Dynamic Energy Planning for Autonomous UAVs

University of Southern Denmark

Dynamic Energy Planning for Autonomous UAVs

A Dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Robotics

by

Adam Seewald

Approved by:

Dr. Ulrik Pagh Schultz, Advisor
Unmanned Aerial Systems Center
University of Southern Denmark

Approved on:

2021

https://doi.org/10.

The typesetting is done using LaTeX. Figures are generated with PGF/TikZ. Fonts are EB Garamond, Helvetica, and Iwona for body, headers, and equations respectively.

First edition, 2021

Published by University of Southern Denmark, in Odense, Denmark

To …

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesentimperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet antelobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectustellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia loremsit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesentimperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet antelobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectustellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia loremsit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Contents

1 Introduction 1

1.1 A Brief History of Aerial Robotics 1

1.2 Motivation 1

1.3 Objective 1

1.4 Outline of the Approach 1

1.5 Applications 1

1.6 Problem Formulation 1

1.6.1 Problem formulation 4

1.7 Structure 4

2 State of the Art 5

2.1 Energy Modeling 5

2.2 Motion Planning 8

2.3 Planning with Dynamics 8

2.4 Planning for Autonomous UAVs 8

2.4.1 Flight controllers 8

2.4.2 Energy models in aerial robotics 8

2.4.3 State estimation in aerial robotics 8

2.4.4 Optimal control in aerial robotics 8

2.5 Planning Computations with Motion 8

2.6 Summary 8

3 Energy Models 9

3.1 Models Classification 9

3.2 Energy Model of the Computations 9

3.2.1 Computational energy of the UAV 9

3.2.2 Measurement layer 9

3.2.3 Hardware platforms and benchmarks 9

3.2.4 Power measurements for embedded boards . . . 9
3.2.5 The `powprofiler` tool . . . 9
3.2.6 Energy-aware design of algorithms . . . 9
3.2.7 ROS middleware . . . 9
3.2.8 Experimental methodology . . . 9
3.3 Battery Model . . . 9
3.3.1 UAV batteries . . . 10
3.3.2 Derivation of differential battery model . . . 10
3.4 Energy Model of the Motion . . . 10
3.4.1 Mechanical energy of the UAV . . . 10
3.4.2 Experimental methodology . . . 10
3.5 Periodic Energy Model . . . 10
3.5.1 Furier series of empirical data . . . 10
3.5.2 Derivation of differential periodic model . . . 10
3.5.3 Derivation of the nominal control . . . 15
3.5.4 Merging computations and motion . . . 16
3.6 Results . . . 18
3.7 Summary . . . 18

4 State Estimation 19
4.1 A Brief History of State Estimation . . . 19
4.2 A Necessary Background on Probability Theory . . . 19
4.3 The Curve Fitting Problem . . . 19
4.3.1 Least square fitting . . . 19
4.3.2 Levenberg-Marquardt algorithm . . . 19
4.4 Periodic Model Estimation . . . 19
4.4.1 Period estimation . . . 19
4.4.2 Minimization of the estiamte error . . . 19
4.4.3 Period and horizon in continuous estimation . . . 19
4.5 Filtering . . . 19
4.5.1 Discrete time Kalman filter . . . 19
4.5.2 Continuous time Kalman filter . . . 20
4.5.3 Nonlinear filtering . . . 20
4.6 Results . . . 20
4.7 Summary . . . 20

5 Guidance 21
5.1 Vector Fields for Guidance . . . 21
5.2 Derivation of the Guidance Action . . . 21
5.2.1 Motion simulations . . . 21

5.2.2 Energy simulations . . . 21
5.3 Alteration of the Path . . . 21
5.4 Results . . . 21
5.5 Summary . . . 21

6 Optimal Control Generation 23
6.1 A Brief History of Optimal Control . . . 24
6.2 Optimization Problems with Dynamics . . . 26
6.2.1 Continuous systems: unconstrained case . . . 26
6.2.2 Continuous systems: constrained case . . . 27
6.2.3 Perturbed systems . . . 27
6.2.4 Multistage systems . . . 27
6.3 Numerical Simulation and Differentiation . . . 27
6.3.1 Euler method . . . 27
6.3.2 Runge-Kutta methods . . . 27
6.3.3 Algorithmic differentiation . . . 27
6.4 Direct Optimal Control Methods . . . 27
6.4.1 Direct single shooting . . . 27
6.4.2 Direct multiple shooting . . . 27
6.4.3 Direct collocation . . . 27
6.5 Numerical Optimization . . . 28
6.5.1 Convexity . . . 28
6.5.2 Optimality conditions . . . 28
6.5.3 First order optimality conditions . . . 28
6.5.4 Second order optimality conditions . . . 28
6.5.5 Sequential quadratic programming . . . 28
6.5.6 Nonlinear interior point methods . . . 28
6.6 Model Predictive Control . . . 28
6.6.1 Output model predictive control . . . 28
6.6.2 Optimal control generation with model predictive control . . . 28
6.7 Results . . . 28
6.8 Summary . . . 28

References 29

Figures

The plan defined as a FSM 2
Definition notation shown on an initial slice of the plan in Figure ??. 3
Detail of the stage Γ_i in the FSM 3

The alteration of the path parameter $c_{1,1}$, the radius of the circle (it corresponds to the alteration of the plan in Figure ??). 16

Summary of the optimal control approach 24

Notation

$\exists$	there exists

Abbreviations

LP	linear program
QP	quadratic program
MPC	model predictive control
NLP	non linear program
UAV	unmanned aerial vehicle
OCP	optimal control problem
BVP	boundary-value problem

Chapter 1

Introduction

Status
General structure with just some dummy text.

A$^{\text{A}}$

1.1 A Brief History of Aerial Robotics

1.2 Motivation

1.3 Objective

1.4 Outline of the Approach

1.5 Applications

1.6 Problem Formulation

Let us adopt the following mathematical notation. Given an integer a, $[a]$ is the set $\{0, 1, \ldots, a\}$, $[a]^+$ the set $[a]/\{0\}$. Bold lower-case letters indicates vectors. $c_{i,j}$ the j-th parameter of the i-th parameters set c_i. $\underline{c}_{i,j}$, $\overline{c}_{i,j}$ are the lower and upper bounds of the parameter $c_{i,j}$.

Let us assume that the path at stage i can be altered with ρ path parameters

$$c_i^\rho := \{c_{i,1}, c_{i,2}, \ldots, c_{i,\rho}\}, \tag{1.1}$$

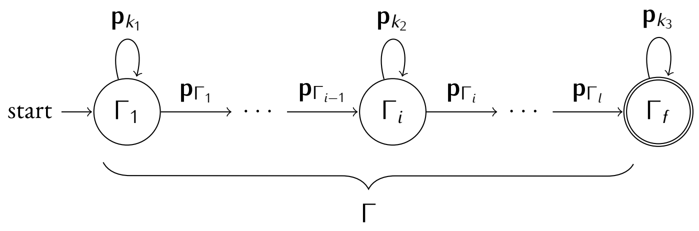

Fig. 1.1. The plan defined as a FSM

and the computations with σ computation parameters

$$c_i^\sigma := \{c_{i,\rho+1}, c_{i,\rho+2}, \dots, c_{i,\rho+\sigma}\}. \tag{1.2}$$

We then express the path as a continuous twice differentiable function $\varphi_i : \mathbb{R}^2 \times \mathbb{R}^\rho \to \mathbb{R}$ of a point and the path parameters. The function returns a metric of the distance between the point and the nominal trajectory. We express the computations as the value of the computation parameters. We discuss the concrete meaning of the value of path parameters in Subsection ??, and computation parameters in Subsection ??.

