Imperial College
London

Numerical Optimal Control with Applications in Aerospace

Yuanbo Nie

January 3, 2021

Supervisors: Dr Eric Kerrigan
Prof Rafael Palacios

Department of Aeronautics
Imperial College London

This thesis is submitted for the degree of
Doctor of Philosophy of Imperial College London

Declaration of originality

The work presented hereafter is based on research carried out by the author at the Imperial College London and it is all the author's own work under supervisors' supervision, except where otherwise acknowledged. Reuse of author's own published works during the PhD degree program are also acknowledged according to publishers' guidelines.

Yuanbo Nie

London, September 2020

Copyright declaration

Abstract

This thesis explores various computational aspects of solving nonlinear, continuous-time dynamic optimization problems (DOPs) numerically. Firstly, a direct transcription method for solving DOPs is proposed, named the integrated residual method (IRM). Instead of forcing the dynamic constraints to be satisfied only at a selected number of points as in direct collocation, this new approach alternates between minimizing and constraining the squared norm of the dynamic constraint residuals integrated along the whole solution trajectories. The method is capable of obtaining solutions of higher accuracy for the same mesh compared to direct collocation methods, enabling a flexible trade-off between solution accuracy and optimality, and providing reliable solutions for challenging problems, including those with singular arcs and high-index differential-algebraic equations.

A number of techniques have also been proposed in this work for efficient numerical solution of large scale and challenging DOPs. A general approach for direct implementation of rate constraints on the discretization mesh is proposed. Unlike conventional approaches that may lead to singular control arcs, the solution of this on-mesh implementation has better numerical properties, while achieving computational speedups. Another development is related to the handling of inactive constraints, which do not contribute to the solution of DOPs, but increase the problem size and burden the numerical computations. A strategy to systematically remove the inactive and redundant constraints under a mesh refinement framework is proposed.

The last part of this work focuses on the use of DOPs in aerospace applications, with a number of topics studied. Using example scenarios of intercontinental flights, the benefits of formulating DOPs directly according to problem specifications are demonstrated, with notable savings in fuel usage. The numerical challenges with direct collocation are also identified, with the IRM obtaining solutions of higher accuracy, and at the same time suppressing the singular arc fluctuations.

To my family, friends and all the people I love.

Acknowledgement

First and foremost, I would like to thank my supervisor, Dr Eric Kerrigan, for the continuous support throughout this research program. I have enjoyed every moment of the more than 500 hours of discussions we had. Your encouragement, guidance, friendship, patience, and professionalism have all been extremely beneficial for me in the tackling of various challenges.

I would like to express my gratitude to Prof Rafael Palacios for the generous support he had offered. I am indebted to him for allowing me to (temporally) leave aerospace applications and have four years of full dedication to tackling dynamic optimization related challenges. This work may only be a small step forward, but I am happy that we are one inch closer to having efficient and reliable methods and tools to address the challenges in aerospace engineering.

Next, I would like to thank all members of the Department of Aeronautics for the memorable journey at Imperial College London. Special thanks to Dr Andrew Wynn, Dr George Papadakis and Dr Thulasi Mylvaganam for providing the opportunity for me to be involved in the development of new lab sessions and online coursework for control-related modules at the department. It was a great pleasure to assist students' understanding of the theories through practical experiences. I have also had the honour to be one of the PhD representatives and would like to thank Prof Paul Robinson, Dr Peter Vincent, Dr Paul Bruce, Lisa Kelly and Molly Ip for their support. During the last 4 years, I have co-supervised 16 Master's projects, and I would like to thank every student for their dedication and friendship.

I would also like to extend my thanks to all my colleagues from the Department of Electrical and Electronic Engineering, who have provided great ideas through multiple discussions. Hereby, I would like to thank Omar Faqir, Ian McInerney, Dr Bulat Khusainov, Dr Andrea Picciau, Dr Luxin Zhang, Harsh Shukla, Dr Paola Falugi, Dr Ercan Atam, Dr Marta Zagorowska and Lucian Nita. Special thanks go to Martin Neuenhofen, who has introduced me to the integrated residual methods and inspired many of the related developments in this work.

