsymbol{K} &= -\tilde{Q}_{uu}^{-1}\tilde{Q}_{ux} \textit{ Feedback Gain} \end{aligned}$$

to compute the value function update. Should the backward pass fail, the regularization parameter is increased and the backward pass restarts. This constitutes another difference to the previously stage-wise chosen μ_k values. In addition, the proposed method only ensures the positive definiteness of the $\boldsymbol{Q_{uu}}$-matrix., thereby not guaranteeing stable feedback gains. The regularized feedfoward and feedback terms are also used to generate the optimal control input. Furthermore, these regularized matrices are considered when obtaining the backward pass equations and the suggested improved value update cancels out the error generated by the regularization. The improved value update is given here without proof

$$\begin{aligned} V(\boldsymbol{x}_k) &= Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) + \frac{1}{2}\boldsymbol{k}^\mathsf{T}Q_{\boldsymbol{uu}}\boldsymbol{k} + \boldsymbol{k}^\mathsf{T}Q_{\boldsymbol{u}} \\ V_{\boldsymbol{x}}(\boldsymbol{x}_k) &= Q_{\boldsymbol{x}} + \boldsymbol{K}^\mathsf{T}Q_{\boldsymbol{uu}}\boldsymbol{k} + \boldsymbol{K}^\mathsf{T}Q_{\boldsymbol{u}} + Q_{\boldsymbol{ux}}^\mathsf{T}\boldsymbol{k} \\ V_{\boldsymbol{xx}}(\boldsymbol{x}_k) &= Q_{\boldsymbol{xx}} + \boldsymbol{K}^\mathsf{T}Q_{\boldsymbol{uu}}\boldsymbol{K} + \boldsymbol{K}^\mathsf{T}Q_{\boldsymbol{ux}} + Q_{\boldsymbol{ux}}^\mathsf{T}\boldsymbol{K} \end{aligned}$$

Similar as in the other methods, the regularization parameter μ should be adapted continuously in accordance with the distance to the local minimum. A regularization schedule is thus combined with the above scheme to help tweak

μ to the minimum value needed for H to be positive definite. Thus, μ is adapted in the main iteration by following the schedule

Increase μ:
$\Delta \leftarrow max(\Delta_0, \Delta \cdot \Delta_0)$
$\mu \leftarrow max(\mu_{min}, \mu \cdot \Delta)$
Decrease μ:
$\Delta \leftarrow max(\frac{1}{\Delta_0}, \frac{\Delta}{\Delta_0})$

$$\mu \leftarrow \begin{cases} \mu \cdot \Delta & \text{if } \mu \cdot \Delta > \mu_{min} \\ 0 & \text{if } \mu \cdot \Delta < \mu_{min} \end{cases}$$

The hyperparameters used are $\mu_{min} = 1^{-6}$ and $\Delta = 1.5$. This method worked in all models albeit the convergence rate losing quadratic properties in the high-dimensional problem.

In conclusion, when it comes to regularization, there is no definitive choice that can be made. The application of these methods often involves experimenting with multiple approaches to determine the most effective one. From experience in high-dimensional tasks, regularization becomes more challenging due to conflicting factors. To ensure the positive definiteness of the $\boldsymbol{H}$-matrix, a large regularization parameter is selected thereby encouraging very small steps, which significantly impacts the convergence speed. In those cases, one might consider iLQG as a more suitable candidate.