Definition 1.6.1 (Stage, plan, triggering, and final point)
The i-th stage *Γ_i at time instant k of a plan Γ is defined*

$$\Gamma_i := \{\varphi_i(\mathbf{p}_k, c_i^\rho), c_i^\sigma \mid \exists\ \mathbf{p}_k,\ \varphi_i(\mathbf{p}_k, c_i^\rho) \in \mathcal{C}_i, \\ \forall j \in [\sigma]^+,\ c_{i,\rho+j} \in \mathcal{S}_{i,j}\},$$

where $\mathcal{C}_i := [\underline{c}_i, \overline{c}_i] \subseteq \mathbb{R}$ is the path constraint set, and $\mathcal{S}_{i,j} := [\underline{c}_{i,\rho+j}, \overline{c}_{i,\rho+j}] \subseteq \mathbb{Z}_{\geq 0}$ the j-th computation constraint set. $\mathbf{p}_k$ is a point of a UAV flying at an altitude $h \in \mathbb{R}_{>0}$ w.r.t. some inertial navigation frame $\mathcal{O}_W$.
The plan *is a finite state machine (FSM) Γ where the state-transition function $s : \bigcup_i \Gamma_i \times \mathbb{R}^2 \to \bigcup_i \Gamma_i$ maps a stage and a point to the next stage*

$$s(\Gamma_i, \mathbf{p}_k) := \begin{cases} \Gamma_{i+1} & \textit{if } \mathbf{p}_k = \mathbf{p}_{\Gamma_i} \\ \Gamma_i & \textit{otherwise} \end{cases}.$$

The point $\mathbf{p}_{\Gamma_i}$ that allows the transition between Γ_i and Γ_{i+1} is called triggering point. *The last triggering point $\mathbf{p}_{\Gamma_l}$ relative to the last stage Γ_l is called* final point.

The altitude h from Definition 1.6.1 might change at different flying phases and under different atmospheric conditions

In Figure 1.2, $\varphi_1, \dots, \varphi_6$ are paths. φ_1 and φ_5 are circles, while φ_2, φ_4, and φ_6 are lines. They are all relative to different stages $\Gamma_1, \Gamma_2, \dots$. The constraints set

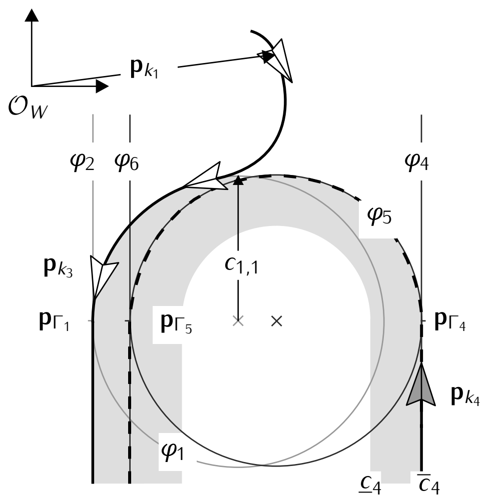

Fig. 1.2. Definition notation shown on an initial slice of the plan in Figure ??.

$\mathcal{C}_1, \mathcal{C}_2, \dots$ forms the area where the paths $\varphi_1, \varphi_2, \dots$ can be altered with the parameters $c_{i,1}, \dots, c_{i,\rho}$ (gray area in the figure). This area is bounded by $\underline{c}_i$, $\overline{c}_i$, and can be different per each stage (in Figure 1.2, the area relative to Γ_4 is bounded by $\underline{c}_4, \overline{c}_4$).

In Figure 1.2, $\mathbf{p}_{\Gamma_1}$ allows the transition between Γ_1 and Γ_2, $\mathbf{p}_{\Gamma_4}$ between Γ_4 and Γ_5, and $\mathbf{p}_{\Gamma_5}$ between Γ_5 and Γ_6.

A slice of the plan in Figure 1.1 shows the transition between the stages with the FSM. The triggering point $\mathbf{p}_{\Gamma_{i-1}}$ allows the transition to the stage Γ_i. The UAV remains in the stage with any generic point $\mathbf{p}_{k_2}$. It eventually enters the stage Γ_{i+1} with the triggering point $\mathbf{p}_{\Gamma_i}$ and so on, until it reaches the final point. The stage Γ_f is the accepting stage (it indicates that the UAV has completed the plan).

$$\begin{tikzcd}
\cdots \arrow[r, "\mathbf{p}_{\Gamma_{i-1}}(c_1^{\rho}, \dots, c_{i-1}^{\rho})"] & \boxed{\Gamma_i} \arrow[loop above, "\mathbf{p}_k"] \arrow[r, "\mathbf{p}_{\Gamma_i}(c_1^{\rho}, \dots, c_i^{\rho})"] & \cdots
\end{tikzcd}$$

Fig. 1.3. Detail of the stage Γ_i in the FSM

Generally, one can express the triggering points in function of the i-th trajectory parameters c_i^{ρ}, or any previous trajectory parameters, propagating the information therein if necessary (see Figure 1.3).

1.6.1 Problem formulation

In order to simplify the problem formulation, we consider some primitive paths. All the other paths are built from these paths with a shift $\mathbf{d} := (x_d, y_d)$.

Given $n \in \mathbb{Z}_{>0}$ $(n < l, l/n \in \mathbb{Z})$ primitive paths $\varphi_1, \dots \varphi_n$, a generic starting point $\mathbf{p}$ and the current levels of the path parameters c_1^ρ, all the other paths $\varphi_{n+1}, \dots, \varphi_l$ are built

$$\begin{aligned}\varphi_{(i-1)n+j}(\mathbf{p} + (i-1)\mathbf{d}, c_1^\rho) - \\ \varphi_{in+j}(\mathbf{p} + i\mathbf{d}, c_1^\rho) = e_j,\end{aligned} \tag{1.3}$$

$\forall i \in [l/n - 1]^+, j \in [n]^+$, where $e_j \in \mathbb{R}$ is the j-th constant difference.

Definition 1.6.2 (Period)
The period $T \in \mathbb{R}_{>0}$ is the time between $\varphi_{(i-1)n+j}$ and φ_{in+j} in Equation (1.3).

The algorithm measures the time between the paths and assumes the initial period is one. The periods might be different for different js due to atmospheric interferences.

One can define the plan using primitive paths or define all the stages explicitly and find n searching the value which satisfies the Equation (1.3). If there is no such value, (e.g., when the plan is composed of only one stage), the period T from Definition 1.6.2 can be determined empirically from energy data (such as these shown in Figure ??).

Problem 1.6.1 (UAV planning problem)
Consider an initial plan Γ from Definition 1.6.1. We are interested in the planning of the parameters c_i, $\forall i \in [l]^+$ and energy constraints and in the guidance of the UAV to the path resulting from such plan.

1.7 Structure

Chapter 2

State of the Art

Status

Pasted energy modeling in Section 2.1 *from* (Seewald, Schultz, Ebeid, et al., 2019), *rest dummy text.*

A ^{A}

2.1 Energy Modeling

Marowka developed a power-metrics based analytical model, to exploit the possibility of considerably increasing energy efficiency by choosing an optimal chip configuration, which we extended to evaluate the impact of different architectural design choices on energy efficiency (Marowka, 2017). Marowka's theoretical contribution shows three processing schemes for heterogeneous computing with a comparison of their respective energy efficiency. Variations in chip configurations are done to investigate the impact on energy, power, and performance. Symmetric processor scheme consists of only a multicore CPU. Asymmetric CPU-GPU processor scheme consists of both CPU and GPU on the hardware side, and of a program running on CPU or GPU, but not on both in the same time interval, on the software side. CPU-GPU simultaneous processing scheme consists of a program running on CPU and GPU simultaneously. Our work extends and starts from Marowka's approach by building an experimental method to the CPU-GPU simultaneous processing scheme.

For evaluating the effects of a battery as an energy source, we used the work done by (R. Rao et al., 2003). Their work summarizes state-of-the-art battery modeling into four classes of models that capture the battery state and its non-linearities. The lowest class contains the physical models that are accurate and model battery state evolution through a set of ordinary and partial differential equations. However, they suffer from a significant level of complexity that reflects on the time needed to produce predictions. The work proceeds by showing empirical models, that predict battery state from empirical trials. The third class consists of abstract models that we incorporated into our approach, in particular, by deriving the equation from the model developed by (Hasan et al., 2018) (they model battery state through an equivalent electrical circuit and its evolution in time). The fourth class consists of mixed models where experimental data are collected and subsequently refined with analytical expressions to determine the models' parameters.