On this occasion, I would also like to thank Prof Max Mulder, Dr Ping Chu and Dr Erik-Jan van Kampen from TU Delft, and Dr Gertjan Looye and Thiemo Kier from DLR for introducing me to the fascinating world of research in aircraft flight control systems design, and for inspiring and encouraging me to set sail to search for the methods and tools to address the aircraft upset recovery challenge, a major motivation for this research work.

Finally, I would also like to thank my family and friends in China, the Netherlands, Germany and the UK. I have had the pleasure to acquire may friends who always encouraged me to keep going despite any difficulties. Among these wonderful people, special thanks to Filipe Buscariolo, Omar Mahfoze, Karim Shawki, and Xingchen Zhou.

I am deeply grateful for my parents' love, support, and encouragement in my pursuit of aerospace engineering since I was a young boy. Also, I would like to thank my Dutch family and my parents-in-law for all the support and love, as well as your understanding for the limited time I had available in the course of this project. Special thanks go to my wife Yi Li, for moving to the UK to join me. Together we welcomed the arrival of my son Shenghang to the world on the last day of 2019.

Thanks to all of you, the last four years have been an incredible journey!

Yuanbo Nie

London, September 2020

Preface

When I started the PhD at Imperial, Eric's senior student, Bulat, was in the process of finishing his PhD. He had dedicated a major part of his PhD to linear MPC before moving to its nonlinear counterpart. I still remember clearly the advice he offered to me: *"do not waste your time on the linear stuff, go directly for the NMPC"*.

The four-year journey seems long at the start but certainly feels much shorter now that it is approaching the finishing line. Dynamic optimization is such a fascinating subject that connects so many different fields of study, that every time I learnt about a new subject, I ended up with ten more areas of unknowns and many ideas to explore.

However, the biggest obstacle to any research may not necessarily be the technical challenge, but rather the strong inertia to keep taking the same approach that one is familiar with, and solve the problems in the way one knows how to solve. I always felt that if I have started on the integrated residual methods a few months earlier, the method, as well as the implementation, would be more mature than what it is now presented. Having said that, I do feel relieved that I am now able to proudly offer my advice to fellow PhD students: *do not waste your time on collocation, go directly for the integrated residual methods.*

Also, I would like to emphasize that this thesis does not represent an aerospace engineer's departure from his original field of interests and moving on to control engineering. Instead, it symbolizes the efforts an aerospace engineer is willing to take when current technologies and tools are no longer able to address the challenges out-of-the-box. This determination is a key driving force for me to navigate through unfamiliar subjects in order to eventually arrive at a future that non-intuitive and non-trivial solutions to challenging aerospace problems can be obtained efficiently and reliably. I hope the work presented in this thesis is a solid step forward.

Yuanbo Nie

London, September 2020

Contents

List of Figures

List of Tables

List of Symbols

Abbreviations

AFP	Auxiliary Feasibility Problem
AWE	Airborne Wind Energy
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BVP	Boundary Value Problem
DAE	Differential-Algebraic Equations
DAIR	Direct Alternating Integrated Residual
DDOP	Discretized Dynamic Optimization Problem
DFO	Derivative Free Optimization
DOP	Dynamic Optimization Problem
FBL	Feedback Linearization
FDI	Fault Detection and Isolation
FPGA	Field Programmable Gate Arrays
HJB	Hamilton-Jacobi-Bellman
HS	Hermite-Simpson
IPM	Interior Point Method
IRM	Integrated Residual Method
IRNS	Integrated Residual Norm Squared (Error)
IRS	Integrated Residual Squared (Error)
KKT	Karush-Kuhn-Tucker
LG	Legendre-Gauss
LGL	Legendre-Gauss-Lobatto
LGR	Legendre-Gauss-Radau
LICQ	Linear Independence Constraint Qualification
LP	Linear Programming
LPV	Linear Parameter Varying
LQR	Linear Quadratic Regulator
MIRNS	Mean Integrated Residual Norm Squared (Error)
MIRS	Mean Integrated Residual Squared (Error)
MPC	Model Predictive Control
MR	Mesh Refinement
NDI	Nonlinear Dynamic Inversion
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problems
ODE	Ordinary Differential Equation
OEP	Optimal Estimation Problems
PBF	Penalty Barrier Finite Element Method
PMP	Pontryagin's Maximum Principle

