

CRANFIELD UNIVERSITY

ON PARK

**TRAJECTORY OPTIMIZATION WITH SPARSE
GAUSS-HERMITE QUADRATURE**

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING

PhD

Academic Year: 2017–2022

Supervisors: Prof. Hyo-Sang Shin, Prof. Antonios Tsourdos
13th April 2023

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING

PhD

Academic Year: 2017–2022

ON PARK

**Trajectory Optimization with Sparse Gauss-Hermite
Quadrature**

Supervisors: Prof. Hyo-Sang Shin, Prof. Antonios Tsourdos
13th April 2023

© Cranfield University 2023. All rights reserved. No part of
this publication may be reproduced without the written
permission of the copyright owner.

Abstract

This thesis aims to innovate knowledge on the trajectory optimization problem for aerospace applications by adopting a numerical integration in optimal control designs. Quadrature point scheme is considered to substitute a derivative of an optimal cost-to-go function and system dynamics approximated by Gaussian Quadrature rule.

Based on the Differential Dynamic Programming (DDP) algorithm, a sequence of optimal control input is derived by sparsely chosen quadrature points with the Smolyak's rule over an exponential weighting function on the Gaussian quadrature, called as Sparse Gauss-Hermite Quadrature (SGHQ). The sampling points propagated via system dynamics computes the mean and covariance of a probability distribution of the value function avoiding a numerical differentiation in the DDP. This approach improves an accuracy and robustness against the numerical calculation in the highly nonlinear environment despite the less number of the quadrature points compared with the fully composed by Gauss-Hermite Quadrature. Besides, the number of sampling points can be determined by Smolyak's rule definitely, while the other sampling point-based approach chooses by heuristic/empirical method such as a trial and error. The proposed method is carried out and validate via numerical simulation: a fixed-wing aircraft controller and missile guidance.

Considering a stochastic environment and control policy, trajectory optimization problem can be extended to a stochastic trajectory optimization and maximum entropy problem by adding an entropy term in a deterministic trajectory optimization problem: a Guided Policy Search (GPS). This entropy term in a stochastic problem enables to prevent a control policy from falling into local minima by maximizing exploration in the given

unknown environment, and improves robustness to respond to a potentially flexible environment. A local policy is updated by DDP framework where the SGHQ-DDP can be implemented to find a mean and covariance of policy distribution by solving the soft Bellman equation. The numerical simulation shows the feasibility of the SGHQ-GPS method under an unknown system dynamics under the fitted model.

Keywords

Optimal Control, Trajectory Optimization, Differential Dynamic Programming, Constrained Differential Dynamic Programming, Unscented Dynamic Programming, Sparse Gauss-Hermite Quadrature, Reinforcement Learning, Guided Policy Search

Contents

Abstract	iii
Contents	v
List of Figures	viii
List of Tables	ix
List of Abbreviations	x
Acknowledgements	xi
1 Introduction	1
1.1 Background	1
1.1.1 Trajectory Optimization	1
1.1.2 Reinforcement Learning	4
1.2 Motivation and Research Objective	6
1.3 List of Publications	8
2 Constrained Differential Dynamic Programming	11
2.1 Differential Dynamic Programming	11
2.1.1 Notation and Problem Definition	12
2.1.2 Backward Pass	13
2.1.3 Forward pass	16
2.2 Constrained Differential Dynamic Programming	18
2.2.1 Augmented Lagrangian Method	20

2.2.2	Constrained Differential Dynamic Programming	21
2.3	Flexible Final Time Differential Dynamic Programming	23
2.4	Numerical Simulation	29
2.4.1	A Fixed-wing aircraft	29
2.4.2	Missile guidance	34
2.5	Conclusion	39
3	Sparse Gauss-Hermite Quadrature CDDP	41
3.1	Unscented Dynamic Programming	41
3.2	Sparse Gauss-Hermite Quadrature	45
3.2.1	Numerical Integration	45
3.2.2	Gauss-Hermite Quadrature	47
3.2.3	Sparse Grid	50
3.3	SGHQ DDP	58
3.4	SGHQ CDDP	59
3.5	Numerical Simulation	61
3.5.1	A fixed-wing aircraft	62
3.5.2	Missile guidance	67
3.6	Conclusion	70
4	SGHQ Guided Policy Search	72
4.1	Stochastic Policy Optimization	72
4.2	Guided Policy Search	77
4.2.1	GPS: Local Policy Update	79
4.2.2	GPS: Global Policy Update	82
4.2.3	GPS: Parameter Update for KL-Divergence	82
4.3	Sparse Gauss-Hermite Quadrature Guided Policy Search	83
4.4	Numerical Simulation	85
4.5	Conclusion	89
5	Conclusions	90
5.1	Conclusion	90

5.2	Future Works	91
A	Dynamics	93
A.1	Fixed-wing Aircraft Dynamics	93
A.2	Missile dynamics	95
B	Guided Policy Search	98
B.1	Gaussian Function	98
B.2	Gaussian Mixture Model	99
B.3	Linear Regression and Conditional Gaussian Function	102
	Bibliography	106

List of Figures

1.1	Reinforcement Learning with an actor critic structure [1]	5
1.2	Thesis outline	9
2.1	Backward and Forward Passes of the DDP	19
2.2	The F/A-18 High Alpha Research Vehicle (HARV) developed by NASA's Dryden flight research center [2]	30
2.3	Result of the (C)DDP F/A-18 control problem	32
2.4	Cost comparison between the DDP and CDDP	33
2.5	Geometry of the kinematics model in the three dimensional plane	34
2.6	Comparison result of the (C)DDP missile guidance	37
2.7	Cost comparison result of the (C)DDP missile guidance	37
2.8	Comparison result of the FFT-(C)DDP missile guidance: state and input trajectories	38
2.9	Comparison result of the FFT-(C)DDP missile guidance: cost and time . .	39
3.1	The example of the Gauss-Hermite Quadrature rule, $n = 2, n_I = 3$	50
3.2	The comparison of the GHQ and the SGHQ rule. $n = 2, L_v=3$	57
3.3	Process comparison of the DDP and SGHQ-DDP	60
3.4	Comparison result of the CDDP algorithms of F/A-18: flat-spin recovery .	64
3.5	Cost comparison of F/A-18: Flat-spin recovery	65
3.6	Comparison result of the FFT-CDDP algorithms of F/A-18: flat-spin re- covery	66
3.7	3D Trajectory of F/A-18: flat-spin recovery	67
3.8	Comparison result of the FFT-CDDP F/A-18: flat-spin recovery	67
3.9	Comparison result of the FFT algorithms in missile guidance: state and input trajectories	69
3.10	Comparison result of the FFT algorithms in missile guidance: cost and time	69
4.1	Process comparison of the GPS and SGHQ-GPS	85
4.2	Trajectory of the GPS for the state and control input of F/A-18	87
4.3	Cost comparison of the GPS	88

List of Tables

1.1	Trajectory Optimization: Direct method vs Indirect method [3, 4]	4
2.1	Simulation parameters of the (C)DDP for F/A-18: optimal regulator . . .	31
2.2	Simulation initial/desired conditions of the F/A-18: optimal regulator . .	31
2.3	Simulation initial/desired conditions of the missile	35
2.4	Simulation parameters of the (C)DDP for the missile	36
2.5	Comparison result of the both (C)DDP and FFT-(C)DDP	38
2.6	Comparison of DDP and its variants	40
3.1	Number of points for SGHQ rule	56
3.2	Number of points for GHQ rule	56
3.3	Simulation initial/desired conditions of the F/A-18: flat spin recovery . .	62
3.4	Simulation parameters of the CDDP for F/A-18: flat spin recovery	63
3.5	Comparison result of F/A-18: flat-spin recovery	67
3.6	Simulation initial/desired conditions of the missile	68
3.7	Comparison result of the missile guidance	68
3.8	Comparison of DDP and sampling point-based DDPs	71
4.1	Simulation parameter of the GPS for F/A-18: optimal regulator	86
4.2	Cost comparison: GPS and SGHQ-GPS	88
A.1	Atmospheric constants	96

List of Abbreviations

MDP	Markov Decision Process
RL	Reinforcement Learning
LQR	Linear Quadratic Regulator
iLQR	iterative LQR
DP	Dynamic Programming
DDP	Differential Dynamic Programming
CDDP	Constrained Differential Dynamic Programming
FFT	Flexible Final Time
UT	Unscented Transform
UDP	Unscented Dynamic Programming
GQ	Gauss Quadrature
GHQ	Gauss-Hermite Quadrature
SGHQ	Sparse Gauss-Hermite Quadrature
GPS	Guided Policy Search
SGHQ-	Sparse Gauss-Hermite Quadrature Guided Policy Search
GPS	

Acknowledgements

Firstly, I would like to express my gratitude to my supervisors, Prof. Hyo-Sang Shin, and Prof. Antonios Tsourdos for their support and supervision during my PhD course. I also thank you Dr. Minguk Seo for the discussions and comments during the review meeting. I acknowledge the finance support from the educational department of the Korean Government.

Chapter 1

Introduction

1.1 Background

1.1.1 Trajectory Optimization

Trajectory optimization is a designing framework to control complicated or nonlinear control applications based on the optimal control theory [1, 5, 6]. The aim of the trajectory optimization is to design a system trajectory for a finite-time horizon minimizing (or maximizing) the corresponding performance index (or cost function) while satisfying given constraint conditions. It generates state and control trajectories from a given initial condition to a target (or desired) condition for a given time horizon, evaluates the performance index and finds a best system trajectory in a *off-line* manner. Over the last decades, trajectory optimization problems are widely utilized in engineering area such as robotics, aerospace and other applications. Robotics area such as multibody arm, Multi Joint dynamics with Ccontact (MuJoCo) physics engine and humanoid robots utilizes to optimize a desired trajectory avoiding obstacles in the workspace [7], to perform a complex tasks using an online trajectory optimization [8, 9], and to reach a moving target with a control limits [10]. Especially, the trajectories of missile, Unmanned Aerial Vehicle (UAV) and autonomous vehicle are planned from a developing a low-level controller to guidance law. Missile trajectories are designed to intercept a stationary target satisfying constraint

conditions such as physical control limit or collision/no-fly zone avoidance [11, 12, 13]. Aircraft and multicopter-type drone are operated by computing trajectories while minimizes its stabilizing error and control effort [14, 15]. A rocket ascent trajectory to deliver a payload [16] and re-entry mission are calculated with a Genetic Algorithm [17], and multi objective trajectory optimization incorporated with an evolutionary algorithm are proposed to consider requirements in real-world [18]. Further details are described in the [19, 20] with the understanding of the numerical optimization and its advantages.

The solver for the trajectory optimization can be categorized: a direct method and indirect method [3, 4, 21], which is summarized briefly in Table 1.1. A direct method parameterizes both state and control input trajectories, and converts the original problem into a Nonlinear Programming Problem (NLP) by discretizing the optimization problem with an implicit integration scheme based on interpolation, which is known as *transcription*. The primary advantage of direct method is that it is much easier to extend to general problems involving path constraints, discontinuous states and control inputs inequalities. The popular approach of the direct method is a direct collocation method of which basic idea is to discretize both the state and control input trajectories into a finite node explicitly and then interpolate between the nodes using Hermitian splines approximated by a sequence of polynomial segments [22, 23, 24]. The trapezoidal and Hermite-Simpson quadrature rule are generally utilized for the collocation point of the numerical interpolation and then the NLP solver is properly used for example, Sequential Quadratic Programming, (SQP), Sparse Nonlinear OPTimizer (SNOPT), and Interior Point OPTimizer (IPOPT) [25, 26, 27]. Lastly, the Pseudo-spectral method, or *global collocation*, is applied in higher polynomials to achieve an exponential rate of convergence (*spectral* convergence rate) using orthogonal polynomial, typically Chebyshev or Gauss-Legendre-based method [4, 28].

Whereas, indirect methods utilizes the first and second order optimality of the necessary and sufficient conditions analytically, then discretizes and solves these conditions and problem numerically. The main advantage of an indirect method has generally more

accuracy and reliable error estimation compared with a direct method and guarantees the local optimality of the solution by the first order necessary condition. One of the most successful indirect trajectory optimization solver in recent years is Differential Dynamic Programming (DDP). DDP is the second order shooting method introduced by Mayne and Jacobson [29, 30] utilizing a local model and a quadratic approximation of the value function. It decomposes the trajectory optimization problem into a one-step sub-optimization problem by Bellman's principle of optimality from the classic Dynamic Programming (DP). DDP deals with the shortcoming of the DP, i.e. the curse of dimensionality by a second order approximation of the system and the value function trajectories around the current value. Its assumption is valid near the current nominal trajectory and can achieve fast quadratic convergence rate [31], hence, it computes a local linear-feedback controller (or control correction) around the trajectory by the first-order optimality condition.

iterative Linear Quadratic Regulator (iLQR), one of variant DDP algorithms, uses the only first-order derivative of the explicit dynamics in the Hessian approximation of the value function [32, 33], which is similar to the relationship between Newton's method and Gauss-Newton method using the only Jacobian [34] and numerically advantageous. Due to its computational efficiency by approximating the cost function and dynamics as a quadratic and linear function, a nonlinear optimal control sequence problem is formulated and solved recursively by LQR problem.

To solve a trajectory optimization problem in an online manner, (or so-called online trajectory optimization), Model Predictive Control (MPC) framework is implemented to utilize robustness to moderate model errors and adaptation ability [35]. MPC is a recursive and iterative online method by iLQR for the predefined finite-horizon optimization problem [8], therefore, MPC is also called as Receding Horizon Control (RHC) method.

These approaches are repeated iteratively and re-optimizes the given performance index at every time steps as improvement in the control input affects on the environment or system dynamics change, however, the demand for computation is challenging as a dimension of the state is increased and the system state needs to be observed fully. To deal

Table 1.1: Trajectory Optimization: Direct method vs Indirect method [3, 4]

Direct method [21, 23]	Remarks	- Parameterize/discretize the state and/or input - Convert into a nonlinear optimization problem. - Solve the Nonlinear Programming Problem, (NLP) by SQP, SNOPT, IPOPT. - Ex. Direct Collocation (local/global collocation)
	Pros/Cons	- Easy to set up and solve the problem. - High degree of flexibility/robustness. - Less precision and high sensitivity. - Computational burden.
Indirect method [1, 36]	Remarks	- Find the necessary/sufficient optimality condition. - Optimize the discretized state and input. - Ex. Differential Dynamic Programming (DDP), iLQR, (N)MPC
	Pros/Cons	- Guarantee a local optimality and better precision. - Difficulties in the mathematical optimality condition. - Less degree of flexibility.

with this practical issue, model-based algorithm is widely used because control policy is approximated by neural network that can restrict its inputs to observable state variables, and system model can be learned which is used in the policy learning. Most of the model-based policy search is carried out by Reinforcement Learning.

1.1.2 Reinforcement Learning

Reinforcement Learning (RL) is one of machine learning algorithms how an agent takes action in a given environment (or system) to maximize a cumulative reward [37]. An agent of RL interacts with the corresponding environment to learn what/how to do an action yielding a best/optimal reward (Fig. 1.1), where an environment is formalized as Markov Decision Process (MDP) which describes a sequential decisions of dynamical system:

$$\begin{aligned}
P_{x,x'}^u &= \Pr\{x'|x, u\} \\
&= \Pr\{x_{t+1}|x_t, u_t, x_{t-1}, u_{t-1}, \dots, x_1, u_1\}
\end{aligned} \tag{1.1}$$

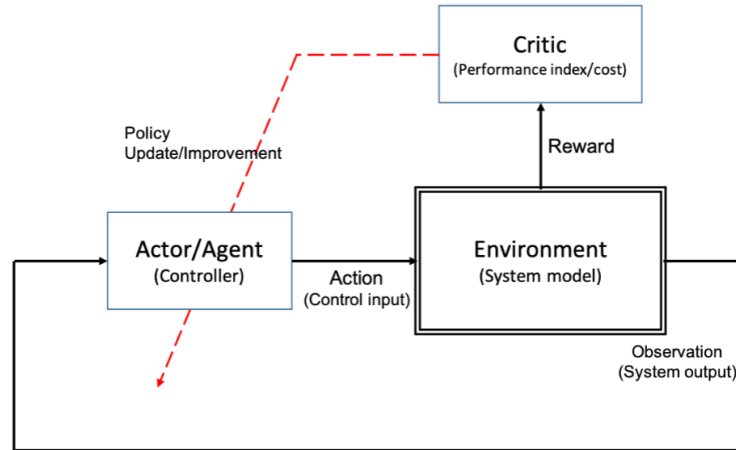


Figure 1.1: Reinforcement Learning with an actor critic structure [1]

It means that transition probability to the next state x' only depends on the current state x and action u not on the history of x and u . The RL is provided by on the Dynamic Programming (DP) based on the Bellman's *principle of optimality* stating that an optimal policy has the property that whatever the previous decision variable has been chosen, the next decision must constitute an optimal policy with regard to the current state resulting from the previous decision [38]. Using those two property, i.e. MDP and DP, a complex problem can be reformulated as a sub-problem satisfying the Bellman equation giving us *recursive relation* which decomposes the value function (or optimal cost-to-go, $V(x)$) into an optimal action of the current step and the optimal value of the remaining step: $V(x) = r(x, u) + V(x')$. Many RL approaches also make use of the action-value function, $Q(x, u)$ satisfying a recursive form of: $Q(x, u) = r(x, u) + Q(x', u')$.

RL can be categorized by two approaches: a model-free RL and a model-based RL. Model-free RL approach finds a optimal policy or solves a Bellman equation online by Temporal Difference (TD) learning for uncertain or partially unknown systems by sensor measurement. The value function is estimated by interacting with the system by bootstrapping using Q-learning and SARSA methods [37, 39]. Especially, recent attempts have been made on RL framework to design optimal control laws for unknown systems using an Adaptive Dynamic Programming (ADP). The main idea of ADP/RL is to find a optimal policy by observing system outputs or fully known state variables. By estimat-

ing the value function by critic, a RL agent (or actor) enables adopting a control policy from a repeated interaction and the value is improved to drive the Bellman error to zero [40, 41, 42, 43], which is an Actor-Critic (AC) structure depicted in Fig.1.1.

When a system model is known, RL just utilizes this system information to find a control policy by solving DP problem, however, issues related to model-based approach is that it is difficult to collect a number of data tuples of state-action pairs when dealing with a real physical system. To handle this issue, model-based approach must requires a model learning process reducing a sample complexity significantly by using system identification tools such as a linear regression with Gaussian Processes (GPs) [44, 45, 46].

In control policy search by RL, the objective is to find a generalized control policy $\pi(u_i|x_i)$ of an agent in an environment even in a stochastic system. Most of policy search uses neural network to approximate the value function or control policy. When a dimension of an environment is large or continuous, it is hard to learn the value function within a finite sampling tuples, furthermore, the parameters of control policy space also become large of infinite. To overcome this drawbacks, a Guided Policy Search (GPS) has been proposed [47, 48] utilizing the trajectory optimization algorithm (classic DDP). GPS solves the DDP problem to find a local control policy by restricting the system trajectory from the previous one by Kullback Liebler Divergence (KL-Divergence) and train a global policy, Adopting KL-Divergence maximizes the information entropy of the trajectory distribution that ensures the fully and widely exploration in the trajectory and prevents falling into a local minima, which improves stability of the policy search algorithm and reduces training iteration. The details of GPS and its variant algorithms are summarized in [47, 49].

1.2 Motivation and Research Objective

Classic DDP is a representative approach of an indirect solver for trajectory optimization using the Bellman's principle of optimality with quadratic convergence properties.

The second order approximated DP by Taylor expansion around a local state and input trajectory finds a locally optimal solution, however, the numerical differentiation in the derivative computation has a poor performance of the convergence especially in a highly nonlinear case or relatively large time step. To avoid a mathematical computation of dynamics derivative and value function, a deterministic sampling-based approach motivated by Unscented Transform (UT) [50] from the Unscented Kalman Filter (UKF) was proposed [51], which is named Unscented Dynamic Programming (UDP) [14]. Instead of UDP, this research utilizes a new quadrature points-based method motivated from 'Sparse Gauss-Hermite Quadrature' (SGHQ) filter to approximate a probability distribution of the gradient and Hessian of the value function substituting the numerical differentiation in classic DDP.

The sampling points for the Gauss-Hermite quadrature are sparsely chosen by Smolyak's rule that alleviates the curse of dimensionality in the point selection for the multi-dimensional system where the number of sampling points is exponentially increased as the system dimension is expanded. Furthermore, the number of the SGHQ point can be adjusted by Smolyak's rule (by level selection) so that the numerical integration accuracy can be improved while the computation load is mitigated compared with the fully-gridded Gauss-Hermite quadrature. The sampling points replace a probability distribution by the sampled mean and covariance of the propagated points directly without derivatives computation, hence an approximation inaccuracy issue in the dynamics by complexity such as a singularity or unstable points can be avoided. In addition, the sampling time issue in terms of discretization errors related to the validity near the perturbation assumption can also be resolved since the nonlinear dynamics is utilized directly without the dynamics Jacobian.

The overall aim of this thesis is to innovate knowledge on the DDP for aerospace applications by a new quadrature points-based method with a SGHQ-rule. This research focuses on developing SGHQ-based DDP algorithm to improve the mathematical stability in the gradient and Hessian computation by avoiding a numerical differentiation.

Lastly, SGHQ-based GPS is proposed which is also derivative free method for a policy search in a stochastic environment. The individual objectives are addressed as follows:

- **O1** Understanding of traditional trajectory optimization and reinforcement learning for control policy search.
- **O2** Understanding of the classic DDP with/without constraint condition and flexible final time algorithm to adjust the finite time horizon of the optimal control problem.
- **O3** Design a SGHQ-based DDP algorithm and its variants considering constraint condition and flexible final time.
- **O4** Implement SGHQ-DDP in the aerospace applications to validate the effectiveness of the SGHQ-rule comparing with other DDPs.
- **O5** Understanding of GPS algorithm and design a SGHQ-GPS algorithm.
- **O6** Verify the feasibility of SGHQ-GPS via numerical simulation.

1.3 List of Publications

A list of papers during the PhD research are reported below.

Book Chapter

O. Park, H. S. Shin and A. Tsourdos, "Nonlinear Model Predictive Control-based Line Tracking Guidance", Advanced in Aerospace Control, 2022. (accepted)

Journal Papers

O. Park, H. S. SHin, A. Tsourdos, and H. Shaoming, 'Trajectory Optimization with Sparse Gauss-Hermite Quadrature', (Temporary title, prepared)

Conference Papers

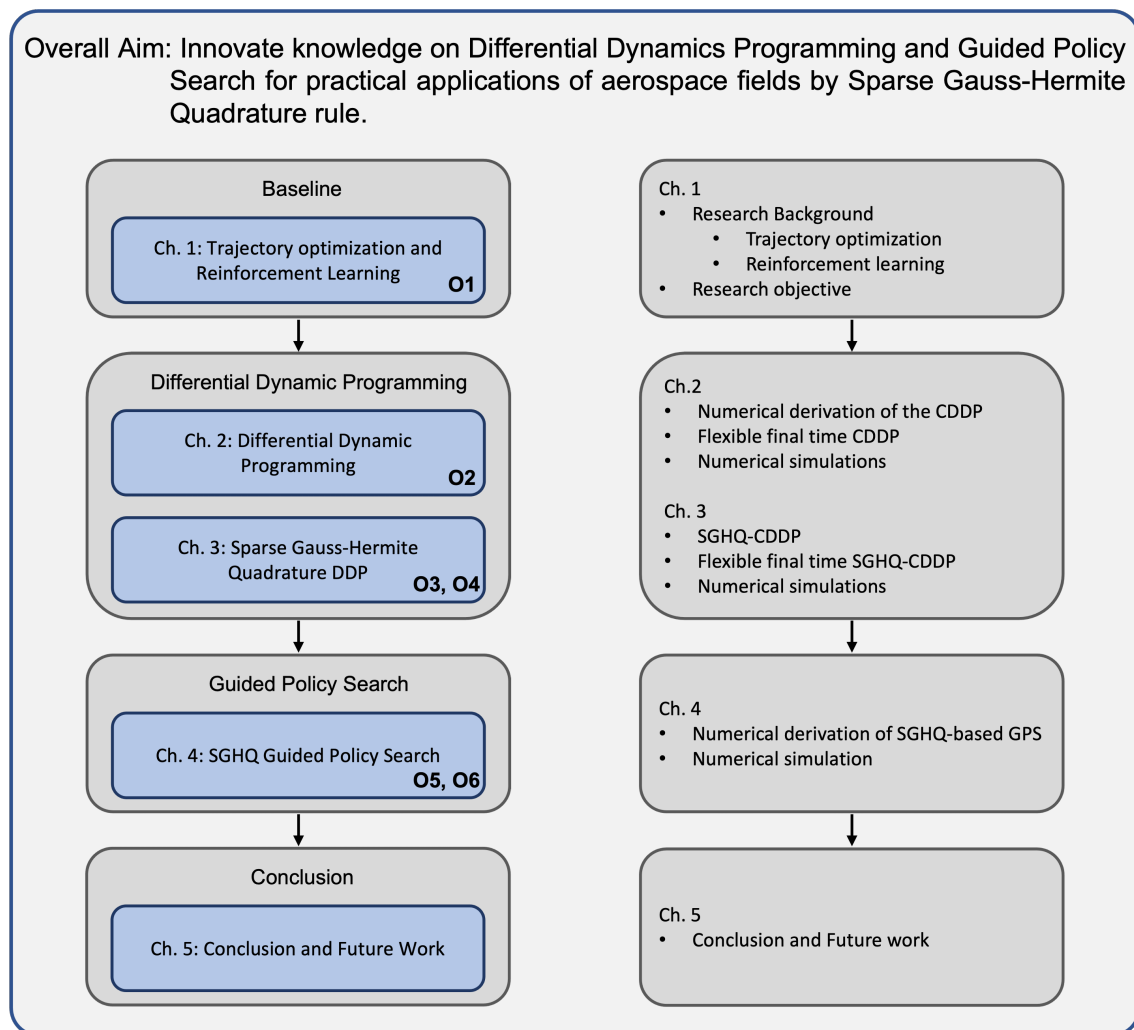


Figure 1.2: Thesis outline

O. Park, H. S. Shin and A. Tsourdos, "Optimal Control using Sparse Gauss-Hermite Quadrature Differential Dynamic Programming", the 13th ASian Control Conference (ASCC), 2022.

O. Park, H. S. Shin, H. I. Lee and A. Tsourdos, "Optimal and Adaptive Control Design using Recursive Least Square with a New Exponential Forgetting Factor", 9th International Conference on Robot Intelligence Technology and Application (RITA), 2021.

O. Park, H. S. Shin and A. Tsourdos, "Linear Quadratic Tracker with Integrator using

Integral Reinforcement Learning", 2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS). IEEE, 2019.

O. Park, H. S. Shin and A. Tsourdos, "Evolutionary Game Theory based Multi-Objective Optimization for Control Allocation of Over-actuated System", International Federation of Automatic Control (IFAC) Automatic Control in Aerospace, 2019.

Chapter 2

Constrained Differential Dynamic Programming

2.1 Differential Dynamic Programming

The Differential Dynamic Programming (DDP) is one of the most powerful and widely used trajectory optimization algorithms to solve a nonlinear optimal control problem by optimizing trajectories of the state and control input based on the optimal control theory [8, 10]. The DDP is also known as a second-order Dynamic Programming (DP) algorithm to compute a local control policy or control input sequence over the given horizon in a recursive way by using the Bellman's optimality principle with a Markov property, which overcomes the shortcoming of the classic DP, i.e. the curse of dimensionality. The main objective of the DDP is to find a decision variable, or an input sequence, by the first and second order derivatives of the dynamics and value function approximated by Taylor series expansions around the nominal state and compute a locally optimal control sequence for the smooth discrete time systems in a recursive manner. Firstly, an initial input sequence properly chosen generates a state trajectory and then the value function and action-value function are approximated as a quadratic function around the nominal trajectory by backward computation from the final time with a control policy update. The

"forward pass" evaluates the computed control policy in the "backward pass" by deviation from the nominal trajectory and decides whether or not to accept the updated policy. This process is repeated until the convergence condition is satisfied [52].

This chapter describes a details of the classic DDP with/without nonlinear constraint conditions by utilizing a Lagrangian method with a quadratic penalty function, or so-called augmented Lagrangian method. Lastly, the final time free DDP proposed recently is also introduced briefly to incorporate into classic DDP.

2.1.1 Notation and Problem Definition

Consider a nonlinear dynamics with a given initial condition, x_1 :

$$\dot{x} = f_c(x_i, u_i) \quad (2.1)$$

where $x \in R^n$ is a state vector and $u \in R^m$ is an control input vector and f_c is a nonlinear and continuous time dynamics. The subscription i means a time step, i.e. $i = [1, 2, \dots, N]$ is equivalent to $i = [t_1, t_2, \dots, t_N]$. The associated performance index, or the so-called cost function, is the sum of the running cost $l(x_i, u_i)$ and the final step cost $l_f(x_N)$:

$$J(x, U) = \sum_{i=1}^{N-1} l(x_i, u_i) + l_f(x_N) \quad (2.2)$$

The main objective is to find a control input sequence $U = [u_1, u_2, \dots, u_{N-1}]$ given the initial state x_{t_1} to minimize the associated performance index:

$$\begin{aligned} U^* &= \min_U J(x_1, U) \\ &= \sum_{i=1}^{N-1} l(x_i, u_i^*) + l_f(x_N) \end{aligned} \quad (2.3)$$

The (state) value function, or the so-called the optimal cost-to-go, at time i can be defined from x_i by substituting the optimal control sequence $U^* = [u_i^*, u_{i+1}^*, \dots, u_{i+N-1}^*]$ into Eq.