System-level optimization techniques, such as dynamic voltage scaling, have been used for lowering the power consumption (Chowdhury and Chakrabarti, 2005; I. Hong et al., 1999; Luo and Jha, 2001). These techniques are available for hardware featuring dynamic voltage scaling and aim to achieve higher energy efficiency by including information about configuration parameters into the scheduler. They however focus on homogeneous systems, unlike our work which is designed to work for heterogeneous systems. This approach to modeling has nevertheless been extended to include GPU features (S. Hong and Kim, 2010), to heterogeneous systems (Bailey et al., 2014), to optimal software partitioning (Goraczko et al., 2008), and by Wu et al. to machine learning techniques (Wu et al., 2015). The work by Wu et al. mostly relies on neural networks and has been introduced only recently in the field of power estimation and modeling for heterogeneous systems. In particular, for a collection of applications, Wu et al. train a neural network by measuring a number of performance counters for different configurations. Even if these techniques perform well for defining static optimization strategies, they are generally not suitable for heterogeneous parallel systems in aerial robotics. In these cases, systems suffer from a considerable level of uncertainty, for which reason a statically defined energy model often would not model the real energy behavior. An overview of energy estimation in the context of machine learning approaches has recently been presented by (García-Martín et al., 2019). They present a literature review motivated by a belief that the machine learning community is unfamiliar with energy models, but do not relate to GPU-featured devices nor in general heterogeneous devices. In contrast, in our work we aim at automating the generation of application-level power estimation models that can be adapted for machine learning algorithms. Our approach does not yet take detailed scheduling decisions into account, unlike the black-box approach for CPU-GPU energy-aware scheduling presented by (Barik et al., 2016): they model the power by relating execution time to power consumption but otherwise do not focus on energy models.

Our approach shares the same principle of differentiating the microcontroller from the companion computer with (Mei et al., 2004, 2005). The controller acts on the actuators and reads the sensors, while the companion computer (can be found with different names in literature, such as secondary or embedded computer), performs computationally heavy operations. A similar approach for mobile robots is presented by (Dressler and Fuchs, 2005). However, both contributions neither elaborate further on computational elements of a heterogeneous platform, such as GPU, nor focus on different robots except the one under analysis.

The majority of other contributions in the literature focus on optimizing motion planning to increase power efficiency. For instance, approaches to minimize UAV power consumption, such as the work by (Kreciglowa et al., 2017), aim to determine the best trajectory generation method for an aerial vehicle to travel from one configuration to another. Uragun suggests the use of power-efficient components (Uragun, 2011): an energy-efficient UAV system can either be built using conceptual product development with emerging technologies or using energy-efficient components. Kanellakis et al. affirm that integrating visual sensors in the UAV ecosystem still lacks solid experimental evaluation (Kanellakis and Nikolakopoulos, 2017). They suggest that to save energy, the available payload for sensing and computing has to be restricted. Our approach towards energy modeling shares a similar principle as the one presented by (Sadrpour et al., 2013a,b) for Unmanned Ground Vehicles or UGVs. They propose a linear regression-based technique in the absence of real measurements and a Bayesian networks-based one in their presence. We used a simplified approximation technique to limit the number of computations needed while focusing rather on an accurate battery prediction.

To validate our approach and quantify its outcomes, we used the models previously developed for fine-grained energy modeling by (Nunez-Yanez and Lore, 2013), and (Nikov et al., 2015) respectively. In summary, fine-grained energy modeling uses hardware event registers to capture the CPU state under a representative workload. The energy-modeling consists of three stages. In data collection, the first stage, a benchmark runs on the platform and data are collected. The second and third stage, data processing and model generation, are performed offline on a different architecture. In these two stages, data are analyzed and a model that predicts possible future usage is generated.

Calore et al. develop an approach for measuring power efficiency for High-Performance Computing or HPC systems (Calore et al., 2015). An external board is used to measure the power consumption, while the data are collected from NVIDIA Jetson TK1 board running one benchmark. Our initial analysis presented at HLPGPU 2019 was made using a similar technique (Seewald, Ebeid, et al., 2019). A shunt resistor and digital multimeter integrated into the external board was used to evaluate the power efficiency. In this paper we extend our experiments to use internal power monitors and address a broader range of platforms. We now build a proper energy model that reflects the computational behavior of the device under study and shows the energy evolution. An early report on our work has been presented in the Team-

Play project's deliverable D4.3 (*Public Deliverables of the TeamPlay Horizon2020 Project* 2019) "Report on Energy, Timing and Security Modeling of Complex Architectures".

2.2 Motion Planning

2.3 Planning with Dynamics

2.4 Planning for Autonomous UAVs

2.4.1 Flight controllers

2.4.2 Energy models in aerial robotics

2.4.3 State estimation in aerial robotics

2.4.4 Optimal control in aerial robotics

2.5 Planning Computations with Motion

2.6 Summary

Chapter 3

Energy Models

Status
Started filling the model from (Seewald, Garcia de Marina, and Schultz, n.d.) (Subsection 3.5.2)*, rest is dummy text.*

A^{A}

3.1 Models Classification

3.2 Energy Model of the Computations

3.2.1 Computational energy of the UAV

3.2.2 Measurement layer

3.2.3 Hardware platforms and benchmarks

3.2.4 Power measurements for embedded boards

3.2.5 The `powprofiler` tool

3.2.6 Energy-aware design of algorithms

3.2.7 ROS middleware

3.2.8 Experimental methodology

3.3 Battery Model

3.3.1 UAV batteries

3.3.2 Derivation of differential battery model

3.4 Energy Model of the Motion

3.4.1 Mechanical energy of the UAV

3.4.2 Experimental methodology

3.5 Periodic Energy Model

3.5.1 Furier series of empirical data

3.5.2 Derivation of differential periodic model

We refer to the instantaneous energy consumption evolution simply as the energy signal. We model the energy using energy coefficients $\mathbf{q} \in \mathbb{R}^m$ that characterize such energy signal. The coefficients are derived from Fourier analysis (the size of the energy coefficients vector m is related to the order of a Fourier series) and estimated using a state estimator.

We prove a relation between the energy signal and the energy coefficients in Lemma 3.5.1. We show after the main results how this approach allows us variability in terms of non-periodic signals.

After having illustrated the energy model, we enhance it with the energy contribution of the path in and of the computations in Subsection 3.5.4.

Let us consider a periodic energy signal of period T, and a Fourier series of an arbitrary order $r \in \mathbb{Z}_{\geq 0}$ for the purpose of modeling of the energy signal

$$h(t) = a_0/T + (2/T)\sum_{j=1}^{r}\big(a_j \cos \omega j t + b_j \sin \omega j t\big), \tag{3.1}$$

where $h : \mathbb{R}_{\geq 0} \to \mathbb{R}$ maps time to the instantaneous energy consumption, $\omega := 2\pi/T$ is the angular frequency, and $a, b \in \mathbb{R}$ the Fourier series coefficients.

The energy signal can be modeled by Equation (3.1) and by the output of a linear model

$$\dot{\mathbf{q}}(t) = A\mathbf{q}(t) + B\mathbf{u}(t), \tag{3.2a}$$

$$y(t) = C\mathbf{q}(t), \tag{3.2b}$$

where $y(t) \in \mathbb{R}$ is the instantaneous energy consumption.

The state $\mathbf{q}(t)$ contains the energy coefficients

$$\mathbf{q}(t) = \begin{bmatrix} \alpha_0(t) & \alpha_1(t) & \beta_1(t) & \cdots & \alpha_r(t) & \beta_r(t) \end{bmatrix}^T, \tag{3.3}$$

where $\mathbf{q}(t) \in \mathbb{R}^m$ with $m = 2r + 1$. The state transition matrix

$$A = \begin{bmatrix} 0 & 0^{1\times 2} & 0^{1\times 2} & \dots & 0^{1\times 2} \\ 0^{2\times 1} & A_1 & 0^{2\times 2} & \dots & 0^{2\times 2} \\ 0^{2\times 1} & 0^{2\times 2} & A_2 & \dots & 0^{2\times 2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0^{2\times 1} & 0^{2\times 2} & 0^{2\times 2} & \dots & A_r \end{bmatrix}, \tag{3.4}$$

where $A \in \mathbb{R}^{m\times m}$. In matrix A, the top left entry is zero, the diagonal entries are $A_1, \dots, A_r$, the remaining entries are zeros. Matrix $0^{i\times j}$ is a zero matrix of i rows and j columns. The submatrices $A_1, A_2, \dots, A_r$ are defined

$$A_j := \begin{bmatrix} 0 & \omega j \\ -\omega j & 0 \end{bmatrix}, \tag{3.5}$$

The output matrix

$$C = (1/T)\begin{bmatrix} 1 & 1 & 0 & \cdots & 1 & 0 \end{bmatrix}, \tag{3.6}$$

where $C \in \mathbb{R}^m$.