B. Simulation

B.1. One-dimensional Dynamics

First, we consider one-dimensional nonlinear stochastic system of the form

$$dx = (\alpha cos(x) + u)\, dt + x^2 dw \tag{71}$$

where the parameter α controls the degree of instability of the system. It is noted how the Brownian noise enters in a multiplicative fashion with the nonlinear state dynamics. The terms of the linearized dynamics required for the SDDP algorithm are obtained easily $A_t = 1 - \alpha sin(x)\delta t$ and $B_t = \delta t$. By computing the quadratic approximation of the dynamics, it is clear that $\tilde{\boldsymbol{\mathcal{N}}} = \tilde{\boldsymbol{\mathcal{G}}} = \tilde{\boldsymbol{\mathcal{L}}} = \tilde{\boldsymbol{\mathcal{Z}}} = \tilde{\boldsymbol{\mathcal{U}}} = 0$, while the remaining non-zero terms are $\tilde{\boldsymbol{\mathcal{F}}} = -4\sigma^2\delta t\, V_{xx}\, x^2$, $\tilde{\boldsymbol{\mathcal{S}}} = -2\sigma^2\delta t\, V_{xx}\, x^3$ and $\tilde{\boldsymbol{\mathcal{M}}} = -4\sigma^2\delta t\, V_{xx}\, x^2$. The goal is to find an optimal policy to steer the state x from the initial state $x_0 = 0$ to the desired target state $x(T) = 3$, while being subject to the stochastic disturbance. The cost function is defined as $J^\pi(\boldsymbol{x}, t) = \mathbb{E}\left[\int_{t=0}^{T} R \cdot u(\tau)^2\, d\tau + \phi_N(\boldsymbol{x}(T))\right]$ with the terminal cost being $\phi_N(\boldsymbol{x}(T)) = Q_f \cdot (x(T) - x_{des,T})^2$. The weighting matrices were selected as $Q_f = 10$ and $R = 10^{-3}$. Again, it is noted that the defined optimal control problem has only control-dependent running cost resulting in no penalization for the state x. The time step is chosen as 20 ms and the time horizon is set to 3 s. The optimal control input and the corresponding optimal state trajectory are shown in Figure 1.

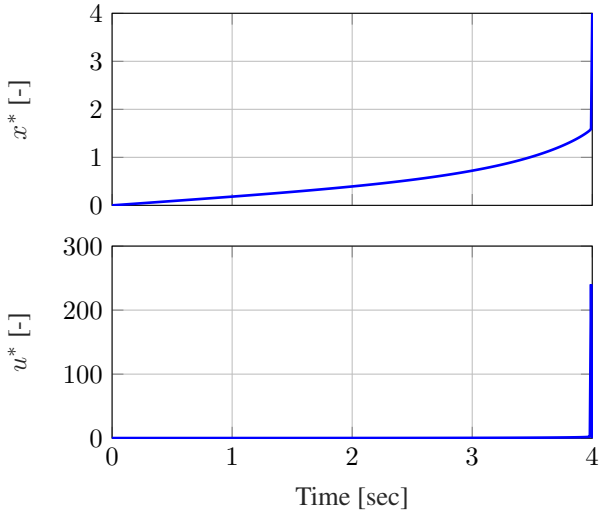

Figure 1: Optimal state and control trajectories for one-dimensional dynamics.

Quadratic Convergence is preserved as the regularization scheme remains inactive and the algorithm converges in five iterations. As the running cost is only control-dependent, the optimal control is approximately zero everywhere except at the end, where the control input is triggered in order for x to reach the desired target.

B.2. Simple Inverted Pendulum

Secondly, the SDDP is tested on a 2 DOF nonlinear stochastic model of an inverted pendulum

$$dx = \begin{pmatrix} x_2 \\ 4sin(x_1) + u \end{pmatrix} dt + \begin{pmatrix} 0 \\ \beta u \end{pmatrix} dw, \tag{72}$$

where the states denote $x_1 = \theta$ and $x_2 = \dot{\theta}$. The stochastic dynamics are control-dependent and represent measurement noise. The measurement noise parameter is set to $\beta = 0.04$. The matrices of the linearized dynamics are given

$$A_t = \begin{pmatrix} 1 & 1 \\ 4cos(x_1) & 1 \end{pmatrix} \delta t$$
$$B_t = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \delta t$$

as well as all terms required for the SDDP

$$\boldsymbol{\mathcal{F}} = \begin{pmatrix} -4sin(x_1) & 0 \\ 0 & 0 \end{pmatrix} \delta t$$
$$\tilde{\boldsymbol{\mathcal{Z}}} = \sigma^2\delta t \begin{pmatrix} 0 & \beta \end{pmatrix}^T V_{xx} \begin{pmatrix} 0 \\ \beta \end{pmatrix}$$
$$\tilde{\boldsymbol{\mathcal{U}}} = \sigma^2\delta t \begin{pmatrix} 0 & \beta \end{pmatrix}^T V_{xx} \begin{pmatrix} 0 \\ \beta u \end{pmatrix}.$$