(2.2):

$$V(x_i) = \min_{U_i^*} J(x, U_i^*) \quad (2.4)$$

The value function of the final time can be defined as $V_{t_N} = l_f(x_{t_N})$. The principle of the dynamic programming reduces the finite horizon problem to a sequence of minimization over a one-step problem by proceeding backward in a time:

$$V(x_i) = \min_{u_i} l(x_i, u_i) + V(f(x_i, u_i)) \quad (2.5)$$

2.1.2 Backward Pass

From the final time cost $l_f(x_N)$, the Bellman's Principle of Optimality can be applied to the given performance index to show the recurrent form of the value function:

$$V(x_N) = l_f(x_N) \quad (2.6)$$

$$\begin{aligned} V(x_i) &= \min_{u_i} \{l(x_i, u_i) + V(x_{i+1})\} \\ &= \min_{u_i} Q(x_i, u_i) \end{aligned} \quad (2.7)$$

where $Q(x_i, u_i) = l(x_i, u_i) + V(x_{i+1})$ is known as the action-value function or Q-function and Eq. (2.7) is the Bellman equation which shows the relation between the current time i and the next time $i + 1$. However, the issue associated with the trajectory optimization is difficult to solve as the value function $V(x_i)$ typically has the nonlinearity of the dynamics. This nonlinearity issue can be prevented by applying the Taylor Series expansion near the

nominal trajectory $(x, u)_i = (\bar{x}, \bar{u})_i + (\delta x, \delta u)_i$:

$$\begin{aligned}
V(x_i) &= V(\bar{x}_i + \delta x_i) \\
&\approx V(\bar{x}_i) + V(\delta x_i) \\
&= V(\bar{x}_i) + \frac{\partial V^T}{\partial x_i}(x_i - \bar{x}_i) + \frac{1}{2}(x_i - \bar{x}_i)^T \frac{\partial^2 V}{\partial x^2}(x_i - \bar{x}_i) \\
\therefore V(\delta x_i) &= \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_i \end{bmatrix}^T \begin{bmatrix} 0 & V_x^T \\ V_x & V_{xx} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \end{bmatrix}
\end{aligned} \tag{2.8}$$

where $\delta x_i = x_i - \bar{x}_i$ is a small perturbation in state and $(\cdot)_{\{\cdot\},\{\cdot\}}$ is a partial derivative with respect to its subscript component. Similar to the value function, action-value function can also be expressed as:

$$\begin{aligned}
Q(x_i, u_i) &= Q(\bar{x}_i + \delta x_i, \bar{u}_i + \delta u_i) \\
&\approx Q(\bar{x}_i, \bar{u}_i) + Q(\delta x_i, \delta u_i) \\
&= Q(\bar{x}_i, \bar{u}_i) + \frac{\partial Q^T}{\partial x_i}(x_i - \bar{x}_i) + \frac{\partial Q^T}{\partial u_i}(u_i - \bar{u}_i) \\
&\quad + \frac{1}{2}(x_i - \bar{x}_i)^T \frac{\partial^2 Q}{\partial x^2}(x_i - \bar{x}_i) + \frac{1}{2}(x_i - \bar{x}_i)^T \frac{\partial^2 Q}{\partial x \partial u}(u_i - \bar{u}_i) \\
&\quad + \frac{1}{2}(u_i - \bar{u}_i)^T \frac{\partial^2 Q}{\partial u \partial x}(x_i - \bar{x}_i) + \frac{1}{2}(u_i - \bar{u}_i)^T \frac{\partial^2 Q}{\partial u^2}(u_i - \bar{u}_i) \\
\therefore Q(\delta x_i, \delta u_i) &= \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}
\end{aligned} \tag{2.9}$$

where $\delta x_i = x_i - \bar{x}_i$, $\delta u_i = u_i - \bar{u}_i$ are small perturbations in state and input, and $Q_{xu} = Q_{ux}^T$. From the (2.9), the optimal control correction term can be computed by taking a

partial derivative with respect to the control input δu_i :

$$\begin{aligned}\delta u_i &= \frac{\partial Q(\delta x_i, \delta u_i)}{\partial \delta u_i} \\ &= Q_u + Q_{ux}\delta x_i + Q_{uu}\delta u_i \\ \therefore \delta u_i &= -Q_{uu}^{-1}(Q_{ux}\delta x_i + Q_u)\end{aligned}\tag{2.10}$$

In order to guarantee the invertibility of Q_{uu} , the regularization term is added as follows:

$$\begin{aligned}\delta u_i &= -(Q_{uu} + \rho I)^{-1}(Q_{ux}\delta x_i + Q_u) \\ &= K_i\delta x_i + d_i\end{aligned}\tag{2.11}$$

where K_i and d_i are feedback gain and feedforward gain, respectively. The recurrent form of the value function, (2.7), can also be represented as a function of small perturbations of the nominal pair $(\bar{x}_i, \bar{u}_i)$:

$$\begin{aligned}Q(\bar{x}_i + \delta x_i, \bar{u}_i + \delta u_i) &= l(\bar{x}_i + \delta x_i, \bar{u}_i + \delta u_i) + V(\bar{x}_{i+1} + \delta x_{i+1}) \\ &= l(\bar{x}_i + \delta x_i, \bar{u}_i + \delta u_i) + V(f_d(\bar{x}_i + \delta x_i, \bar{u}_i + \delta u_i)) \\ Q(\delta x_i, \delta u_i) &\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}^T \begin{bmatrix} 0 & l_x^T & l_u^T \\ l_x & l_{xx} & l_{xu} \\ l_u & l_{ux} & l_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix} \\ &\quad + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}^T \begin{bmatrix} 0 & (f_{dx}^T V'_x)^T & (f_{du}^T V'_x)^T \\ f_{dx}^T V'_x & f_{dx}^T V'_{xx} f_{dx} & f_{dx}^T V'_{xx} f_{du} \\ f_{du}^T V'_x & f_{du}^T V'_{xx} f_{dx} & f_{du}^T V'_{xx} f_{du} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}\end{aligned}\tag{2.12}$$

where $x_{i+1} = f_d(x_i, u_i)$ is the discretized continuous-time dynamics by simple Euler method and $(\cdot)'$ means the value at the next time step. Thus, the following equations are satisfied:

$$\begin{aligned}
Q_x &= l_x + f_{dx}^T V'_x \\
Q_u &= l_u + f_{du}^T V'_x \\
Q_{xx} &= l_{xx} + f_{dx}^T V'_{xx} f_{dx} + V'_x f_{dxx} \\
Q_{xu} &= l_{xu} + f_{dx}^T V'_{xx} f_{du} + V'_x f_{dxu} \\
Q_{uu} &= l_{uu} + f_{du}^T V'_{xx} f_{du} + V'_x f_{duu}
\end{aligned} \tag{2.13}$$

The high order tensor term, i.e. the second order derivatives of the dynamics $f_{d..}$, are often ignored due to the expensive computation, resulting in the iLQR [32, 33]. By substituting the obtained control input (2.11) into (2.9) shows the following equation:

$$\begin{aligned}
Q(\delta x_i, \delta u_i) &= \begin{bmatrix} 1 \\ \delta x_i \\ K_i \delta x_i + d_i \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \\ K_i \delta x_i + d_i \end{bmatrix} \\
&= V(\delta x_i)
\end{aligned} \tag{2.14}$$

the Jacobian and Hessian of the V and expected change in cost, ΔV are:

$$\begin{aligned}
V_x &= Q_x + K_i^T Q_{uu} d_i + K_i^T Q_u + Q_{xu} d_i \\
V_{xx} &= Q_{xx} + K_i^T Q_{uu} K_i + K_i^T Q_{ux} + Q_{xu} K_i \\
\Delta V &= d_i^T Q_u + \frac{1}{2} d_i^T Q_{uu} d_i
\end{aligned} \tag{2.15}$$

The process of the backward pass is summarized in Algorithm 1.

2.1.3 Forward pass

Once the backward pass is completed, the computed optimal control policy is evaluated by generating a new trajectory X from the nominal trajectory $(\bar{X}, \bar{U})$, which is called as a

Algorithm 1 DDP: Backward Pass

```

1: Initialize  $V_{x_N} = l_{x_N}$ ,  $V_{xx_N} = l_{xx_N}$  from Eq. (2.6)
2: for  $i = N-1:-1:1$  do
3:   Q function update: Eq.(2.13), Eq.(2.14)
4:   if  $Q_{uu}$  is invertible then
5:     Compute the control policy: Eq.(2.11)
6:     Update the value function: Eq.(2.15)
7:   else
8:     Increase  $\rho$  by Eq.(2.20) and go to line 2
9:   end if
10: end for

```

forward pass:

$$\begin{aligned}
x_1 &= \bar{x}_1 \\
u_i &= \bar{u}_i + \delta u_i \\
&= \bar{u}_i + \alpha d_i + K_i(x_i - \bar{x}_i) \\
x_{i+1} &= f(x_i, u_i)
\end{aligned} \tag{2.16}$$

where x_i , u_i are the updated trajectories and α is a backtracking line-search parameter initially set on the feedforward gain, d_i , by 1 and then reduced iteratively to make ensure an adequate reduction in cost, because the optimal control correction (2.11) is computed based on the gradient descent approach by partial derivative with respect to the control input δu_i . A line-search parameter, α must be added in the forward pass to satisfy decreasing in cost.

$$z = \frac{J(\bar{X}, \bar{U}) - J(X, U)}{-\Delta V(\alpha)} \tag{2.17}$$

where

$$\Delta V(\alpha) = \sum_{i=1}^{N-1} d_i^T Q_{ui} + \frac{1}{2} \sum_{i=1}^{N-1} d_i^T Q_{uu} d_i \tag{2.18}$$

and z is the cost reduction ratio comparing the actual and expected reduction. If z is in the interval $[\beta_1 \ \beta_2]$, which is usually set by $[1e-4 \ 10]$, the updated control correction δu_i^* in the backward pass and the new trajectories are accepted. If not, the line-search parameter

Algorithm 2 DDP: Forward Pass

```

1: while do
2:   Initialize line-search parameter  $\alpha=1$ .
3:   Initialize nominal state  $x_{t_1} = \bar{x}_{t_1}$ .
4:   for  $i = 1:N$  do
5:     Evaluate the updated control input Eq.(2.11) using Eq.(2.16)
6:   end for
7:   Compute the new cost  $J$ : Eq.(2.2)
8:   if  $\bar{J} - J > 0$  then
9:     break
10:  else
11:    Update line-search parameter,  $\alpha$  by Eq.(2.19)
12:  end if
13: end while

```

α is updated by

$$\alpha = \gamma_\alpha \alpha \quad (2.19)$$

where $0 < \gamma_\alpha < 1$ is the scaling factor of the line-search parameter typically chosen by $\gamma_\alpha = 0.5$, which is described in the Algorithm 2 [10, 52]. Lastly, the regularization parameter ρ is also adjusted by regulating parameter $1 \leq \gamma_\rho \leq 2$.

$$\rho = \begin{cases} \gamma_\rho \rho & Q_{uu} \text{ is not invertible} \\ \frac{1}{\gamma_\rho} \rho & Q_{uu} \text{ is invertible} \end{cases} \quad (2.20)$$

These 'backward' and 'forward' passes are repeated (Fig. 2.1) until the terminal conditions of the DDP are satisfied such as cost tolerance, feedforward gain, and maximum iteration (Algorithm 3) [52].

2.2 Constrained Differential Dynamic Programming

One of the advantages of the optimization problems is that an equality or inequality constraint can be involved in the problem formulation for which a function $f(x, u)$ is to be minimized or maximized subject to satisfy a constraint function, $g(x, u)$. The DDP can also incorporate the constraint function into the problem formulation [52, 53, 54]

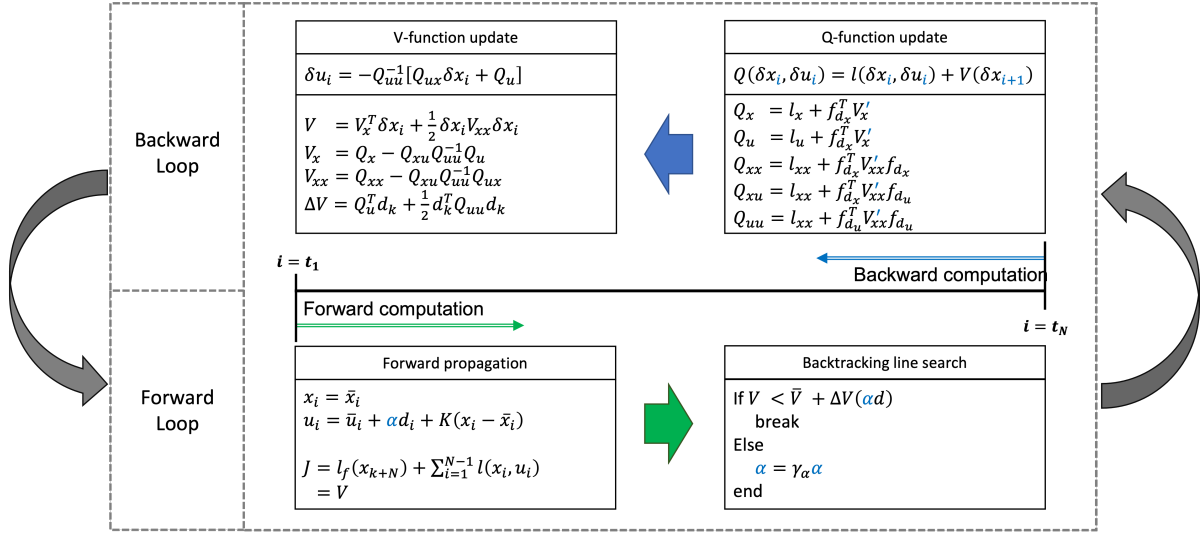


Figure 2.1: Backward and Forward Passes of the DDP

Algorithm 3 Differential Dynamic Programming

```

1: for iter = 1 : MaxIter do
2:   Differentiation the cost and dynamics near the nominal trajectory
3:   for i = 1:N do
4:     Differentiate:  $l(x_i, u_i), f(x_i, u_i)$ 
5:   end for
6:   DDP: Backward Pass by Algorithm 1
7:   DDP: Forward Pass by Algorithm 2
8:   if Forward Pass is done then
9:     Update nominal trajectories  $\bar{X} \leftarrow X, \bar{U} \leftarrow U$ ,
10:    Decrease  $\rho$  by Eq.(2.20)
11:    if  $\bar{J} - J < \varepsilon$  then
12:      break
13:    end if
14:     $\bar{J} \leftarrow J$ 
15:  else
16:    Increase  $\rho$  by Eq.(2.20)
17:    if  $\rho > \rho_{max}$  then
18:      break
19:    end if
20:  end if
21: end for

```

To deal with the constraint condition, the Lagrangian-based method is generally utilized in the equality constraint problem. In addition to the Lagrangian method, a penalty term can also be considered for the inequality condition, which is called as an Augmented Lag-

rangian method. By applying the second order Taylor series expansions to the associated function with equality or inequality constraints expressed by a function of quadratic form, the approximates are then implemented on the deviation of the nominal trajectory.

2.2.1 Augmented Lagrangian Method

Consider the following optimization problem with constraint conditions that is represented as a function of state and input vectors:

$$\begin{aligned} J(x, U) = \min_u \quad & \sum_{i=1}^{N-1} l(x_i, u_i) + l_f(x_N) \\ \text{s.t.} \quad & g_i(x_i, u_i) \leq 0 \end{aligned} \quad (2.21)$$

where $g(x, u) \in R^p$ is a nonlinear equality or inequality constraint function enforcing on . The augmented performance index, so-called Lagrangian function, can be defined with a penalty term and a Lagrange multiplier associated with the constraint condition [55]:

$$\begin{aligned} J_a(X, U, \lambda, \mu) &= \sum_{i=1}^{N-1} l(x_i, u_i) + \lambda_i^T g_i(x_i, u_i) + \frac{1}{2} g_i(x_i, u_i)^T I_{\mu_i} g_i(x_i, u_i) \\ &\quad + l_f(x_N) + \lambda_N^T g_N(x_N) + \frac{1}{2} g_N(x_N)^T I_{\mu_N} g_N(x_N) \\ &= \sum_{i=1}^{N-1} l_a(x_i, u_i, \lambda_i, \mu_i) + l_{af}(x_N, \lambda_N, \mu_N) \end{aligned} \quad (2.22)$$

where $\lambda \in R^p$ is a Lagrange multiplier, and $I_{\mu} \in R^{p \times p}$ is a diagonal matrix with a penalty parameter, $\mu \in R^p$. The activation condition of the inequality $g(x_i, u_i)$ is:

$$I_{\mu_i}^j = \begin{cases} 0 & g_i^j(x_i, u_i) \leq 0 \wedge \lambda_i^j = 0, \quad j \in R^p \\ \mu_i^j & \text{otherwise} \end{cases} \quad (2.23)$$

where superscript j is the j th component of $g_i(x_i, u_i)$. The update for the λ can be derived

based on the necessary condition:

$$\lambda_i^{j+1} = \begin{cases} \lambda_i^j + \mu_i^j g(x_i, u_i) & \text{equality constraint} \\ \max(0, \lambda_i^j + \mu_i^j g(x_i, u_i)) & \text{inequality constraint} \end{cases} \quad (2.24)$$

and the penalty parameter is updated:

$$\mu_i^{j+1} = C_\mu \mu_i^j \quad (2.25)$$

where a scheduling factor, C_μ , is generally chosen by $C_\mu > 1$.

2.2.2 Constrained Differential Dynamic Programming

To develop Constrained DDP, (CDDP), the Lagrangian function, Eq.(2.22) is used to compute the new value function and action-value function:

$$J_a(X, U, \lambda, \mu) = \sum_{i=1}^{N-1} l_a(x_i, u_i, \lambda_i, \mu_i) + l_{af}(x_N, \lambda_N, \mu_N) \quad (2.26)$$

where the augmented running cost and the final time cost are defined:

$$\begin{aligned} l_a(x_i, u_i, \lambda_i, \mu_i) &= l(x_i, u_i) + \lambda_i^T g(x_i, u_i) + \frac{1}{2} g(x_i, u_i)^T I_{\mu_i} g(x_i, u_i) \\ l_{af}(x_N, \lambda_N, \mu_N) &= l_f(x_N) + \lambda_N^T g(x_N) + \frac{1}{2} g(x_N)^T I_{\mu_N} g(x_N) \end{aligned} \quad (2.27)$$

With the augmented cost $\hat{l}$, the value function and action-value function can be modified as:

$$\begin{aligned} J_a(X, U^*, \lambda, \mu) &= \sum_{i=t}^{N-1} l_a(x_i, u_i^*) + l_{af}(x_N) \\ \therefore J_a(X, U^*, \lambda, \mu) &= V_a(x_i) \\ &= l_a(x_i, u_i^*) + V_a(x_{i+1}) \end{aligned} \quad (2.28)$$

$$Q_a(x_i, u_i^*) = l_a(x_i, u_i^*) + V_a(x_{i+1}) \quad (2.29)$$

Algorithm 4 CDDP: Backward Pass

```

1: Initialize  $V_{ax_N} = l_{af_{x_N}}, V_{axx_N} = l_{af_{xx_N}}$  from Eq.(2.28).
2: for  $i = N-1:-1:1$  do
3:    $Q_a$  function update: Eq.(2.31) using Eq.(2.27)
4:   if  $Q_{auu}$  is invertible then
5:     Compute the control policy: Eq.(2.32)
6:     Update the value function: Eq.(2.33)
7:   else
8:     Increase  $\rho$  by Eq.(2.20) go to line 2
9:   end if
10: end for

```

Similar to the DDP, the new value function and action-value function can also be approximated by the first and second order Taylor series expansions:

$$\begin{aligned}
 V_a(x_i + \delta x_i) &\approx V_a(x_i) + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_i \end{bmatrix}^T \begin{bmatrix} 0 & V_{ax}^T \\ V_{ax} & V_{axx} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \end{bmatrix} \\
 Q_a(x_i + \delta x_i, u_i + \delta u_i) &\approx Q_a(x_i, u_i) + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}^T \begin{bmatrix} 0 & Q_{ax}^T & Q_{au}^T \\ Q_{ax} & Q_{axx} & Q_{axu} \\ Q_{au} & Q_{aux} & Q_{auu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_i \\ \delta u_i \end{bmatrix}
 \end{aligned} \tag{2.30}$$

Thus, the following equations are also satisfied:

$$\begin{aligned}
 Q_{ax} &= l_{ax} + f_{dx}^T V_{ax}' \\
 Q_{au} &= l_{au} + f_{dx}^T V_{ax}' \\
 Q_{axx} &= l_{axx} + f_{dx}^T V_{axx}' f_{dx} + V_{ax}' f_{dxx} \\
 Q_{axu} &= l_{axu} + f_{dx}^T V_{axx}' f_{du} + V_{ax}' f_{dxu} \\
 Q_{auu} &= l_{auu} + f_{du}^T V_{axx}' f_{du} + V_{ax}' f_{duu}
 \end{aligned} \tag{2.31}$$

The modification gives the optimal control correction:

$$\begin{aligned}
 \delta u_i &= -(Q_{auu} + \rho I)^{-1} (Q_{aux} \delta x_i + Q_{ax}) \\
 &= K_{ai} \delta x_i + d_{ai}
 \end{aligned} \tag{2.32}$$

Algorithm 5 CDDP: Forward Pass

```

1: while do
2:   Initialize line-search parameter  $\alpha=1$ .
3:   Initialize nominal state  $x_{t_1} = \bar{x}_{t_1}$ .
4:   for  $i = 1:N$  do
5:     Evaluate the updated control input by Eq.(2.32) using Eq.(2.16)
6:   end for
7:   Compute the new cost  $J$  by Eq.(2.26)
8:   if  $\bar{J}_a - J_a > 0$  then
9:     break
10:  else
11:    Update line-search parameter,  $\alpha$  by Eq.(2.19)
12:  end if
13: end while

```

and the updated value function:

$$\begin{aligned}
V_{ax} &= Q_{ax} + K_{ai}^T Q_{auu} d_{ai} + K_{ai}^T Q_{au} + Q_{axu} d_{ai} \\
V_{axx} &= Q_{axx} + K_{ai}^T Q_{auu} K_{ai} + K_{ai}^T Q_{aux} + Q_{axu} K_{ai} \\
\Delta V_a &= d_{ai}^T Q_{au} + \frac{1}{2} d_{ai}^T Q_{auu} d_{ai}
\end{aligned} \tag{2.33}$$

Lastly, the augmented Lagrangian parameter for the constraint condition, λ and μ , are updated using the Eq.(2.24), Eq.(2.25).

2.3 Flexible Final Time Differential Dynamic Programming

Section 2.1 and 2.2 solved a trajectory optimization problem in a fixed final time framework without/with constraint conditions, respectively. The one of the difficulties in the DDP or optimal control problem is that once the time horizon N -step is determined, it can not be handled explicitly. Furthermore, the information of the finite time horizon influencing on the performance must be selected by heuristic manner in advance. To deal with the final time issue in the finite-horizon problem, the static optimization-based approach reformulated from the dynamics optimization problem is derived to find an optimal time

Algorithm 6 Constrained Differential Dynamic Programming

```

1: for  $iter = 1 : \text{MaxIter}$  do
2:   Differentiate the augmented cost and dynamics near the nominal trajectory
3:   for  $i = 1:N$  do
4:     Compute the constraint condition: Eq.(2.23) and Eq.(2.24)
5:     Compute the cost function: Eq.(2.27)
6:     Differentiate:  $l_a(x_i, u_i, \lambda_i, \mu_i)$ ,  $f(x_i, u_i)$ 
7:   end for
8:   CDDP: Backward Pass by Algorithm 4
9:   CDDP: Forward Pass by Algorithm 5
10:  if Forward Pass is done then
11:    Update nominal trajectories:  $X \leftarrow \bar{X}$ ,  $U \leftarrow \bar{U}$ ,
12:    Decrease  $\rho$  by Eq.(2.20)
13:    if  $J_a - \bar{J}_a < \varepsilon$  then
14:      break
15:    end if
16:     $J \leftarrow \bar{J}_a$ 
17:  else
18:    Increase  $\rho$  by Eq.(2.20)
19:    if  $\rho > \rho_{max}$  then
20:      break
21:    end if
22:  end if
23:  for  $i = 1:N$  do
24:    if Constraint condition:  $g(x_i, u_i) \leq 0$  then
25:      update  $\lambda$  by Eq.(2.24)
26:    else
27:      update  $\mu$  by Eq.(2.25)
28:    end if
29:  end for
30: end for

```

increment, which is called as a Flexible Final Time (FFT) algorithm adjusting the final time adaptively [56, 57]. The free final time DDP incorporating the FFT algorithm into the trajectory optimization problem termed as FFT-DDP is presented utilizing the Taylor series approximation on the value function with respect to the final time step motivated from the above observations.

Assume that the value function or cost-to-go at the final time t_N is a function of state and time variable, then the approximation around the nominal trajectory at the final time can

be obtained by Taylor series expansion similar to the Eq.(2.9):

$$\begin{aligned} V(x_N, t_N) &= V(\bar{x}_N + \delta x_N, \bar{t}_N + \delta t_N) \\ &\approx V(\bar{x}_N, \bar{t}_N) + V(\delta x_N, \delta t_N) \end{aligned} \quad (2.34)$$

From the Eq.(2.34), the variation of the $V(\delta x_N, \delta t_N)$ around the $(\bar{x}_N, \bar{t}_N)$ is:

$$V(\delta x_N, \delta t_N) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_N \\ \delta t_N \end{bmatrix}^T \begin{bmatrix} 0 & V_{x_N}^T & V_{t_N}^T \\ V_{x_N} & V_{x_N x_N} & V_{x_N t_N} \\ V_{t_N} & V_{t_N x_N} & V_{t_N t_N} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_{t_N} \\ \delta t_{t_N} \end{bmatrix} \quad (2.35)$$

To find a gradient $(\cdot)_{\{\cdot\}}$ at the final time, t_N , the chain rule is utilized:

$$\begin{aligned} V_{t_N} &= \left(\frac{\partial x_N}{\partial t_N} \right)^T \frac{\partial V}{\partial x_N} \\ &= \dot{x}_N^T \frac{\partial V}{\partial x_N} \\ &= f_c(x_N, u_N)^T V_{x_N} \end{aligned} \quad (2.36)$$

where $\dot{x} = f_c(x, u)$ is a nonlinear continuous dynamics and $u_N = u_{N-1}$ is applied to implement the time derivative of the system at the final time. The Hessian terms $(\cdot)_{\{\cdot\}}$ of the value function can also be found using the matrix calculus, $\frac{\partial}{\partial x}(f^T y) = f_x^T y + y_x^T f$:

$$\begin{aligned} V_{x_N t_N} &= \frac{\partial}{\partial x_N} \left(\frac{\partial V}{\partial t_N} \right) \\ &= \frac{\partial}{\partial x_N} \left\{ f_c(x_N, u_N)^T V_{x_N} \right\} \\ &= \frac{\partial}{\partial x_N} \left\{ f_c(x_N, u_N)^T \right\} V_{x_N} + V_{x_N x_N}^T f_c(x_N, u_N) \end{aligned} \quad (2.37)$$

$$\begin{aligned}
V_{t_N t_N} &= \frac{\partial}{\partial t_N} \left(\frac{\partial V}{\partial t_N} \right) \\
&= \frac{\partial}{\partial t_N} \left\{ f_c(x_N, u_N)^T V_{x_{t_N}} \right\} \\
&= \frac{\partial}{\partial t_N} \left\{ f_c(x_N, u_N)^T \right\} V_{x_N} + V_{x_N t_N}^T f_c(x_N, u_N)
\end{aligned} \tag{2.38}$$

If the continuous dynamics $\dot{x} = f_c(x, u)$ is a time invariant system, the following relation can be satisfied:

$$\frac{\partial}{\partial t_N} \left\{ f_c(x_N, u_N) \right\} = 0 \tag{2.39}$$

Thus, Eq. (2.38) can be rewritten:

$$\begin{aligned}
V_{t_N t_N} &= \frac{\partial}{\partial t_N} \left\{ \cancel{f_c(x_N, u_N)^T} \right\} V_{x_N} + V_{x_N t_N}^T f_c(x_N, u_N) \\
&= V_{x_N t_N}^T f_c(x_N, u_N)
\end{aligned} \tag{2.40}$$

By taking a partial derivative of $V(\delta x_N, \delta t_N)$, Eq.(2.34), with respect to the final time t_N as the control input correction is computed by Eq.(2.10), the increment of the final time δt_N can be updated:

$$\begin{aligned}
\frac{\partial}{\partial t_N} V(\delta x_N, \delta t_N) &= V_{t_N} + V_{t_N x_N} \delta x_N + V_{t_N t_N} \delta t_N \\
&= 0 \\
\therefore \delta t_N^* &= -V_{t_N t_N}^{-1} (V_{t_N x_N} \delta x_{t_N} + V_{t_N}) \\
&= -[V_{t_N t_N} + \rho_{t_N}]^{-1} (V_{t_N x_N} \delta x_{t_N} + V_{t_N})
\end{aligned} \tag{2.41}$$

where δt_N is an optimal correction of the final time, t_N , and $V_{t_N t_N}$, $V_{t_N x_N}$ and V_N are computed from the Eq.(2.38), Eq.(2.37) and Eq.(2.36), respectively. The regularization parameter (ρ_{t_N}) of the $V_{t_N t_N}$ can also be added in the Eq.(2.41) for the numerical stability of

Algorithm 7 Flexible Final Time DDP: Backward Pass

```

1: Initialize  $V_{x_N}, V_{x_N x_N}$  by Eq.(2.44)
2: for  $i=N-1:-1:1$  do
3:   Q function update: Eq.(2.13).
4:   if  $Q_{uu}$  is invertible then
5:     Compute the control policy: Eq.(2.11)
6:     Update the value function: Eq.(2.15)
7:   else
8:     Increase  $\rho$  by Eq.(2.20) go to line 2
9:   end if
10: end for

```

the inversion to ensure the invertibility as Eq.(2.11).

$$\rho_{t_N} = \begin{cases} \gamma_{t_N} \rho_{t_N} & \text{Forward pass is not done} \\ \frac{1}{\gamma_{t_N}} \rho_{t_N} & \text{Forward pass is done} \end{cases} \quad (2.42)$$

Hence, the updated variation of the value function at the final time, $V(\bar{x}_N + \delta x_N, \bar{t}_N + \delta t_N^*)$ is represented by substituting the time increment, Eq.(2.41), back into Eq.(2.34):

$$\begin{aligned} V(\bar{x}_N + \delta x_N, \bar{t}_N + \delta t_N) &\approx V(\bar{x}_N, \bar{t}_N) + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_N^T \\ \delta t_N^T \end{bmatrix}^T \begin{bmatrix} 0 & V_{x_N}^T & V_{t_N}^T \\ V_{x_N} & V_{x_N x_N} & V_{x_N t_N} \\ V_{t_N} & V_{t_N x_N} & V_{t_N t_N} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_N \\ \delta t_N \end{bmatrix} \\ &= V(\bar{x}_N, \bar{t}_N) + [V_{x_N} - V_{x_N t_N} V_{t_N t_N}^{-1} V_{t_N}] \delta x_N \\ &\quad + \frac{1}{2} \delta x_N^T [V_{x_N x_N} - V_{x_N t_N} V_{t_N t_N}^{-1} V_{t_N x_N}] \delta x_N - \frac{1}{2} V_{t_N} V_{t_N t_N}^{-1} V_{t_N} \delta t_N \end{aligned} \quad (2.43)$$

Consequently, the updated Jacobian and Hessian of the value function V at the final time, t_N , becomes

$$\begin{aligned} V_{x_N}^N &= V_{x_N} - V_{x_N t_N} [V_{t_N t_N} + \rho_{t_N}]^{-1} V_{t_N} \\ V_{x_N x_N}^N &= V_{x_N x_N} - V_{x_N t_N} [V_{t_N t_N} + \rho_{t_N}]^{-1} V_{t_N x_N} \end{aligned} \quad (2.44)$$

From the Bellman equation, Eq.(2.7), the following action-value function can also be

given by:

$$\begin{aligned}
Q(x_{N-1}, u_{N-1}) &= l(x_{N-1}, u_{N-1}) + V(x_N, t_N) \\
Q(x_{N-2}, u_{N-2}) &= l(x_{N-2}, u_{N-2}) + V(x_{N-1}) \\
&\vdots \\
Q(x_1, u_1) &= l(x_1, u_1) + V(x_2)
\end{aligned} \tag{2.45}$$

and its Jacobian and Hessian are also satisfied by Eq.(2.9).