The linear model in Equation (3.2) allows us to include the control in the model of Equation (3.1).

Lemma 3.5.1 (Signal, output equality)
Suppose control $\mathbf{u}$ is a zero vector, matrices A, C are described by Equation (3.3), and the initial guess $\mathbf{q}_0$ is

$$\mathbf{q}_0 = \begin{bmatrix} a_0 & a_1/2 & b_1/2 & \cdots & a_r/2 & b_r/2 \end{bmatrix}^T.$$

Then, the signal h in Equation (3.1) is equal to the output y in Equation (3.2).

Proof. We propose a formal proof of the lemma. The proof justifies the choice of the items of the matrices A, C and of the initial guess $\mathbf{q}_0$ in Equation (3.3). We write these elements such that the coefficients of the series $a_0, \dots, b_r$ are the same as the coefficients of the state $\alpha_0, \dots, \beta_r$.

Let us re-write the Fourier series expression in Equation (3.1) in its complex form with the well-known Euler's formula

$$e^{it} = \cos t + i \sin t. \tag{3.7}$$

With $t = \omega j t$, we find the expression for

$$\cos \omega j t = (e^{i\omega j t} + e^{-i\omega j t})/2 \tag{3.8a}$$
$$\sin \omega j t = (e^{i\omega j t} - e^{-i\omega j t})/(2i) \tag{3.8b}$$

by substitution of $\sin \omega j t$ and $\cos \omega j t$ respectively. This leads Kuo, 1967

$$\begin{aligned} h(t) = a_0/T + (1/T)\sum_{j=1}^{r} e^{i\omega j t}(a_j - ib_j) + \\ (1/T)\sum_{j=1}^{r} e^{-i\omega j t}(a_j + ib_j), \end{aligned} \tag{3.9}$$

where i is the imaginary unit.

The solution at time t can be expressed

$$\mathbf{q}(t) = e^{At}\mathbf{q}_0. \tag{3.10}$$

Both the solution and the system in Equation (3.2) are well established expressions derived using standard textbooks Kuo, 1967; Ogata, 2002. To solve the matrix exponential e^{At}, we use the eigenvectors matrix decomposition method Moler and Van Loan, 2003.

The method works on the similarity transformation

$$A = VDV^{-1}. \tag{3.11}$$

The power series definition of e^{At} implies Moler and Van Loan, 2003

$$e^{At} = Ve^{Dt}V^{-1}. \tag{3.12}$$

We consider the non-singular matrix V, whose columns are eigenvectors of A

$$V := \begin{bmatrix} v_0 & v_1^0 & v_1^1 & \dots & v_r^0 & v_r^1 \end{bmatrix}. \tag{3.13}$$

We then consider the diagonal matrix of eigenvalues

$$D = \mathrm{diag}(\lambda_0, \lambda_1^0, \lambda_1^1, \dots, \lambda_r^0, \lambda_r^1). \tag{3.14}$$

λ_0 is the eigenvalue associated to the first item of A. λ_j^0, λ_j^1 are the two eigenvalues associated with the block A_j. We can write

$$Av_j = \lambda_j v_j \;\; \forall j = \{1, \dots, m\}, \tag{3.15}$$

and

$$AV = VD. \tag{3.16}$$

We apply the approach in terms of Equation (3.2), under the assumptions made in the lemma (the control is a zero vector)

$$\dot{\mathbf{q}}(t) = A\mathbf{q}(t). \tag{3.17}$$

The linear combination of the initial guess and the generic solution

$$F\mathbf{q}(0) = \gamma_0 v_0 + \sum_{k=0}^{1}\sum_{j=1}^{r} \gamma_j v_j^k, \tag{3.18a}$$

$$F\mathbf{q}(t) = \gamma_0 e^{\lambda_0 t} v_0 + \sum_{k=0}^{1}\sum_{j=1}^{r} \gamma_j e^{\lambda_j t} v_j^k, \tag{3.18b}$$

where

$$F = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \tag{3.19}$$

is a $F \in \mathbb{R}^m$ column vector of ones.

Let us consider the second expression in Equation (3.18). It represents the linear combination of all the coefficients of the state at time t. It can also be expressed in the following form

$$\begin{aligned} F\mathbf{q}(t)/T = \gamma_0 e^{\lambda_0 t} v_0/T + (1/T)\sum_{j=1}^{r} \gamma_j e^{\lambda_j^0 t} v_j^0 + \\ (1/T)\sum_{j=1}^{r} \gamma_j e^{\lambda_j^1 t} v_j^1. \end{aligned} \tag{3.20}$$

We proof that the eigenvalues λ and eigenvectors V are such that Equation (3.20) is equivalent to Equation (3.9).

The matrix A is a block diagonal matrix, so we can express its determinant as the multiplication of the determinants of its blocks

$$\det(A) = \det(0)\det(A_1)\det(A_2)\cdots\det(A_r). \tag{3.21}$$

We proof the first determinant and the others separately.

Thereby we start by proofing that the first terms of the Equations (3.9–3.20) match. We find the eigenvalue from $\det(0) = 0$, which is $\lambda_0 = 0$. The corresponding eigenvector can be chosen arbitrarily

$$(0 - \lambda_0)v_0 = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}, \tag{3.22}$$

$\forall v_0$, thus we choose

$$v_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}. \tag{3.23}$$

We find the value γ_0 of the vector γ so that the terms are equal

$$\gamma_0 = \begin{bmatrix} a_0 & 0 & \cdots & 0 \end{bmatrix}. \tag{3.24}$$

Then, we proof that all the terms in the sum of both the Equations (3.9–3.20) match.

For the first block A_1, we find the eigenvalues from

$$\det(A_1 - \lambda I) = 0. \tag{3.25}$$

The polynomial $\lambda^2 + \omega^2$, gives two complex roots–the two eigenvalues

$$\lambda_1^0 = i\omega, \tag{3.26a}$$

$$\lambda_1^1 = -i\omega. \tag{3.26b}$$

The eigenvector associated with the eigenvalue λ_1^0 is

$$v_1^0 = \begin{bmatrix} 0 & -i & 1 & 0 & \cdots & 0 \end{bmatrix}^T. \tag{3.27}$$

The eigenvector associated with the eigenvalue λ_1^1 is

$$v_1^1 = \begin{bmatrix} 0 & i & 1 & 0 & \cdots & 0 \end{bmatrix}^T. \tag{3.28}$$

Again, we find the values γ_1 of the vector γ such that the equivalences

$$\begin{cases} e^{i\omega t}(a_1 - ib_1) & = \gamma_1 e^{i\omega t} v_1^0 \\ e^{-i\omega t}(a_1 + ib_1) & = \gamma_1 e^{i\omega t} v_1^1 \end{cases} \tag{3.29}$$

hold. They hold for

$$\gamma_1 = \begin{bmatrix} b_1 & a_1 \end{bmatrix}. \tag{3.30}$$

The proof for the remaining $r - 1$ blocks is equivalent.

The initial guess is build such that the sum of the coefficients is the same in both the signals. In the output matrix, the frequency $1/T$ accounts for the period in Equations (3.9–3.20) and Equation (3.1). At time instant zero, the coefficients b_j are not present and the coefficients a_j are doubled for each $j = 1, 2, \ldots, r$ (thus we multiply by a half the corresponding coefficients in $\mathbf{q}_0$). To match the outputs $h(t) = y(t)$, or equivalently

$$F\mathbf{q}(t)/T = C\mathbf{q}(t), \tag{3.31}$$

we define

$$C := (1/T)\begin{bmatrix} 1 & 1 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{3.32}$$

We thus conclude that the signal and the output are equal, hence the lemma holds.