QP	Quadratic Programming
RFCS	Reconfigurable Flight Control System
RHC	Receding Horizon Control
RK	Runge-Kutta
RTI	Real-Time Iterations
SEIR	The Susceptible - Exposed - Infectious - Recovered Model
SHC	Shrinking Horizon Control
SIR	The Susceptible - Infectious - Recovered Model
SOP	Static Optimization Problem
SQP	Sequential Quadratic Programming
UAV	Unmanned Aerial Vehicle
VHC	Variable Horizon Control

Subscripts

$_i$	Variables corresponds to point i
$_l$	The l^{th} inequality path constraint of the DOP
$_L$	The lower bound
$_{tol}$	The error tolerance
$_U$	The upper bound
$_\xi$	The ξ^{th} dynamic constraint of the DOP

Superscripts

*	Corresponding to the optimal solution
$^{(k)}$	Variables corresponding to mesh interval k
$'$	The numerical derivatives w.r.t time

Symbols (Greek)

β_i	Basis functions for the approximation functions
χ	Parameterized states
$\hat{\chi}$	Values of state variables on data point of the refined mesh, evaluated from approximate solutions with the previous mesh design
Δt	Time differences between t_0 and t_f
$\Delta t^{(k)}$	Time differences between mesh nodes of interval k
δ	Dirac delta function
$\digamma_i$	Evaluation of dynamic equation at data point i
ϵ	Absolute local constraint violation error
η	Absolute local error
ε	Dynamic equation residuals
ε_J	A small number prescribing the acceptable increase in objective value, for posterior regularization
Λ	Multipliers corresponding to discretized constraint equations
λ	Costate / adjoint corresponding to dynamic constraints
μ	Costate / adjoint corresponding to inequality path constraints
ν	Multiplier / adjoint corresponding to boundary constraints

γ	Inequality constraints in DDOP
Φ	Mayer cost
ϕ	Boundary constraints
Π	Interval buffer to ensure correct identification of potentially active constraints
ψ	Equality constraints in DDOP
ϖ	Weighting function in weighted residual methods
ϱ	MIRS error
τ_i	Time corresponding to i^{th} data point
Θ	Threshold parameter for normalized multipliers, to determine active status
ϑ	The ratio between regularization terms and the original objective, for simultaneous regularization
$\hat{\upsilon}$	Values of free variables on data point of the refined mesh, evaluated from approximate solutions with the previous mesh design
υ	Parameterized free variables
$\hat{\Xi}$	Absolute local constraint violation error computed at the grid points of the refined mesh
Ξ	Slack variable for the AFP
ζ	Absolute local error for individual dynamic equations

Symbols (Roman)

$\mathcal{A}$	A discretization-dependent constant matrix
a_i	Coefficient for the approximation function
$\mathcal{C}$	Full set of discretized constraint equations
c	Inequality path constraints
$\mathcal{D}$	A matrix containing time variables
d	A small perturbation
f	System dynamics in the form of ODEs
g	System dynamics in the form of DAEs
K	Number of mesh intervals
k	Index for intervals
$\mathcal{H}$	Hamiltonian
J	Objective
$\hat{K}$	Number of intervals of the refined mesh
$\mathcal{L}$	Lagrangian
l	Lagrange cost
M	Number of intervals to be subdivided due to large absolute local error in each mesh refinement iteration
m	Number of free variables in DOP
M_{max}	Maximum number of intervals allowed to be subdivided due to large absolute local error in each mesh refinement iteration
$\hat{N}$	Number of grid points of the refined mesh
$\hat{N}^{(k)}$	Number of data points in interval k of the refined mesh