Algorithm 8 Flexible Final Time Differential Dynamic Programming

```

1: for  $iter = 1 : \text{MaxIter}$  do
2:   Differentiate the cost and dynamics near the nominal trajectory
3:   for  $i = 1 : N$  do
4:     Differentiate:  $l_a(x_i, u_i), f_c(x_i, u_i)$ 
5:   end for
6:   Backward Pass by FFT-DDP Algorithm 7
7:   Forward Pass by DDP Algorithm 2
8:   if Forward Pass is done then
9:     Compute the time increment:  $\delta t_N$  by Eq.(2.41)
10:    Compute the horizon step increment:  $\delta N = \delta t_N / dt$ 
11:    if  $\delta N > 0$  then
12:      Extend the trajectory:  $x_{i_n+1} = f_d(x_{i_n}, u_{i_n}), i_n = 1, 2, \dots, \delta N$ 
13:      Update the trajectory,  $\bar{X}$  and  $\bar{U}$ , from the expanded trajectory.
14:    else
15:      Truncate the trajectory:  $\bar{X} = [x_1, x_2, \dots, x_{N+\delta N}], \bar{U} = [u_1, u_2, \dots, u_{N-1+\delta N}]$ 
16:    end if
17:    Compute the new cost by  $\bar{X}, \bar{U}$ :  $\bar{J}$ 
18:    Update nominal trajectories:  $X \leftarrow \bar{X}, U \leftarrow \bar{U}$ ,
19:    Decrease  $\rho$  by Eq.(2.20)
20:    Decrease  $\rho_{t_N}$  by Eq.(2.42)
21:    if  $J - \bar{J} < \varepsilon$  then
22:      break
23:    end if
24:     $J \leftarrow \bar{J}$ 
25:  else
26:    Increase  $\rho$  by Eq.(2.20)
27:    Increase  $\rho_{t_N}$  by Eq.(2.42)
28:    if  $\rho > \rho_{max}$  then
29:      break
30:    end if
31:  end if
32: end for

```

2.4 Numerical Simulation

This section provides the result of the DDP algorithm through numerical simulations involving two applications: a control law design for a fixed-wing aircraft and a missile guidance problem. The objective of the first case is to stabilize a fixed-wing aircraft which is a simple optimal regulator problem of the nonlinear system. The feedforward and feedback gains of the aircraft inputs such as δ_{thr} , δ_{ele} , δ_{ail} and δ_{rud} are obtained by the DDP with the physical limit $\delta_{\{\cdot\}}_{min} \leq \delta_{\{\cdot\}} \leq \delta_{\{\cdot\}}_{max}$. The second case assumes that a three dimensional air-to-surface impact-angle guidance problem is considered with a missile dynamics modelled by a kinematic and ideal point mass model. The longitudinal and lateral acceleration commands are designed by DDP algorithm with a bounded acceleration, $|a_{y,z}| \leq |a_{y,z}|_{max}$ with a lower level controller. The classic DDP utilizes the projection method to find a feasible control input to handle a control input constraint by:

$$u = \min[u_{max}, \max[u_{min}, u]] \quad (2.46)$$

All the system information required for the DDP implementation are fully available.

2.4.1 A Fixed-wing aircraft

A fixed-wing aircraft considered in this thesis is F/A-18 High Angle of Attack Research Vehicle (HARV) developed by NASA shown in the Fig. 2.2. This vehicle is introduced for the high Angle of Attack (AOA) region over the post-stall and spin maneuverability of the high AOA flight envelope. The equation of motion for the aircraft is expressed by Newton's second law for rigid-body with aerodynamic model obtained from the NASA technical report [58, 59, 2]. The more details of the F/A-18 and dynamics modelling are described in the Appendix A.1.



Figure 2.2: The F/A-18 High Alpha Research Vehicle (HARV) developed by NASA's Dryden flight research center [2]

Firstly, the state and control input vector for the (C)DDP problem can be defined by:

$$\begin{aligned} x &= [V_T, \alpha, \beta, p, q, r, \phi, \theta]^T \in \mathbb{R}^{n \times 1} \\ u &= [\delta_{thr}, \delta_{el}, \delta_{er}, \delta_a, \delta_r]^T \in \mathbb{R}^{m \times 1} \end{aligned} \quad (2.47)$$

where $[V_T, \alpha, \beta]^T$ are the velocity, Angle of Attack (AOA), side slip angle (or Angle of Sideslip, AOS), $[p, q, r]^T$ are the angular velocity with respect to its body axis and $[\phi, \theta]^T$ are roll and pitch attitude angle, respectively. The control input vector is organized by throttle, left/right elevator, aileron and rudder as with a conventional aircraft. The performance index to stabilize a conventional aircraft to a trim point can be formulated as follows:

$$\begin{aligned} J &= \min_U \frac{1}{2} (x_N - x_d)^T S_N (x_N - x_d) + \frac{1}{2} \sum_{i=1}^{N-1} \{ (x_i - x_d)^T Q (x_i - x_d) + (u_i - u_d)^T R (u_i - u_d) \} \\ \text{s.t. } & u_{min} \leq u \leq u_{max} \end{aligned} \quad (2.48)$$

where $U = [u_1, u_2, \dots, u_{N-1}]^T \in \mathbb{R}^{m \times (N-1)}$ is an input sequence for the given finite horizon N , and x_d is the trim point of the state to be stabilized, x_{tr} and u_d is the corresponding

trim control input, u_{tr} . The physical constraint of the control input can be presented by inequality constraint function $g(x, u)$ by:

$$\begin{aligned} g(x_i, u_i) &= |u_i - u^+| < u^- \\ &= |u_i - u^+| - u^- < 0 \end{aligned} \quad (2.49)$$

where $u^+ = \frac{(u_{max} + u_{min})}{2}$, $u^- = \frac{(u_{max} - u_{min})}{2}$ are used for the mathematical formulation of the inequality constraint meanwhile the classic DDP utilizes the naive clamping by Eq. (2.46). The Q , R , and S_N are the weighting matrices for the state and control input of the running cost and that for the final time state, and the Augmented Lagrangian parameter, λ and μ are also initialized, which are presented in the Table 2.1.

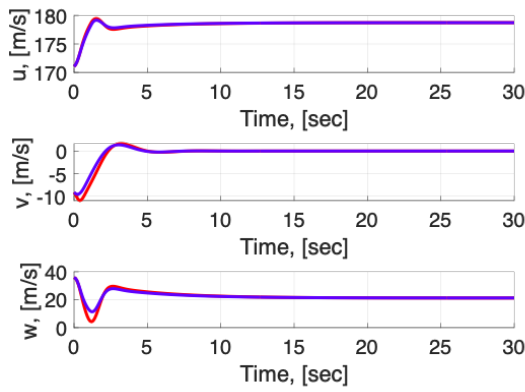
Table 2.1: Simulation parameters of the (C)DDP for F/A-18: optimal regulator

	Parameter	Value
DDP	Sampling time, dt	0.1s
	Time horizon, N	300
	Q	$I_{n \times n}$
	R	$I_{m \times m}$
	S_N	$10Q$
CDDP	Lagrange multiplier, λ_{ini}	1
	Penalty parameter, μ_{ini}	10^{-2}
	Scheduling factor, C_μ	2

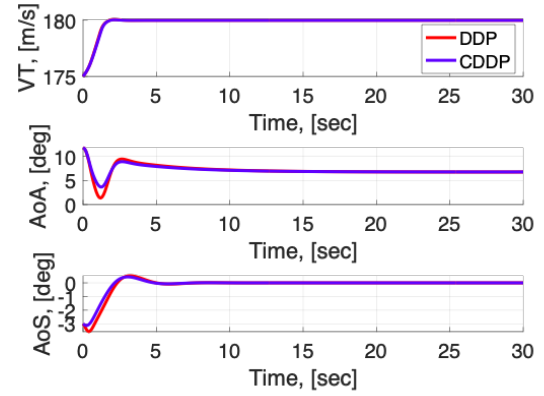
The initial condition deviated from the trim condition and its desired values (trim values) are also listed in the Table 2.2.

Table 2.2: Simulation initial/desired conditions of the F/A-18: optimal regulator

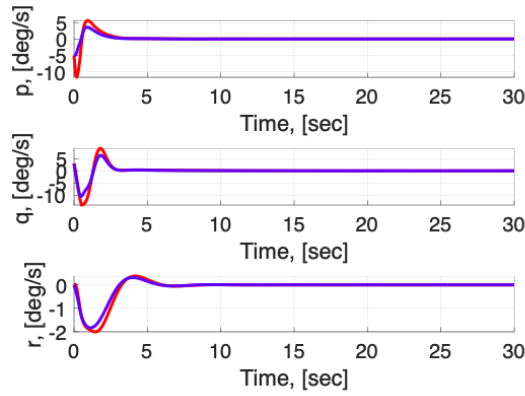
	Parameter	Value
Initial condition	Velocity, V_T	175 [m/s]
	AOA/AOS, $[\alpha, \beta]^T$	$[11.83^\circ, -3^\circ]^T$
	Angular velocity, $[p, q, r]^T$	$[-5, 3, 0]^T$ [deg/s]
	Angular attitude, $[\phi, \theta]^T$	$[-3^\circ, 7.83^\circ]^T$
	Control inputs, $[\delta_{thr}, \delta_{el}, \delta_{er}, \delta_a, \delta_r]^T$	$0_{m \times 1}$
Trim condition	Velocity, V_T	180 [m/s]
	AOA/AOS, $[\alpha, \beta]^T$	$[6.83^\circ, 0^\circ]^T$
	Angular velocity, $[p, q, r]^T$	$[0, 0, 0]^T$ [deg/s]
	Angular attitude, $[\phi, \theta]^T$	$[0^\circ, 6.83^\circ]^T$
	Control inputs, $[\delta_{thr}, \delta_{el}, \delta_{er}, \delta_a, \delta_r]^T$	$[28\%, -0.55^\circ, -0.55^\circ, 0^\circ, 0^\circ]^T$



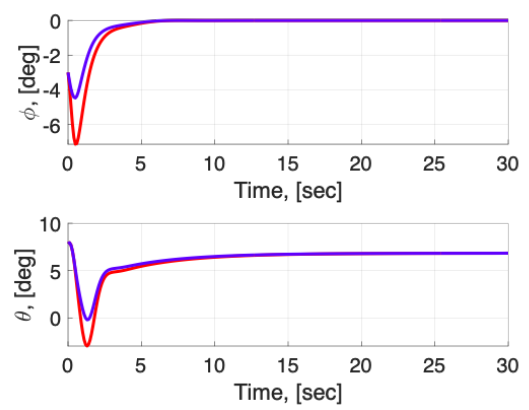
(a) Time history of state: body velocity



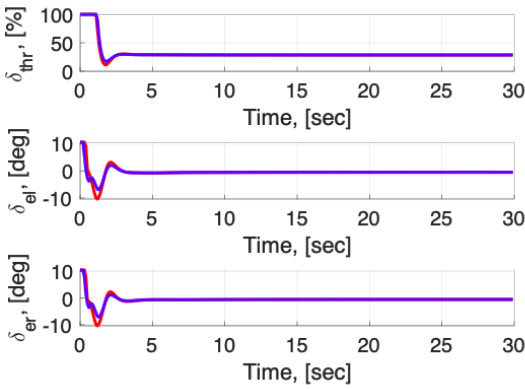
(b) Time history of state: velocity, AoA, and AoS



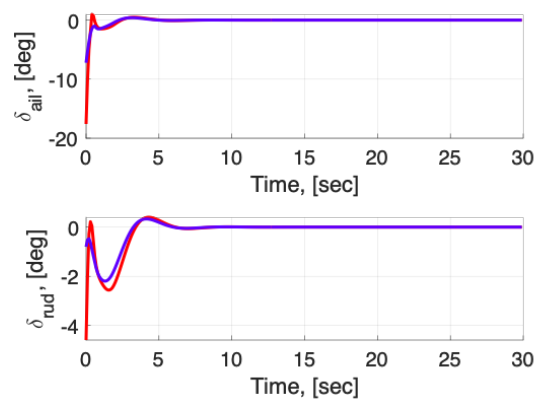
(c) Time history of state: Angular rate



(d) Time history of state: Euler angle



(e) Time history of input: throttle, left/right elevator



(f) Time history of input: aileron, rudder

Figure 2.3: Result of the (C)DDP F/A-18 control problem

The Fig. 2.3 shows the system state and input responses of the optimal regulating problem of the DDP and CDDP. It is clear that the state and control input of the both DDP and CDDP are regulated to the trim value completely within the given time horizon in the

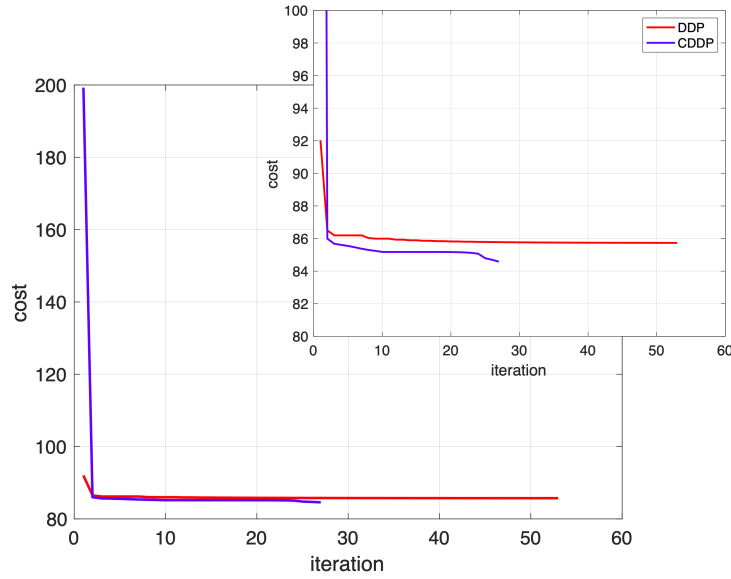


Figure 2.4: Cost comparison between the DDP and CDDP

53 and 27 DDP/CDDP iterations, respectively. Although a naive clamping on the input constraint is utilized in the classic DDP, the state and input trajectories of the DDP can also be stabilized to the desired value because inequality constraint is deactivated during the entire horizon excepted for the initial phase to track the desired value. It can also be found in the Fig. 2.3b, Fig. 2.3c and Fig. 2.3d that the state trajectories of the CDDP have less deviation from the desired value to be stabilized compared to that of the DDP. It is the associated result that the constrained control inputs by physical limit are penalized by Lagrangian term by Eq. (2.23), (2.24) and (2.25), which results in the Fig. 2.4 where the CDDP has more efficient to handle the constraint condition than the classic DDP.

2.4.2 Missile guidance

A three-dimensional air-to-surface impact-angle guidance problem shown in Fig.2.5 is considered by the following equation [11, 12]:

$$\begin{aligned}
 \dot{X}_m &= V_m \cos \gamma_m \cos \psi_m \\
 \dot{Y}_m &= V_m \cos \gamma_m \sin \psi_m \\
 \dot{Z}_m &= -V_m \sin \gamma_m \\
 \dot{V}_m &= \frac{F_T - D}{m} - g \sin \gamma_m \\
 \dot{\gamma}_m &= -\frac{a_z + g \cos \gamma_m}{V_m} \\
 \dot{\psi}_m &= \frac{a_y}{V_m \cos \gamma_m}
 \end{aligned} \tag{2.50}$$

where V_m , γ_m and ψ_m are the velocity, flight path angle and azimuth angle of the kinematic model and $[X_m, Y_m, Z_m]^T$ is the position vector with respect to the inertia frame, respectively.

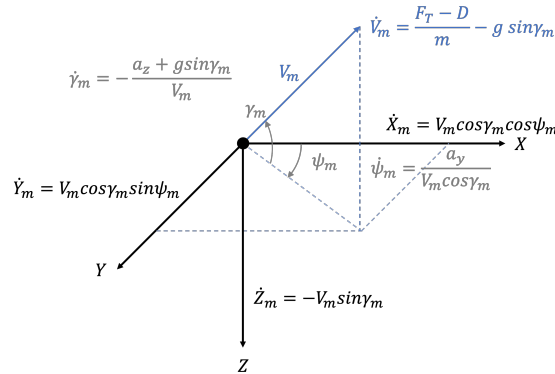


Figure 2.5: Geometry of the kinematics model in the three dimensional plane

The m refers the mass of the vehicle and g is the gravitational acceleration and the aerodynamic drag force D and thrust of which more details such as aerodynamic coefficient, thrust and mass are described in the Appendix A.2. Note that this chapter assumes a time-invariant system with constant parameters given in the Table 2.3 to demonstrate the feasibility of the DDP algorithm on the guidance problem. The missile is designed to intercept a stationary ground target located at the origin point, $O = [0, 0, 0]^T$, of the NED -coordinate

Table 2.3: Simulation initial/desired conditions of the missile

	Parameter	Value
Missile parameter [12]	Mass	135 [kg]
	Drag coefficient, $[C_{D0}, C_{Di}]^T$	$[0.02, 0]^T$
	Thrust, F_T	0 [N]
Initial condition	Position, $[X_m, Y_m, Z_m]^T$	$[5000m, 5000m, -5000m]^T$
	Velocity, V_m	300 [m/s]
	Angle, $[\gamma_m, \psi_m]^T$	$[0^\circ, 100^\circ]^T$
	Control input, $[a_{my}, a_{mz}]^T$	$[0, 0]^T$ [m/s ²]
Target value	Position, $[X_m, Y_m, Z_m]^T$	$[0m, 0m, 0m]^T$
	Velocity, V_m	free
	Angle, $[\gamma_m, \psi_m]^T$	$[-60^\circ, 250^\circ]^T$

system (or XYZ-coordinate system) from the initial position, $pos_m = [X_m, Y_m, Z_m]^T$ with an initial angle γ_m, ψ_m .

Firstly, the state and input vector can be defined by $x = [X_m, Y_m, Z_m, V_m, \gamma_m, \psi_m]^T \in \mathbb{R}^{n \times 1}$ and $u = [a_y, a_z]^T \in \mathbb{R}^{m \times 1}$, respectively. Then, the objective function used in this simulation is formulated as:

$$J = \min_U \frac{1}{2} (x_N - x_d)^T S_N (x_N - x_d) + \frac{1}{2} \sum_{i=1}^{N-1} \{ (x_i - x_d)^T Q (x_i - x_d) + u_i^T R u_i \} \quad (2.51)$$

s.t. $|u_i| \leq u_{max}$

where $U = [u_1, u_2, \dots, u_{N-1}]^T \in \mathbb{R}^{m \times (N-1)}$ is an input trajectory for the given time N sequence, and x_d is a desired point with a velocity and impact angle. The initial and terminal conditions of the vehicle are presented in the Table 2.3 and the weighting matrices for the running cost and terminal cost are also given in the Table 2.4 with hyperparameters of the Augmented Lagrangian method in the CDDP. From the input constraint of the problem in the CDDP, the inequality function can be defined by $g(x_i, u_i) = |u_i| - u_{max}$ where the maximum accelerational command are set by 50 m/s², while the DDP uses Eq.(2.46). The initial input sequence $u_{1:N-1} = [0, 0]^T$ for the initial (C)DDP iteration is then used to generate an uncontrolled trajectory.

The Fig. 2.6 shows the results of the DDP and CDDP algorithms including the flight trajectory (Fig. 2.6a), time history of the state and control input (Fig. 2.6b ~ 2.6d),

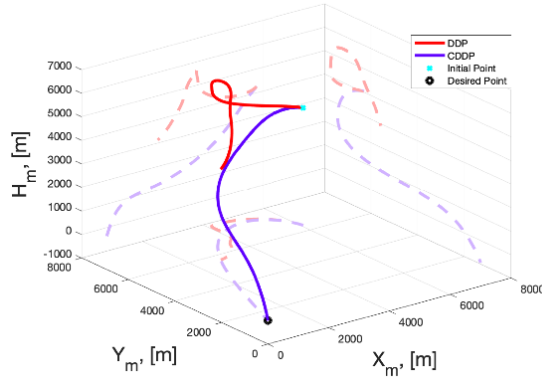
Table 2.4: Simulation parameters of the (C)DDP for the missile

	Parameter	Value
DDP	Sampling time, dt	0.1s
	Time horizon, N	300
	Q	$10^{-4}I_{n \times n}$
	R	$I_{m \times m}$
	S_N	$diag[10^2 I_{3 \times 3}, 10^{-2}, 10^8 I_{2 \times 2}]$
CDDP	Lagrange multiplier, λ	1
	Penalty parameter, μ	0.1
	Scheduling factor, C_μ	2

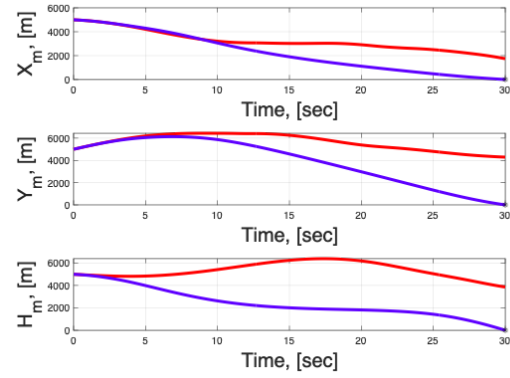
and cost convergence is also demonstrated in the Fig. 2.7 to show the efficiency of the CDDP compared with that of the classic DDP. From the Augmented Lagrangian method, the constraint condition of the control input is formulated in the cost by the penalty and Lagrangian terms that find the solution located on the boundary when the control input exceeds the control limit, while the classic DDP restricts the violated control input naively.

Hence, the CDDP shows the better performance in the resultant trajectory (Fig.2.6), and the cost (Fig.2.7) compared with the DDP due to the characteristic of the method handling the constraint condition when the constraint condition is required to be satisfied.

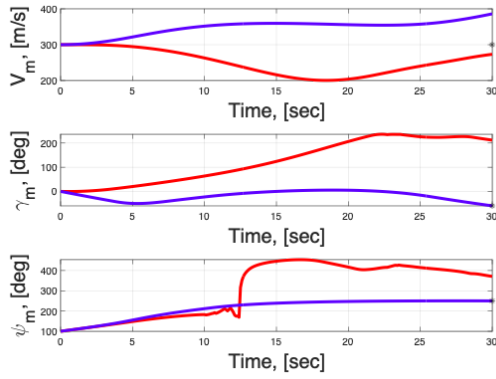
The following simulation verifies the free final time problem by FFT algorithm to evaluate the FFT-(C)DDP algorithm. The weighting matrices of the cost function formulated by Eq.(2.51) and initial condition are also equal to the Table 2.3 and Table 2.4 with an initial final time $t_{Nini} = 25s$. The simulation result given in the Fig. 2.8 presents that the FFT algorithm satisfies the desired condition by solving the trajectory optimization problem, even though the classic DDP fails to intercept the stationary target on the ground in a given fixed-final time depicted in Fig.2.6. That is, the FFT algorithm provides both the DDP and CDDP with the capability to generate the the state and input trajectories by adjusting the final time t_N . When we also compare the cost function by Fig. 2.9a and Table 2.5, the CDDP-based algorithm has less cost and fast convergence rate compared with the classic DDP algorithm. Lastly, the final time t_N is rather extended, however, the FFT algorithm enables to solve the problem with the less cost. The simulation results demonstrate that the DDP algorithm can be applied to find an optimal solution from a low



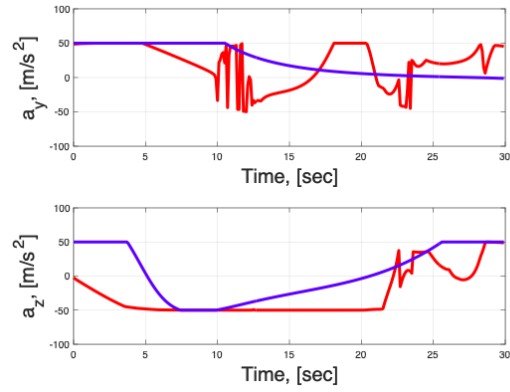
(a) Flight trajectory



(b) Time history of the position



(c) Time history of the state trajectory



(d) Time history of the input trajectory

Figure 2.6: Comparison result of the (C)DDP missile guidance

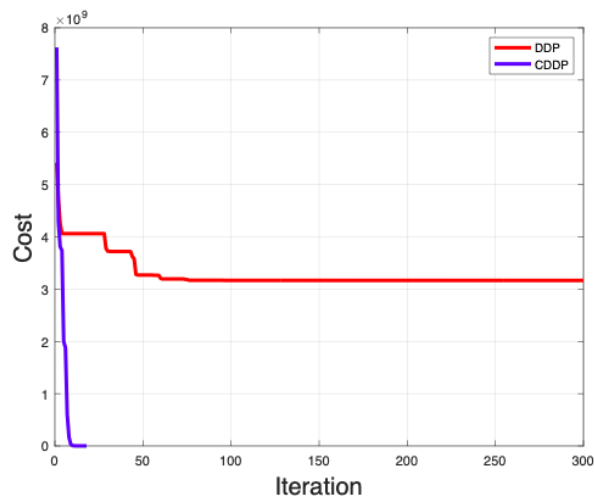


Figure 2.7: Cost comparison result of the (C)DDP missile guidance

level controller to guidance law.

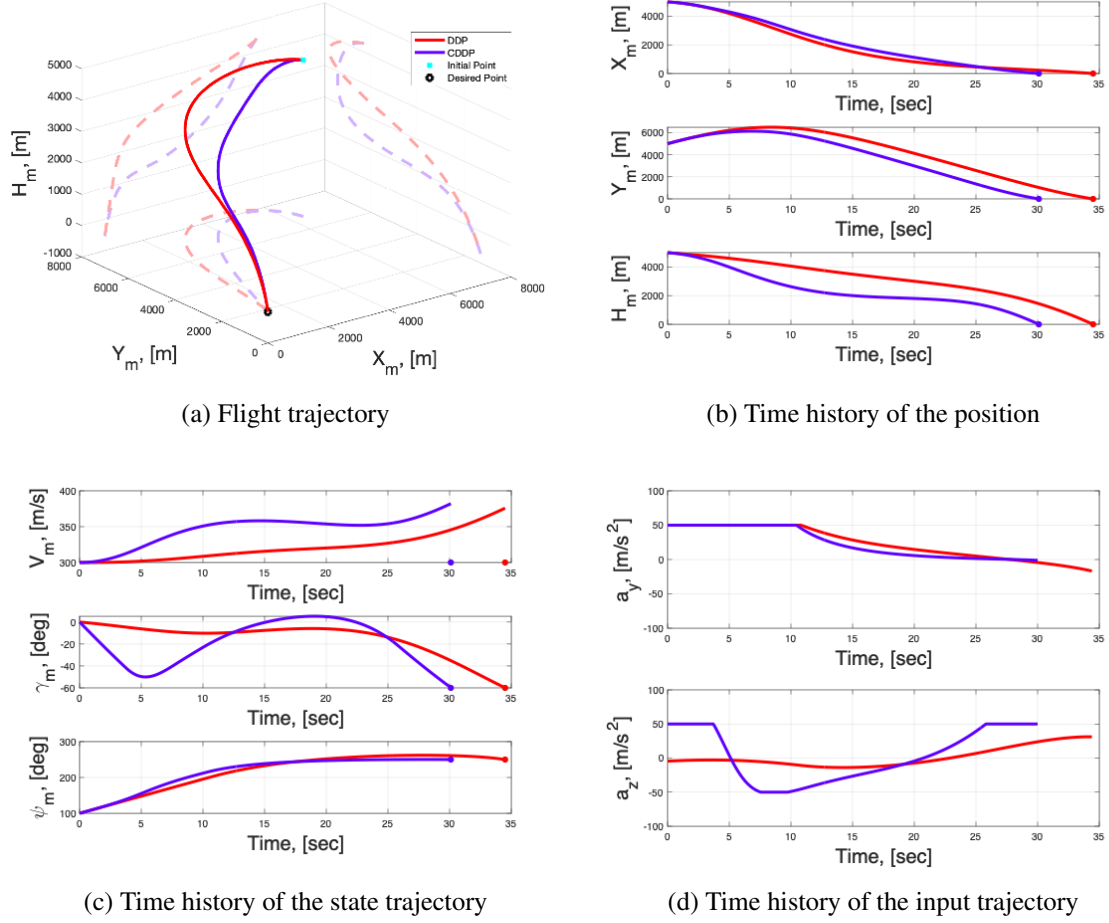


Figure 2.8: Comparison result of the FFT-(C)DDP missile guidance: state and input trajectories

Table 2.5: Comparison result of the both (C)DDP and FFT-(C)DDP

	Initial	Final Time, t_{Nini}	Final Time, t_N	Iteration	Cost
DDP		30s	30s	300 (max)	3.16×10^9
CDDP		30s	30s	50	890470
FFT-DDP		25s	34.5s	156	890713
FFT-CDDP		25s	30.1s	22	870186

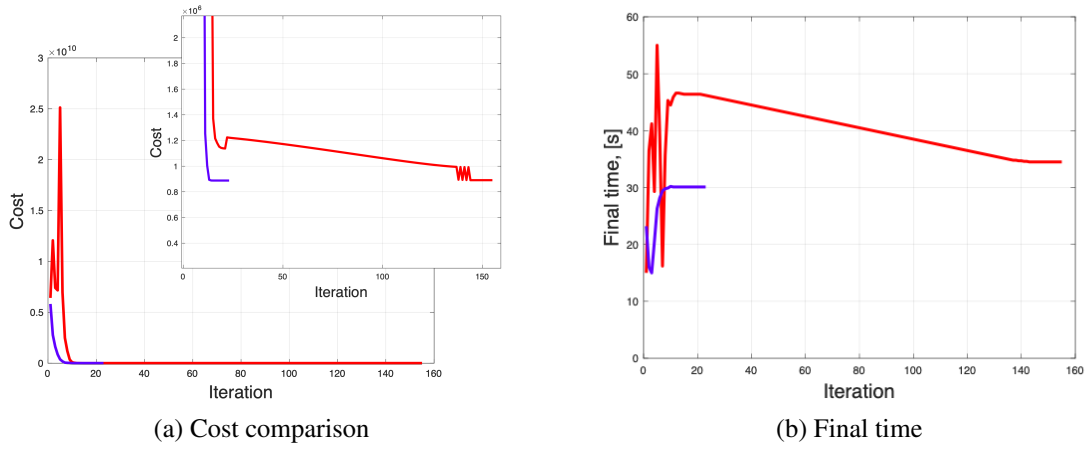


Figure 2.9: Comparison result of the FFT-(C)DDP missile guidance: cost and time

2.5 Conclusion

This chapter revisits the preliminaries of the DDP, and FFT-based DDP algorithms for a final time fixed/free problem, respectively and summarizes the algorithms in the Table 2.6 briefly. A constraint condition is incorporated into the DDP framework by Lagrangian method with a penalty function, which is the CDDP algorithm based on the Augmented Lagrangian method. The FFT algorithm is applied to the DDP framework for the final time free problem adjusting the time horizon by truncating or extending the final time and its trajectories and constraint conditions are also implemented in the FFT algorithm, called FFT-CDDP algorithm. The numerical simulations show the advantages of the variants of the DDP algorithm by incorporating each others to complement its weakness points. The CDDP deals with the constraint condition of the state and input trajectories by penalizing the violated conditions actively and the FFT algorithm gives an adaptive final time that can achieve a proper performance in the given trajectory optimization problem by adjusting time horizon. These variants of the DDP will be utilized to propose a new sampling point-based algorithm in the next chapter.

Table 2.6: Comparison of DDP and its variants

	DDP	CDDP	FFT algorithm
Pros	(Baseline algorithm)	Handling a constraint	Adaptive final time
Cons	Discretization error Fixed final time Handling a constraint	Discretization error Fixed final time	Discretization error Handling a constraint

Chapter 3

Sparse Gauss-Hermite Quadrature

CDDP

3.1 Unscented Dynamic Programming

The Differential Dynamic Programming (DDP) is the second order numerical method to find an optimal control correction by approximating the cost function and dynamics around the nominal trajectory with quadratic convergence properties using the Taylor series expansion. A mathematical computation of the cost and dynamics is unavoidable to obtain those derivatives in the backward pass. Furthermore, as a complexity of the system is increased, the accuracy of the dynamics and value function derivative brings a computational burden. To avoid the numerical differentiation, the Unscented Dynamic Programming (UDP) has been proposed which is a sampling-based DDP method utilizing the concept of the Unscented Transform (UT) [50] from the Unscented Kalman Filter (UKF) [51, 14, 54]. The UT propagates a sampled point called *sigma points* through a nonlinear dynamic (or called a nonlinear coordinate transform) to compute a sampled mean and covariance of a probability distribution that replaces the gradient and Hessian of the action-value function in Eq.(2.9). The transformed points are then used to approximate a new cost-to-go function at the earlier time step during the 'backward pass' in DDP.