■

We note for practical reasons that the signal would still be periodic with another linear combination of coefficients. For instance,

$$C := d\begin{bmatrix} 1 & 0 & 1 & \cdots & 0 & 1 \end{bmatrix}, \tag{3.33}$$

or

$$C := d\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \tag{3.34}$$

for a constant value $d \in \mathbb{R}$.

3.5.3 Derivation of the nominal control

Let us suppose that at time instant t the plan reached the i-th stage Γ_i and the control

$$\mathbf{c}_i(t) = \begin{bmatrix} c_i^{\rho}(t) & c_i^{\sigma}(t) \end{bmatrix}^T, \tag{3.35}$$

where $\mathbf{c}_i(t) \in \mathbb{R}^n$ with $n := \rho + \sigma$ differs from the nominal control $\mathbf{u}(t)$ in Equation (3.2). We include the control in the nominal control exploiting the following observation.

Observation
We observe that:

- *A change in path parameters affects the energy indirectly. It alters the time when the UAV reaches the final point* $\mathbf{p}_{\Gamma_i}$.
- *A change in computation parameters affects the energy directly. It alters the instantaneous energy consumption as more computations require more power (and vice versa).*

We use this information later in the algorithm to check that the battery discharge time is greater and replan the path parameters accordingly. We replan the computation parameters to maximize the instantaneous energy consumption against the maximum battery discharge rate.

The nominal control is

$$\mathbf{u}(t) := \hat{\mathbf{u}}(t) - \hat{\mathbf{u}}(t-1), \tag{3.36}$$

where $\hat{\mathbf{u}}(t)$ is defined as the energy estimate of a given control sequence at time instant t, $\hat{\mathbf{u}}(t-1)$ at the previous time instant $t-1$

$$\hat{\mathbf{u}}(t) := \mathrm{diag}(\nu_i)\mathbf{c}_i(t) + \tau_i. \tag{3.37}$$

The input matrix is then

$$B = \begin{bmatrix} 0^{1\times\rho} & 1 & \cdots & 1 \\ 0^{1\times\rho} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0^{1\times\rho} & 0 & \cdots & 0 \end{bmatrix}, \tag{3.38}$$

where $B \in \mathbb{R}^{m\times n}$ contains zeros except the first row where the first ρ columns are still zeros and the remaining σ are ones.

$\hat{\mathbf{u}}(t)$ is a stage-dependent scale transformation with

$$\nu_i = \begin{bmatrix} \nu_i^\rho & \nu_i^\sigma \end{bmatrix}^T, \tag{3.39a}$$

$$\tau_i = \begin{bmatrix} \tau_i^\rho & \tau_i^\sigma \end{bmatrix}^T, \tag{3.39b}$$

scaling factors quantifying the contribution to the plan of a given parameter in terms of time for the first ρ parameters, and power for the remaining σ (we use the same notation for the path and computation scaling factors as for the parameters).

The nominal control $\mathbf{u}(t)$ is then the difference of these contributions of two consecutive controls $\mathbf{u}(t-1)$, $\mathbf{u}(t)$ applied to the system.

$B\mathbf{u}(t)$ merely includes the difference in power into the model in Equation (3.2).

3.5.4 Merging computations and motion

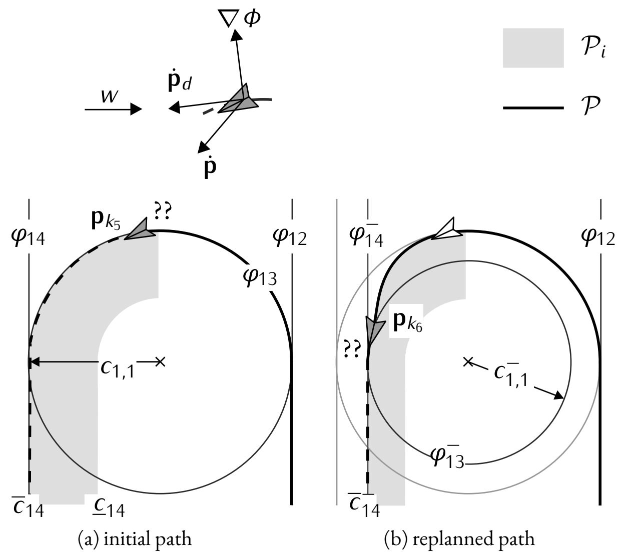

Fig. 3.1. The alteration of the path parameter $c_{1,1}$, the radius of the circle (it corresponds to the alteration of the plan in Figure ??).

The set

$$\mathcal{P}_i := \{\mathbf{p}_k \mid \varphi_i(\mathbf{p}_k, c_i^\rho) \in \mathcal{C}_i\}, \tag{3.40}$$

delimits the area where the i-th path φ_i is free to evolve using the path parameters c_i^ρ (the gray area in Figure 3.1). φ_i is a function of the two coordinates and the path parameters, and is equal to zero when a point $\mathbf{p}_k$ is on the path. Physically, this means the UAV is flying exactly over the nominal trajectory. The path parameters allows to change the path. They are a way to alter the nominal trajectory in the initial plan and thus alter the energy by changing the flying time in the example in Figure ??. In fact, the algorithm uses the set from Equation (3.40) to find the path parameters such that the plan consisting of flying φ_i has the highest energy, while still respecting the constraints. In Figure 3.1, the parameter radius of the circle $c_{1,1}$ is replanned as, e.g., averse atmospheric conditions do not allow to terminate the plan.

We derive the new position $\mathbf{p}_{k+1}$ computing the vector field $\nabla\varphi_i := \begin{bmatrix}\partial\varphi_i/\partial x & \partial\varphi_i/\partial y\end{bmatrix}^T$, and the direction to follow in the form of velocity vector de2017guidance

$$\dot{\mathbf{p}}_d(\mathbf{p}_k) := E\nabla\varphi_i - k_e\varphi_i\nabla\varphi_i, \;\; E = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \tag{3.41}$$

where E specifies the rotation (it influence the tracking direction), and $k_e \in \mathbb{R}_{\geq 0}$ the gain to adjusts the speed of convergence. The direction the velocity vector $\dot{\mathbf{p}}_d$ is pointing at is generally different from the course heading $\dot{\mathbf{p}}$ due to the atmospheric interferences (wind $w \in \mathbb{R}$ in the top of Figure 3.1).

The scaling factors for the path parameters from Equation (3.36) are derived empirically. For the example in Figure 3.1, we can obtain the scaling factor $\nu_{1,1}$ measuring the time needed to compute the path with the lowest configuration $\underline{c}_1$, $\underline{t}$ and the highest $\overline{t}$. The variation of the control hence results in an approximate measure of the plans' time variation with factors

$$\begin{aligned} \nu_{i,j} &= \left((\overline{t} - \underline{t})/(\overline{c}_{i,j} - \underline{c}_{i,j})\right)/\rho, \\ \tau_{i,j} &= \left(\underline{c}_{i,j}(\underline{t} - \overline{t})/(\overline{c}_{i,j} - \underline{c}_{i,j}) + \underline{t}\right)/\rho, \end{aligned} \tag{3.42}$$

$\forall j \in [\rho]^+$. Moreover, let the factors be zero when the parameters set $c_i^\rho = \{\emptyset\}$.

Let us recall from Definition 1.6.1 that the i-th stage Γ_i of the plan Γ contains the computation parameters which characterize the computations. We estimate the energy cost of these computations using `powprofiler`, the open-source modeling tool adapted from earlier work on computational energy analysis Seewald, Schultz, Ebeid, et al., 2019; Seewald, Schultz, Roeder, et al., 2019, and energy estimation of a fixed-wing UAV Seewald, Garcia de Marina, Midtiby, et al., 2020.

For this purpose, we assume the UAV carries an embedded board that runs the computations. Our tool measures the instantaneous energy consumption of a subset of possible computation parameters within the computation constraint sets and builds an energy model: a linear interpolation, one per each computation.