The remaining terms $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{Z}} = \tilde{\boldsymbol{\mathcal{M}}} = \tilde{\boldsymbol{\mathcal{N}}} = \tilde{\boldsymbol{\mathcal{G}}} = \tilde{\boldsymbol{\mathcal{F}}} = \tilde{\boldsymbol{\mathcal{L}}} = \tilde{\boldsymbol{\mathcal{S}}} = 0$ equate to zero. The goal is to 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). The cost function is defined as $J^{\pi}(\boldsymbol{x}, t) = \mathbb{E}\left[\int_{t=0}^{T} R \cdot u(\tau)^2 \, d\tau + \phi_N(\boldsymbol{x}(T))\right]$ with the terminal cost being $\phi_N(\boldsymbol{x}(T)) = (x(T) - x_{des,T})^T Q_f (x(T) - x_{des,T})$. The weighting matrices were selected as $Q_f = \begin{pmatrix} 10 & 0 \\ 0 & 0 \end{pmatrix}$ and $R = 10^{-2}$. The time step is selected as $\delta t = 20\,\text{ms}$ and the time horizon is $T = 4\,\text{s}$. The optimal state trajectories with the optimal input sequence are shown in Figure 2.

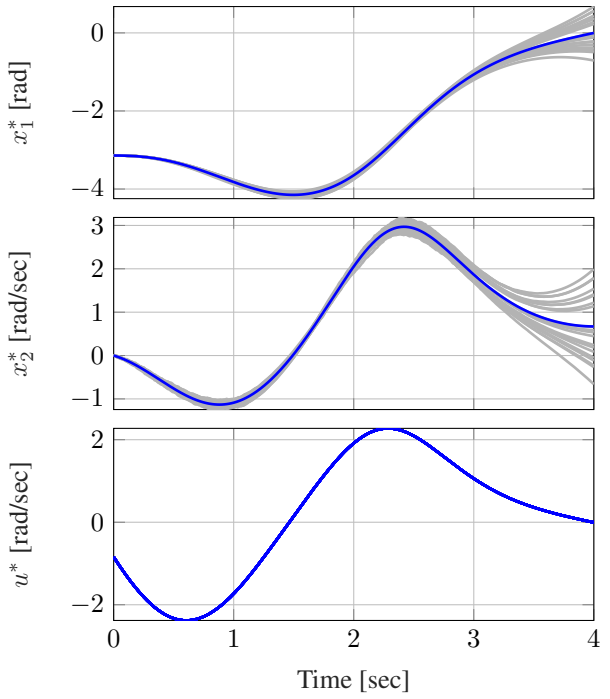

Figure 2: Multiple realizations of the optimal states trajectories (grey) and the optimal control trajectory for the simple inverted pendulum model. The state trajectories in blue is the mean over 500 samples.

The convergence plot is presented in Figure 3.

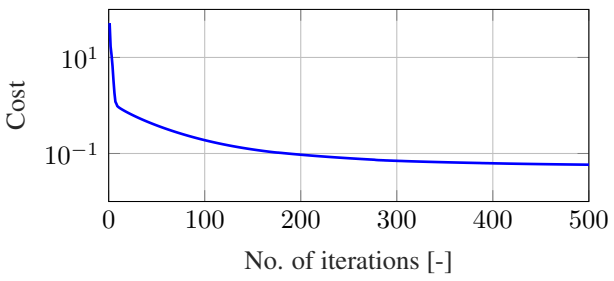

Figure 3: Convergence plot for the simple inverted pendulum.