Firstly, to compute the Hessian of the Q-function, a set of $(n + m)$ *sigma points* is generated using the UT relation [50].

$$\begin{aligned} \begin{bmatrix} \tilde{x}_{i+1}^j \\ \tilde{u}_i^j \end{bmatrix} &= \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} + \beta \Gamma(:, j), \quad j = 1, 2, \dots, n + m \\ \begin{bmatrix} \tilde{x}_{i+1}^j \\ \tilde{u}_i^j \end{bmatrix} &= \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} - \beta \Gamma(:, j), \quad j = n + m + 1, n + m + 2, \dots, 2(n + m) \end{aligned} \quad (3.1)$$

where $\Gamma(:, j) \in R^{(n+m) \times 1}$ is the j^{th} column vector of the Cholesky decomposition of the following equation with a scaling factor β :

$$\Gamma = chol \left(\begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & l_{uu} \end{bmatrix}_i^{-1} \right) \quad (3.2)$$

The *sigma points* $[\tilde{x}_{i+1}^T \ \tilde{u}_i^T]^T$ are then propagated backward through the dynamics $\tilde{x}_i = f^-(\tilde{x}_{i+1}, \tilde{u}_i)$. Using the propagated sampled points, the Hessian term in Eq.(2.9) can be computed:

$$\begin{aligned} \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix}_i &= \begin{bmatrix} f_x^T V'_{xx} f_x & f_x^T V'_{xx} f_u \\ f_u^T V'_{xx} f_x & f_u^T V'_{xx} f_u + l_{uu} \end{bmatrix}_i + \begin{bmatrix} l_{xx} & l_{xu} \\ l_{ux} & 0_{m \times m} \end{bmatrix}_i \\ &= \begin{bmatrix} f_x & f_u \end{bmatrix}_i^T \begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & l_{uu} \end{bmatrix}_i \begin{bmatrix} f_x & f_u \end{bmatrix}_i + \begin{bmatrix} l_{xx} & l_{xu} \\ l_{ux} & 0_{m \times m} \end{bmatrix}_i \\ &= G_i + \begin{bmatrix} l_{xx} & l_{xu} \\ l_{ux} & 0_{m \times m} \end{bmatrix}_i \end{aligned} \quad (3.3)$$

where the second derivative term, G_i , can be decomposed by following equation:

$$G_i = \begin{bmatrix} f_x^T V'_{xx} f_x & f_x^T V'_{xx} f_u \\ f_u^T V'_{xx} f_x & f_u^T V'_{xx} f_u + l_{uu} \end{bmatrix}_i \quad (3.4)$$

$$= \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^T \begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & l_{uu} \end{bmatrix}_i \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i$$

$$\therefore G_i^{-1} = \left(\begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^{-1} \right) \begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & l_{uu} \end{bmatrix}_i^{-1} \left(\begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^{-1} \right)^T \quad (3.5)$$

$$= \bar{F}_i^{-1} \bar{P}^{-1} \bar{F}_i^{-T}$$

If the Cholesky decomposition $G_i^{-1} = \Lambda \Lambda^T$ and $\bar{P} = \Gamma \Gamma^T$ can be applied, where Λ and Γ are the unique lower triangular matrices, then, the following equation can also be satisfied as:

$$\Lambda \Lambda^T = \left(\bar{F}^{-1} \right) \Gamma \Gamma^T \left(\bar{F}^{-1} \right)^T \quad (3.6)$$

$$\therefore \Lambda = \bar{F}^{-1} \Gamma$$

This means that the column vector of Γ can be propagated backward through the dynamics $\bar{F}^{-1}$, which is equivalent to the propagation of the covariance, $P' = \left(\frac{\partial f}{\partial x} \right) P \left(\frac{\partial f}{\partial x} \right)^T$, and the nonlinear transformation $f(x_i, u_i)$ as well. Thus, G_i^{-1} can be obtained by computing the covariance matrix using the backward propagated sample point, $[\tilde{x}_i^T \ \tilde{u}_i^T]^T$.

$$G_i^{-1} = \frac{1}{2\beta^2} \sum_{j=1}^{2(n+m)} \left(\begin{bmatrix} \tilde{x}_i^j \\ \tilde{u}_i^j \end{bmatrix} - \begin{bmatrix} \mu_i^x \\ \mu_i^u \end{bmatrix} \right) \left(\begin{bmatrix} \tilde{x}_i^j \\ \tilde{u}_i^j \end{bmatrix} - \begin{bmatrix} \mu_i^x \\ \mu_i^u \end{bmatrix} \right)^T \quad (3.7)$$

where $\begin{bmatrix} \mu_i^{xT} & \mu_i^{uT} \end{bmatrix}^T$ are the sampled mean value of the propagated variables, $\begin{bmatrix} \tilde{x}_i^T & \tilde{u}_i^T \end{bmatrix}^T$

respectively:

$$\begin{bmatrix} \mu_i^x \\ \mu_i^u \end{bmatrix} = \frac{1}{2(n+m)} \sum_{j=1}^{2(n+m)} \begin{bmatrix} \tilde{x}_i^j \\ \tilde{u}_i^j \end{bmatrix} \quad (3.8)$$

To find a gradient of the Q-function Eq.(2.9), the numerical differentiation method is used.

Define the propagated points, ζ , by projecting V'_x onto the *sigma points*, $\tilde{x}_{i+1}$, as follows:

$$\begin{aligned} \zeta_i &= (V'_x)^T \tilde{x}_{i+1} \\ &\approx (V'_x)^T \begin{bmatrix} f_x & f_u \end{bmatrix} \begin{bmatrix} \tilde{x}_i \\ \tilde{u}_i \end{bmatrix} \\ \therefore \zeta_i^T &= \begin{bmatrix} \tilde{x}_i^T & \tilde{u}_i^T \end{bmatrix} \begin{bmatrix} f_x^T \\ f_u^T \end{bmatrix} V'_x \end{aligned} \quad (3.9)$$

Then, the principle of the finite difference between the *sigma points* is applied as:

$$d = D \begin{bmatrix} f_x^T V'_x \\ f_u^T V'_x \end{bmatrix} \quad (3.10)$$

where D and d are the differences of the sample points and the defined points g :

$$d = \begin{bmatrix} (\zeta_i^1 - \zeta_i^{n+m+1})^T \\ \vdots \\ (\zeta_i^{n+m} - \zeta_i^{2(n+m)})^T \end{bmatrix} \quad (3.11)$$

$$D = \begin{bmatrix} (\tilde{x}_i^1 - \tilde{x}_i^{n+m+1})^T & (\tilde{u}_i^1 - \tilde{u}_i^{n+m+1})^T \\ \vdots & \vdots \\ (\tilde{x}_i^{n+m} - \tilde{x}_i^{2(n+m)})^T & (\tilde{u}_i^{n+m} - \tilde{u}_i^{2(n+m)})^T \end{bmatrix} \quad (3.12)$$

Thus, the gradient terms in Q-function Eq.(2.9) can be computed by solving a linear

system Eq.(3.10):

$$\begin{aligned} \begin{bmatrix} Q_x \\ Q_u \end{bmatrix}_i &= \begin{bmatrix} l_x \\ l_u \end{bmatrix}_i + \begin{bmatrix} f_x^T V'_x \\ f_u^T V'_x \end{bmatrix} \\ &= \begin{bmatrix} l_x \\ l_u \end{bmatrix}_i + D_i^{-1} d_i \end{aligned} \quad (3.13)$$

Thus, the approximation of the Q-function, Eq.(2.13), can be obtained by substituting the gradient and Hessian with the numerical computation, Eq.(3.13) and Eq.(3.7) by Algorithm 9 in the backward pass.

Algorithm 9 UDP: Backward Pass

- 1: Initialize $V_{xN} = l_{xN}$, $V_{xxN} = l_{xxN}$
 - 2: **for** $i=N-1:-1:1$ **do**
 - 3: $\Gamma_i = chol \left(\begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & l_{uu} \end{bmatrix}_i \right)$
 - 4: Unscented Transform by Eq.(3.1)
 - 5: Backward propagation of the *sigma points*: $\tilde{x}_i = f^-(\tilde{x}_{i+1}, \tilde{u}_i)$
 - 6: Update Q-function Eq.(2.13) and Eq.(2.14) using the gradient and Hessian: Eq.(3.13), Eq.(3.7)
 - 7: **if** Q_{uu} is invertible **then**
 - 8: Compute the control policy: Eq.(2.11)
 - 9: Update the value function: Eq.(2.15)
 - 10: **else**
 - 11: Increase ρ by Eq.(2.20) go to line 2
 - 12: **end if**
 - 13: **end for**
-

3.2 Sparse Gauss-Hermite Quadrature

3.2.1 Numerical Integration

Numerical integration, or so-called '*quadrature*', is a method to calculate the approximates of an integral of a function $f(x)$ by using numerical techniques when the integrand $f(x)$ is only known at the certain points or finding an integral of the $f(x)$ is even complicated via symbolic methods analytically [60]. Sometimes, numerical integration is easier

to compute the approximates of the integrand that is too complicated to evaluate mathematically, therefore the numerical integration methods are useful and essential in many practical situations. There are two typical methods: Newton-Cotes method and Gauss method. The first approach is the most common numerical integration method replacing a integrand $f(x)$ with a polynomial:

$$I = \int_a^b f(x)dx \simeq \int_a^b f_{n_p}(x)dx \quad (3.14)$$

where $f_{n_p}(x) = a_0 + a_1x + \dots + a_{n_p-1}x^{n_p-1} + a_{n_p}x^{n_p}$ is a polynomial and n_p is the order of the polynomial. The integral can be approximated using a series of polynomial or data segment with the boundary points of the interval. The first-order method of polynomial (*trapezoidal rule*) is geometrically equivalent to the area of the trapezoid under the straight line generated by the boundary points, The higher-order polynomials (*Simpson's rule*) makes use of extra points midway between the boundary points by dividing the more segment of the area to enhance the accuracy.

The second method is *Gauss Quadrature* (GQ) employing the quadrature points, x determined from the integration interval that makes a more accurate integral estimate results. It yields exact values of integrations for polynomials of degree up to $2n_I - 1$, with n_I quadrature points. It has better efficiency and simplicity and produces more exact results because the base condition of the fixed point (or boundary point) is removed while location of the boundary point used in the polynomial-based method is predetermined (hence it is an open quadrature rule). The basic estimate of the Gauss quadrature over a finite interval $[a \ b]$ can be represented as a weighted sum of the function with a finite number of sample points:

$$\begin{aligned} I &= \int_a^b f(t)dt \\ &\Rightarrow \int_{-1}^1 f(x)dx \simeq \sum_{k=1}^{n_I} w_k f(x_k) \end{aligned} \quad (3.15)$$

where the interval of the integration is converted into $[-1 \ 1]$, x_k and w_k are integration

points (or quadrature points) and its corresponding weights, respectively, which are determined according to the complexity of the function, $f(x)$ to be integrated.

3.2.2 Gauss-Hermite Quadrature

Gauss-Hermite Quadrature (GHQ) is a GQ over the interval $[-\infty \infty]$. An integration of an arbitrary function $f(x)$ over an exponential weighting function is evaluated as a weighted sum over a set of sample points known as GHQ points:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \simeq \sum_{k=1}^{n_I} w_k f(\gamma_k) \quad (3.16)$$

where γ_i is the roots of the Hermite polynomial, $H_{n_I}(x)$, and its corresponding weights, w_i are given by:

$$w_k = \frac{2^{n_I-1} n_I! \sqrt{\pi}}{n_I^2 [H_{n_I-1}(x_i)]^2}, \quad \sum_{k=1}^{n_I} w_k = 1 \quad (3.17)$$

The Hermite polynomial function, $H_{n_I}(x)$, is given by Rodrigues's Formula:

$$H_{n_I}(x) = (-1)^{n_I} e^{x^2} \frac{d^{n_I}}{dx^{n_I}} (e^{-x^2}) \quad (3.18)$$

Instead of finding quadrature point from the above root-finding method, the tridiagonal method is presented, which is stable and mathematically/computationally better approach [61, 62]. For the univariate GHQ rule with respect to the univariate standard Gaussian distribution, $N(x; 0, 1)$, with n_I quadrature points, the expected value of the function $f(x)$ can be approximated as:

$$\begin{aligned} I(f) &= \int_{-\infty}^{\infty} f(x) N(x; 0, 1) dx \\ &\simeq \sum_{k=1}^{n_I} w_k f(\gamma_k) \end{aligned} \quad (3.19)$$

The quadrature point γ_k and the weights w_k ($1 \leq k \leq n_I - 1$) can be calculated from the tridiagonal matrix $\mathbf{L}$: $\mathbf{L}_{k,k+1} = \mathbf{L}_{k+1,k} = \sqrt{k/2}$:

$$\mathbf{L} = \begin{bmatrix} 0 & \sqrt{1/2} & 0 & 0 & \\ \sqrt{1/2} & 0 & \sqrt{2/2} & 0 & \dots \\ 0 & \sqrt{2/2} & 0 & \sqrt{3/2} & \\ 0 & 0 & \sqrt{3/2} & 0 & \\ & \vdots & & & \ddots \end{bmatrix} \quad (3.20)$$

The corresponding weight w_k is calculated by v_{1k} that is the first element of the k^{th} normalized eigenvector of $\mathbf{L}$. i.e.

$$\begin{cases} \gamma_1 = 0, & w_1 = 1 & i = 1 \\ \gamma_k = \sqrt{2}\varepsilon_k, & w_k = (v_k)^2 & 1 < k \leq n_I - 1 \end{cases} \quad (3.21)$$

where the univariate GHQ rule with n_I points is exact up to the $(2n_I - 1)$ th or less order of polynomials of f [61].

To represent the multivariate GHQ problem, $N(x; 0, I_n)$, the univariate GHQ rule can be extended by the tensor product rule, since each quadrature points are uncorrelated [62].

$$\begin{aligned} I(f) &= \int_{-\infty}^{\infty} f(x) N(x; 0, I_n) dx \\ &\approx \sum_{l_j=1}^{n_I} w_{l_j} \cdots \sum_{l_n=1}^{n_I} w_{l_n} f(\gamma_{l_1}, \dots, \gamma_{l_n}) \\ &= \sum_{l=1}^{(n_I)^n} w_l f(\gamma_l) \end{aligned} \quad (3.22)$$

where

$$\begin{aligned} \gamma_l &= [\gamma_{l_1}, \dots, \gamma_{l_n}]^T \\ w_l &= \prod_{j=1}^n w_{l_j} \end{aligned} \quad (3.23)$$

The multivariate GHQ rule is also exact up to the $(2n_I - 1)$ th order of all polynomials [61], For example, consider the two dimensional, $n = 2$, Gauss-Hermite Quadrature (GHQ) rule with 3 points, $n_I = 3$. The symmetric tridiagonal matrix $\mathbf{L}$, $\mathbf{L}_{k,k+1} = \mathbf{L}_{k+1,k} = \sqrt{k/2}$ is given as:

$$\mathbf{L} = \begin{bmatrix} 0 & \sqrt{1/2} & 0 \\ \sqrt{1/2} & 0 & \sqrt{2/2} \\ 0 & \sqrt{2/2} & 0 \end{bmatrix} \quad (3.24)$$

The eigen-value (ε) and eigen-vector (v) of the $\mathbf{L}$ can be computed by:

$$\begin{aligned} \varepsilon &= -\frac{\sqrt{3}}{\sqrt{2}}, 0, \frac{\sqrt{3}}{\sqrt{2}} \\ v &= \begin{bmatrix} -\frac{1}{\sqrt{6}} & \frac{\sqrt{2}}{\sqrt{3}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \end{aligned} \quad (3.25)$$

The quadrature point, γ , and the corresponding weight, w , can then be obtained by Eq.(3.21):

$$\begin{aligned} \gamma &= \sqrt{2}\varepsilon \\ &= -\sqrt{3}, 0, \sqrt{3} \\ w &= (v_{1,1:3})^2 \\ &= \left(-\frac{1}{\sqrt{6}}\right)^2, \left(\frac{\sqrt{2}}{\sqrt{3}}\right)^2, \left(\frac{1}{\sqrt{6}}\right)^2 \\ &= \frac{1}{6}, \frac{4}{6}, \frac{1}{6} \end{aligned} \quad (3.26)$$

Hence, for a $n = 2$ system with a $n_I = 3$ GH quadrature points, the set of the quadrature points will be $(n_I)^n = 3^2 = 9$ number of points and the associated weights, which are

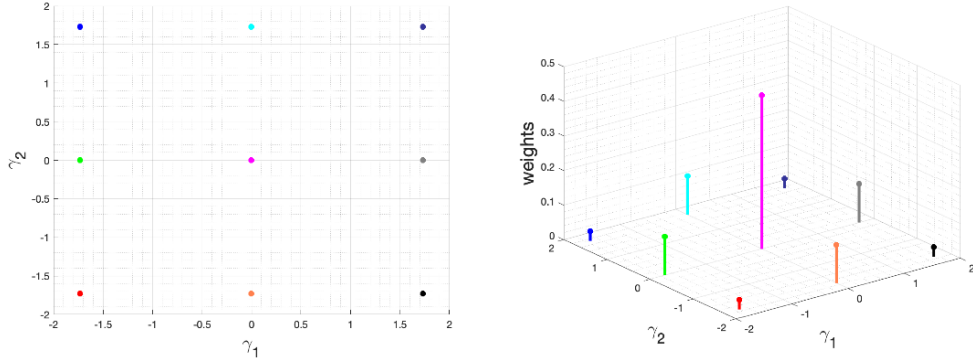


Figure 3.1: The example of the Gauss-Hermite Quadrature rule, $n = 2$, $n_I = 3$.

combinations of an one-dimensional GHQ point and their weights:

$$\begin{aligned}
 \begin{pmatrix} (\gamma_1, \gamma_1) & (\gamma_1, \gamma_2) & (\gamma_1, \gamma_3) \\ (\gamma_2, \gamma_1) & (\gamma_2, \gamma_2) & (\gamma_2, \gamma_3) \\ (\gamma_3, \gamma_1) & (\gamma_3, \gamma_2) & (\gamma_3, \gamma_3) \end{pmatrix} &= \begin{pmatrix} (-\sqrt{3}, -\sqrt{3}) & (-\sqrt{3}, 0) & (-\sqrt{3}, \sqrt{3}) \\ (0, -\sqrt{3}) & (0, 0) & (0, \sqrt{3}) \\ (\sqrt{3}, -\sqrt{3}) & (\sqrt{3}, 0) & (\sqrt{3}, \sqrt{3}) \end{pmatrix} \\
 \begin{pmatrix} w_1 w_1 & w_1 w_2 & w_1 w_3 \\ w_2 w_1 & w_2 w_2 & w_2 w_3 \\ w_3 w_1 & w_3 w_2 & w_3 w_3 \end{pmatrix} &= \begin{pmatrix} (\frac{1}{6})(\frac{1}{6}) & (\frac{1}{6})(\frac{4}{6}) & (\frac{1}{6})(\frac{1}{6}) \\ (\frac{4}{6})(\frac{1}{6}) & (\frac{4}{6})(\frac{4}{6}) & (\frac{4}{6})(\frac{1}{6}) \\ (\frac{1}{6})(\frac{1}{6}) & (\frac{1}{6})(\frac{4}{6}) & (\frac{1}{6})(\frac{1}{6}) \end{pmatrix}
 \end{aligned} \tag{3.27}$$

However, the number of the quadrature point (γ_i) is increased exponentially $(n_I)^n$ with a system dimension n , i.e. 'curse of dimensionality issue'. To handle this issue, the Sparse-grid method is utilized.

3.2.3 Sparse Grid

From the previous subsection 3.2.1 and 3.2.2, the GHQ rule is efficient for the univariate integration, however, the multivariate GHQ suffers from the curse of the dimensionality as the number of the points is increased exponentially by $(n_I)^n$. Hence, most of the GHQ rule is implemented in the low dimensional cases. To cope with the curse of dimensionality of the GHQ product rule, the sparse grid quadrature rule is derived based

on the Smolyak's rule that utilizes a linear combination of lower level tensor products of univariate quadrature rules to approximate multivariate quadrature [63, 64].

$$\begin{aligned}
 I(f) &= \int_{-\infty}^{\infty} f(x) N(x; 0, I_n) dx \\
 &\approx I_{n,L}(f) \\
 &= \sum_{q=L-n}^{L-1} (-1)^{L-1-q} (\mathbf{C}_{L-1-q}^{n-1}) \sum_{\Omega \in \mathbb{N}_q^n} (I_{l_1} \otimes I_{l_2} \otimes \cdots \otimes I_{l_n})(f)
 \end{aligned} \tag{3.28}$$

where the $I_{n,L}(f)$ is an approximate of the numerical integration of an n dimensional system, f with the accuracy level L . $\mathbf{C}$ denotes the binomial coefficient, i.e. $\mathbf{C}_k^n = \frac{n!}{(n-k)!k!}$ of which $!$ means the factorial operator, $n! = n \times (n-1) \times \dots \times 2 \times 1$, and $\otimes$ denotes a tensor product. $\Omega \triangleq [i_1, i_2, \dots, i_n]$ is an accuracy level sequence and the q is the auxiliary parameter for the mathematical notation. An available set of i_p is given by the set of accuracy level sequences, $\mathbb{N}_q^n$, indicating the possible accuracy level i_p for each univariate quadrature:

$$\mathbb{N}_q^n = \begin{cases} \Omega : \sum_{j=1}^n l_j = n + q, & \text{for } q \geq 0 \\ \emptyset & \text{for } q < 0 \end{cases} \tag{3.29}$$

where the parameter q is given as $L - n \leq q \leq L - 1$ and $\emptyset$ denotes an empty set. The Eq.(3.28) can then be written in an explicit form:

$$I_{n,L}(f) = \sum_{q=L-n}^{L-1} (-1)^{L-1-q} (\mathbf{C}_{L-1-q}^{n-1}) \sum_{\Omega \in \mathbb{N}_q^n} \sum_{q_{s_1} \in X_{l_1}} \sum_{q_{s_2} \in X_{l_2}} \cdots \sum_{q_{s_n} \in X_{l_n}} f(q_{s_1}, q_{s_2}, \dots, q_{s_n}) w_{s_1} w_{s_2} \cdots w_{s_n} \tag{3.30}$$

where X_{l_j} is the univariate quadrature point with the accuracy level l_j . $q_{s_j} = [q_{s_1}, q_{s_2}, \dots, q_{s_n}] \in X_{l_j}$ and w_j , ($1 \leq j \leq n$), are a Sparse Gauss-Hermite quadrature (SGHQ) point and its weights associated with q_{s_j} , respectively. The final set of the SGHQ points is given as:

$$X_{n,L} = \bigcup_{q=L-n}^{L-1} \bigcup_{\Omega \in \mathbb{N}_q^n} (X_{l_1} \otimes X_{l_2} \otimes \cdots \otimes X_{l_n}) \tag{3.31}$$

where $\cup$ denotes the union operator of the individual SGHQ points. For the better understanding of the details in the SGHQ rule, recall the example of the $n = 2, L = 3$, from the previous subsection 3.2.2. The possible accuracy level $\mathbb{N}_q^n$ can be calculated by Eq.(3.29) with the available auxiliary parameter, q that would be $q = 1, 2$ from $L - n \leq q \leq L - 1$. For $q = 1$:

$$\begin{aligned}\mathbb{N}_q^n &= \mathbb{N}_1^2 = \sum_{p=1}^n l_p = n + q \\ &\Rightarrow \sum_{p=1}^2 l_p = l_1 + l_2 = 3 \\ &\therefore (l_1, l_2) = \{(1, 2), (2, 1)\}\end{aligned}\tag{3.32}$$

and for $q = 2$:

$$\begin{aligned}\mathbb{N}_q^n &= \mathbb{N}_2^2 = \sum_{p=1}^n l_p = n + q \\ &\Rightarrow \sum_{p=1}^2 l_p = l_1 + l_2 = 4 \\ &\therefore (l_1, l_2) = \{(1, 3), (2, 2), (3, 1)\}\end{aligned}\tag{3.33}$$

From the Eq.(3.31), the final set of the point can be computed by tensor product, $\otimes$:

$$\begin{aligned}X_{2,3} &= \bigcup_{q=L-n}^{L-1} \bigcup_{\Omega \in \mathbb{N}_q^n} (X_{l_1} \otimes X_{l_2} \otimes \dots \otimes X_{l_n}) \\ &= \underline{(X_1 \otimes X_2) \cup (X_2 \otimes X_1)}_{q=1} \bigcup \underline{(X_1 \otimes X_3) \cup (X_2 \otimes X_2) \cup (X_3 \otimes X_1)}_{q=2}\end{aligned}\tag{3.34}$$

where the number of the univariate point set is chosen by $2^L - 1$. i.e. for l_1 :

$$X_1 = 0, w_1 = 1\tag{3.35}$$

for l_2 : $2^2 - 1 = 3$ obtained from the Eq.(3.26):

$$\begin{aligned} X_2 &= -\sqrt{3}, 0, \sqrt{3} \\ w_2 &= \frac{1}{6}, \frac{4}{6}, \frac{1}{6} \end{aligned} \quad (3.36)$$

for l_3 : $2^3 - 1 = 7$. The symmetric tridiagonal matrix $\mathbf{L}$ is given by $\mathbf{L}_{i,i+1} = \mathbf{L}_{i+1,i} = \sqrt{i/2}$:

$$\mathbf{L} = \begin{bmatrix} 0 & \sqrt{\frac{1}{2}} & 0 & & 0 & 0 \\ \sqrt{\frac{1}{2}} & 0 & \sqrt{\frac{2}{2}} & \cdots & 0 & 0 \\ 0 & \sqrt{\frac{2}{2}} & 0 & & \vdots & \\ & \vdots & & \ddots & & \\ 0 & 0 & 0 & \cdots & 0 & \sqrt{\frac{6}{2}} \\ 0 & 0 & 0 & \cdots & \sqrt{\frac{6}{2}} & 0 \end{bmatrix} \quad (3.37)$$

The eigen-value (ε) and eigen-vector (v) of the above $\mathbf{L}$ can be computed using the 'eig' built-in function of the MATLAB:

$$\begin{aligned} \varepsilon &= \pm 2.6520, \pm 1.6736, \pm 0.8163, 0 \\ v &= \begin{bmatrix} -0.0234 & 0.1754 & 0.4900 & 0.6761 & 0.4900 & 0.1754 & 0.0234 \\ 0.0878 & -0.4151 & -0.5657 & 0 & 0.5657 & 0.4151 & 0.0878 \\ -0.2163 & 0.5706 & 0.1153 & -0.4781 & 0.1153 & 0.5706 & 0.2163 \\ 0.3967 & -0.4408 & 0.3851 & 0 & -0.3851 & 0.4408 & 0.3967 \\ -0.5566 & 0.0275 & -0.3221 & 0.4140 & -0.3221 & 0.0275 & 0.5566 \\ 0.5787 & 0.3652 & -0.1781 & 0 & 0.1781 & -0.3652 & 0.5787 \\ -0.3780 & -0.3780 & 0.3780 & -0.3780 & 0.3780 & -0.3780 & 0.3780 \end{bmatrix} \end{aligned} \quad (3.38)$$

and the univariate quadrature point and the corresponding weight can then be obtained by Eq.(3.21):

$$\begin{aligned}
 X_3 &= \sqrt{2}\varepsilon \\
 &= \pm 3.7504, \pm 2.3668, \pm 1.1544, 0 \\
 v &= (w_{1,1:7})^2 \\
 &= (-0.0234)^2, (0.1754)^2, (0.4900)^2, (0.6761)^2, (0.4900)^2, (0.1754)^2, (0.0234)^2
 \end{aligned} \tag{3.39}$$

Thus, the final set of the quadrature points, $X_{n=2, L_v=3}$, can be obtained by Eq.(3.34) including all the combination of the $\mathbb{N}_1^2$ and $\mathbb{N}_2^2$.