The computations are implemented by software components, e.g., Robot Operating System (ROS) nodes in a ROS-based system Quigley et al., 2009. The user implements these nodes such that they change the computational load according to node-specific ROS parameters–the computation parameters. In a generic software component system, the user maps the computational load to the arguments Seewald, Schultz, Roeder, et al., 2019. In both cases, with ROS Zamanakos et al., 2020 or with generic software components system Seewald, Schultz, Roeder, et al., 2019, the tool performs automatic modeling. For instance, if the computation is an object detector, a computation parameter $c_{1,2}$ might correspond to frames-per-second (fps) rate. The tool then measures power according to the detection frequency.

We note that while the path can differ for each stage, the tasks remain the same. However, the user can inhibit or enable a computation varying its computation constraint set.

Let us define $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as the instantaneous computational energy consumption value obtained using the tool.

The scaling factors add the computational energy component to the model in Equation (3.2). They are derived similarly to Equation (3.42)

$$\nu_{i,j} = (g(\overline{c}_{i,j}) - g(\underline{c}_{i,j}))/(\overline{c}_{i,j} - \underline{c}_{i,j}), \tag{3.43a}$$

$$\tau_{i,j} = \underline{c}_{i,j}(g(\underline{c}_{i,j}) - g(\overline{c}_{i,j}))/(\overline{c}_{i,j} - \underline{c}_{i,j}) + g(\underline{c}_{i,j}), \tag{3.43b}$$

$\forall j \in [\rho + 1, \rho + \sigma]$. Moreover, let the factors be zero when the parameters set $c_i^\sigma = \{\emptyset\}$.

3.6 Results

3.7 Summary

Chapter 4

State Estimation

Status
General structure with just some dummy text.

A A,

4.1 A Brief History of State Estimation

4.2 A Necessary Background on Probability Theory

4.3 The Curve Fitting Problem

4.3.1 Least square fitting

4.3.2 Levenberg-Marquardt algorithm

4.4 Periodic Model Estimation

4.4.1 Period estimation

4.4.2 Minimization of the estiamte error

4.4.3 Period and horizon in continuous estimation

4.5 Filtering

4.5.1 Discrete time Kalman filter

4.5.2 Continuous time Kalman filter

4.5.3 Nonlinear filtering

4.6 Results

4.7 Summary

Chapter 5

Guidance

Status
General structure with just some dummy text.

A A

5.1 Vector Fields for Guidance

5.2 Derivation of the Guidance Action

5.2.1 Motion simulations

5.2.2 Energy simulations

5.3 Alteration of the Path

5.4 Results

5.5 Summary

Chapter 6

Optimal Control Generation

Status

Initial version up to Subsection 6.2.1. From Subsection 6.2.2 dummy text.

This chapter provides essential theoretical background on optimal control theory necessary to derive an optimal configuration of the path and computations of the flying UAV. It solves the problem posed in Section 1.6 and illustrate an algorithm that generates the optimal configuration dynamically. The algorithm relies on a modern optimal control technique known as model predictive control (MPC), where the optimal control trajectory is evaluated on a receding horizon for each optimization step (Rawlings et al., 2017).

Optimal control deals with finding optimal ways to control a dynamic system (Sethi, 2019). It determines the control signal–the evolution in time of the decision variables–such that the model satisfies the dynamics and simultaneously optimizes a performance index (Kirk, 2004).

Many optimization problems originating in fields such as robotics, economics, and aeronautics can be formulated as optimal control problems (OCPs) (Von Stryk and Bulirsch, 1992). Optimization is often called mathematical programming (Nocedal and Wright, 2006) a term that means finding ways to solve the optimization problem. One can often find programming in this context in terms such as linear program (LP), quadratic program (QP), and nonlinear program (NLP). NLPs is the class of optimization problems that we use to derive the optimal configuration. OCPs can be seen as optimization problems with the added difficulty of continuous dynamics. The latter is to be integrated over the optimization horizon using numerical simulation. In the algorithm, we formulate the dynamic planning problem as an OCP that we solve with a numerical method: we transform the OCP in an NLP using numerical simulation and solve the NLP using numerical optimization, as proposed in (Rawlings et al., 2017). The process is illustrated in Fig. 6.1.

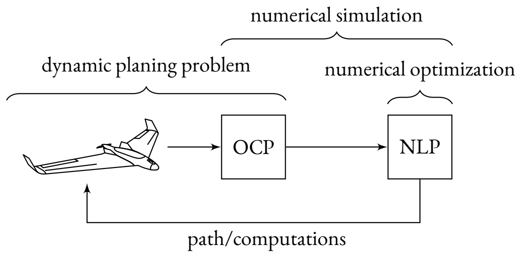

Fig. 6.1. Summary of the optimal control approach. The problem is formulated as an OCP, into finite-dimensional discrete NLP using numerical simulation. NLP is solved using numerical optimization and the optimal configuration for a given time horizon is returned to the UAV. The following horizon is evaluated again in a technique known as MPC.

A typical performance measure for an OCP is built such that the system: reaches a target set $\mathcal{Q}_f$ in minor time, reaches a given final state $\mathbf{q}_f$ with minimum deviation, maintains the state evolution as close as possible to a given desired evolution, or reaches the target set with the minimum control expenditure effort (Kirk, 2004). In energy planning, it is desired to focus on the latter performance measure.

The outline of the chapter is as follows. After a brief history of optimal control, we introduce formally the OCP subject to continuous dynamics. We then illustrate numerical simulation approaches to convert the infinite-dimensional continuous dynamics into finite-dimensional discrete dynamics. We formulate later in the chapter the dynamic planning problem for the optimal configuration of the path and computations with proper constraints. Finally, we illustrate MPC to solve OCP on a receding horizon. We propose an algorithm to solve such OCP using a numerical method on the horizon along with the analysis of its practical feasibility.

The chapter builds on the rest of the work as follows. In the OCP formulation, we use the estimated state from Chapter 4 of a perfect model in Chapter 3 to solve the planning problem in Section 1.6 and guide the UAV with the obtained optimal configuration from this chapter with the technique in Chapter 5.

6.1 A Brief History of Optimal Control

Optimal control originates from the calculus of variations (Sethi, 2019), based on the work of Euler and Lagrange. Calculus of variations solves the problem of determining the arguments of an integral, such that its value is maximum (or minimum). The equivalent problem in calculus is to determine the argument of a function where the function is maximum (or minimum). The work by Euler and Lagrange was later extended by Legendre, Hamilton, and Weierstrass (Paulen and Fikar, 2016). It has gained a renewed interest in the mid-twentieth century, as modern calculators offered practical ways of solving some OCPs for nonlinear and time-varying systems that were earlier impracticable (Bryson and Ho, 1975).

The conversion of the calculus of variation problems in OCPs requires the addition of the control variable to the dynamics (Sethi, 2019).

There are numerous methods to analytically and numerically solve these continuous time OCPs, although analytical solution is often impracticable except for very limited state dimensions (Rawlings et al., 2017). In the early day of optimal control, some analytical solutions were proposed with dynamic programming (Bellman, 1957), and with maximum (or minimum) principle (Pontryagin et al., 1962).

In computer science dynamic programming is fundamental to compute optimal solutions, yet it's original form was developed to solve optimal control problems (LaValle, 2006). Dynamic programming in optimal control theory is based on a partial differential equation of the performance index named Hamilton-Jacobi-Bellman (HJB) equation, which is solved either analytically for small dimensional state space problems, or numerically (Rawlings et al., 2017). Dynamic programming can be shown to be equivalent to the principle (Paulen and Fikar, 2016). The principle is related to HJB equation in that it provides optimality conditions an optimal trajectory must satisfy (LaValle, 2006). HJB offer sufficient conditions for optimality while the principle necessary; yet it is useful to find suitable candidates for optimality (LaValle, 2006).

All the numerical approaches discretize infinite-dimensional problems at a certain point (Rawlings et al., 2017).

A first class of these approaches solves the optimality conditions in continuous time using first-order necessary conditions of optimality from the principle (Böhme and Frank, 2017). This is done by algebraic manipulation using an expression that is similar to the HJB equation, and results in a boundary-value problem (BVP) (Rawlings et al., 2017). The class is commonly referred to as the indirect methods. The BVP is solved by discretization at the very end (Rawlings et al., 2017) and/or gradient-based resolution (Paulen and Fikar, 2016).