N	Number of grid points
n	Number of states in DOP
$N^{(k)}$	Number of data points in interval k
n_c	Number of inequality path constraints in DOP
n_g	Number of DAEs in DOP
n_p	Number of static parameters in DOP
n_q	Number of boundary constraints in DOP
n_z	Number of decision variables
$n_{\mathcal{C}}$	Number of discretized constraint equations
p	Static parameters
$\bar{Q}$	Weighting matrix for states in quadratic regulation cost formulation
$\bar{Q}_f$	Weighting matrix for terminal state in quadratic regulation cost formulation
$Q^{(k)}$	Number of quadrature points in interval k
q_i	Time corresponding to i^{th} quadrature point
$\bar{R}$	Weighting matrix for inputs in quadratic regulation cost formulation
$\mathcal{R}$	IRNS error computed using higher order quadrature rules
r	IRNS error
$\bar{S}$	Weighting matrix for cross-terms in quadratic regulation cost formulation
s_k	Normalized time corresponding to mesh node k
ρ	The penalty weight for simultaneous regularization
t	Time
t_0	Initial time
t_f	Terminal time
t_k	Time corresponding to mesh node k
$\tilde{u}$	Approximation of the trajectory of free variables
u	Trajectory of free variables
$\mathcal{W}$	A diagonal matrix for the weighting of residuals of different dynamic constraints
w	Interval-dependent quadrature weights
$\dot{\tilde{x}}$	Approximation of the state derivative trajectory
$\tilde{x}$	Approximation of the state trajectory
x	Continuous state trajectory
x_0	The initial condition for state variables
$\hat{\mathcal{Z}}$	Interpolated solution on the refined mesh
$\mathcal{Z}$	Tuple of optimization decision variables for the discretized DOP
$\mathcal{Z}_:$	DDOP decision variable vector
$\tilde{z}$	Tuple of approximate solution to DOP
z	Tuple of optimization decision variables for the continuous-time DOP

Part I

Introduction and Background Information

Chapter 1

Introduction

An optimization problem describes the problem of searching for the best solution from all feasible ones. A *feasible solution* is a set of values of the *decision variables* that satisfies all of the constraints in the problem formulation, with *constraints* being a set of conditions that the solutions must fulfil. The solution, according to a certain objective (also known as cost) reaching its minimum or maximum value, is known as an *optimal solution.*

In the last decades, the search for the optimal solution has been integral to many aspects of science, engineering and economics, and profoundly changed the way how challenges are approached and addressed in related fields. Generally speaking, the optimization problems can be categorized as *static optimization problems* (SOPs) and *dynamic optimization problems* (DOPs), depending on the nature of the decision variables.

With discrete decision variables, the finite-dimensional optimization problem is an SOP with the objective being a function of these decision variables. In DOPs, some of the decision variables can be continuous trajectories expressed in the form of functions, hence the objective and constraints become functionals and the optimization problems become infinite-dimensional. Often, the dynamic decision variables in DOPs are functions of time, leading to variants of DOPs such as *optimal control problems* (OCPs), parameter estimation problems, as well as design synthesis problems where dynamic models are employed.

Continuous-time DOPs are typically expressed in the Bolza form, with,

$$\min_{x,u,p,t_0,t_f} \Phi(x(t_0), t_0, x(t_f), t_f, p) + \int_{t_0}^{t_f} L(x(t), u(t), t, p)\ dt \tag{1-1a}$$

subject to

$$\dot{x}(t) = f(x(t), u(t), t, p),\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{1-1b}$$

$$g(x(t), \dot{x}(t), u(t), t, p) = 0,\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{1-1c}$$

$$c(x(t), \dot{x}(t), u(t), t, p) \leq 0,\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{1-1d}$$

$$\phi(x(t_0), t_0, x(t_f), t_f, p) = 0, \tag{1-1e}$$

where $x : \mathbb{R} \to \mathbb{R}^n$ is the continuous state trajectory of the system, $u : \mathbb{R} \to \mathbb{R}^m$ is the trajectory of free variables (typically the control inputs, disturbances and noise), $p \in \mathbb{R}^{n_p}$ are some static parameters, $t_0 \in \mathbb{R}$ and $t_f \in \mathbb{R}$ are the initial and final time. Φ is the Mayer cost functional Φ: $\mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}$, and L is the Lagrange cost functional $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}$. The objective functional (1-1a) is often denoted by a single variable J with optimal solution denoted as J^*. The superscript * will be consistently used in this work to refer to a variable corresponding to the optimal solution.