For $q = 1$:

$$\begin{aligned}
 X_1 \otimes X_2 &= (0, -\sqrt{3}), (0, 0), (0, \sqrt{3}) \\
 X_2 \otimes X_1 &= (-\sqrt{3}, 0), (0, 0), (\sqrt{3}, 0)
 \end{aligned} \tag{3.40}$$

For $q = 2$:

$$\begin{aligned}
 X_1 \otimes X_3 &= (0, -3.7504), (0, -2.3668), (0, -1.1544), (0, 0), \\
 &\quad (0, 1.1544), (0, 2.3668), (0, 3.7504) \\
 X_2 \otimes X_2 &= (-\sqrt{3}, -\sqrt{3}), (-\sqrt{3}, 0), (-\sqrt{3}, \sqrt{3}), \\
 &\quad (0, -\sqrt{3}), (0, 0), (0, \sqrt{3}), \\
 &\quad (\sqrt{3}, -\sqrt{3}), (\sqrt{3}, 0), (\sqrt{3}, \sqrt{3}) \\
 X_3 \otimes X_1 &= (-3.7504, 0), (-2.3668, 0), (-1.1544, 0), (0, 0), \\
 &\quad (0, 1.1544), (0, 2.3668), (0, 3.7504)
 \end{aligned} \tag{3.41}$$

Thus, the resultant set of the SGHQ points is:

$$\begin{aligned} \therefore X_{2,3} = \{ & (0,0), (0,-\sqrt{3}), (0,\sqrt{3}), (-\sqrt{3},0), (\sqrt{3},0), \\ & (0,-3.7504), (0,-2.3668), (0,-1.1544), (0,1.1544), (0,2.3668), (0,3.7504) \\ & (-\sqrt{3},-\sqrt{3}), (-\sqrt{3},\sqrt{3}), (\sqrt{3},-\sqrt{3}), (\sqrt{3},\sqrt{3}) \\ & (-3.7504,0), (-2.3668,0), (-1.1544,0), (1.1544,0), (2.3668,0), (3.7504,0) \} \end{aligned} \quad (3.42)$$

The corresponding weights are computed by following relation from Eq. (3.28):

$$w_s = (-1)^{L-1-q} \mathbf{C}_{L-1-q}^{n-1} \prod_{j=1}^n w_{l_j} \quad (3.43)$$

For $q = 1$:

$$\begin{aligned} w &= (-1)^{3-1-1} \mathbf{C}_{3-1-1}^{2-1} \prod_{j=1}^2 w_{l_j} \\ X_1 \otimes X_2 : & (-1) \left[1 \left(\frac{1}{6} \right), 1 \left(\frac{4}{6} \right), 1 \left(\frac{1}{6} \right) \right] \\ X_2 \otimes X_1 : & (-1) \left[\left(\frac{1}{6} \right) 1, \left(\frac{4}{6} \right) 1, \left(\frac{1}{6} \right) 1 \right] \end{aligned} \quad (3.44)$$

For $q = 2$:

$$\begin{aligned} X_1 \otimes X_3 : & [1(0.0005), 1(0.0308), 1(0.2401) \\ & 1(0.4571), 1(0.2401), 1(0.0308), 1(0.0005)] \\ X_2 \otimes X_2 : & \left[\left(\frac{1}{6} \right) \left(\frac{1}{6} \right), \left(\frac{1}{6} \right) \left(\frac{4}{6} \right), \left(\frac{1}{6} \right) \left(\frac{1}{6} \right) \right. \\ & \left. \left(\frac{4}{6} \right) \left(\frac{1}{6} \right), \left(\frac{4}{6} \right) \left(\frac{4}{6} \right), \left(\frac{4}{6} \right) \left(\frac{1}{6} \right) \right. \\ & \left. \left(\frac{1}{6} \right) \left(\frac{1}{6} \right), \left(\frac{1}{6} \right) \left(\frac{4}{6} \right), \left(\frac{1}{6} \right) \left(\frac{1}{6} \right) \right] \\ X_3 \otimes X_1 : & [(0.0005)1, (0.0308)1, (0.2401)1 \\ & (0.4571)1, (0.2401)1, (0.0308)1, (0.0005)1] \end{aligned} \quad (3.45)$$

Table 3.1: Number of points for SGHQ rule

Level \ Dim.	1	2	3	4	5	...	n
1	1	1	1	1	1	...	1
2	3	5	7	9	11	...	$2n + 1$
3	7	21	37	57	81	...	$2n^2 + 6n + 1$

Table 3.2: Number of points for GHQ rule

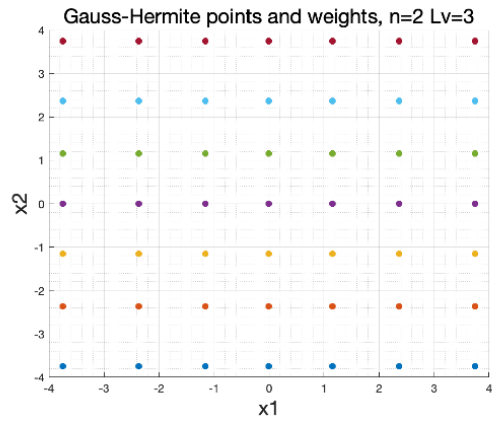
Level \ Dim.	1	2	3	4	5	...	n
1	1	1	1	1	1	...	1
2	3	9	27	81	243	...	$(2^2 - 1)^n$
3	7	49	343	2401	7203	...	$(2^3 - 1)^n$

where the weights for the already existed points are then updated by:

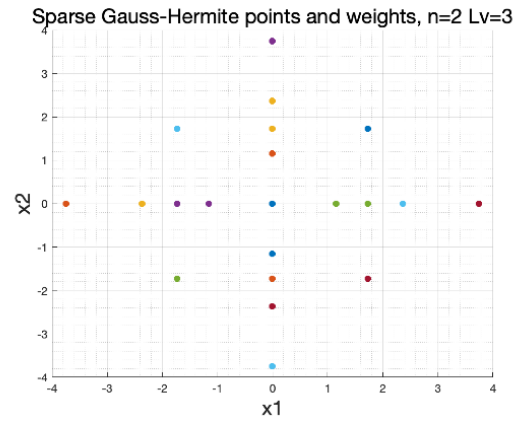
$$w_s = w_s + (-1)^{L-1-q} \mathbf{C}_{L-1-q}^{n-1} \prod_{j=1}^n w_{l_j}$$

$$\begin{aligned}
(0, -\sqrt{3}) &: -\frac{1}{6} + \frac{4}{36} = 0.0556 \\
(0, 0) &: -2 \left(\frac{4}{6} \right) + 2(0.4571) + \frac{11}{36} = 0.0253 \\
(0, \sqrt{3}) &: -\frac{1}{36} + \frac{4}{36} = 0.0556 \\
&\vdots
\end{aligned} \tag{3.46}$$

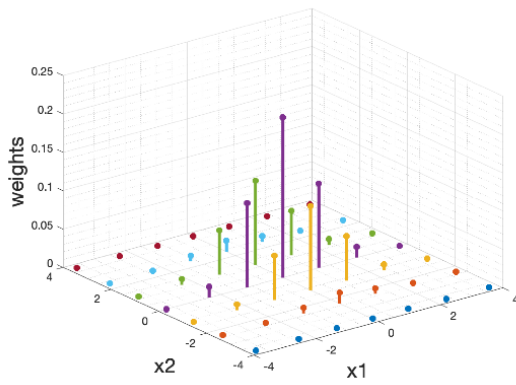
The result points and its weights are illustrated in the following Fig. 3.2 which contains the 21 points, meanwhile the 49 points are used for the that of the GHQ rule by tensor product $X_3 \otimes X_3$. It is obvious from the above figure and tables that the number of the final set of the quadrature points for the SGHQ rule is not increased exponentially compared with the that of the GHQ rule for the higher dimension integration, thus, the curse of the dimensionality issue of the GHQ rule can be alleviated.



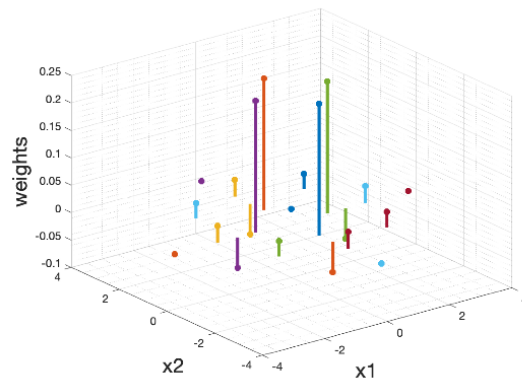
(a) Conventional tensor Product rule, $X_3 \otimes X_3$



(b) Sparse Grid $X_{2,3}$



(c) GHQ points and weights



(d) SGHQ points and weights

Figure 3.2: The comparison of the GHQ and the SGHQ rule. $n = 2$, $L_v=3$

3.3 SGHQ DDP

The SGHQ-DDP also utilizes a set of sampling points generated by the SGHQ rule with a specified accuracy level L and manner m_L : $2L - 1$ or $2^L - 1$. To apply a numerical quadrature based on the Gauss-Hermite Quadrature, the new quadrature points are required to be set by linear transformation (or coordinate rotation) to convert into a normal distribution with a zero mean and unit covariance:

$$\begin{aligned} I &= \int f(x) N(x; \bar{x}, \Sigma_n) dx \\ &= \int f(\bar{\mathbf{x}} + \sqrt{\Sigma} y) N(y; 0_n, I_n) dy \\ &\approx \sum_{l=1}^{(n_l)^n} w_l f(\bar{\mathbf{x}} + \sqrt{\Sigma} \mathbf{y}_l) \end{aligned} \quad (3.47)$$

which can be seen as 'integral by substitution' by changing an integral variable, i.e. $\mathbf{y} = \sqrt{\Sigma}^{-1}(x - \bar{\mathbf{x}})$ with the factorized matrix of the covariance given by $\Sigma = \sqrt{\Sigma}(\sqrt{\Sigma})^T$: Hence, the new points can be computed by the following relation:

$$\begin{bmatrix} \tilde{x}_{i+1}^j \\ \tilde{u}_i^j \end{bmatrix} = \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} + \Gamma \chi^j, \quad j = 1, 2, \dots, M \quad (3.48)$$

where the augmented vector $[\tilde{x}_{i+1}^T, \tilde{u}_i^T]^T$ is new quadrature points, Γ is the lower triangular matrix of the Cholesky decomposition by Eq.(3.2), and χ is a quadrature point by SGHQ-rule. The new quadrature points are then propagated backward through a nonlinear dynamics, $\tilde{x}_i = f^-(\tilde{x}_{i+1}, \tilde{u}_i)$. Similar to the UDP, the backward propagated quadrature points $[\tilde{x}_i^T, \tilde{u}_i^T]^T$ are then used to find the Hessian of Q-function by computing the covariance:

$$G_i^{-1} = \sum_{j=1}^M w^j \left(\begin{bmatrix} \tilde{x}_i^j \\ \tilde{u}_i^j \end{bmatrix} - \begin{bmatrix} \bar{x}_i \\ \bar{u}_i \end{bmatrix} \right) \left(\begin{bmatrix} \tilde{x}_i^j \\ \tilde{u}_i^j \end{bmatrix} - \begin{bmatrix} \bar{x}_i \\ \bar{u}_i \end{bmatrix} \right)^T \quad (3.49)$$

where w^j is a corresponding weight to the quadrature point sampled from the SGHQ rule, and $\bar{x}$, $\bar{u}$ are the mean value of its points, respectively:

$$\begin{bmatrix} \mu_i^{\tilde{x}} \\ \mu_i^{\tilde{u}} \end{bmatrix} = \sum_{j=1}^M w_{\chi}^j \begin{bmatrix} \tilde{x}_i^j \\ \tilde{u}_i^j \end{bmatrix} \quad (3.50)$$

The gradient of the Q-function can also be computed using the principle of the finite-difference:

$$\begin{bmatrix} (\tilde{x}_i^1 - \mu_i^{\tilde{x}})^T & (\tilde{u}_i^1 - \mu_i^{\tilde{u}})^T \\ (\tilde{x}_i^2 - \mu_i^{\tilde{x}})^T & (\tilde{u}_i^2 - \mu_i^{\tilde{u}})^T \\ \vdots & \vdots \\ (\tilde{x}_i^M - \mu_i^{\tilde{x}})^T & (\tilde{u}_i^M - \mu_i^{\tilde{u}})^T \end{bmatrix} \begin{bmatrix} f_x^T V_x' \\ f_u^T V_x' \end{bmatrix} = \begin{bmatrix} \zeta^1 - \bar{\zeta} \\ \zeta^2 - \bar{\zeta} \\ \vdots \\ \zeta^M - \bar{\zeta} \end{bmatrix} \quad (3.51)$$

where $\bar{\zeta}$ is propagated by Eq.(3.9) and the mean of the propagated points are given:

$$\begin{aligned} \tilde{\zeta}_i &= (V_x')^T \tilde{x}_{i+1}^j \\ &= (V_x')^T \begin{bmatrix} f_x \tilde{x}_i \\ f_u \tilde{u}_i \end{bmatrix} \\ \bar{\zeta}_i &= (V_x')^T \mu_{i+1}^{\tilde{x}} \end{aligned} \quad (3.52)$$

The pseudo code for the backward pass of the SGHQ-DDP is given in the Algorithm 10 and the process comparison between the DDP and SGHQ-DDP is shown in the Fig. 3.3.

3.4 SGHQ CDDP

As the CDDP is derived using the Augmented Lagrangian method to consider an equality/inequality constraint in the previous chapter, the SGHQ DDP can also be formulated in a CDDP framework by defining an augmented cost with a constraint condition where the gradient and Hessian of the Q-function are replaced by SGHQ rule. Recall the augmented performance index, or so-called the Lagrangian function Eq.(2.26) and the augmented

Algorithm 10 SGHQ DDP: Backward Pass

- 1: Initialize $V_{x_N} = l_{x_N}$, $V_{xx_N} = l_{xx_N}$
- 2: **for** $i=N-1:-1:1$ **do**
- 3: $\Gamma = chol \left(\begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & l_{uu} \end{bmatrix}_i \right)$
- 4: Generate a set of quadrature points by Eq.(3.48)
- 5: Backward propagation of the new quadrature points: $\tilde{x}_i = f^{-}(\tilde{x}_{i+1}, \tilde{u}_i)$
- 6: Update Q-function Eq.(2.13) and Eq.(2.14) using the gradient and Hessian: Eq.(3.52), Eq.(3.49)
- 7: **if** Q_{uu} is invertible **then**
- 8: Compute the control policy: Eq.(2.11)
- 9: Update the value function: Eq.(2.15)
- 10: **else**
- 11: Increase ρ by Eq.(2.20) go to line 2
- 12: **end if**
- 13: **end for**

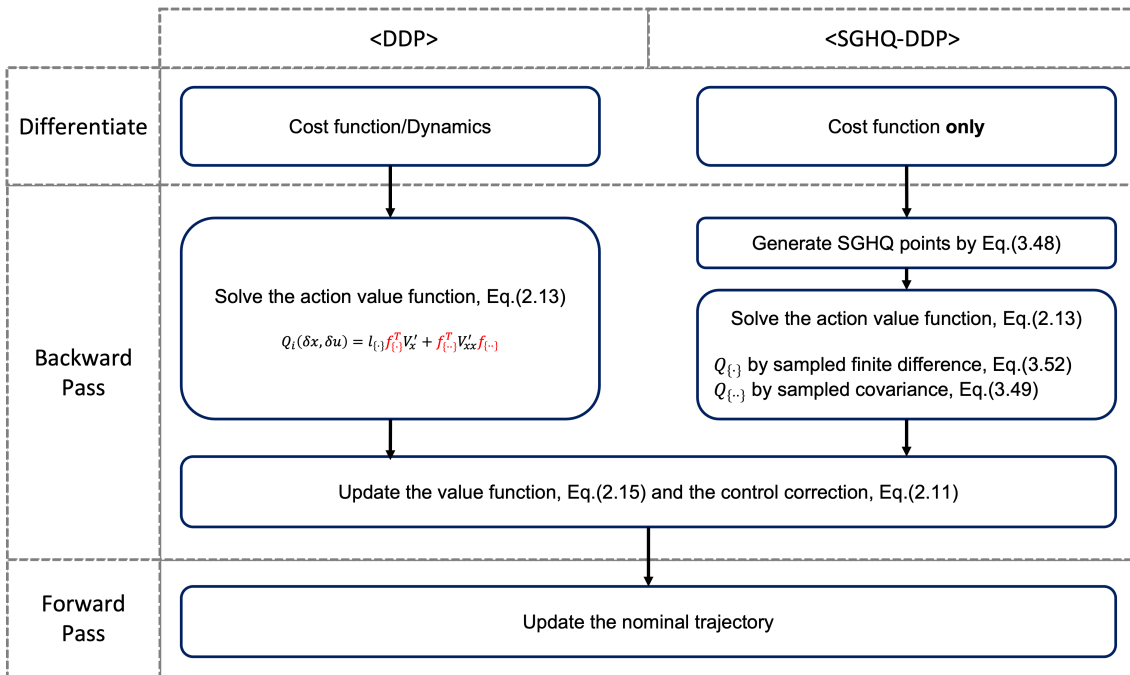


Figure 3.3: Process comparison of the DDP and SGHQ-DDP

running cost Eq.(2.27), the Bellman equation can then be represented based on the CDDP:

$$\begin{aligned}
 \begin{bmatrix} Q_{axx} & Q_{axu} \\ Q_{aux} & Q_{auu} \end{bmatrix}_i &= \begin{bmatrix} f_x^T V'_{axx} f_x & f_x^T V'_{axx} f_u \\ f_u^T V'_{axx} f_x & f_u^T V'_{axx} f_u + l_{auu} \end{bmatrix}_i + \begin{bmatrix} l_{axx} & l_{axu} \\ l_{aux} & 0_{m \times m} \end{bmatrix}_i \\
 &= \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^T \begin{bmatrix} V'_{axx} & 0_{n \times m} \\ 0_{m \times n} & l_{auu} \end{bmatrix}_i \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i + \begin{bmatrix} l_{axx} & l_{axu} \\ l_{aux} & 0_{m \times m} \end{bmatrix}_i
 \end{aligned}
 \tag{3.53}$$

Therefore, the second derivative term of the Q_a -function, G_{ai} , can be defined:

$$\begin{aligned}
 G_{ai} &= \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^T \begin{bmatrix} V'_{axx} & 0_{n \times m} \\ 0_{m \times n} & l_{auu} \end{bmatrix}_i \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i \\
 \therefore G_{ai}^{-1} &= \begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^{-1} \begin{bmatrix} V'_{axx} & 0_{n \times m} \\ 0_{m \times n} & l_{auu} \end{bmatrix}_i^{-1} \left(\begin{bmatrix} f_x & f_u \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}_i^{-1} \right)^T \quad (3.54) \\
 &= \bar{F}_i^{-1} \bar{P}_{ai}^{-1} \bar{F}_i^{-T}
 \end{aligned}$$

If the Cholesky decomposition of the $G_{ai}^{-1} = \Lambda_a \Lambda_a^T$ and $\bar{P}_{ai} = \Gamma_a \Gamma_a^T$ are satisfied as described in the UDP, where Λ_a and Γ_a are the unique lower triangular matrices, then, the following equation can also be derived as:

$$\begin{aligned}
 \Lambda_a \Lambda_a^T &= \left(\bar{F}_i^{-1} \right) \Gamma_a \Gamma_a^T \left(\bar{F}_i^{-1} \right)^T \\
 \therefore \Lambda_a &= \bar{F}_i^{-1} \Gamma_a
 \end{aligned} \quad (3.55)$$

where Eq.(3.55) forms a nonlinear transformation technique to approximate Λ_a through the dynamics, $\bar{F}_i^{-1}$. Thus, G_a^{-1} can then be approximated by Eq.(3.49), Eq.(3.50). The gradient of the Q_a -function is also be calculated by principle of the finite-difference using Eq.(3.52) with the augmented value V_{ax} and V_{axx} .

3.5 Numerical Simulation

This section demonstrates the effectiveness of the SGHQ-based (C)DDP algorithm. The numerical simulations are also carried out with the F/A-18 and missile dynamics used in the section 2.4, however, the considered simulation scenarios are more highly nonlinear cases for the real implementations. For the first fixed-wing aircraft case, a flat-spin recovery scenario is considered, which is an autorotation or uncommanded roll along the longitudinal axis with a downward rotating motion around the vertical axis [65]. The

objective is to stabilize its airspeed, rate of rotation (or angular velocity) and attitude by disrupting the spinning motion, decreasing the AoA below the critical AoA and unstalling the wing. The second missile guidance also assumes the interception of the stationary target on the ground with a time-varying parameter of the missile dynamics, such as mass, aerodynamic coefficient, and thrust to consider the boost phase maneuver. All the information including both of aircraft and missile dynamics of the post-stall aerodynamics and time-varying parameters, respectively, are also fully available.

3.5.1 A fixed-wing aircraft

The performance index is formulated by Eq. (2.48) with a physical constraint of the control input by Eq. (2.49) for an inequality constraint condition, $g(x, u) \leq 0$ with the corresponding state x , and control input, u defined by Eq. (2.47). The initial condition of the flat spin is listed on the Table 3.3 [66]. The parameters of the (C)DDP, i.e. time

Table 3.3: Simulation initial/desired conditions of the F/A-18: flat spin recovery

	Parameter	Value
Initial condition	Velocity, V_T	80.89 [m/s]
	AOA/AOS, $[\alpha, \beta]^T$	$[71.63^\circ, 2.4^\circ]^T$
	Angular velocity, $[p, q, r]^T$	$[-28.40, -3.28, -86.76]^T$ [deg/s]
	Angular attitude, $[\phi, \theta]^T$	$[1.36^\circ, -17.36^\circ]^T$
	Control inputs, $[\delta_{thr}, \delta_{el}, \delta_{er}, \delta_a, \delta_r]^T$	$0_{m \times 1}$
Trim condition	Velocity, V_T	180 [m/s]
	AOA/AOS, $[\alpha, \beta]^T$	$[6.83^\circ, 0^\circ]^T$
	Angular velocity, $[p, q, r]^T$	$[0, 0, 0]^T$ [deg/s]
	Angular attitude, $[\phi, \theta]^T$	$[0^\circ, 6.83^\circ]^T$
	Control inputs, $[\delta_{thr}, \delta_{el}, \delta_{er}, \delta_a, \delta_r]^T$	$[28\%, -0.55^\circ, -0.55^\circ, 0^\circ, 0^\circ]^T$

horizon, weighting matrices and Augmented Lagrangian parameters are also presented in the Table 3.4, where the weighting function of the state value in the running cost, Q , has more priority on the angular rotation recovery attenuating the angular rate $\omega^b = [p, q, r]^T$ than any other state regulation, which is highly recommended by Federal Aviation Administration (FAA) [65].

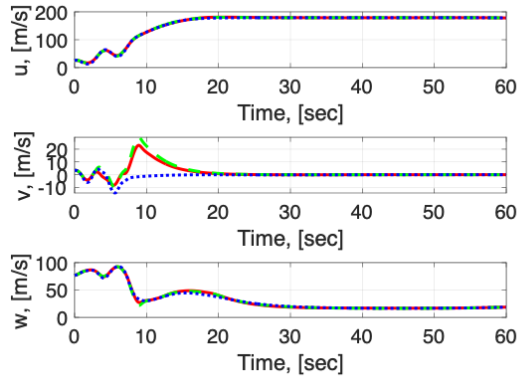
The Fig. 3.4 and Fig. 3.5 represent the aircraft state and input trajectories of the Con-

Table 3.4: Simulation parameters of the CDDP for F/A-18: flat spin recovery

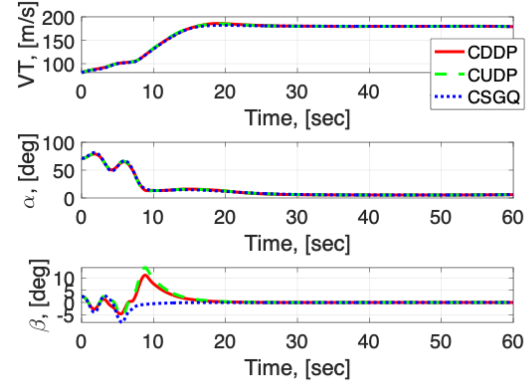
	Parameter	Value
DDP	Sampling time, dt	0.1s
	Time horizon, N	600
	Q	$diag([10^{-2}, 10^2 I_{2 \times 2}, 10^3 I_{3 \times 3}, 10 I_{2 \times 2}])$
	R	$I_{m \times m}$
	S_N	$10Q$
CDDP	Lagrange multiplier, λ_{ini}	1
	Penalty parameter, μ_{ini}	10^{-2}
	Scheduling factor, C_μ	1.2

strained DDP, (CDDP), Constrained UDP, (CUDP) and SGHQ-CDDP algorithms for a lower level of the control law for flat-spin recovery. All the three algorithms regulate the state and control input variables in 20s, and the trajectories then converge to their desired values. As shown in the Fig. 3.4e and Fig. 3.4f, all the algorithms use the bang-bang type maximum control input to decrease the angular velocity ω^b [65]. All the cases have similar responses attenuating the angular velocity rapidly and then stabilizing the other angular attitudes ($[\phi, \theta]^T$) and velocity (v^b), however, aileron and rudder have different responses that leads to the abrupt change in the lateral motion, roll attitude (ϕ) and sideslip angle (β), which are shown in the Fig. 3.4b and Fig. 3.4d. When it comes to the three CDDP algorithms, the SGHQ-CDDP takes more iteration during the numerical optimization, however, it has the least cost compared with the CDDP and CUDP presented in the Fig. 3.8 and the Table 3.7. Therefore, all of the CDDP-based algorithms can be applied to design a recovery control system and the SGHQ-based approach shows a slightly better performance.

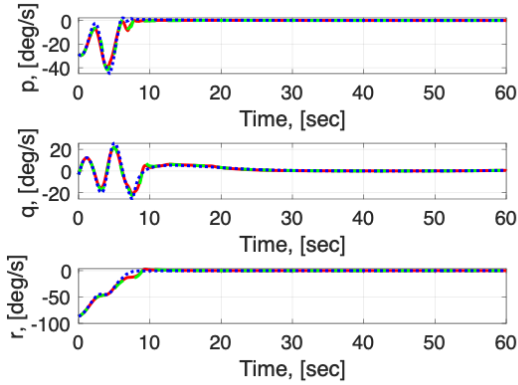
Unlike in the optimal regulating problem of the low level controller system, stall, spin or unusual attitude of an aircraft should be recovered or stabilized within a minimum time. The characteristics of these flight conditions have lower maneuverability even make loss of its controllability that can lead to a catastrophic condition, hence, it can be expected that the FFT algorithm effects on the this problem by optimizing a final time. The following figures and table demonstrate the result of the FFT-CDDP algorithms. The state and input sequences of all algorithms are given in the Fig. 3.6 and 3D trajectories and its altitude



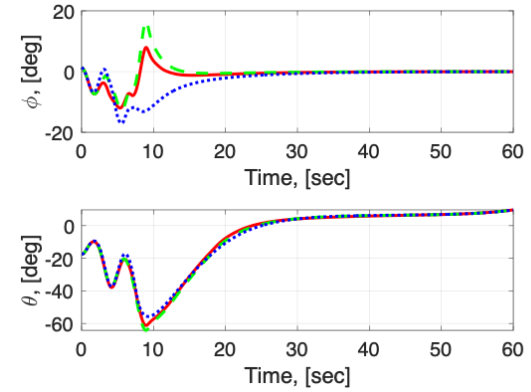
(a) Time history of the state: body velocity



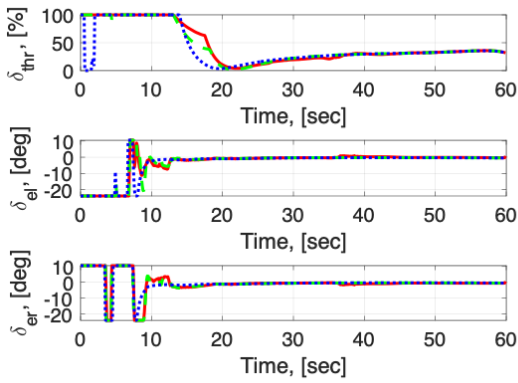
(b) Time history of the state: velocity, AoA, AoS



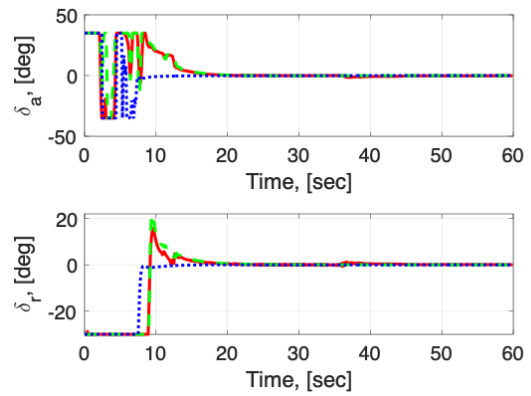
(c) Time history of the state: angular velocity



(d) Time history of the state: Euler angle



(e) Time history of the state: throttle, left/right elevator



(f) Time history of the input: aileron and rudder

Figure 3.4: Comparison result of the CDDP algorithms of F/A-18: flat-spin recovery

loss are shown in Fig. 3.7. It is clearly observed that the the conventional DDP and SGHQ-based FFT algorithms recovers from a flat-spin by attenuating its angular velocity in the first 10s, which is a similar result of the previous result given in the Fig. 3.4,

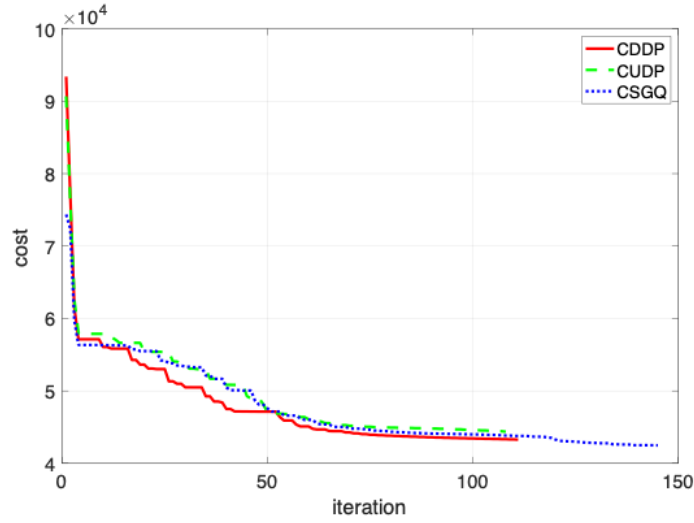
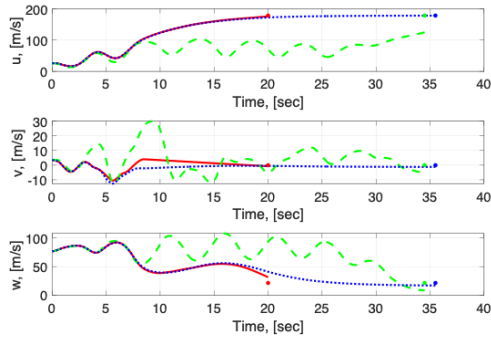
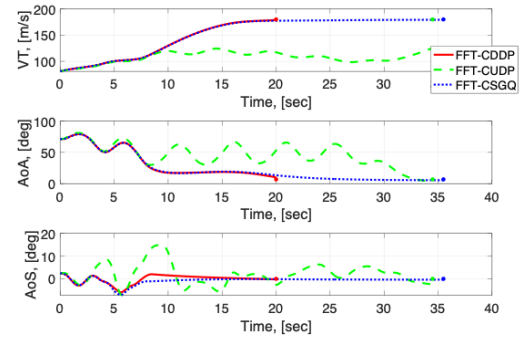


Figure 3.5: Cost comparison of F/A-18: Flat-spin recovery

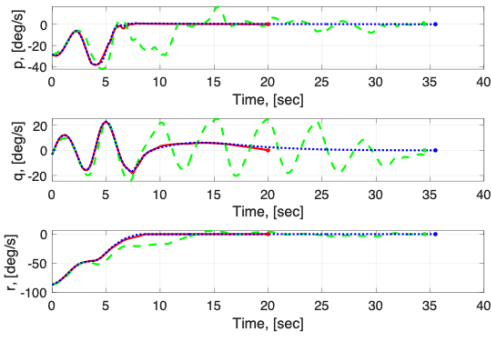
while the UDP-based algorithm fails to regulate its angular rate and stabilize its attitude and input. In addition, the only SGHQ-based algorithm has ability to regulate its unusual attitude to the desired values after stabilizing angular velocity stage in 35.5s. Even though the CDDP successes in the regulating angular rates in a shortest time, its trajectory does not converge to the desired value in the attitude and the roll and pitch attitude $([\phi, \theta]^T)$ have a nose and left-wing down motion shown in Fig. 3.6d. Lastly, the performance of the FFT algorithms are compared in the Fig. 3.8 and Table 3.5. When it comes to the altitude loss comparison, the trajectories of the DDP and UDP are extended using the feedforward K_i and feedback d_i gains of the each final time that is described in the Algorithm 8 for the explicit comparison given the same time horizon. It can be definitely seen from the Fig. 3.7 that the SGHQ-based algorithm has the minimum altitude loss compared with the that of DDP and UDP, summarized in the Table 3.5. This is a noticeable result that to minimize altitude loss is a significant parameter to prevent fatal aircraft accident as flat spin is a downward rotation motion with the loss of control. In terms of the cost, both CDDP and SGHQ decrease its cost obviously and the final costs are comparatively similar value even though the time horizon of the CDDP reduces to the least small time, which is the minimum saturation of the time increment, $\delta_{t_{Nmin}}$. This means that the SGHQ algorithm has a longer time 'N'-step in the running cost $l_a(x_i, u_i)$ and a better tracking



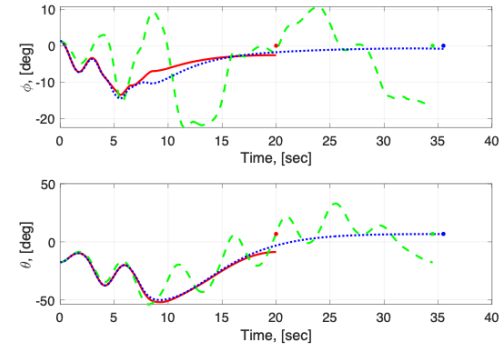
(a) Time history of the state: body velocity



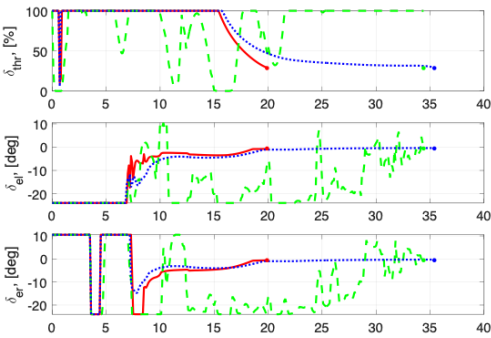
(b) Time history of the state: velocity, AoA, AoS



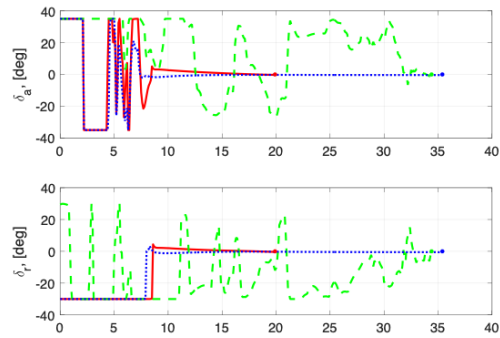
(c) Time history of the state: angular velocity



(d) Time history of the state: Euler angle



(e) Time history of the state: throttle, left/right elevator



(f) Time history of the input: aileron and rudder

Figure 3.6: Comparison result of the FFT-CDDP algorithms of F/A-18: flat-spin recovery

performance of the SGHQ algorithm can be expected if a given time horizon for both algorithms are equivalent. Thus, the SGHQ has more efficiency in the nonlinear case even with the FFT algorithm.