B.3. Simple Parafoil Dynamics

Lastly, we consider a high-dimensional parafoil homing trajectory optimization problem. Let $x_1(t) = x(t)$ be the position in x-direction, $x_2(t) = y(t)$ the position in y-direction, $x_3(t) = \theta(t)$ the turn angle and $x_4(t) = z(t)$ the position in z-direction. Assuming no disturbance is acting on the parafoil, a simple 4-DOF can be constructed using the following set of differential equations

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \frac{dx_3}{dt} \\ \frac{dx_4}{dt} \end{bmatrix} = \begin{bmatrix} v \cdot cos(x_3(t)) \\ v \cdot sin(x_3(t)) \\ u(t) \\ -r \end{bmatrix}, \tag{73}$$

where v denotes the horizontal velocity. For realistic values, we assume $v = 15\frac{m}{s}$. This system is weakly actuated as only the heading direction of the wind can be manipulated. The above model can be reduced to a 3-DOF by decoupling the equation for x_4. Furthermore, we extend the obtained reduced order model to include wind gusts as an external disturbance acting on both x_1 and x_2

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \frac{dx_3}{dt} \end{bmatrix} = \begin{bmatrix} v \cdot cos(x_3(t)) + w_x(t) \\ v \cdot sin(x_3(t)) + w_y(t) \\ u(t) \end{bmatrix}, \tag{74}$$

where $w_x(t)$ and $w_y(t)$ denote the wind intensity in $\frac{m}{s}$ in both x- and y-direction, respectively. We assume that $w_x(t)$ and $w_y(t)$ are both generated by a Dryden wind turbulence model, where normally distributed, white noise signals are passed through the following filter

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

We treat the turbulence length scale L, the turbulence intensity σ^2 as tuning parameters. These tuning parameters are functions of the altitude h. Assuming a low-altitude level (< 1000 feet), realistic values are summarized in Table 2.

Table 2: Realistic Values for L and σ

Turbulence length scale	Turbulence intensity
L = h (h: Altitude in feet)	$\sigma^2 = 0.1 \cdot W_{20}$

Hereby, denotes W_{20} the wind speed at 20 feet (6 m) in $\frac{m}{s}$. Typically for light turbulence, the wind speed at 20 feet is 15 knots ($\approx 30\ \frac{m}{s}$); for moderate turbulence, the wind speed is 30 knots ($\approx 60\ \frac{m}{s}$), and for severe turbulence, the wind speed is 45 knots ($\approx 90\ \frac{m}{s}$).

To include the wind filter in the overall model, it is required to obtain a transfer function $G(s)|_{s=\mathrm{i}\omega}$, such that $\Phi(\omega) = |G(\mathrm{i}\omega)|^2$. The obtained transfer function is

$$G(s) = \sigma \cdot \sqrt{\frac{L}{\pi v}} \cdot \frac{1 + \cdot\sqrt{3}\frac{L}{v}s}{1 + \left(\frac{L}{v}s\right)^2}.$$

A state space realization of order 2 can be obtained resulting in

$$\frac{d\xi(t)}{dt} = A_\xi \cdot \xi(t) + B_\xi \cdot z(t)$$
$$w(t) = C_\xi \cdot \xi(t),$$

where $\xi(t) \in \mathbb{R}^{2\times 1}$ is a vector of unknown states, $z(t) \in \mathbb{R}^{1\times 1}$ is the gaussian white noise and w(t) denotes the wind intensity. For simplification purposes, we assume constant and identical system matrices A_ξ, B_ξ and C_ξ for both x- and y-direction. An extension to the obtained model would be to incorporate a Kalman Filter, estimating the system matrices of the turbulence model.