The system dynamics are enforced as equality constraints, in the form of ordinary differential equations (ODEs) with $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^n$ and differential-algebraic equations (DAEs) with $g : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g}$. Equations (1-1b) and (1-1c) can be referred together as the dynamic equations. c is the inequality path constraint ($c : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$), and ϕ is the boundary condition ($\phi : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_q}$).

For the numerical solution of DOPs, *transcription methods* are often employed. The process takes the vast solution space of DOPs (infinite-dimensional) and defines low dimensional manifolds where approximated problems can be expressed as *discretized dynamic optimization problems* (DDOPs) and solved efficiently. This is the main focus of this thesis; however, in this introductory chapter, the role of DOPs in control systems design will be first discussed in Section 1-1 to help motivate this research, highlighting the importance to formulate DOPs according to problem specifications. The research objectives are presented in Section 1-2. Section 1-3 details on the organization of the thesis and Section 1-4 explains the computation environments and the notation conventions for the example problems demonstrated in this work.

1-1 DOPs in control systems design

1-1-1 Conventional implementation: optimal guidance, navigation and control

The diagram in Figure 1-1 is commonly regarded as the overall architecture of control system design. Conventional optimal control and estimation methods follow this system structure, leading to the fields of optimal guidance, optimal control (in a narrow sense of regulation) and

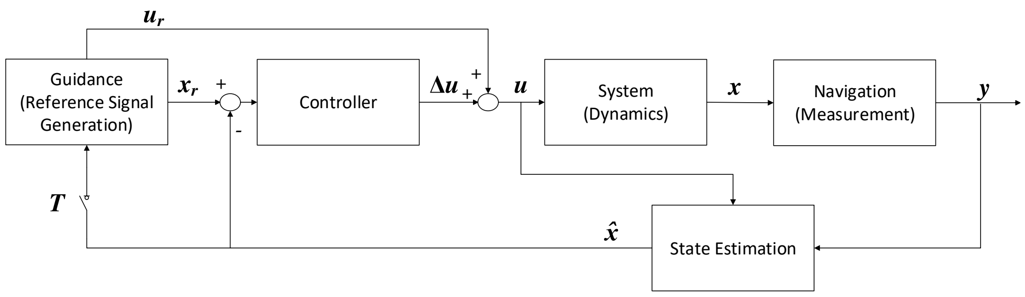


Figure 1-1: Overview of a control system, with y the output variable, $\hat{x}$ the estimated state, x_r the state reference, u_r the input reference, Δu the input increment, T the trigger for the regeneration of reference trajectory

optimal estimation. It is possible to design each of these components as a DOP, i.e. solving three DOPs for their respective purposes, or use a combination of different methods (e.g. use optimal guidance together with other schemes for navigation and control). This section focuses on the role of optimal guidance and optimal control in such a closed-loop setup.

Optimal control

The terminology of 'control' has the narrow sense of regulation, hence is often (inadvertently) used to infer stabilization or set-point tracking tasks. As a result, in traditional optimal control studies, people often associate optimal control with the classical formulation of quadratic regularization. For example, in linear quadratic regulator (LQR) design with state-feedback for stabilization and disturbance rejection, the following infinite horizon quadratic cost is minimized:

$$J = \int_{t_0}^{\infty} x(t)^\top \bar{Q} x(t) + u(t)^\top \bar{R} u(t) + 2x(t)^\top \bar{S} u(t)\, dt.$$

$\bar{Q} \succeq 0 \in \mathbb{R}^{n\times n}$, $\bar{R} \succ 0 \in \mathbb{R}^{m\times m}$ and $\bar{S} \in \mathbb{R}^{m\times n}$ are weighting matrices addressing the relative trade-offs between different variables and different terms. Another example would be in model predictive control (MPC) where set-point tracking tasks require the minimization of a finite horizon quadratic cost (with an optional terminal cost on $x(t_f)$)

$$J = \int_{t_0}^{t_f} (x(t) - x_r(t))^\top \bar{Q} (x(t) - x_r(t)) + u(t)^\top \bar{R} u(t)\, dt + x(t_f)^\top \bar{Q}_f x(t_f),$$

with $x_r \in \mathbb{R}^n$ a reference trajectory to be tracked, and $\bar{Q}_f \succeq 0 \in \mathbb{R}^{n\times n}$ the weighting matrix for the terminal state cost.