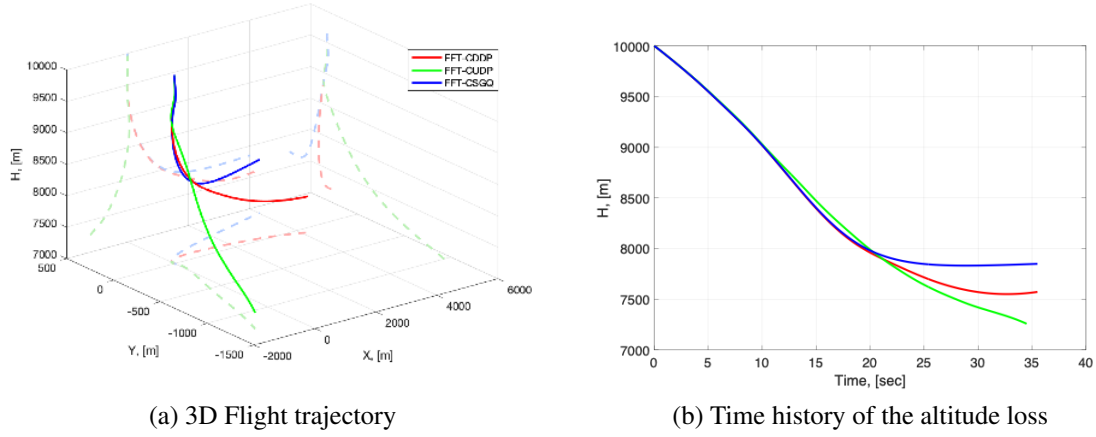


Figure 3.7: 3D Trajectory of F/A-18: flat-spin recovery

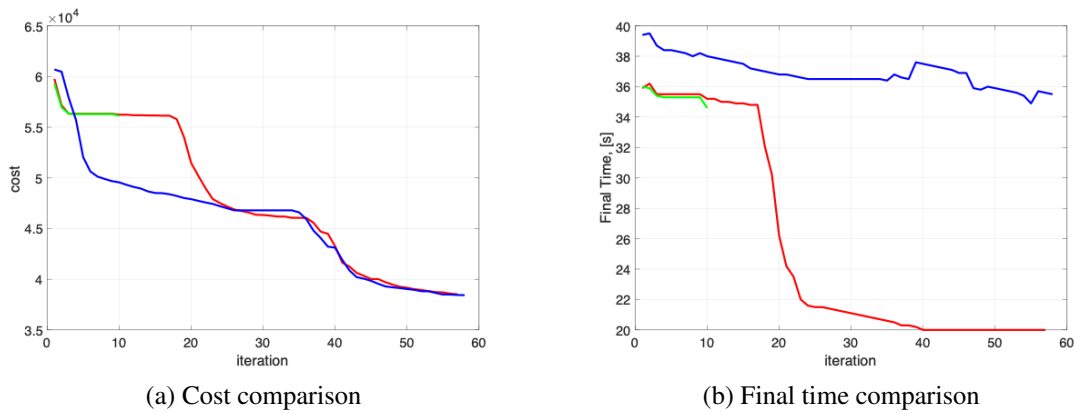


Figure 3.8: Comparison result of the FFT-CDDP F/A-18: flat-spin recovery

Table 3.5: Comparison result of F/A-18: flat-spin recovery

	Ini. Final Time, t_{Nini}	Final Time, t_N	Iteration	Cost	Altitude Loss
CDDP	40s	20s	57	38515.7	2741.69m
CUDP	40s	34.5s	10	56115.7	2427.59m
SGHQ-CDDP	40s	35.8s	58	38424.8	2150.68m

3.5.2 Missile guidance

The missile dynamics used in this subsection is the same as the previous subsection given by Eq. (2.50) and Fig. 2.5, however, time-varying parameters are considered: mass, aerodynamic coefficient, thrust force, and the acceleration command limit is set by $100 [m/s^2]$ due to the accelerating phase [11, 67]. The performance index and inequality condition set by the control input are formulated by Eq. (2.51), and the state and control input

vector are also equivalent to subsection 2.4.2. The initial and desired conditions for the simulation are given in the Table 3.6 and the parameters for the CDDP and FFT algorithm are also given by Table 2.4. All the results of the CDDP, CUDP and SGHQ-CDDP with

Table 3.6: Simulation initial/desired conditions of the missile

	Parameter	Value
Missile parameter [11]	Mass	(Appendix A.2)
	Drag coefficient, $[C_{D0}, C_{Di}]^T$	
Initial condition	Thrust, F_T	$[5000m, 5000m, -5000m]^T$
	Position, $[X_m, Y_m, Z_m]^T$	
	Velocity, V_m	
	Angle, $[\gamma_m, \psi_m]^T$	
Target value	Control input, $[a_{my}, a_{mz}]^T$	$[0, 0]^T [m/s^2]$
	Position, $[X_m, Y_m, Z_m]^T$	$[0m, 0m, 0m]^T$
	Velocity, V_m	$300 [m/s]$
	Angle, $[\gamma_m, \psi_m]^T$	$[-90^\circ, -240^\circ]^T$

the FFT algorithm are shown in Fig. 3.9. As the state trajectory is shown in the Fig. 3.9a, Fig. 3.9b and Fig. 3.9c, all the CDDP algorithms intercept the target with the desired impact angles, successfully. Even though the weighting for the velocity in the running cost, $Q(4,4)$, is set by a small value for the free final velocity, the velocity condition is also satisfied by accelerating the velocity in the boost phase and decelerating by aerodynamic force (by drag) while the terminal condition of the velocity is not satisfied in the time-invariant case from the subsection 2.4.2. Moreover, the SGHQ-based algorithm

Table 3.7: Comparison result of the missile guidance

	Initial Final Time, t_{Nini}	Final Time, t_N	Iteration	Cost
CDDP	30s	24.1s	24	767733
CUDP	30s	24.3s	24	770321
SGHQ-CDDP	30s	18.6s	25	696472

takes the least time (Fig. 3.10b) with the minimum cost by accelerating its acceleration in the initial phase plotted in the Fig. 3.9d although the DDP iteration number is similar to the other algorithm, summarized in the Table 3.7 and Fig. 3.10a. From the subsection 2.4.2, it is confirmed that the FFT algorithm improves the ability to optimize a trajectory optimization problem, and the SGHQ-based algorithm provides the better performance in

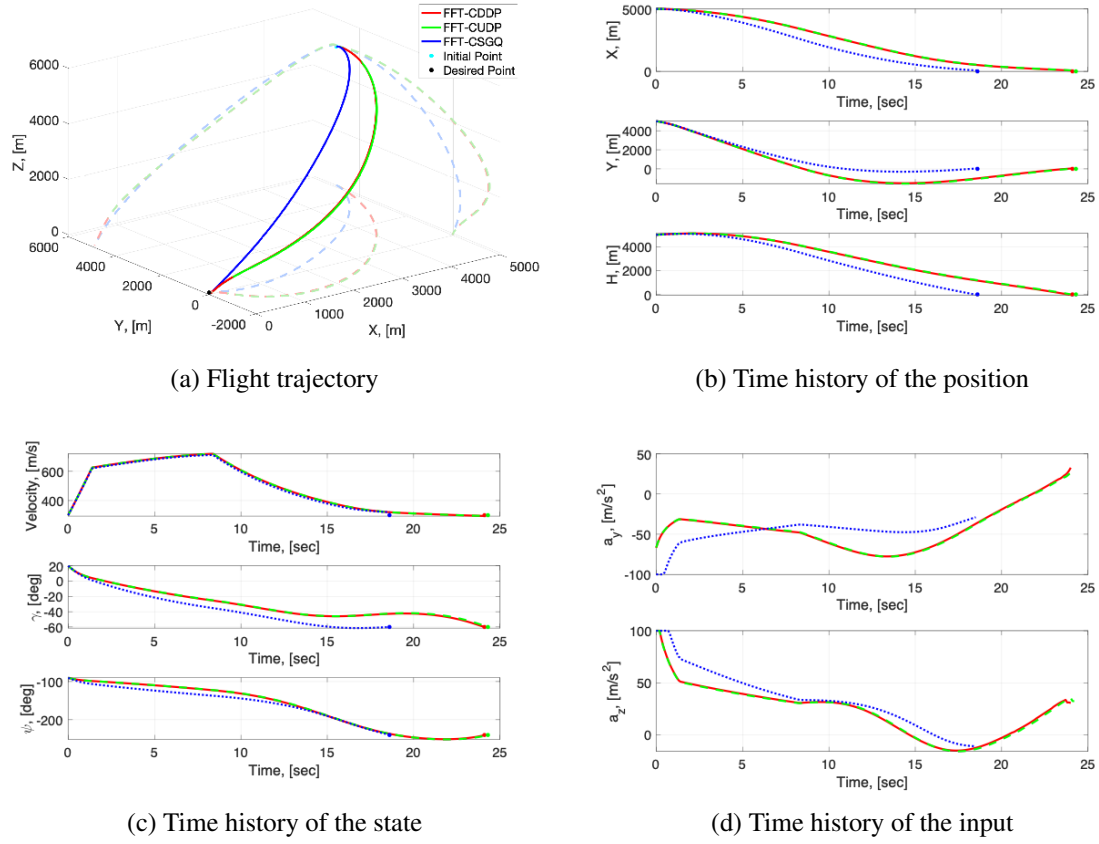


Figure 3.9: Comparison result of the FFT algorithms in missile guidance: state and input trajectories

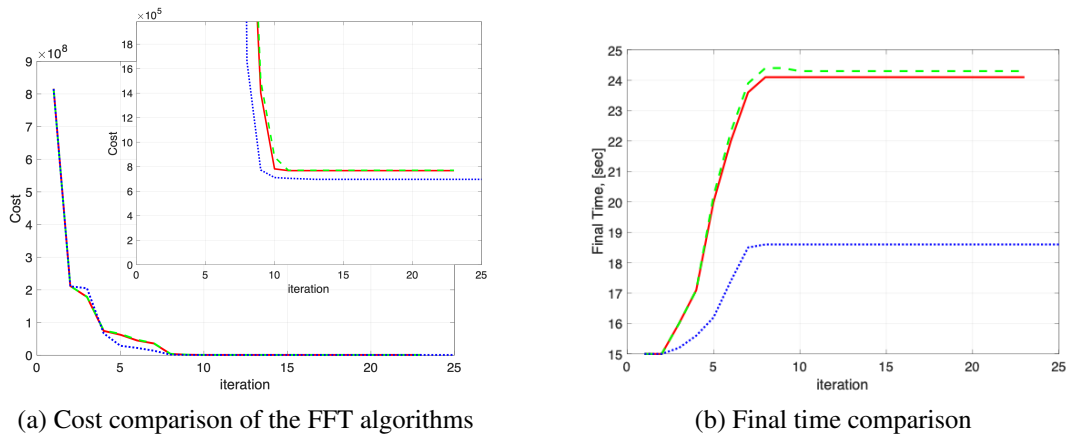


Figure 3.10: Comparison result of the FFT algorithms in missile guidance: cost and time the nonlinear trajectory optimization problem.

3.6 Conclusion

This chapter demonstrates the quadrature point-based DDP algorithm especially utilizing the Sparse Gauss-Hermite Quadrature rule sampling in accordance with the Smolyak's rule called as the SGHQ rule. The idea is come from the unscented transform of the Unscented Kalman Filter replacing the gradient and Hessian of the action-value function by numerical computation of a probability distribution, which are compared with the baseline algorithm and the UDP in the Table 3.8. The transformed points sampled by the SGHQ rule mimics the gradient and Hessian of the Bellman equation in the backward computation of the DDP framework, which achieves a less approximation error. This also enables the update of the Bellman equation in the backward computation to find an optimal control correction without the differentiation of the cost, action-value function and dynamics. Moreover, the SGHQ can adjust the level of the quadrature rule that affects on the accuracy of the numerical integration so that the more quadrature points can be utilized to approximate the probability distribution. Even though the more quadrature points are selected by a higher level of the Sparse grid rule that means it would have more *sigma point* of the UDP, it still has less number of the fully-gridded Gauss-Hermite quadrature points and resolves the curse of dimensionality issue.

Furthermore, the FFT-based SGHQ-CDDP algorithm is also formulated for the free final time trajectory optimization problem. The derived FFT-SGHQ-CDDP is implemented in the aerospace applications for the low level controller and guidance law of a fixed-wing aircraft and missile guidance, respectively. The numerical simulations show that the integrated SGHQ-CDDP algorithm with the FFT algorithm has a better performance and accuracy and is applicable to the final time free problem where the time horizon should be minimized or adjusted for a successful implementation without the numerical discretization in the dynamics.

Table 3.8: Comparison of DDP and sampling point-based DDPs

	DDP	UDP	SGHQ-DDP
Pros	(Baseline algorithm)	Derivative-free	Derivative-free Point level selection
Cons	Table 2.6	Fixed-sampling points	Rule for level selection

Chapter 4

SGHQ Guided Policy Search

4.1 Stochastic Policy Optimization

Previous chapter 2 and 3 described a trajectory optimization problem solved by Differential Dynamic Programming (DDP) algorithm and its variant of sampling point-based approaches: Unscented Dynamic Programming (UDP) and Sparse Gauss-Hermite Quadrature DDP (SGHQ-DDP). The DDP algorithm addresses a single locally optimized trajectory near a nominal (or previous) trajectory approximated by Taylor series expansion, while policy search or inverse optimal control (or Inverse Reinforcement Learning) handles a various of possible trajectories in the framework of stochastic optimization problem. The DDP solves a deterministic system model and control input, however, a stochastic optimization deals with a propabilistic system model $p(x_{i+1}|x_i, u_i)$ and a stochastic control policy $p(u_i|x_i)$ conditioned on x_i . Hence, a number of available trajectories need to be generated to make a trajectory distribution so the control policy that can learn a policy distribution not for a single possible trajectory but a task execution. To solve a stochastic optimization problem by finding a control policy, the classic optimization problem Eq.(2.2) should be converted to the expected cost by $\mathbb{E}[\sum_{i=1}^{N-1} l(x_i, u_i) + l_f(x_N)]$, and then an entropy term is augmented in this expected cost to obtain trajectory distributions as wide as possible [48]: $\mathbb{E}[\sum_{i=1}^{N-1} l(x_i, u_i) + \log(u_i|x_i) + l_f(x_N)]$, which is called as

a soft optimality or maximum entropy control problem. [68, 69]

Consider a stochastic trajectory optimization problem of which objective is to minimize the expected cost with respect to a control policy $\pi(u_i|x_i)$ below:

$$J = \mathbb{E}_{\tau \sim p(\tau)} \sum_{i=1}^{N-1} \{l(x_i, u_i) + \log \pi(u_i|x_i)\} + l_f(x_N) \quad (4.1)$$

where τ is a system trajectory from an initial condition (x_1, u_1) , i.e. $\tau = \{x_1, u_1, x_2, u_2, \dots, x_N\}$, and $\log \pi(u_i|x_i)$ is a Gaussian control policy expressed by multivariate Gaussian function with mean $\mu_i^u \in R^m$ and covariance $S_i \in R^{m \times m}$:

$$\begin{aligned} \pi(u_i|x_i) &= N(u_i|\mu_i^u, S_i) \\ &= \frac{1}{\sqrt{(2\pi)^m \det(S_i)}} \exp\left\{-\frac{1}{2}(u_i - \mu_i^u)^T S_i^{-1} (u_i - \mu_i^u)\right\} \end{aligned} \quad (4.2)$$

The mean of the linear Gaussian controller can be set by $\mu_i^u = \bar{u}_i + d_i + K_i(x_i - \bar{x}_i)$ as shown in the DDP, Eq.(2.16). From the Bellman's principle of optimality which states that an optimal control policy has the property that whatever the initial condition and decision procedure are, the remaining decisions must constitute of an optimal policy starting from the state resulting from the last decision, the given cost function Eq.(4.1) can be expressed in a recurrent form of the value function as shown in the value function from the Eq.(2.5):

$$\begin{aligned} J &= \mathbb{E}_{\tau_{u_1} \sim \pi(u_1|x_1)} \left[\sum_{i=1}^{N-1} \{l(x_i, u_i) + \log \pi(u_i|x_i)\} + l_f(x_N) \right] \\ &= V(x_1) \end{aligned} \quad (4.3)$$

which is called as a soft state-value function where the $\log \pi(u_i|x_i)$ term is augmented in the deterministic trajectory optimization problem Eq.(2.2). To apply the Dynamic Programming (DP) in the soft state-value function, the Eq.(4.3) is rearranged at the time step

k with a Bayes rule:

$$\begin{aligned}
V(x_k) &= \mathbb{E}_{\tau_{u_k} \sim p(\tau_{u_k}|x_k)} \left[\sum_{i=k}^{N-1} \frac{\{l(x_i, u_i) + \log \pi(u_i|x_i)\}}{=\tilde{l}(x_i, u_i)} + l_f(x_N) \right] \\
&= \mathbb{E}_{\tau_{u_k} \sim p(\tau_{u_k}|x_k)} \left[\log \pi(u_k|x_k) + \sum_{i=k}^{N-1} \frac{\{l(x_i, u_i) + \log \pi(u_{i+1}|x_{i+1})\}}{=l_{\pi_i}} + l_f(x_N) \right] \\
&= \int_{\tau_{u_k}} \log \pi(u_i|x_i) p(\tau_{u_k}|x_k) d\tau_{u_k} + \int_{\tau_{u_k}} \left[\sum_{i=k}^{N-1} \{l_{\pi_i}\} + l_f(x_N) \right] p(\tau_{u_k}|x_k) d\tau_{u_k} \\
&= \int_{u_k} \int_{\tau_{x_{k+1}}} \log \pi(u_i|x_i) p(\tau_{x_{k+1}}|x_k, u_k) d\tau_{x_{k+1}} \pi(u_k|x_k) du_k \\
&\quad + \underbrace{\int_{u_k} \int_{\tau_{x_{k+1}}} \left[\sum_{i=k}^{N-1} \{l_{\pi_i}\} + l_f(x_N) \right] p(\tau_{x_{k+1}}|x_k, u_k) d\tau_{x_{k+1}} \pi(u_k|x_k) du_k}_{=Q(x_k, u_k)} \\
\therefore V(x_k) &= \mathbb{E}_{u_k \sim \pi(u_k|x_k)} \left[\log \pi(u_k|x_k) + Q(x_k, u_k) \right]
\end{aligned}$$

where $Q(x_k, u_k) = \mathbb{E}_{\tau_{x_{k+1}} \sim p(\tau_{x_{k+1}} | x_k, u_k)} \left[\sum_{i=k}^{N-1} \{l(x_i, u_i) + \log \pi(u_{i+1} | x_{i+1})\} + l_f(x_N) \right]$ is a soft action-value function. Rearrange the soft action-value function as described in the soft value function Eq.(4.4):

$$\begin{aligned}
Q(x_k, u_k) &= \int_{\tau_{x_{k+1}}} \left[\sum_{i=k}^{N-1} \{l(x_i, u_i) + \log \pi(u_{i+1} | x_{i+1})\} + l_f(x_N) \right] p(\tau_{x_{k+1}} | x_k, u_k) d\tau_{x_{k+1}} \\
&= \int_{\tau_{x_{k+1}}} l(x_k, u_k) + \left[\sum_{i=k+1}^{N-1} \tilde{l}(x_i, u_i) + l_f(x_N) \right] p(\tau_{x_{k+1}} | x_k, u_k) d\tau_{x_{k+1}} \\
&= \int_{x_{k+1}} \int_{\tau_{u_{k+1}}} l(x_k, u_k) p(\tau_{u_{k+1}} | x_{k+1}) d\tau_{u_{k+1}} p(x_{k+1} | x_k, u_k) dx_{k+1} \\
&\quad + \underbrace{\int_{x_{k+1}} \int_{\tau_{u_{k+1}}} \left[\sum_{i=k+1}^{N-1} \tilde{l}(x_i, u_i) + l_f(x_N) \right] p(\tau_{u_{k+1}} | x_{k+1}) d\tau_{u_{k+1}} p(x_{k+1} | x_k, u_k) dx_{k+1}}_{=V(x_{k+1})} \\
&= l(x_k, u_k) + \int_{x_{k+1}} V(x_{k+1}) p(x_{k+1} | x_k, u_k) dx_{k+1} \\
\therefore Q(x_k, u_k) &= l(x_k, u_k) + \mathbb{E}_{x_{k+1} \sim p(x_{k+1} | x_k, u_k)} [V(x_{k+1})]
\end{aligned} \tag{4.5}$$

Hence, the recursive relation between the current time i and the next time $i + 1$ called the

soft Bellman equation can also be found in a stochastic form using Eq.(4.4) and Eq.(4.5):

$$\begin{aligned} V(x_k) &= \mathbb{E}_{u_k \sim \pi(u_k|x_k)} \left[\log \pi(u_k|x_k) + Q(x_k, u_k) \right] \\ &= \mathbb{E}_{u_k \sim \pi(u_k|x_k)} \left[\log \pi(u_k|x_k) + l(x_k, u_k) + \mathbb{E}_{x_{k+1} \sim p(x_{k+1}|x_k, u_k)} [V(x_{k+1})] \right] \end{aligned} \quad (4.6)$$

From the Bellman's principle of the optimality, the following equation can be satisfied:

$$V^*(x_k) = \min_{\pi} \mathbb{E}_{u_k \sim \pi(u_k|x_k)} \left[\log \pi(u_k|x_k) + l(x_k, u_k) + \mathbb{E}_{x_{k+1} \sim p(x_{k+1}|x_k, u_k)} [V^*(x_{k+1})] \right] \quad (4.7)$$

To solve the stochastic trajectory optimization, the second order Taylor series expansion near a nominal trajectory (x_k, u_k) can be applied in the Eq.(4.2) with a multivariate Gaussian function Eq.(4.2):

$$\begin{aligned} V(\delta x_k) &= \mathbb{E}_{\delta \mu_k^u \sim \pi(\delta \mu_k^u | \delta x_k)} \left[-\frac{1}{2} \{ (\delta u_k - \delta \mu_k^u)^T S_k^{-1} (\delta u_k - \delta \mu_k^u) + \log(2\pi)^m + \log \det(S_k) \} \right. \\ &\quad \left. + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} \right] \end{aligned} \quad (4.8)$$

The recursive relation Eq.(2.12) can also be satisfied to express the gradient and the Hessian of the Q-function and a partial derivative with respect to the mean and the covariance of the control input gives us an optimal control input as:

$$\begin{aligned} \delta \mu_k^u &= \frac{\partial V_k}{\partial \delta \mu_k^u} \\ &= Q_u + Q_{ux} \delta x_k + Q_{uu} \delta \mu_k^u \\ \therefore \delta \mu_k^u &= -Q_{uu}^{-1} (Q_{ux} \delta x_k + Q_u) \\ &= -[Q_{uu} + \rho I]^{-1} (Q_{ux} \delta x_k + Q_u) \\ &= K_k \delta x_k + d_k \end{aligned} \quad (4.9)$$

The mean of the Gaussian control policy is equivalent to the optimal control correction of

the DDP, Eq.(2.11) and a regularization parameter ρ can also be implemented to guarantee the invertibility of Q_{uu} as well.

$$\begin{aligned} S_k &= \frac{\partial V_k}{\partial S_k} \\ &= -\frac{1}{2}S_k^{-1} + \frac{1}{2}Q_{uu} \\ \therefore S_k &= Q_{uu}^{-1} \end{aligned} \quad (4.10)$$

Thus, the soft action-value function can be rewritten

$$\begin{aligned} Q(\delta x_k, \delta u_k) &= -\frac{1}{2}\{tr(I_{m \times m}) + \log(2\pi)^m + \log \det(S_k)\} \\ &\quad + \mathbb{E}_{\delta u_k \sim \pi(\delta u_k | \delta x_k)} \left[\frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} \right] \\ &= -\frac{1}{2}\{tr(I_{m \times m}) + \log(2\pi)^m + \log \det(S_k)\} \\ &\quad + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} + \frac{1}{2}tr\{Q_{uu}S_k\}^{S_k=Q_{uu}^{-1}} \end{aligned} \quad (4.11)$$

By substituting the obtained mean of the optimal control correction Eq.(4.9) into the Eq.(4.11) shows the following equation:

$$\begin{aligned} Q(\delta x_k, \delta u_k) &= -\frac{1}{2}\{\log(2\pi)^m + \log \det(S_k)\} \\ &\quad + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ K_k \delta x_k + d_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ K_k \delta x_k + d_k \end{bmatrix} \\ &= -\frac{1}{2}\{\log(2\pi)^m + \log \det(S_k)\} + \delta x_k^T [Q_x + K_k^T Q_u + Q_{xu}d_k + K_k^T Q_{uu}d_k] \\ &\quad + \frac{1}{2}\delta x_k^T [Q_{xx} + Q_{xu}K_k + K_k^T Q_{ux} + K_k^T Q_{uu}K_k] + d_k^T Q_u + \frac{1}{2}d_k^T Q_{uu}d_k \end{aligned} \quad (4.12)$$

The gradient and Hessian of the soft state-value function and the expected change in cost are:

$$\begin{aligned} V_x &= Q_x + K_k^T Q_u + Q_{xu} d_k + K_k^T Q_{uu} d_k \\ V_{xx} &= Q_{xx} + Q_{xu} K_k + K_k^T Q_{ux} + K_k^T Q_{uu} K_k \\ \Delta V &= d_k^T Q_u + \frac{1}{2} d_k^T Q_{uu} d_k \end{aligned} \quad (4.13)$$

The gradient and Hessian of the V function are also equivalent to the result of the DDP Eq.(2.15), thus, the Gaussian policy can be implemented in a DDP framework to find mean and covariance of the optimal control policy.

$$\therefore \pi^*(u_i|x_i) = N(\bar{u}_i + K_i(x_i - \bar{x}_i) + d_i, Q_{uu}^{-1}) \quad (4.14)$$

4.2 Guided Policy Search

Consider a stochastic trajectory optimization problem with an inequality constraint condition [70]:

$$J = \mathbb{E}_{\tau \sim p(\tau)} \sum_{i=1}^{N-1} l(x_i, u_i) + l_f(x_N), \quad \text{s.t. } D_{KL}(p(\tau) || \bar{p}(\tau)) \leq \varepsilon \quad (4.15)$$

where τ is a trajectory from the initial condition (x_1, u_1) , i.e. $\tau = \{x_1, u_1, x_2, u_2, \dots, x_N\}$ and $p(\tau)$ is a trajectory distribution induced by the local policy $p(u_i|x_i)$ with a dynamics distribution $p(x_{i+1}|x_i, u_i)$:

$$p(\tau) = p(x_1) \prod_{i=1}^{N-1} p(x_{i+1}|x_i, u_i) p(u_i|x_i) \quad (4.16)$$

D_{KL} is a Kullback-Leibler (KL) divergence of the trajectories between the trajectories generated by the previous policy $\bar{p}(u_i|x_i)$ and the current policy $p(u_i|x_i)$, respectively. It prevents trajectory deviation too far from the previous trajectory distribution $\bar{p}(\tau)$ by a

boundary ε :

$$\begin{aligned}
D_{KL}(p(\tau)||\bar{p}(\tau)) &= \mathbb{E}_{\tau \sim p(\tau)} \left[\log p(\tau) - \log \bar{p}(\tau) \right] \\
&= \mathbb{E}_{\tau \sim p(\tau)} \left[\log p(x_1) + \sum_{i=1}^{N-1} \{ \log p(x_{i+1}|x_i, u_i) + \log p(u_i|x_i) \} \right. \\
&\quad \left. - \left[\log \bar{p}(x_1) + \sum_{i=1}^{N-1} \{ \log \bar{p}(x_{i+1}|x_i, u_i) + \log \bar{p}(u_i|x_i) \} \right] \right] \\
&= \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=1}^{N-1} \log p(u_i|x_i) - \log \bar{p}(u_i|x_i) \right] \\
&= \sum_{i=1}^{N-1} \mathbb{E}_{(x_i, u_i) \sim p(x_i, u_i)} \left[\log p(u_i|x_i) - \log \bar{p}(u_i|x_i) \right]
\end{aligned} \tag{4.17}$$

where the inequality condition of the KL divergence between the two trajectories is converted to the KL divergence between the control policies. Hence, the Eq. (4.15) can be reformulated by Lagrangian method:

$$\begin{aligned}
L(\tau, \eta) &= \sum_{i=1}^{N-1} \mathbb{E}_{(x_i, u_i)} \left[l(x_i, u_i) \right] + l_f(x_N) + \eta (D_{KL}(p(u_i|x_i)||\bar{p}(u_i|x_i)) - \varepsilon) \\
&= \sum_{i=1}^{N-1} \mathbb{E}_{(x_i, u_i)} \left[l(x_i, u_i) - \eta \log \bar{p}(u_i|x_i) \right] + l_f(x_N) + \eta \log p(u_i|x_i) - \eta \varepsilon \tag{4.18} \\
&= \sum_{i=1}^{N-1} \mathbb{E}_{(x_i, u_i)} \left[\frac{1}{\eta} l(x_i, u_i) - \log \bar{p}(u_i|x_i) \right] + \frac{1}{\eta} l_f(x_N) + \log p(u_i|x_i) - \varepsilon
\end{aligned}$$

where η is a Lagrangian multiplier for the given KL divergence of the trajectories and ε is an inequality bounded condition. To solve the above the Lagrangian problem, define a surrogate running cost:

$$\begin{aligned}
\tilde{l}(x_i, u_i) &= \frac{1}{\eta} l(x_i, u_i) - \log \bar{p}(u_i|x_i) \\
\tilde{l}_f(x_N) &= \frac{1}{\eta} l_f(x_N)
\end{aligned} \tag{4.19}$$

then, the Lagrangian function, Eq. (4.18), can be rewritten as:

$$L(\tau, \eta) = \min \sum_{i=1}^{N-1} \mathbb{E}_{(x_i, u_i)} \left[\tilde{l}(x_i, u_i) + \log p(u_i|x_i) \right] + \tilde{l}_f(x_N) - \varepsilon \quad (4.20)$$

which is exactly equivalent to the Gaussian policy optimization problem Eq.(4.1). The multiple local policies $p(u_i|x_i)$ are updated for each trajectories using DDP or iLQR by solving the soft Bellman equation , and the global policy $\pi(u_i|x_i)$ is then computed using a general supervised learning such as deep neural network [71] and GMM fit [70] optimizing the following KL-Divergence:

$$\min_{\pi(u_i|x_i)} D_{KL}(\pi(u_i|x_i)||p(u_i|x_i)) \quad (4.21)$$

To simplify the whole process of the GPS, the Mirror Descent GPS, (MDGPS) is proposed [70] by replacing the previous policy $\bar{p}(u_i|x_i)$ with a global policy $\pi(u_i|x_i)$ in the problem formulation Eq.(4.15), which minimizes the expected cost under the trajectory of the policy distribution given by:

$$J = \sum_{i=1}^{N-1} l(x_i, u_i) + l_f(x_N), \quad \text{s.t. } D_{KL}(p(u_i|x_i)||\pi(u_i|x_i)) \leq \varepsilon \quad (4.22)$$

MDGPS decomposes the solving process Eq.(4.22) into the two steps: C-step and S-step. The C-step is a 'control phase' that computes local policies $p(u_i|x_i)$ and the S-step is a 'supervise phase' that optimizes global policy $\pi(u_i|x_i)$ by Eq.(4.21). This research utilizes the modified version of GPS, MDGPS, for the simplicity.