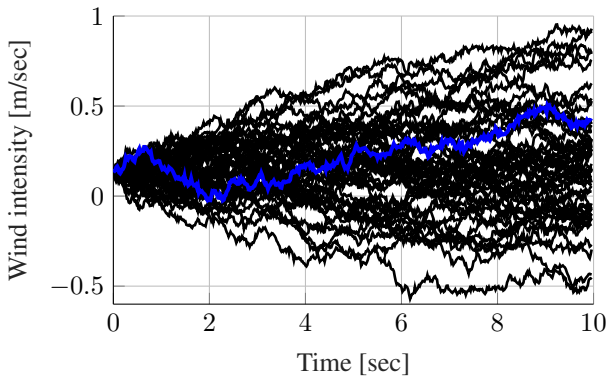

Figure 4: Instances of the wind profiles generated by the Dryden Filter, with one realization highlighted in blue.

The overall stochastic nonlinear 7-DOF model can now be formulated. Again, we let $x_1(t) = x(t)$ be the position in x-direction, $x_2(t) = y(t)$ the position in y-direction and $x_3(t) = \theta(t)$ the turn angle. Furthermore, we redefine $x_4(t) = \xi_{x,1}(t)$ and $x_5(t) = \xi_{x,2}(t)$ to be the unknown states of the first filter generating disturbance in x-direction. Similarly we let $x_6(t) = \xi_{y,1}(t)$ and $x_7(t) = \xi_{y,2}(t)$ be the unknown states of the second filter generating disturbance in y-direction. The overall model is given independent of the state space realization of the wind filter

$$\begin{bmatrix} dx_1 \\ dx_2 \\ dx_3 \\ dx_4 \\ dx_5 \\ dx_6 \\ dx_7 \end{bmatrix} = \begin{bmatrix} v \cdot cos(x_3(t)) + C_{\xi_{1,1}} x_4(t) + C_{\xi_{1,2}} x_5(t) \\ v \cdot sin(x_3(t)) + C_{\xi_{1,1}} x_6(t) + C_{\xi_{1,2}} x_7(t) \\ u(t) \\ A_{\xi_{1,1}} x_4(t) + A_{\xi_{1,2}} x_5(t) \\ A_{\xi_{2,1}} x_4(t) + A_{\xi_{2,2}} x_5(t) \\ A_{\xi_{1,1}} x_6(t) + A_{\xi_{1,2}} x_7(t) \\ A_{\xi_{2,1}} x_6(t) + A_{\xi_{2,2}} x_7(t) \end{bmatrix} \delta t + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ B_{\xi,1} & 0 \\ B_{\xi,2} & 0 \\ 0 & B_{\xi,1} \\ 0 & B_{\xi,2} \end{bmatrix} dw. \tag{75}$$

It is noted that the obtained stochastic differential equation is found to have stochastic dynamics of order zero. We provide the derivative terms required for the SDDP

$$A_t = \begin{pmatrix} 0 & 0 & -vsin(x_3) & C_{1,1} & C_{2,1} & 0 & 0 \\ 0 & 0 & vcos(x_3) & 0 & 0 & C_{1,1} & C_{2,1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{1,1} & A_{1,2} & 0 & 0 \\ 0 & 0 & 0 & A_{2,1} & A_{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_{1,1} & A_{1,2} \\ 0 & 0 & 0 & 0 & 0 & A_{2,1} & A_{2,2} \end{pmatrix}$$

$$B_t = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T \delta t$$

$$\mathcal{F} = \begin{pmatrix} 0 & 0 & -vcos(x_3)V_{x,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & -vsin(x_3)V_{x,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