Optimal guidance

Dynamic optimization for non-regulation type tasks has existed for decades under the framework of *optimal guidance* or *trajectory optimization.* Examples include the design of spacecraft orbit transfer, swingby and reentry trajectories, as well as minimum time or energy flight profiles for various types of aircraft. A good collection of optimal guidance problems in available in [19], and for certain examples, additional insights [17] are provided.

Partially due to computational limitations, the optimal guidance solutions are mostly calculated offline. Often the solutions to the DOPs are implemented as suitable reference signals, x_r or u_r, for lower-level controllers to track. For many applications, an open-loop implementation of this guidance sequence would be sufficient. However, there also exist many applications requiring frequent updates to the guidance solutions due to uncertainties in the form of unmodelled dynamics and external disturbances.

Another option to handle uncertainty is to implement optimal guidance solutions as feedback policies. For sufficiently simplified problems, it is possible to analytically derive the optimality conditions (e.g. the turning and coasting flight trajectories [200]), and thereafter, feedback policies in the form of *guidance laws* can be designed and implemented with minimal online computation overhead. However, for more complicated problems, numerical schemes may be needed to find the DOP solutions. In these cases, based on the characteristics of the obtained solutions, usually simplified guidance laws that can mimic similar solution behaviours are designed for online implementation.

One example would be the supersonic aircraft minimum-time-to climb problem [131, 145], with the optimal climb profile indicated in Figure 1-2 simplified into the dashed lines representing the following five segments, which are suitable to design simplified guidance laws accordingly:

1. an acceleration reaching high-subsonic speeds,
2. a constant airspeed or Mach number (depending on flight regime) climb,
3. a zoom dive: exchange altitude for speed, accelerating along the constant specific energy line until reaching maximum specific excess power,
4. a climb along the maximum specific excess power line,
5. a zoom climb: exchange speed for altitude, decelerating along the constant specific energy line until reaching the final target flight conditions.

The simplifications would lead to trajectories that no longer coincide with the obtained DOP solution. For example, the 5-segment approximation of the minimum-time-to-climb problem has resulted in about 36% increase in the time needed to climb to a target altitude of 65600 ft

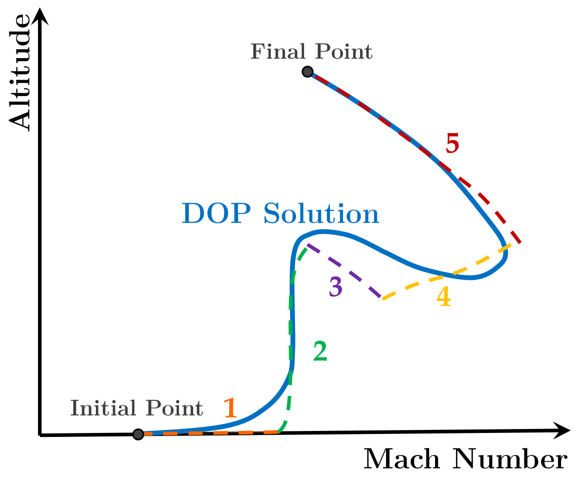


Figure 1-2: Comparison of exact and energy-state minimum time-to-climb paths, adapted from [33]

[33]. Therefore, degraded strategies of this type should strictly be called *efficient guidance* instead of optimal guidance.

With additional information from the solution process, the numerical DOP solutions can also be implemented as feedback policies by solving a simplified companion optimal control problem of lower complexity. Commonly used methods are the extremal field approach, dynamic programming, and the neighbouring extremals approach with perturbation feedback control [32].