4.2.1 GPS: Local Policy Update

The local policy update, C-step, solves the defined Lagrangian in Eq.(4.20) using DDP algorithm with the soft state-value function Eq.(4.4) and the soft action-value function

Eq.(4.5) that are presented based on the surrogate cost defined in Eq.(4.19).

$$\begin{aligned}
 V(x_i) &= \mathbb{E}_{u_i \sim p(u_i|x_i)} \left[\log p(u_i|x_i) + Q(x_i, u_i) \right] \\
 Q(x_i, u_i) &= \tilde{l}(x_i, u_i) + \mathbb{E}_{x_{i+1} \sim p(x_{i+1}|x_i, u_i)} [V(x_{i+1})] \\
 \therefore V(x_i) &= \mathbb{E}_{u_i \sim p(u_i|x_i)} \left[\log p(u_i|x_i) + \tilde{l}(x_i, u_i) + \mathbb{E}_{x_{i+1} \sim p(x_{i+1}|x_i, u_i)} [V(x_{i+1})] \right]
 \end{aligned} \tag{4.23}$$

To find an optimal control correction, the second order Taylor series expansion near the nominal trajectory (x_i, u_i) can also be applied as described in Eq.(4.8):

$$\begin{aligned}
 V(\delta x_i) &= \mathbb{E}_{\delta \mu_i^u \sim p(\delta \mu_i^u | \delta x_i)} \left[-\frac{1}{2} \{ (\delta u_i - \delta \mu_i^u)^T S_i^{-1} (\delta u_i - \delta \mu_i^u) + \log(2\pi)^m + \log \det(S_i) \} \right. \\
 &\quad + \begin{bmatrix} \delta x_i^T & \delta u_i^T \end{bmatrix} \begin{bmatrix} \tilde{l}_x + f_x^T V'_x \\ \tilde{l}_u + f_u^T V'_x \end{bmatrix} \\
 &\quad \left. + \frac{1}{2} \begin{bmatrix} \delta x_i^T & \delta u_i^T \end{bmatrix} \begin{bmatrix} \tilde{l}_{xx} + f_x^T V'_{xx} f_x & \tilde{l}_{xu} + f_x^T V'_{xx} f_u \\ \tilde{l}_{ux} + f_u^T V'_{xx} f_x & \tilde{l}_{uu} + f_u^T V'_{xx} f_u \end{bmatrix} \begin{bmatrix} \delta x_i \\ \delta u_i \end{bmatrix} \right]
 \end{aligned} \tag{4.24}$$

The gradient and Hessian approximation of the surrogate cost expressed by Eq.(4.19) are rewritten with the mean of the global control policy, $\delta \mu_i^\pi = K_i^\pi \delta x_i + d_i^\pi$ [47]:

$$\text{gradient} : \frac{1}{\eta} \begin{bmatrix} \tilde{l}_{x,i} \\ \tilde{l}_{u,i} \end{bmatrix} + \begin{bmatrix} K_i^{\pi T} S_i^{\pi-1} d_i^\pi \\ -S_i^{\pi-1} d_i^\pi \end{bmatrix} \tag{4.25}$$

$$\text{Hessian} : \frac{1}{\eta} \begin{bmatrix} \tilde{l}_{xx,i} & \tilde{l}_{xu,i} \\ \tilde{l}_{ux,i} & \tilde{l}_{uu,i} \end{bmatrix} + \begin{bmatrix} K_i^{\pi T} S_i^{\pi-1} K_i^\pi & -K_i^{\pi T} S_i^{\pi-1} \\ -S_i^{\pi-1} d_i^\pi & S_i^{\pi-1} \end{bmatrix} \tag{4.26}$$

where the second term of each gradient and Hessian comes from the global Gaussian control policy, $\log \pi(u_i|x_i)$. The mean and covariance for the current Gaussian control

policy, $\log p(u_i|x_i)$, is obtained by taking partial derivatives by Eq.(4.9) and (4.10):

$$\begin{aligned}\delta\mu_i^u &= -Q_{uu}^{-1}[Q_{ux}\delta x_i + Q_u] \\ S_i &= Q_{uu}^{-1}\end{aligned}\tag{4.27}$$

where

$$\begin{aligned}Q_x &= \tilde{l}_x + f_x^T V'_x \\ Q_u &= \tilde{l}_u + f_u^T V'_x \\ Q_{xx} &= \tilde{l}_{xx} + f_x^T V'_{xx} f_x \\ Q_{xu} &= \tilde{l}_{xu} + f_x^T V'_{xx} f_u \\ Q_{uu} &= \tilde{l}_{uu} + f_u^T V'_{xx} f_u\end{aligned}\tag{4.28}$$

The dynamics information for the classic trajectory optimization is essential to find an optimal control policy by the approximate of the action-value function Eq.(2.13) and even for the sampling point-based (C)DDP to propagate the sampled point in the UDP (section 3.1) and the SGHQ (section 3.2) algorithms, however, most of system dynamics might not be fully known. Hence, model-based RL is utilized under the unknown environment (or system dynamics) with a model learning. The model-based method must learn a global model that is challenging and infeasible task within a short time while model-free method requires amount of data tuple of interaction time. Due to a complexity of a global model learning, linear regression method is utilized to fit a local dynamics with a Gaussian Mixture Model (GMM) as a prior data, which makes possible to fit a dynamics, $p(x_{i+1}|x_i, u_i) = N(f_{x,i}x_i + f_{u,i}u_i + f_{c,i}, \Sigma_i)$, even if the number of samples is less than the dimension of the system [72, 73, 74]. The details of the GMM are described in the appendix B.2 and B.3.

4.2.2 GPS: Global Policy Update

In practice, the global policy update, S-step, optimizes the following KL-Divergence which is formulated in the problem Eq.(4.22) [75, 74]:

$$\begin{aligned} \pi &\leftarrow \min_{\mu^\pi, \Sigma^\pi} D_{KL}(\pi(u_i|x_i)||p(u_i|x_i)) \\ &\leftarrow \min_{\mu^\pi, \Sigma^\pi} \sum tr(S_i^{-1}\Sigma_i^\pi) - \log|\Sigma_i^\pi| + (\mu_i^\pi - \mu_i^u)^T S_i^{-1}(\mu_i^\pi - \mu_i^u) \end{aligned} \quad (4.29)$$

where $\pi(u_i|x_i) = N(\mu_i^\pi, \Sigma_i^\pi)$ and $p(u_i|x_i) = N(\mu_i^u, S_i)$. For the simplicity, the global policy $\pi(u_i|x_i)$ is also assumed by $\delta\mu_i^\pi = K_i^\pi \delta x + d_i^\pi$ fitted by GMM [70], which is a form of a local policy $p(u_i|x_i)$.

4.2.3 GPS: Parameter Update for KL-Divergence

Lastly, the Lagrange multiplier for the KL-divergence (η), and the step size (ϵ) are then updated. The Lagrange multiplier for the inequality constraint can be updated by Dual Gradient Descent (DGD) method, which is a gradient descent-based approach: $\eta \leftarrow \eta + \alpha_\eta \frac{\partial L}{\partial \eta}$ [76, 48]. In [72], however, a dual variable η is recursively updated by maximizing η until the KL-divergence inequality constraint condition is satisfied by step size ϵ , which alleviates the KL-divergence formulated by equality constraint in the classic GPS:

Algorithm 11 GPS: η update

```

1: if  $D_{KL}(p(u_i|x_i)||\pi(u_i|x_i)) > \epsilon$  then
2:    $\eta_{max} \leftarrow \eta$ 
3:    $\eta \leftarrow \max(0.1\eta_{max}, \sqrt{\eta_{min}\eta_{max}})$ 
4: else
5:    $\eta_{min} \leftarrow \eta$ 
6:    $\eta \leftarrow \max(10\eta_{min}, \sqrt{\eta_{min}\eta_{max}})$ 
7: end if

```

The boundary condition of KL-divergence ϵ is updated by evaluating the difference between the predicted change in the cost of the local policy with the fitted local model and actual

cost sampling from the current policy [47]:

$$\varepsilon' = \varepsilon \frac{l_{k-1}^k - l_{k-1}^{k-1}}{2(l_{k-1}^k - l_k^k)} \quad (4.30)$$

where the subscription and superscription in the cost l means the GPS iteration of the dynamics and cost, respectively. The modified step adjustment in the MDGPS [70] is proposed to constrain the updated policy $p(u_i|x_i)$ from the C-step against the global policy $\pi(u_i|x_i)$:

$$\varepsilon' = \varepsilon \frac{l_{k-1}^k - l_{k-1}^{k-1,\pi}}{2(l_{k-1}^k - l_k^{k,\pi})} \quad (4.31)$$

4.3 Sparse Gauss-Hermite Quadrature Guided Policy Search

As SGHQ-rule substitutes the Jacobian and Hessian of the Q-function with a numerical mean and covariance obtained by sampling points, in the previous Chap. 3, the local policy update can also be computed by SGHQ-rule in the backward pass. The quadrature points are obtained by Eq.(3.48) with a specified accuracy level L_v for each trajectories and the gradient and Hessian of the Q-function are then computed by Eq.(3.52) and Eq.(3.49). Instead of using a linear local dynamics $\delta x_i = f_{dx}\delta x_i + f_{du}\delta u_i$ for a backward propagation of the quadrature points and the Hessian computation Eq.(3.49), a variant of local dynamics is utilized to propagate the sampling points backward:

$$\begin{aligned} x_{i+1} &= f_d(x_i, u_i) \\ &= f_d(\bar{x}_i, \bar{u}_i) + \frac{\partial f_d}{\partial x_i}(x_i - \bar{x}_i) + \frac{\partial f_d}{\partial u_i}(u_i - \bar{u}_i) \\ &= f_{dx}x_i + f_{du}u_i + \underbrace{f_d(\bar{x}_i, \bar{u}_i) - f_{dx}\bar{x}_i - f_{du}\bar{u}_i}_{=f_{dc}} \\ &= f_{dx}x_i + f_{du}u_i + f_{dc} \end{aligned} \quad (4.32)$$

Replacing the running cost $l(x_i, u_i)$ in the problem formulation by the surrogate cost defined in Eq.(4.19), the C-step of the GPS for local policy is computed. Inequality condition such as limits of state and control input $x_{min} \leq x \leq x_{max}$, $|u| \leq u_{max}$ can also

be involved in the surrogate cost as described in the CDDP algorithm by Augmented Lagrangian method:

$$J = \sum_{i=1}^{N-1} l(x_i, u_i) + l_f(x_N), \quad \text{s.t.} \quad \begin{cases} g(x_i, u_i) \leq 0 \\ D_{KL}(p(u_i|x_i) || \pi(u_i|x_i)) \leq \varepsilon \end{cases} \quad (4.33)$$

Surrogate cost including a KL-divergence can also be expanded by augmenting an inequality constraint $g(x_i, u_i)$ as:

$$\tilde{l}(x_i, u_i) = \frac{1}{\eta} \left\{ l(x_i, u_i) + \lambda_i^T g(x_i, u_i) + \frac{1}{2} g(x_i, u_i)^T I_{\mu_i} g(x_i, u_i) \right\} - \log \pi(u_i|x_i) \quad (4.34)$$

The entire C-step (local policy update) is described in Algorithm 12 and the process comparison GPS and SGHQ-GPS are shown in Fig. 4.1 as:

Algorithm 12 SGHQ Guided Policy Search: Backward Pass

```

1: for n=1:TrajMax do
2:   Compute surrogate cost:  $\tilde{l}(x_i, u_i)$  by Eq.4.34
3:   gradient: Eq.(4.25)
4:   Hessian: Eq.(4.26)
5:   Initialize the value function:  $V_{x_{N,n}} = \tilde{l}_{x_{N,n}}, V_{xx_{N,n}} = \tilde{l}_{xx_{N,n}}$ 
6:   for i=N-1:-1:1 do
7:      $\Gamma = chol \left( \begin{bmatrix} V'_{xx} & 0_{n \times m} \\ 0_{m \times n} & \tilde{l}_{uu} \end{bmatrix}_{i,n} \right)$ 
8:     Generate a set of quadrature points by Eq.(3.48)
9:     Backward Propagation of the sigma points using  $\tilde{x}_{i,n} = f_{dx}^{-1}[x_{i+1,n} - (f_{du}u_{i,n} + f_{dc})]$ 
10:    Update Q-function using the gradient and Hessian: Eq.(3.49) and (3.52)
11:    if  $Q_{uu}$  is invertible then
12:      Compute the control policy: Eq.(4.9) and (4.10)
13:      Update the value function: Eq.(4.13)
14:    else
15:      Increase  $\rho$  go to line 6
16:    end if
17:  end for
18: end for

```

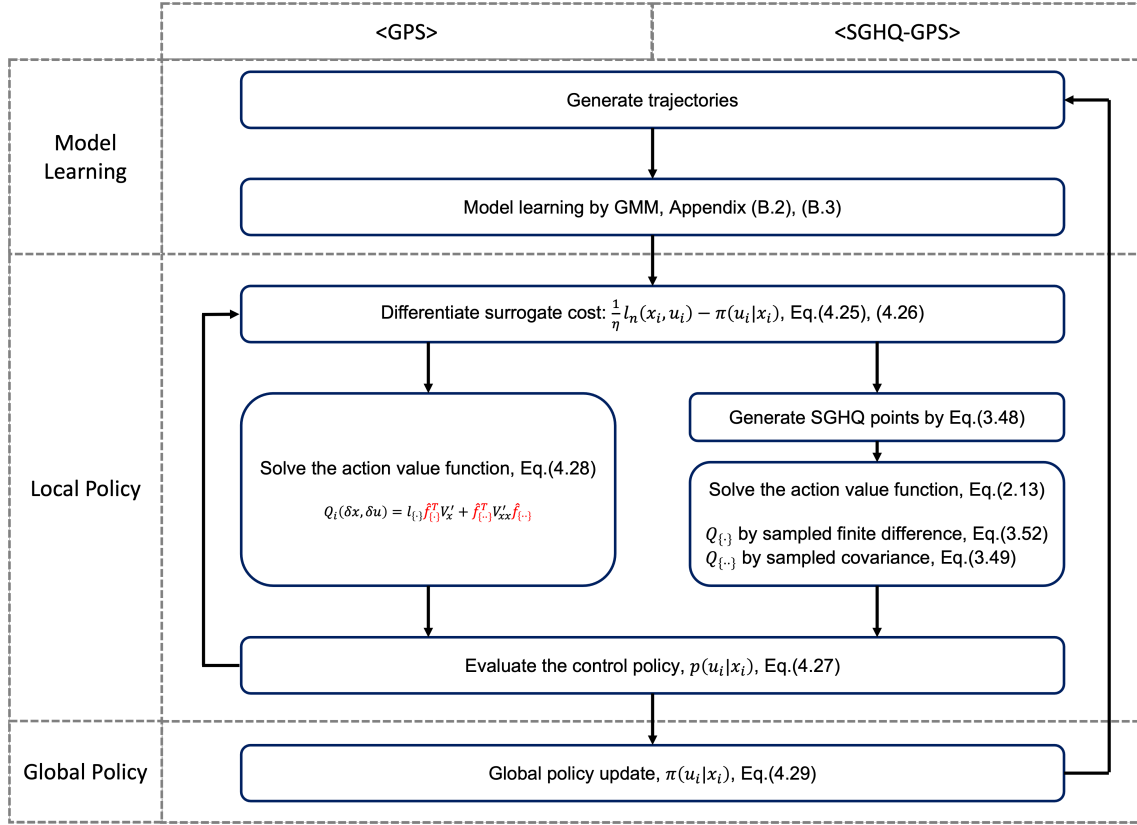


Figure 4.1: Process comparison of the GPS and SGHQ-GPS

4.4 Numerical Simulation

The numerical simulation is carried out with a F/A-18 to show the feasibility of the SGHQ-rule in the GPS algorithm. The aim of controller designing is to stabilize its attitude of which scenario is equivalent to the section 2.4, however, the dynamics model used in the backward pass of the C-step is a learned model by linear regression with GMM. The objective function of the GPS for fixed-wing aircraft stabilization is formulated as:

$$J = \min_U \mathbb{E}_{(x_i, u_i)} \frac{1}{2} (x_N - x_d)^T S_N (x_N - x_d) + \frac{1}{2} \sum_{i=1}^{N-1} \{ (x_i - x_d)^T Q (x_i - x_d) + (u_i - u_d)^T R (u_i - u_d) \}$$

$$\text{s.t.} \begin{cases} u_{min} \leq u_i \leq u_{max} \\ D_{KL}(p(u_i|x_i) || \pi(u_i|x_i)) \end{cases}$$
(4.35)

where the DDP parameter Q , R , S_N , and the desired value (x_d, u_d) are weighting function and the trim state and control input, respectively, which are defined in the Table 2.1 and 2.2 of the section 2.4. The control limit is also formulated in a $g(x_i, u_i) \leq 0$ form by Eq.(2.49). The gradient and Hessian of the surrogate cost Eq.(4.25) and (4.26) are:

$$\text{gradient} : \frac{1}{\eta} \begin{bmatrix} Q(x_i - x_{tr}) \\ R(u_i - u_{tr}) \end{bmatrix} + \begin{bmatrix} K_i^{\pi T} S_i^{\pi-1} d_i^{\pi} \\ -S_i^{\pi-1} d_i^{\pi} \end{bmatrix} \quad (4.36)$$

$$\text{Hessian} : \frac{1}{\eta} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} K_i^{\pi T} S_i^{\pi-1} K_i^{\pi} & -K_i^{\pi T} S_i^{\pi-1} \\ -S_i^{\pi-1} d_i^{\pi} & S_i^{\pi-1} \end{bmatrix} \quad (4.37)$$

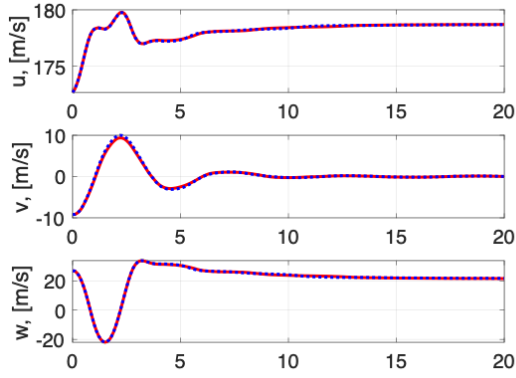
The initial GPS parameter is also listed in the Table 4.1, and the pseudo code for the SGHQ-GPS is described in Algorithm as follows 13.

Algorithm 13 SGHQ-Guided Policy Search

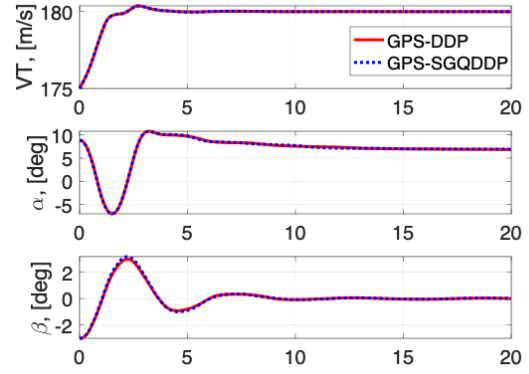
- 1: Initialize GPS and DDP parameters.
 - 2: **for** iter = 1:MaxIter **do**
 - 3: **for** n=1:TrajMax **do**
 - 4: Generate trajectories $\tau_{n=1:nMax} = [x_1, u_1, \dots, x_N]$ from an initial $x_1 \sim p(x_1)$.
 - 5: **end for**
 - 6: Model learning by linear regression with GMM by B.2 and B.3.
 - 7: **for** $i_n = 1 : i_{nMax}$ **do**
 - 8: C-step: find each local policies $p_n(u_i|x_i)$ by Algorithm 12.
 - 9: Forward pass: evaluate the updated local policies by Algorithm 2.
 - 10: η update by Algorithm 11.
 - 11: **end for**
 - 12: S-step: find a global policy $\pi(u_i|x_i)$ by subsection:4.2.2.
 - 13: Adjust KL-divergence by Eq.(4.31).
 - 14: Check break condition by DDP in Algorithm 3.
 - 15: **end for**
-

Table 4.1: Simulation parameter of the GPS for F/A-18: optimal regulator

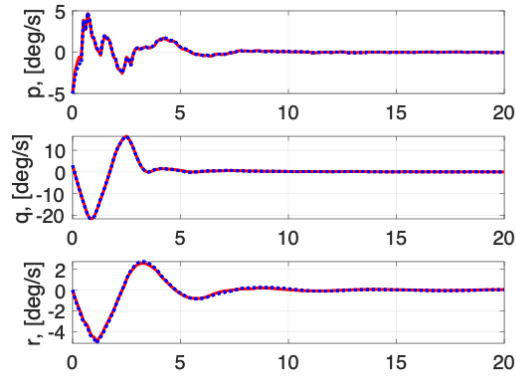
	Parameter	Value
GPS	Lagrange multiplier, η	1
	Scheduling factor, η_μ	1.6
	Initial step size of D_{KL} , ϵ	10^2
	Number of trajectories, TrajMax	$n_x + n_u + n_x + 10$



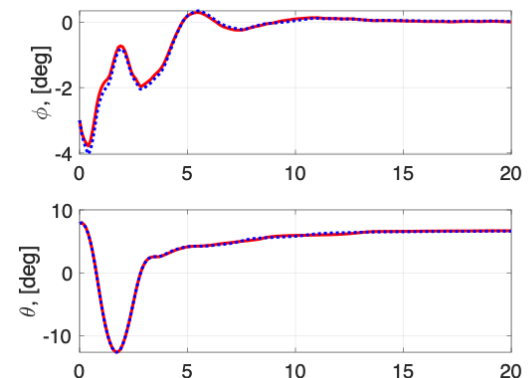
(a) Time history of the state: body velocity



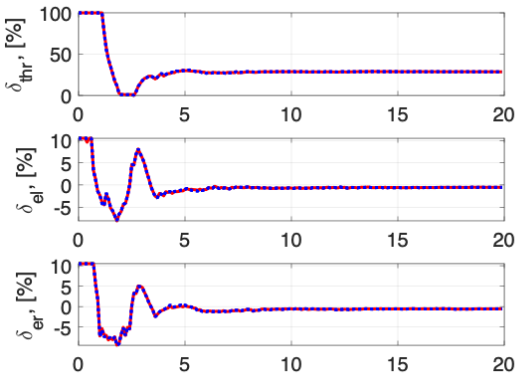
(b) Time history of the state: velocity, AoA, AoS



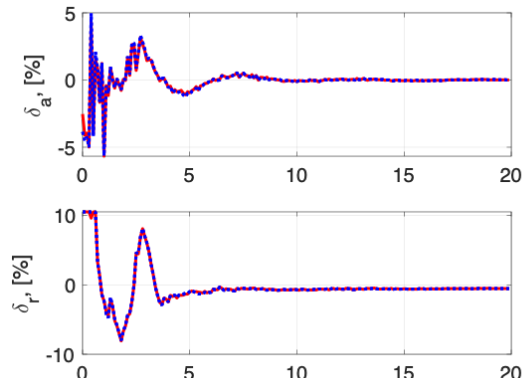
(c) Time history of the state: angular velocity



(d) Time history of the state: Euler angle



(e) Time history of the state: throttle, left/right elevator



(f) Time history of the input: aileron and rudder

Figure 4.2: Trajectory of the GPS for the state and control input of F/A-18

The Fig. 4.2 and 4.3 show the system response of the global policy resulted from the SGHQ-GPS algorithm. The result gives us that both state and control inputs are stabilized to the trim value completely within 10 GPS iteration even though the system model is

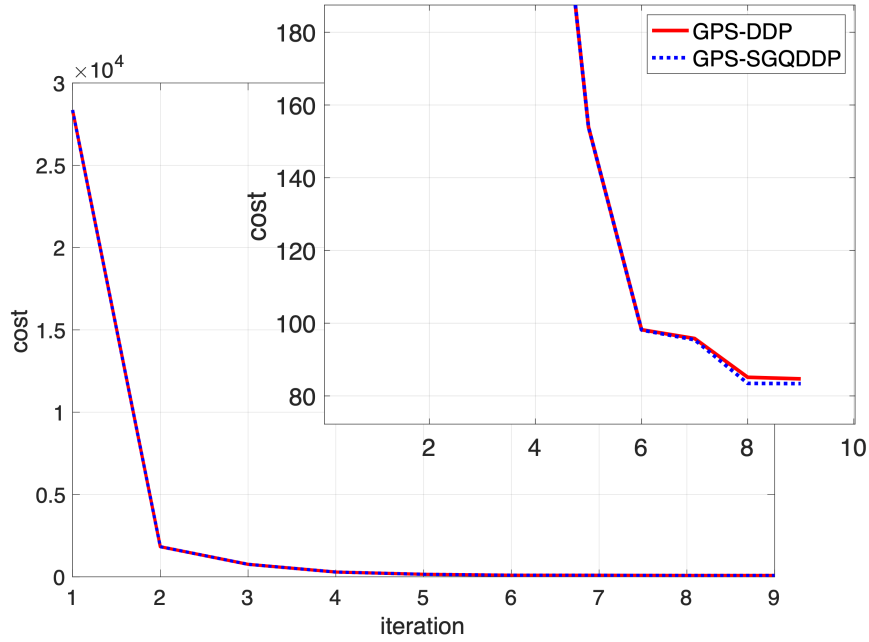


Figure 4.3: Cost comparison of the GPS

also learned by GMM during the trajectory generation phase. When we compare the result with the CDDP in Fig.2.3, the result of the GPS and SGHQ-GPS also show a similar tendency even the GPS algorithms does not use the true dynamics but utilize the learned GMM model. However, the GPS algorithms have an oscillating input responses and state trajectory as well due to the Gaussian distribution that is used to identify an unknown system information to be approximated: $u_i = \bar{u}_i + d_i + K_i(x_i - \bar{x}_i) + \Sigma_i^u$, $\Sigma_i^u \sim N(0, Q_{uu_i}^{-1})$, where an additional noise term is consistently added to the mean control input which still adds an additional cost term. Although the cost of the SGHQ-GPS has similar value compared with that of the deterministic trajectory optimization (DDP, Chap. 2), the SGHQ-GPS converges to near the optimal cost in 10 GPS iterations shown in Fig. 4.3 and Table 4.2. This result also presents the availability of the SGHQ-DDP in the GPS algorithm.

Table 4.2: Cost comparison: GPS and SGHQ-GPS

	GPS	SGHQ-GPS
Cost	84.6	83.38

4.5 Conclusion

This chapter revisits the GPS algorithm and its variant (MDGPS) to expand a deterministic trajectory optimization problem based on the DDP algorithm to a stochastic trajectory optimization problem using a sampling points-based method. In the previous chapter, a system model and its local approximate are directly used to solve an trajectory optimization problem, while the GPS learns a local model expressed by linear system and then utilizes the fitted model by linear regression model with a GMM to solve a soft Bellman equation in the local policy update step (C-step). The sampling point-based DDP algorithm, SGHQ-DDP described in the chapter 3, is applied to the GPS framework to find an optimal control policy in the C-step. It is verified from the numerical simulation that the sampling-based approach is feasible to solve a stochastic trajectory optimization problem to solve the soft-Bellman equation replacing by the approximates of the gradient and Hessian. It also means that an another sampling points-based method, i.e. UDP algorithm, can be implemented in the GPS framework. There is also a limitation that the learned model obtained by GMM is a linear which means the accuracy of the backward pass in the GPS and SGHQ-GPS would not different as both algorithms use the identical GMM model.

Chapter 5

Conclusions

5.1 Conclusion

This thesis proposed a sampling-based trajectory optimization algorithm based on the Differential Dynamic Programming (DDP) framework by replacing a numerical computation of the gradient and Hessian of the value function with a numerical integration method. The sparse-grid rule was integrated with the Gauss-Hermite Quadrature (GHQ), so-called SGHQ, to alleviate a computational load in the sampling point generation, i.e. curse of dimensionality issue and utilizes the numerical integration using the concept of the Unscented Transform (UT) without a partial derivative of dynamics directly. This can avoid an accuracy issue of approximation in the complex dynamics such as highly nonlinear case, singularity and unstable point from derivatives.

Especially, this research focused on to design a control system for a flight control law stabilizing its attitude and guidance law of a fixed-wing aircraft and missile via numerical simulations from a linear to highly nonlinear stabilizing control system, respectively. Constraint condition and final time free problems were also dealt with by Augmented Lagrangian method (Constrained DDP, CDDP) and adjustment of the incremental change in final time (Flexible Final Time, FFT). The SGHQ-based CDDP with a flexible final time shown the best performance from the result of the simulation, particularly nonlin-

ear case because the sampling points-based method does not compute derivative of the dynamics numerically, which avoids an obtaining approximates as expected.

The stochastic trajectory optimization was also handled by Guided Policy Search (GPS) where a SGHQ rule is applied in the local policy search process by DDP. Each trajectories were optimized by computing a optimal control satisfying the trajectory distance between the current and previous defined by Kullback-Liebler Divergence (KL-Divergence). The GPS with SGHQ-DDP under unknown environment was also carried out with a fitted dynamics model and it validate the availability of SGHQ rule in the GPS.

In practice, both deterministic and stochastic trajectory optimization are generally carried out in a *off-line* manner with the assumption that the dynamics information is fully given. Indeed, the 1st necessary condition of the optimality should be computed, which would be affected by sensor frequency in practical environment, while the SGHQ-DDP utilizes the sampled mean and covariance using the propagated sampling points via the nonlinear transform through dynamics, directly. In other words, the sampling rate issue by discretization time step determined that is related to the validity of the perturbation assumption in the discretization could be avoided. As the feasibility, effectiveness and potential benefits of the SGHQ rule were evaluated from the numerical simulations, the SGHQ-based sampling point trajectory solvers allows to overcome the limitation of the numerical computation.