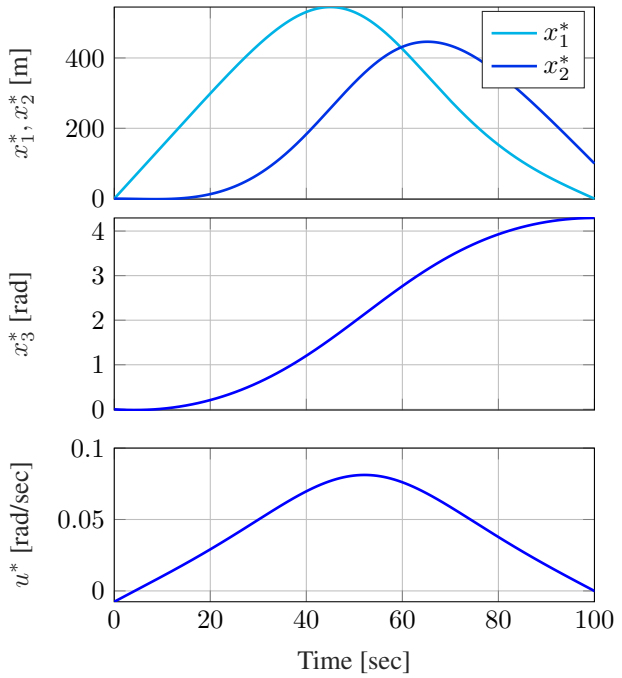

Figure 5: Upper plot presents the mean of the states x_1 and x_2 indicating the location in the x-y-plane. Middle plot shows the mean of the state x_3 representing the angle. Bottom plot exhibits the optimal turn rate.

and the remaining terms are zero. The goal is to find a control input sequence to steer the the parafoil from the initial state ($x_{10} = 0$, $x_{20} = 0$) to the state ($x_T = 0$, $x_T = 100$). The cost function is defined as $J^\pi(\boldsymbol{x}, t) = \mathbb{E}\left[\int_{t=0}^{T} R \cdot u(\tau)^2 \, d\tau + \phi_N(\boldsymbol{x}(T))\right]$ with the terminal cost being $\phi_N(\boldsymbol{x}(T)) = (x(T) - x_{des,T})^T Q_f (x(T) - x_{des,T})$. The weighting matrix Q_f was selected to penalize only the states x_1 and x_2 with the following weights $Q_{f,11} = 1$ and

$Q_{f,22} = 1$, while the running penalty was chosen to be $R = 10^{-2}$. The time step is selected as $\delta t = 20\,\mathrm{ms}$ and the time horizon is $T = 100\,\mathrm{s}$. During simulation, it was observed how the SDDP converged slowly due to inaccuracies from the model approximation. Thus, we have reduced the order of the expanded dynamics to include only first-order terms (κ = 0), which resulted in much quicker convergence. The mean of the optimal state trajectories as well as the obtained optimal control input are shown in Figure 5.

VI. Limitations

This section conducts a theoretical analysis regarding the limitations of SDDP and evaluates the proposed algorithm for general nonlinear stochastic systems.

First, in [8] the author claims that the cubic and quartic correction terms emerging during the computation of second order term of the cost-to-go cancel out. In fact, these terms are completely neglected in the author's computation as the author utilizes a quadratically expanded Bellman equation, which effectively lead us to neglect these higher order dynamics. Inclusion of such terms would require a fourth order expansion of the Bellman equation.

Secondly, we consider the parafoil homing trajectory example. The stochastic differential equation only has constant stochastic dynamics $F(x, u) = F$. This leads all correction terms arising in the derivatives of the state-action value function to be zero. This means that in the case of constant stochastic disturbance the proposed algorithm is reduced to the classical DDP and stochastic dynamics are treated as additive noise, which is not considered in the optimization. The "blindness" of these approaches to additive noise thus still remains.

Thirdly, the proposed SDDP framework incorporates stochastic disturbances by introducing stage-wise correction terms to the state-action value function derivatives in order to optimize the control input. These correction terms do not convey any information about future disturbances leaving the algorithm blind to them.

More importantly, due to its model-based nature the proposed SDDP algorithm does not take into account model uncertainties, which are ubiquitous in stochastic systems. One way to consider these model inaccuracies is to move the analysis to the belief space as in [6]. This would improve also computational complexity for high-dimensional application tasks as regularization of covariance matrices is simpler due to their positive semi-definiteness.