1-1-2 Issues with the conventional approach

When control systems are designed and implemented for real-world applications, there are always the ultimate mission objective and design specifications to fulfil: passenger airlines should bring the passenger from the origin to destination on time using minimum fuel, and racing cars should complete all the laps in minimum time. When the regulation type of controller designs are included in the loop, the way these controllers handle deviations to the trajectory (e.g. due to disturbances) will lead to an inherent loss of optimality. In addition, in spite of the name “optimal control”, methods that use the classical quadratic regulation formulation can rarely achieve the optimal performance according to specifications of a tracking task.

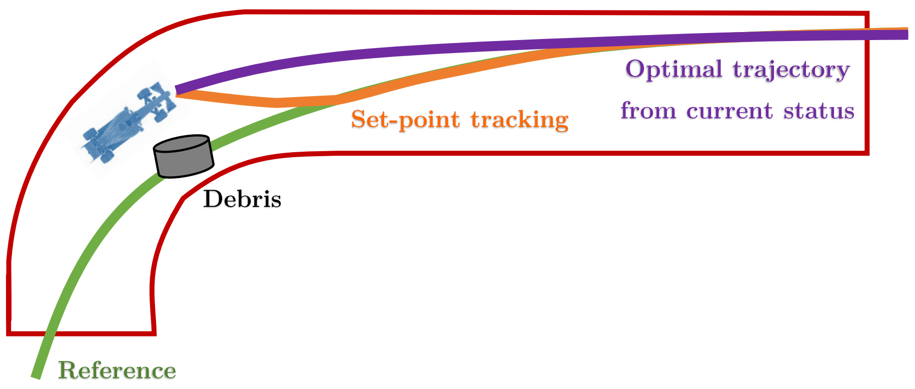


Figure 1-3: Illustration for a racing car at a turn

Loss of optimality with set-point tracking of a pre-computed reference

With regulation control, all deviations from the target are treated equally; however, their contribution to the overall task completion (objective) may be different. This can be demonstrated with a simple example of a racing car going through the turns, in Figure 1-3. Suppose that the vehicle uses a set-point tracking controller to track the reference trajectory, generated beforehand using optimal guidance. Upon entry at one of the corners, some debris blocks the way, so the car has to enter the turn wider. Upon passing the obstacle, a set-point based strategy would be to aggressively return to the originally planned path, in the effort to reduce deviations from the reference.

If a DOP is formulated directly corresponding to the mission specifications and solved online, the solution may take advantage of the fact that the car is at a wider position and likely slower condition, hence can generate the optimal exit strategy accordingly. In short, even if the reference profile is optimal at the time of computation, due to uncertainties, the real-time execution of it with regulation controllers will not necessarily be.

Issues of quadratic cost formulation

For regulation tasks such as disturbance rejection and set-point tracking, an optimal control method using classical quadratic regulation cost formulations will not necessarily result in better control efficiency or performance in comparison to other control strategies that are not optimization-based. The problem lies fundamentally in the objective formulation.

Firstly, the square of the input variable $u^2(t)$ does not represent energy in the majority of applications, and there are two aspects of this argument:

- The idea that $u^2(t)$ needs to be penalized for efficiency may be more valid for some applications than others. For example, in chemical engineering, $u(t)$ may represent the use of expensive chemical products. However, in aerospace flight controls applications, the intention is not really to penalize flight control surface movements, as long as they are inside the specified allowances.
- A solution that minimizes the integral of $u^2(t)$ is not the same as a solution that minimizes energy consumption. Consider another aerospace example, the study in Section 14-2-1 demonstrates that the solutions which minimize the integral of the square of instantaneous fuel flow, thrust and throttle settings all have led to higher fuel usage, as much as 4.5% (more than 5000 kg, sufficient for a Boeing 737 to complete a 1.5 hr flight) for a single transcontinental flight in comparison to the solution that directly minimizes the reduction in aircraft weight due to fuel consumption.

Therefore, if the objective formulation needs to embed a trade-off between the performance metric and the energy usage, the actual formulation of energy consumption should be used instead of $u^2(t)$.