5.2 Future Works

The feasibility of the SGHQ-GPS is verified in the Chap.4 to optimize a control policy modeled by linear-Gaussian policy: $\pi(u_i|x_i) \sim N(\bar{u}_i + d_i + \bar{K}_i(x_i - \bar{x}_i), \Sigma_i^u)$ conditioned on state x_i . The local control policy is obtained by DDP that optimizes each trajectories by stochastic version of the Bellman equation (or soft Bellman equation) by computing a mean and covariance of the locally linear and optimal Gaussian control policy even under the unknown environment with a fitted system model with a GMM. However, the

proposed method by Chap. 4 has a limitation that both of GPS and sampling point-based GPS utilize the trained model obtained by GMM which is a linearized model. As we compared the result from the Fig. 4.3 and Table 4.2, the SGHQ-GPS does not improve the performance of the stochastic trajectory optimization problem significantly.

In the future work, it may be valuable to extend the use of the SGHQ-GPS to highly nonlinear cases to show the appropriacy of the sampling point-based approach using a nonlinear trained model. This means that the trained nonlinear model can be utilized in the point propagation using the nonlinear transform concept and the better performance and accuracy can be expected in the nonlinear case without the dynamics Jacobian. Furthermore, the final time free method (flexible final time) based approach can also be solved to adjust the given time horizon in the SGHQ-GPS framework that enables to truncate the redundant time horizon or extend a given short time to solve the stochastic trajectory optimization problem.

Appendix A

Dynamics

A.1 Fixed-wing Aircraft Dynamics

The considered fixed-wing aircraft model is F/A-18 HARV (High Angle of attack Research Vehicle, HARV) that is a research vehicle introduced for the high Angle of Attack (AOA) region over the post-stall and spin maneuverability of the high AOA flight envelope [2]. The aerodynamics coefficient and its derivatives are provided by a parameterized table by AOA. The motion of the aircraft is expressed by Newton's law [59].

Force equations:

$$\dot{v}^b = \frac{F_{tot}}{m} - \omega^b \times v^b \quad (\text{A.1})$$

Moment equations:

$$\dot{\omega}^b = J_b^{-1} (M_{tot} - \omega^b \times J_b \omega^b) \quad (\text{A.2})$$

where an inertia matrix J_b is:

$$J_b = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{zx} & 0 & I_{zz} \end{bmatrix} \quad (\text{A.3})$$

Kinematic equations:

$$\begin{aligned}\dot{\phi} &= p + \tan \theta (q \sin \phi + r \cos \phi) \\ \dot{\theta} &= q \cos \phi - r \sin \phi\end{aligned}\tag{A.4}$$

where $v^b = [u, v, w]^T$ and $\omega^b = [p, q, r]^T$ are a velocity and an angular velocity in a body frame, respectively. $F_{tot} = [F_x, F_y, F_z]^T$ and $M_{tot} = [M_x, M_y, M_z]^T = [l, m, n]^T$ denote total external force and moment vector, respectively. The force equation A.1 can also be rewritten in an aircraft velocity $V_T = \|v^b\|_2$, AOA ($\alpha = \tan^{-1}(w/u)$) and Angle of Sideslip (AOS, $\beta = \sin^{-1}(v/V_T)$) terms as follows:

$$\begin{aligned}\dot{V}_T &= \frac{u\dot{u} + v\dot{v} + w\dot{w}}{V_T} \\ \dot{\alpha} &= \frac{u\dot{w} - w\dot{u}}{u^2 + w^2} \\ \dot{\beta} &= \frac{V_T\dot{v} - v\dot{V}_T}{V_T^2 \cos \beta}\end{aligned}\tag{A.5}$$

where $\frac{d}{dt}[u, v, w]^T$ is given from the Eq.(A.1). The aerodynamic coefficient for the force equation A.1 and moment equation A.2 can be expressed as follows:

Aerodynamic force coefficient:

$$\begin{aligned}C_L &= C_{L_0} + C_{L_q} \left(\frac{bq}{2V_T} \right) + C_{L_{\delta_{el}}} \delta_{el} + C_{L_{\delta_{er}}} \delta_{er} \\ C_D &= C_{D_0} + C_{D_q} \left(\frac{bq}{2V_T} \right) + C_{D_{\delta_{el}}} \delta_{el} + C_{D_{\delta_{er}}} \delta_{er} \\ C_Y &= C_{Y_\beta} \beta + C_{Y_p} \left(\frac{bp}{2V_T} \right) + C_{Y_r} \left(\frac{br}{2V_T} \right) + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_{el}}} \delta_{el} + C_{Y_{\delta_{er}}} \delta_{er} + C_{Y_{\delta_r}} \delta_r\end{aligned}\tag{A.6}$$

Aerodynamic moment coefficient:

$$\begin{aligned}C_l &= C_{l_\beta} \beta + C_{l_p} \left(\frac{bp}{2V_T} \right) + C_{l_r} \left(\frac{br}{2V_T} \right) + C_{l_{\delta_a}} \delta_a + C_{l_{\delta_{el}}} \delta_{el} + C_{l_{\delta_{er}}} \delta_{er} + C_{l_{\delta_r}} \delta_r \\ C_m &= C_{m_0} + C_{m_q} \left(\frac{cq}{2V_T} \right) + C_{m_{\delta_{el}}} \delta_{el} + C_{m_{\delta_{er}}} \delta_{er} \\ C_n &= C_{n_\beta} \beta + C_{n_p} \left(\frac{bp}{2V_T} \right) + C_{n_r} \left(\frac{br}{2V_T} \right) + C_{n_{\delta_a}} \delta_a + C_{n_{\delta_{el}}} \delta_{el} + C_{n_{\delta_{er}}} \delta_{er} + C_{n_{\delta_r}} \delta_r\end{aligned}\tag{A.7}$$

Control input variable δ_{el} , δ_{er} , δ_a , δ_r are left/right elevator, aileron and rudder, respectively, and their limites are also given by:

$$\begin{aligned} -10.5^\circ &\leq \delta_{el,er} \leq 24^\circ \\ -35^\circ &\leq \delta_a \leq 35^\circ \\ -30^\circ &\leq \delta_r \leq 30^\circ \end{aligned} \tag{A.8}$$

And the engine thrust is modelled by:

$$F_{thr} = T_{max} \delta_{thr} \tag{A.9}$$

where $0 \leq \delta_{thr} \leq 1$ is a throttle of the engine.

A.2 Missile dynamics

A aerodynamic drag is consisted of the zero-lift drag and induced drag as follows:

$$D = D_0 + D_i \tag{A.10}$$

where

$$\begin{aligned} D_0 &= QS_m C_{d_0} \\ D_i &= \frac{C_{d_i} m^2 (a_y^2 + a_z^2)}{QS_m} \\ Q &= \frac{1}{2} \rho_{air} V_m^2 \end{aligned} \tag{A.11}$$

Table A.1: Atmospheric constants

Parameter	Value
Ratio of Specific heats, γ	1.403
Gas constant, R	287.26 [J/kg K]
Lapse rate, L_R	0.0065 [K/m]
Temperature at sea level, T_0	288.16 [K]
Height of Troposphere, h_{tr}	11,000 [m]
Gravitational acceleration, g	9.806 [m/s^2]
Reference area, S_m	1 [m^2]

C_{d_0} is the zero-lift drag coefficient, C_{d_i} is the induced drag coefficient:

$$C_{d_0} = \begin{cases} 0.02 & M < 0.93 \\ 0.02 + 0.2(M - 0.93) & M < 1.03 \\ 0.04 + 0.06(M - 1.03) & M < 1.10 \\ 0.0442 - 0.007(M - 1.10) & M \geq 1.10 \end{cases} \quad (A.12)$$

$$C_{d_i} = \begin{cases} 0.2 & M < 1.15 \\ 0.2 + 0.246(M - 1.15) & M \geq 1.15 \end{cases} \quad (A.13)$$

Mach number, M , is obtained by:

$$M = \frac{V_m}{V_{ss}} = \frac{V_m}{\sqrt{\gamma RT}}, \quad (A.14)$$

where atmospheric constants, (γ, R) are listed in the table A.2 and the temperature is computed by following relation:

$$T = T_0 - L_R \max(0, \min(h_{tr}, h)) \quad (A.15)$$

The air density is given by a function of altitude, h :

$$\rho_{air}(h) = 1.15579 - 1.058 \times 10^{-4}h + 3.725 \times 10^{-9}h^2 - 6.0 \times 10^{-14}h^3, \quad h \in [0, 20,000m] \quad (\text{A.16})$$

Lastly, the thrust and the mass of the missile vehicle are given:

$$F_T = \begin{cases} 33,000 & 0 \leq t \leq 1.5 \\ 7,500 & 1.5 \leq t \leq 8.5 \\ 0 & 8.5 \leq t \end{cases} \quad (\text{A.17})$$

$$m = \begin{cases} 135 - 14.53t & 0 \leq t \leq 1.5 \\ 113.205 - 3.31t & 1.5 \leq t \leq 8.5 \\ 90.035 & 8.5 \leq t \end{cases} \quad (\text{A.18})$$

Appendix B

Guided Policy Search

B.1 Gaussian Function

The Gaussian distribution, (or known as the normal distribution) is one of widely used probability distributions for continuous variable [77]. The general form of the univariate Gaussian distribution is:

$$N(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \quad (\text{B.1})$$

where μ is the mean and σ^2 is the variance. Multivariate Gaussian distribution for a multi-dimensional variable also takes the form:

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \quad (\text{B.2})$$

where $\mu \in R^m$ and $\Sigma \in R^{m \times m}$ are mean and covariance of x , respectively and $|\cdot|$ is a determinant of covariance matrix.

B.2 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a parameterized probabilistic density model represented by a weighted sum of Gaussian distributions, which is commonly used as a parametric model of the probability distribution of features [78]. GMM is estimated from training data by the Expectation-Maximization (EM) algorithm of Maximum a Posteriori (MAP) estimation from a well-trained prior model [79].

Consider a GMM given by a weighted sum of K components (or clusters) of Gaussian distribution as:

$$p(z) = \sum_{i=1}^K w_i N_i(z|\mu_i, P_i) \quad (\text{B.3})$$

where μ_i and P_i are the mean and covariance of the corresponding Gaussian distribution and K is the number of the Gaussian clusters and $N(x|\mu, \Sigma)$ satisfies the Eq. (B.2). A weighting parameter satisfies the following relation:

$$\sum_{i=1}^K w_i = 1, \quad 0 \leq w_i \leq 1 \quad (\text{B.4})$$

Parameters for GMM can be updated by solving the following constrained optimization problem that maximizes the likelihood of the GMM given the training data with a assumption that the set of training data $z = [z_1, z_2, \dots, z_T]$ has independence between the data, which is Maximum Likelihood (ML) estimation [80]:

$$\max_{\mu_i, P_i, w_i} \prod_{l=1}^T \log p(z_l), \quad s.t. \sum_{i=1}^K w_i = 1 \quad (\text{B.5})$$

The constrained optimization problem Eq.(B.5) is converted into unconstrained optimization by applying Lagrangian method as:

$$\max_{\mu_i, P_i, w_i, \lambda} L = \sum_{l=1}^T \log \left\{ \sum_{i=1}^K w_i N_i(z_l|\mu_i, P_i) \right\} + \lambda \left(1 - \sum_{i=1}^K w_i \right) \quad (\text{B.6})$$

Taking a partial derivative with respect to the GMM parameters, μ_i, P_i and w_i

Mean, μ :

$$\frac{\partial L}{\partial \mu_i} = \sum_{l=1}^T \left\{ \frac{w_i N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \right\} P_i^{-1} (z_l - \mu_i) = 0 \quad (\text{B.7})$$

$$\therefore \mu_i = \frac{\sum_{l=1}^T \gamma_i^l z_l}{\sum_{l=1}^T \gamma_i^l}$$

where

$$\gamma_i^l = \frac{w_i N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \quad (\text{B.8})$$

Covariance, P_i :

$$\frac{\partial L}{\partial P_i} = \sum_{l=1}^T \left\{ \frac{w_i N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \right\} \left(\frac{1}{2} P_i^{-1} (z_l - \mu_i) (z_l - \mu_i)^T P_i^{-1} - \frac{1}{2} P_i^{-1} \right) = 0 \quad (\text{B.9})$$

$$P_i = \frac{\sum_{l=1}^T \gamma_i^l (z_l - \mu_i) (z_l - \mu_i)^T}{\sum_{l=1}^T \gamma_i^l}$$

Weight, w_i :

$$\begin{aligned} \frac{\partial L}{\partial w_i} &= \sum_{l=1}^T \left\{ \frac{N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \right\} - \lambda = 0 \\ &= \sum_{i=1}^K w_i \sum_{l=1}^T \left\{ \frac{N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \right\} - \lambda \sum_{i=1}^K w_i \\ &= \sum_{i=1}^K \sum_{l=1}^T \gamma_i^l - \lambda \\ \therefore \lambda &= T \quad \left(\because \sum_{i=1}^K \gamma_i^l = 1 \right) \end{aligned} \quad (\text{B.10})$$

Multiplying w_i on the both sides gives:

$$\begin{aligned}\frac{\partial L}{\partial w_i} &= \sum_{l=1}^T \left\{ \frac{N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \right\} - \lambda \\ &= \sum_{l=1}^T \left\{ w_i \frac{N_i(z_l | \mu_i, P_i)}{\sum_{i=1}^K w_i N_i(z_l | \mu_i, P_i)} \right\} - w_i T \\ \therefore w_i &= \frac{1}{T} \sum_{l=1}^T \gamma_i^l\end{aligned}\tag{B.11}$$

Thus, the mean and covariance of the GMM represented by each cluster mean μ_i and covariance P_i with weight w_i are computed by:

$$\begin{aligned}\mu^{GMM} &= \sum_{i=1}^K w_i \mu_i \\ P^{GMM} &= \sum_{i=1}^K w_i \{ P_i + (\mu_i - \mu)(\mu_i - \mu)^T \}\end{aligned}\tag{B.12}$$

Lastly, the GMM data is considered as a prior information to provide the normal-inverse-Wishart prior [81] for the mean and covariance of the local linear Gaussian model [74]. The posteriori estimate of mean and covariance are:

$$\mu_z^{MAP} = \frac{m\mu_0 + n_0\mu_z}{m + n_0}\tag{B.13}$$

$$P_{zz}^{MAP} = \frac{\Phi + NP_{zz} + \frac{Nm}{N+m}(\mu_z - \mu_0)(\mu_z - \mu_0)^T}{N + n_0}\tag{B.14}$$

where μ_z and P_{zz} are empirical mean and covariance of the training data and μ_0 and Φ are set by the mean and covariance of the GMM, μ^{GMM} , Φ^{GMM} and m , n_0 are 1, respectively.

B.3 Linear Regression and Conditional Gaussian Function

Local model is fitted by linear regression [82] using a conditional Gaussian distribution [77]. Consider a multivariate Gaussian distribution that the conditional distribution of one set is Gaussian if two sets of variables are jointly Gaussian [77, 83]. Suppose z is a n_z -dimensional vector with Gaussian distribution $N(z|\mu_z, P_z)$:

$$z = \begin{bmatrix} a \\ b \end{bmatrix} \quad (\text{B.15})$$

where $a \in R^{n_a}$ and $b \in R^{n_b}$. The mean and covariance of the vector z are given by:

$$\mu_z = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix} \quad (\text{B.16})$$

$$\Sigma_z = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \quad (\text{B.17})$$

From the product rule of probability, conditional distribution can be inferred from the joint distribution $p(z) = p(a, b)$ [77]:

$$\begin{aligned} p(a, b) &= p(a) \alpha \\ \int p(a, b) da &= \alpha \int p(a) da \quad \int p(a) da = 1 \\ \int p(a) p(b) da &= \alpha \\ \therefore p(b) &= \alpha \end{aligned} \quad (\text{B.18})$$

Therefore,

$$\begin{aligned}
 p(a, b) &= p(a)p(b) \\
 p(a) &= \frac{p(a, b)}{p(b)} \\
 &= p(a|b)
 \end{aligned} \tag{B.19}$$

which means that probability distribution of a is equivalent to the probability distribution of $p(a|b)$.

Define a precision matrix $\Lambda = \Sigma^{-1}$ for the convenience:

$$\Lambda_z = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix} \tag{B.20}$$

and consider an exponential term of the Gaussian probability function $p(z)$:

$$\begin{aligned}
 & -\frac{1}{2} \begin{bmatrix} (a - \mu_a)^T & (b - \mu_b)^T \end{bmatrix} \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}^{-1} \begin{bmatrix} a - \mu_a \\ b - \mu_b \end{bmatrix} \\
 &= -\frac{1}{2} \left\{ (a - \mu_a)^T \Sigma_{aa}^{-1} (a - \mu_a) + (a - \mu_a)^T \Sigma_{ab}^{-1} (b - \mu_b) \right. \\
 & \quad \left. + (b - \mu_b)^T \Sigma_{ba}^{-1} (a - \mu_a) + (b - \mu_b)^T \Sigma_{bb}^{-1} (b - \mu_b) \right\}
 \end{aligned} \tag{B.21}$$

and conditional probability $p(a|b)$:

$$\begin{aligned}
 & -\frac{1}{2} (a - \mu_{a|b})^T \Sigma_{a|b}^{-1} (a - \mu_{a|b}) \\
 &= -\frac{1}{2} \left\{ a^T \Sigma_{a|b}^{-1} a - a^T \Sigma_{a|b}^{-1} \mu_{a|b} - \mu_{a|b}^T \Sigma_{a|b}^{-1} a + \mu_{a|b}^T \Sigma_{a|b}^{-1} \mu_{a|b} \right\}
 \end{aligned} \tag{B.22}$$

From the 'completing the square' in terms of ' a ', the following mean and covariance relations are satisfied

Covariance:

$$\Sigma_{a|b}^{-1} = \Lambda_{aa} \tag{B.23}$$

Mean:

$$\begin{aligned}
\Sigma_{a|b}^{-1} \mu_{a|b} &= \{\Lambda_{aa} \mu_a - \Lambda_{ab}(b - \mu_b)\} \\
\mu_{a|b} &= \Sigma_{a|b} \{\Lambda_{aa} \mu_a - \Lambda_{ab}(b - \mu_b)\} \\
&= \mu_a - \Lambda_{aa}^{-1} \Lambda_{ab}(b - \mu_b) \\
&= \mu_a - \Lambda_{aa}^{-1} \left\{ -\Sigma_{aa}^{-1} \Sigma_{ab} (\Sigma_{bb} - \Sigma_{ba} \Sigma_a^{-1} \Sigma_{ab})^{-1} \right\} (b - \mu_b) \\
\therefore \mu_{a|b} &= \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1} (b - \mu_b)
\end{aligned} \tag{B.24}$$

where Schur complement is used to compute $(\Sigma^{-1} = \Lambda)$:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \tag{B.25}$$

Replacing a and b with x_{k+1} and $[x_k, u_k]$, respectively, shows:

$$\begin{aligned}
\mu_{x_{k+1}|x_k, u_k} &= \mu_{x_{k+1}} + \Sigma_{x_{k+1}x_k u_k} \Sigma_{x_k u_k x_k u_k}^{-1} (x_k u_k - \mu_{x_k u_k}) \\
\Sigma_{x_{k+1}|x_k u_k} &= \Sigma_{x_{k+1}x_{k+1}} - \Sigma_{x_{k+1}x_k u_k} \Sigma_{x_k u_k x_k u_k}^{-1} \Sigma_{x_k u_k x_{k+1}}
\end{aligned} \tag{B.26}$$

The local model is an approximate of the global model $p(x_{i+1}|x_k, u_k) = N(f(x_k, u_k), \Sigma_k)$ near the nominal trajectory:

$$p(x_{k+1}|x_k, u_k) = N(f_{xu_k}[x_k \ u_k]^T + f_{c_k}, \Sigma_k) \tag{B.27}$$

where $x \in R^{n_x \times 1}$, $u \in R^{n_u \times 1}$, $f_{xu} \in R^{n_x \times (n_x + n_u)}$, $f_c \in R^{n_x \times 1}$ and $\Sigma \in R^{n_x \times n_x}$. The state-action marginals $p(x_k, u_k)$ can be computed by forward propagating Gaussian distribution in time:

$$\begin{aligned}
\mu_{x_{k+1}} &= f_{xu_k}[\mu_{x_k} \ \mu_{u_k}]^T + f_{c_k} \\
\Sigma_{x_{k+1}} &= f_{xu_k}^T \Sigma_{x_k u_k} f_{xu_k} + \Sigma_k
\end{aligned} \tag{B.28}$$

Incorporating Eq.(B.28) into Eq.(B.26):

$$\begin{aligned}
 \mu_{x_{k+1}|x_k, u_k} &= f_{xu_k}[\mu_{x_k} \ \mu_{u_k}]^T + f_{c_k} + \Sigma_{x_{k+1}x_k u_k} \Sigma_{x_k u_k x_k u_k}^{-1} (x_k u_k - \mu_{x_k u_k}) \\
 &= f_{xu_k}[x_k \ u_k] + f_{c_k} \\
 \therefore f_{xu_k} &= \Sigma_{x_{k+1}x_k u_k} \Sigma_{x_k u_k x_k u_k}^{-1}
 \end{aligned} \tag{B.29}$$

From the Eq.(B.28):

$$\begin{aligned}
 f_{c_k} &= \mu_{x_{k+1}} - f_{xu_k}[\mu_{x_k} \ \mu_{u_k}]^T \\
 \Sigma_k &= \Sigma_{x_{k+1}} - f_{xu_k}^T \Sigma_{x_k u_k} f_{xu_k}
 \end{aligned} \tag{B.30}$$

Bibliography

- [1] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.
- [2] K. W. Iliff, *Flight-Determined, Subsonic, Lateral-Directional Stability and Control Derivatives of the Thrust-Vectoring F-18 High Angle of Attack Research Vehicle (HARV), and Comparisons to the Basic F-18 and Predicted Derivatives*. NASA, 1999.
- [3] O. Von Stryk and R. Bulirsch, “Direct and indirect methods for trajectory optimization,” *Annals of operations research*, vol. 37, no. 1, pp. 357–373, 1992.
- [4] A. V. Rao, “A survey of numerical methods for optimal control,” *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.
- [5] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.
- [6] D. S. Naidu, *Optimal control systems*. CRC press, 2002.
- [7] A. Malik, T. Henderson, and R. J. Prazenica, “Trajectory generation for a multibody robotic system using the product of exponentials formulation,” in *AIAA Scitech 2021 Forum*, p. 2016, 2021.
- [8] 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*, pp. 4906–4913, IEEE, 2012.

- [9] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Koley, and E. Todorov, “An integrated system for real-time model predictive control of humanoid robots,” in *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*, pp. 292–299, IEEE, 2013.
- [10] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175, IEEE, 2014.
- [11] P. Kee, L. Dong, and C. Siong, “Near optimal midcourse guidance law for flight vehicle,” in *36th AIAA Aerospace Sciences Meeting and Exhibit*, p. 583, 1998.
- [12] S. He, H.-S. Shin, and A. Tsourdos, “Computational guidance using sparse gauss-hermite quadrature differential dynamic programming,” *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 13–18, 2019.
- [13] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.
- [14] Z. Manchester and S. Kuindersma, “Derivative-free trajectory optimization with unscented dynamic programming,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 3642–3647, IEEE, 2016.
- [15] B. K. Plancher, “Parallel and constrained differential dynamic programming for fast robotic motion planning,” Master’s thesis, Harvard University Cambridge, Massachusetts, 2018.
- [16] P. Gath, K. Well, and K. Mehlem, “Automatic initial guess generation for ariane 5 dual payload ascent trajectory optimization,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 4589, 2000.

- [17] X. Wei, L. Liu, Y. Wang, and Y. Yang, “Reentry trajectory optimization for a hypersonic vehicle based on an improved adaptive fireworks algorithm,” *International Journal of Aerospace Engineering*, vol. 2018, 2018.
- [18] R. Chai, A. Savvaris, A. Tsourdos, Y. Xia, and S. Chai, “Solving multiobjective constrained trajectory optimization problem by an extended evolutionary algorithm,” *IEEE transactions on cybernetics*, vol. 50, no. 4, pp. 1630–1643, 2018.
- [19] E. Almasri and M. K. Uyguroğlu, “Trajectory optimization in robotic applications, survey of recent developments,” 2021.
- [20] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, “A review of optimization techniques in spacecraft flight trajectory design,” *Progress in aerospace sciences*, vol. 109, p. 100543, 2019.
- [21] H. Waschl, I. Kolmanovsky, M. Steinbuch, and L. Del Re, *Optimization and optimal control in automotive systems*, vol. 455. Springer, 2014.
- [22] C. R. Hargraves and S. W. Paris, “Direct trajectory optimization using nonlinear programming and collocation,” *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.
- [23] M. Kelly, “An introduction to trajectory optimization: How to do your own direct collocation,” *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.
- [24] B. Jackson, “Exploring quadrature rules in direct collocation methods for trajectory optimization,” tech. rep., Tech. Rep, 2018.
- [25] M. Heinkenschloss and D. Ridzal, “A matrix-free trust-region sqp method for equality constrained optimization,” *SIAM Journal on Optimization*, vol. 24, no. 3, pp. 1507–1541, 2014.
- [26] P. E. Gill, W. Murray, and M. A. Saunders, “Snopt: An sqp algorithm for large-scale constrained optimization,” *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

- [27] L. T. Biegler and V. M. Zavala, “Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization,” *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.
- [28] I. M. Ross and M. Karpenko, “A review of pseudospectral optimal control: From theory to flight,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.
- [29] D. Mayne, “A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems,” *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.
- [30] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. No. 24, Elsevier Publishing Company, 1970.
- [31] L.-Z. Liao and C. A. Shoemaker, “Convergence in unconstrained discrete-time differential dynamic programming,” *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.
- [32] W. Li and E. Todorov, “Iterative linear quadratic regulator design for nonlinear biological movement systems,” in *ICINCO (1)*, pp. 222–229, Citeseer, 2004.
- [33] 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.*, pp. 300–306, IEEE, 2005.
- [34] J. Nocedal and S. Wright, “Numerical optimization. 2nd edn springer,” *New York*, 2006.
- [35] D. Q. Mayne, M. M. Seron, and S. Raković, “Robust model predictive control of constrained linear systems with bounded disturbances,” *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.
- [36] D. G. Hull, *Optimal control theory for applications*. Springer Science & Business Media, 2013.

- [37] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [38] R. Bellman, “Dynamic programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [39] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.
- [40] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, “Adaptive optimal control for continuous-time linear systems based on policy iteration,” *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.
- [41] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers,” *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012.
- [42] O. Park, H. Shin, and A. Tsourdos, “Linear quadratic tracker with integrator using integral reinforcement learning,” in *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pp. 31–36, IEEE, 2019.
- [43] O. Park, H.-S. Shin, H.-I. Lee, and T. Antonios, “Optimal and adaptive control design using recursive least square with a new exponential forgetting factor,” in *International Conference on Robot Intelligence Technology and Applications*, pp. 116–128, Springer, 2022.
- [44] S. M. Khansari-Zadeh and A. Billard, “Bm: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models,” in *2010 IEEE international conference on robotics and automation*, pp. 2381–2388, IEEE, 2010.
- [45] S. Levine, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search (2015),” *arXiv preprint arXiv:1501.05611*, 2015.
- [46] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, “Combining model-based and model-free updates for trajectory-centric reinforcement

- learning,” in *International conference on machine learning*, pp. 703–711, PMLR, 2017.
- [47] S. Levine and V. Koltun, “Guided policy search,” in *International conference on machine learning*, pp. 1–9, PMLR, 2013.
- [48] S. Levine, *Motor skill learning with local trajectory methods*. Stanford University, 2014.
- [49] J. Du, J. Fu, and C. Li, “Guided policy search methods: A review,” in *Journal of Physics: Conference Series*, vol. 1748, p. 022039, IOP Publishing, 2021.
- [50] S. Julier and J. K. Uhlmann, “A general method for approximating nonlinear transformations of probability distributions,” 1996.
- [51] E. A. Wan and R. Van Der Merwe, “The unscented kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pp. 153–158, Ieee, 2000.
- [52] T. A. Howell, B. E. Jackson, and Z. Manchester, “Altro: A fast solver for constrained trajectory optimization,” in *RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE, 2019) pp*, pp. 7674–7679.
- [53] Z. Xie, C. K. Liu, and K. Hauser, “Differential dynamic programming with nonlinear constraints,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 695–702, IEEE, 2017.
- [54] B. Plancher, Z. Manchester, and S. Kuindersma, “Constrained unscented dynamic programming,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5674–5680, IEEE, 2017.
- [55] S. S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

- [56] H. Hong, A. Maity, and F. Holzapfel, “Free final-time constrained sequential quadratic programming–based flight vehicle guidance,” *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 1, pp. 181–189, 2021.
- [57] A. Maity, R. Padhi, S. Mallaram, G. M. Rao, and M. Manickavasagam, “A robust and high precision optimal explicit guidance scheme for solid motor propelled launch vehicles with thrust and drag uncertainty,” *International Journal of Systems Science*, vol. 47, no. 13, pp. 3078–3097, 2016.
- [58] B. S. Kim, Y. D. Kim, H. C. Bang, M. J. Tahk, and S. K. Hong, *Flight dynamics and control*. Kyungmoon-sa, Seoul, 2004.
- [59] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.
- [60] S. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw Hill, 2011.
- [61] I. Arasaratnam, S. Haykin, and R. J. Elliott, “Discrete-time nonlinear filtering algorithms using gauss–hermite quadrature,” *Proceedings of the IEEE*, vol. 95, no. 5, pp. 953–977, 2007.
- [62] K. Ito and K. Xiong, “Gaussian filters for nonlinear filtering problems,” *IEEE transactions on automatic control*, vol. 45, no. 5, pp. 910–927, 2000.
- [63] B. Jia, M. Xin, and Y. Cheng, “Sparse gauss-hermite quadrature filter with application to spacecraft attitude estimation,” *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 367–379, 2011.
- [64] B. Jia, M. Xin, and Y. Cheng, “Sparse-grid quadrature nonlinear filtering,” *Automatica*, vol. 48, no. 2, pp. 327–341, 2012.
- [65] F. A. Administration, *Airplane Flying Handbook (FAA-H-8083-3A)*. Skyhorse Publishing Inc., 2011.

- [66] D. Kim, G. Oh, Y. Seo, and Y. Kim, “Reinforcement learning-based optimal flat spin recovery for unmanned aerial vehicle,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 4, pp. 1076–1084, 2017.
- [67] H.-G. Kim and H. J. Kim, “Backstepping-based impact time control guidance law for missiles with reduced seeker field-of-view,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 1, pp. 82–94, 2018.
- [68] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.
- [69] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.
- [70] W. H. Montgomery and S. Levine, “Guided policy search via approximate mirror descent,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [71] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.
- [72] S. Levine and P. Abbeel, “Learning neural network policies with guided policy search under unknown dynamics,” *Advances in neural information processing systems*, vol. 27, 2014.
- [73] V. Kumar, E. Todorov, and S. Levine, “Optimal control with learned local models: Application to dexterous manipulation,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 378–383, IEEE, 2016.
- [74] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

- [75] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, “Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search,” in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 528–535, IEEE, 2016.
- [76] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [77] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.
- [78] D. A. Reynolds, “Gaussian mixture models,” *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.
- [79] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.
- [80] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, “Finite mixture models,” *Annual review of statistics and its application*, vol. 6, pp. 355–378, 2019.
- [81] Z. Zhang, “A note on wishart and inverse wishart priors for covariance matrix,” *Journal of Behavioral Data Science*, vol. 1, no. 2, pp. 119–126, 2021.
- [82] S. Weisberg, *Applied linear regression*, vol. 528. John Wiley & Sons, 2005.
- [83] J. S. Bendat and A. G. Piersol, *Random data: analysis and measurement procedures*. John Wiley & Sons, 2011.