On the contrary, modern optimal control often first discretize an the control and state variables in the OCP to a finite dimensional optimization problem (usually NLP), which is then solved with numerical optimization (using gradient-based techniques). This other class of numerical approaches is referred to as direct methods. Some direct methods are single and multiple shooting and collocation methods. We employ direct methods in this chapter.

Modern OCPs are often solved on a finite and receding horizon using an approximation of the true dynamics using MPC techniques. It is a more systematic technique which allow to control a model by re-optimizing the OCP repeatedly (Paulen and Fikar, 2016; Poe and Mokhatab, 2017). It takes into account external interferences by re-estimating the model's state (with techniques that we introduced in Chapter 4). MPC is extensively treated in modern optimal control literature (Camacho and Alba, 2007; Kwon and Han, 2005; Rawlings et al., 2017; Rossiter, 2004; Wang, 2009).

6.2 Optimization Problems with Dynamics

6.2.1 Continuous systems: unconstrained case

Given a state variable $\mathbf{q}$ composed of m states and a control variable $\mathbf{u}$ composed of n controls, the state variable dynamics at a given time instant t can be described by a differential model

$$\dot{\mathbf{q}}(t) = f(\mathbf{q}(t), \mathbf{u}(t), t), \quad \mathbf{q}(t_0) = \mathbf{q}_0 \text{ given}, \quad \forall t \in [t_0, T], \tag{6.1}$$

where $t_0 \in \mathbb{R}_{\geq 0}$ is a given initial time instant, and $\mathbf{q}_0 \in \mathbb{R}^m$ a given initial state guess. The latest can be derived empirically from a previous execution or using some initial sensor data. $f : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^m$ maps the current state, control and time to the next state. The notations for $\dot{\mathbf{q}}(t) := d\mathbf{q}(t)/dt$, $\mathbf{q}$, and $\mathbf{u}$ are the same from Chapter 3. The function f is assumed to be continuously differentiable. Physically, Equation (6.1) specifies the instantaneous change in state variable of a perfect model with no disturbances.

If the control trajectory $\mathbf{u}(t_0), \mathbf{u}(t_1), \ldots, \mathbf{u}(T - \Delta t)$ is known for a given time horizon $t_0 \leq t \leq T$, the model in Equation (6.1) can be derived to obtain the state trajectory $\mathbf{q}(t_0), \mathbf{q}(t_1), \ldots, \mathbf{q}(T)$, where Δt is the instantaneous change in time. The last state at the final time instant T is derived from the last control at the time instant $T - \Delta t$. The state trajectory has indeed one item more than the control trajectory.

Optimal control finds a control trajectory which maximizes (or minimizes) a performance index

$$L = l_f(\mathbf{q}(T), T) + \int_{t_0}^{T} l(\mathbf{q}(t), \mathbf{u}(t), t)\, dt, \tag{6.2}$$

where l, l_f are given instantaneous and final cost functions. The instantaneous cost function maps state, controls, and time to a value that quantifies the cost of a given instant $l : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$. The final cost function maps the state and time to a value which quantifies the cost of the final instant $l_f : \mathbb{R}^m \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$. The performance index $L \in \mathbb{R}$ is then the sum of all the contribution on the time horizon.

Performance index found in (Bryson and Ho, 1975) is also found in literature as cost function in (Simon, 2006; Stengel, 1994), objective function in (S. S. Rao, 2019; Rawlings et al., 2017; Sethi, 2019), or performance measure (Kirk, 2004).

The control variable is usually constrained

$$\mathbf{u}(t) \in \mathcal{U}(t), \quad \forall t \in [t_0, T], \tag{6.3}$$

where $\mathcal{U}(t) \subseteq \mathbb{R}^m$ is the control constraint set. It delimits all the feasible values of the control for the horizon. There can be different control constraint sets for different instants.

The Equations (6.1–6.3) forms unconstrained OCPs. These problems are formalized

$$\begin{aligned}
&\max_{\mathbf{u}(\mathbf{t}) \in \mathcal{U}(\mathbf{t})} l_f(\mathbf{q}(T), T) + \int_{t_0}^{T} l(\mathbf{q}(t), \mathbf{u}(t), t)\, dt, \\
&\text{s.t. } \dot{\mathbf{q}}(t) = f(\mathbf{q}(t), \mathbf{u}(t), t) \\
&\mathbf{q}(t_0) = \mathbf{q}_0 \text{ given.}
\end{aligned} \tag{6.4}$$

The evolution of the model is used to derive an optimal control trajectory $\mathbf{u}(t)$ from an initial guess of the state $\mathbf{q}_0$ and the horizon. This initial simplistic controller does not represent a realistic scenario. The controller implies that the horizon is known. However, it is often the case that only the initial time step of the horizon $[t_0, T]$ is known. In the model from Chapter 3 it is indeed unknown apriori when the UAV plan terminates. Moreover the controller does not include any constraint on the state $\mathbf{q}$, although UAVs are often bounded by strict battery requirements. Lastly, the optimal control generated with such controller is static given the initial state and the horizon. It is unrealistic to assume that the state of the UAV travelling the optimal control $\mathbf{u}$ does not change for instants $t_0 + \Delta t, t_0 + 2\Delta t, \ldots, T$.

All these initial assumption (known final time step, absence of state constraints, static optimal control law) will be eased in the remaining of the chapter.

6.2.2 Continuous systems: constrained case

6.2.3 Perturbed systems

6.2.4 Multistage systems

6.3 Numerical Simulation and Differentiation

6.3.1 Euler method

6.3.2 Runge-Kutta methods

6.3.3 Algorithmic differentiation

6.4 Direct Optimal Control Methods

6.4.1 Direct single shooting

6.4.2 Direct multiple shooting

6.4.3 Direct collocation

6.5 Numerical Optimization

6.5.1 Convexity

6.5.2 Optimality conditions

6.5.3 First order optimality conditions

6.5.4 Second order optimality conditions

6.5.5 Sequential quadratic programming

6.5.6 Nonlinear interior point methods

6.6 Model Predictive Control

6.6.1 Output model predictive control

6.6.2 Optimal control generation with model predictive control

6.7 Results

6.8 Summary

References

Bailey, P. E., D. K. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. R. De Supinski (2014). “Adaptive configuration selection for power-constrained heterogeneous systems”. In: *2014 43rd International Conference on Parallel Processing*. IEEE, pp. 371–380 (cit. on p. 6).

Barik, R., N. Farooqui, B. T. Lewis, C. Hu, and T. Shpeisman (2016). “A black-box approach to energy-aware scheduling on integrated CPU-GPU systems”. In: *Proceedings of the 2016 International Symposium on Code Generation and Optimization*. ACM, pp. 70–81 (cit. on p. 6).

Bellman, R. E. (1957). *Dynamic Programming*. Princeton: Princeton University Press (cit. on p. 25).

Böhme, T. J. and B. Frank (2017). “Indirect Methods for Optimal Control”. In: *Hybrid Systems, Optimal Control and Hybrid Vehicles: Theory, Methods and Applications*. Cham: Springer International Publishing, pp. 215–231 (cit. on p. 25).

Bryson, A. E. and Y.-C. Ho (1975). *Applied optimal control: optimization, estimation and control*. Hemisphere Publishing Corporation (cit. on pp. 25, 26).

Calore, E., S. F. Schifano, and R. Tripiccione (2015). “Energy-performance tradeoffs for HPC applications on low power processors”. In: *European Conference on Parallel Processing*. Springer, pp. 737–748 (cit. on p. 7).

Camacho, E. F. and C. B. Alba (2007). *Model predictive control*. London: Springer-Verlag (cit. on p. 26).

Chowdhury, P. and C. Chakrabarti (2005). “Static task-scheduling algorithms for battery-powered DVS systems”. In: *IEEE transactions on very large scale integration (VLSI) systems* 13.2, pp. 226–237 (cit. on p. 6).

Dressler, F. and G. Fuchs (2005). “Energy-aware operation and task allocation of autonomous robots”. In: *Proceedings of the Fifth International Workshop on Robot Motion and Control, 2005. RoMoCo’05.* IEEE, pp. 163–168 (cit. on p. 7).

García-Martín, E., C. F. Rodrigues, G. Riley, and H. Grahn (2019). “Estimation of energy consumption in machine learning”. In: *Journal of Parallel and Distributed Computing* 134, pp. 75–88 (cit. on p. 6).