Lastly, the hard terminal constraint $\mathbb{E}[x(T)] = x_T$ imposed by the SDDP might not be suitable for all stochastic systems, especially systems with large uncertainties. Replacing these hard constraints with soft constraints on the moments of the terminal state would allow for a wider application to stochastic systems.

VII. Conclusion

In summary, this report presents an analysis of a modified version of DDP for general nonlinear stochastic systems, as described in [8]. Previous works related to stochastic systems are compared to the proposed SDDP. The report provides a conceptual derivation of the DDP algorithm, drawing upon the Bellman principle, in a concise manner. Furthermore, an extension to stochastic systems is derived, highlighting the fundamental distinctions from the classical DDP approach. The computation of the expectation terms is examined in detail. Additionally, unavoidable implementational aspects were thoroughly examined, with one of these aspects being the integration of a line search technique. Another significant aspect discussed extensively is the regularization, where we present the different methods found in the literature, assess their effectiveness and evaluate their trade-offs. Finally, the developed algorithm is evaluated using three models with varying degrees of model orders and stochastic disturbances. Notably, one of the models involves a 7-degree-of-freedom parafoil homing trajectory optimization problem. Through the practical applications, the capabilities and limitations of the algorithm are demonstrated. Building upon the simulations, a theoretical analysis of the limitations of SDDP is conducted, and potential improvements for future work are proposed.

Appendices

A. Derivation of the Backward-pass Equations

In order to establish the expressions for the backward-pass, we substitute the optimal control policy from Eqn.(14) in the expanded Q-function and simplify the obtained expression to arrive at the following expression

$$\tilde{Q}^*(\boldsymbol{x}_k) \approx Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k^*) + \delta\boldsymbol{x}^\mathsf{T}[Q_{\boldsymbol{x}} - Q_{\boldsymbol{xu}}Q_{\boldsymbol{xu}}^{-1}Q_{\boldsymbol{u}}] + \frac{1}{2}\delta\boldsymbol{x}^\mathsf{T}[Q_{\boldsymbol{x}} - Q_{\boldsymbol{xu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{ux}}]\delta\boldsymbol{x}. \tag{76}$$

According to the Bellman equation in discrete form, the above expression is equivalent to the value function evaluated for state $\boldsymbol{x}_k$. Per our previous assumption of a second order approximation of the value function in Eqn.(21), the coefficients with same power are equated and we obtain the equations governing the backpropagation.

$$\boxed{\begin{aligned} V(\boldsymbol{x}_k) &= Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) - \frac{1}{2}Q_{\boldsymbol{u}}^\mathsf{T}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{u}} \\ V_{\boldsymbol{x}}(\boldsymbol{x}_k) &= Q_{\boldsymbol{x}} - Q_{\boldsymbol{xu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{u}} \\ V_{\boldsymbol{xx}}(\boldsymbol{x}_k) &= Q_{\boldsymbol{xx}} - Q_{\boldsymbol{xu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{ux}} \end{aligned}}$$

REFERENCES

[1] D. Jacobson and D. Mayne. *Differential Dynamic Programming*. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company, 1970.

[2] L.-Z. Liao and C. Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control*, 36(6):692–706, 1991.

[3] L.-Z. Liao and C. Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control*, 36(6):692–706, 1991.

[4] L.-Z. Liao and C. A. Shoemaker. Advantages of differential dynamic programming over newton"s method for discrete-time optimal control problems. 1992.

[5] C.-K. Ng, L.-Z. Liao, and D. Li. A globally convergent and efficient method for unconstrained discrete-time optimal control. *Journal of Global Optimization*, 23:401–421, 08 2002.

[6] Y. Pan and E. Theodorou. Probabilistic differential dynamic programming. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[7] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, 2012.

[8] E. Theodorou, Y. Tassa, and E. Todorov. Stochastic differential dynamic programming. In *Proceedings of the 2010 American Control Conference*, pages 1125–1132, 2010.

[9] E. Todorov and W. 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.*, pages 300–306 vol. 1, 2005.