Goraczko, M., J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao (2008). “Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems”. In: *2008 45th ACM/IEEE Design Automation Conference*. IEEE, pp. 191–196 (cit. on p. 6).

Hasan, A., M. Skriver, and T. A. Johansen (2018). “Exogenous kalman filter for state-of-charge estimation in lithium-ion batteries”. In: *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, pp. 1403–1408 (cit. on p. 6).

Hong, I., D. Kirovski, G. Qu, M. Potkonjak, and S. M. B (1999). “Power optimization of variable-voltage core-based systems”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18.12, pp. 1702–1714 (cit. on p. 6).

Hong, S. and H. Kim (2010). “An integrated GPU power and performance model”. In: *ACM SIGARCH Computer Architecture News*. Vol. 38. 3. ACM, pp. 280–289 (cit. on p. 6).

Kanellakis, C. and G. Nikolakopoulos (2017). “Survey on computer vision for UAVs: Current developments and trends”. In: *Journal of Intelligent & Robotic Systems* 87.1, pp. 141–168 (cit. on p. 7).

Kirk, D. E. (2004). *Optimal control theory: an introduction*. Courier Corporation (cit. on pp. 23, 24, 26).

Kreciglowa, N., K. Karydis, and V. Kumar (2017). “Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation”. In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 656–662 (cit. on p. 7).

Kuo, B. (1967). *Automatic Control Systems*. Electrical engineering series. Prentice-Hall (cit. on p. 12).

Kwon, W. H. and S. H. Han (2005). *Receding horizon control: model predictive control for state models*. London: Springer-Verlag (cit. on p. 26).

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press (cit. on p. 25).

Luo, J. and N. K. Jha (2001). “Battery-aware static scheduling for distributed real-time embedded systems”. In: *Proceedings of the 38th annual Design Automation Conference*. ACM, pp. 444–449 (cit. on p. 6).

Marowka, A. (2017). “Energy-aware modeling of scaled heterogeneous systems”. In: *International Journal of Parallel Programming* 45.5, pp. 1026–1045 (cit. on p. 5).

Mei, Y., Y.-H. Lu, Y. C. Hu, and C. G. Lee (2004). “Energy-efficient motion planning for mobile robots”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*. Vol. 5. IEEE, pp. 4344–4349 (cit. on p. 6).

— (2005). “A case study of mobile robot’s energy consumption and conservation techniques”. In: *ICAR’05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE, pp. 492–497 (cit. on p. 6).

Moler, C. and C. Van Loan (2003). “Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later”. In: *SIAM review* 45.1, pp. 3–49 (cit. on p. 12).

Nikov, K., J. L. Nunez-Yanez, and M. Horsnell (2015). “Evaluation of hybrid run-time power models for the ARM big. LITTLE architecture”. In: *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing*. IEEE, pp. 205–210 (cit. on p. 7).

Nocedal, J. and S. Wright (2006). *Numerical optimization*. Springer Science & Business Media (cit. on p. 23).

Nunez-Yanez, J. and G. Lore (2013). “Enabling accurate modeling of power and energy consumption in an ARM-based System-on-Chip”. In: *Microprocessors and Microsystems* 37.3, pp. 319–332 (cit. on p. 7).

Ogata, K. (2002). *Modern Control Engineering*. Prentice Hall (cit. on p. 12).

Paulen, R. and M. Fikar (2016). “Solution of Optimal Control Problems”. In: *Optimal Operation of Batch Membrane Processes*. Cham: Springer International Publishing, pp. 37–56 (cit. on p. 25).

Poe, W. A. and S. Mokhatab (2017). “Process Control”. In: *Modeling, Control, and Optimization of Natural Gas Processing Plants*. Ed. by W. A. Poe and S. Mokhatab. Boston: Gulf Professional Publishing, pp. 97–172 (cit. on p. 25).

Pontryagin, L. S., E. Mishchenko, V. Boltyanskii, and R. Gamkrelidze (1962). *The mathematical theory of optimal processes*. Wiley (cit. on p. 25).

Public Deliverables of the TeamPlay Horizon2020 Project (2019). `https://teamplay-h2020.eu/index.php?page=deliverables`. Accessed: 2019-08-25 (cit. on p. 8).

Quigley, M., K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng (2009). “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2, p. 5 (cit. on p. 17).

Rao, R., S. Vrudhula, and D. N. Rakhmatov (2003). “Battery modeling for energy aware system design”. In: *Computer* 36.12, pp. 77–87 (cit. on p. 5).

Rao, S. S. (2019). *Engineering optimization: theory and practice*. John Wiley & Sons (cit. on p. 26).

Rawlings, J. B., D. Q. Mayne, and M. Diehl (2017). *Model predictive control: theory, computation, and design*. Vol. 2. Madison, Wisconsin: Nob Hill Publishing (cit. on pp. 23–26).

Rossiter, J. A. (2004). *Model-based predictive control: a practical approach*. Boca Raton, Florida: CRC press (cit. on p. 26).

Sadrpour, A., J. Jin, and A. G. Ulsoy (2013a). “Mission Energy Prediction for Unmanned Ground Vehicles Using Real-time Measurements and Prior Knowledge”. In: *Journal of Field Robotics* 30.3, pp. 399–414 (cit. on p. 7).

Sadrpour, A., J. Jin, and A. G. Ulsoy (2013b). “Experimental validation of mission energy prediction model for unmanned ground vehicles”. In: *2013 American Control Conference*. IEEE, pp. 5960–5965 (cit. on p. 7).

Seewald, A., E. Ebeid, and U. P. Schultz (2019). “Dynamic Energy Modelling for SoC Boards: Initial Experiments”. In: *HLPGPU 2019: High-Level Programming for Heterogeneous and Hierarchical Parallel Systems*, p. 4 (cit. on p. 7).

Seewald, A., H. Garcia de Marina, H. S. Midtiby, and U. P. Schultz (2020). “Mechanical and Computational Energy Estimation of a Fixed-Wing Drone”. In: *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. IEEE, pp. 135–142 (cit. on p. 17).

Seewald, A., H. Garcia de Marina, and U. P. Schultz (n.d.). *Energy-Aware Dynamic Planning Algorithm for Autonomous UAVs*. In preparation (available online: `https://github.com/adamseew/iros-2021`) (cit. on p. 9).

Seewald, A., U. P. Schultz, E. Ebeid, and H. S. Midtiby (2019). “Coarse-Grained Computation-Oriented Energy Modeling for Heterogeneous Parallel Embedded Systems”. In: *International Journal of Parallel Programming*, pp. 1–22 (cit. on pp. 5, 17).

Seewald, A., U. P. Schultz, J. Roeder, B. Rouxel, and C. Grelck (2019). “Component-based computation-energy modeling for embedded systems”. In: *Proceedings Companion of the 2019 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*. ACM, pp. 5–6 (cit. on pp. 17, 18).

Sethi, S. P. (2019). *Optimal Control Theory: Applications to Management Science and Economics*. Cham: Springer International Publishing (cit. on pp. 23–26).

Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons (cit. on p. 26).

Stengel, R. F. (1994). *Optimal control and estimation*. Courier Corporation (cit. on p. 26).

Uragun, B. (2011). “Energy efficiency for unmanned aerial vehicles”. In: *2011 10th International Conference on Machine Learning and Applications and Workshops*. Vol. 2. IEEE, pp. 316–320 (cit. on p. 7).

Von Stryk, O. and R. Bulirsch (1992). “Direct and indirect methods for trajectory optimization”. In: *Annals of operations research* 37.1, pp. 357–373 (cit. on p. 23).

Wang, L. (2009). *Model predictive control system design and implementation using Matlab*. London: Springer Science & Business Media (cit. on p. 26).

Wu, G., J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou (2015). “GPGPU performance and power estimation using machine learning”. In: *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, pp. 564–576 (cit. on p. 6).

Zamanakos, G., A. Seewald, H. S. Midtiby, and U. P. Schultz (2020). “Energy-Aware Design of Vision-Based Autonomous Tracking and Landing of a UAV”. In: *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. IEEE, pp. 294–297 (cit. on p. 18).