Some may argue that $u^2(t)$ represents *control effort* hence it has virtue in avoiding control saturation and suppressing undesirable fast oscillations in the input signal. The counter-argument to this claim would be that, instead of achieving these items through a vaguely designed trade-off process, one should formally formulate the actuator saturation limits and enforce rate constraints for variations in input trajectory. Progresses made in suppressing singular arc fluctuations will also help to eliminate the need of adding $u^2(t)$ terms purely to avoid singular arcs. Generally speaking, a singular arc is an interval in the DOP solution where the optimality conditions yield no information about the optimal control trajectory. Further details are provided in Section 2-3-5 and Chapter 11.

Next, the focus can be shifted to the trade-off process embedded in the classical quadratic cost formulation. Firstly, J can be considered as the scalarization of a multi-objective optimization problem using the weighting method: the objectives are combined into a weighted sum. Secondly, for individual terms, the weights ($\bar{Q}$, $\bar{R}$, $\bar{S}$, etc.) are all tuning parameters reflecting the preferences of the decision-maker. Therefore, although this formulation has the numerical advantage that single objective optimization techniques can be used for the solution of the corresponding DOPs, the drawback is also remarkable: at both levels of trade-offs, the final solution is heavily influenced by the vaguely designed weight allocation, posing the risk of yielding unpredictable and unacceptable results [194].

In terms of the overall task, the solution to an optimization problem can be only as good as the formulation of the objective function. This can be seen in results presented in Example 2 of [117], for a combined change of altitude and airspeed of an F16 aircraft. The nonlinear

MPC controller achieves lower cost value for the quadratic objective function by "*trading off altitude errors for better velocity tracking and uses large control values*". However, this low value of cost is only with respect to the set of weights that were tuned heuristically, thus do not necessarily reflect the optimality with regard to the task to be completed. In fact, the linear MPC reached the target altitude faster with smaller control inputs.

In a similar scenario, climbing from 2000 ft to 10000 ft or to 35000 ft should ideally lead to the same response for the expedition of the initial climb. But in the quadratic regulation cost function, the trade-off of this altitude tracking error with respect to the tracking of other states (e.g. velocity deviation), input suppression and input rate suppression will all be different for the two cases. These are all potential sources for unexpected results in a very dynamic but safety-critical operating environment. Designing and checking the weights for the entire flight envelope and all possible scenarios can be an expensive and time-consuming task.

1-1-3 Formulating DOPs based on problem specifications and solving directly

To ensure the mission targets can be reached while fully respecting all limits, the formulation of DOPs must be based on the ultimate design specifications of the problem. In other words, it is important to formulate and solve the right problem.

First and foremost, it is important to identify the objectives and constraints in the problem. Most real-world problems have the ultimate goal in certain forms of minimum-time or minimum-energy. If the problem has multiple objectives, they must be prioritized based on the requirement of problem specifications. All conditions that would invalidate a DOP solution must be included as constraints to ensure their satisfaction. Hence for properly formulated DOPs, any local optimum (to a certain extent, even any feasible solution) should be considered as suitable for implementation.

For example, the aircraft climb problem mentioned earlier can be defined as a minimum-time or minimum-fuel problem, subject to constraints regarding flight dynamics, the terminal states (e.g. cruise altitude and velocity), flight envelope (incl. aerodynamics, structures) limits, passenger/crew comfort limits, airspace restrictions, and maximum duration and fuel usage limits.

Despite the benefits, formulating the DOPs directly based on problem specification can sometimes lead to optimization problems that are difficult to solve, making the solution process more challenging. Although reformulation of the problem and some ad hoc fixes can be helpful in overcoming many of the difficulties, these posterior measures can become increasingly difficult to implement for the solution of DOPs online in a closed-loop.

1-2 Research objectives

The main objective of this research is determined to be

To efficiently obtain reliable and highly accurate solutions to dynamic optimization problems that are formulated directly based on specifications of real-world problems, with minimum usage of reformulations and ad hoc fixes to the problem.

The detailed research goals to be accomplished are as follows:

- To develop of a comprehensive toolbox for the solution of DOPs, where different existing methods can be tested for their respective advantages and disadvantages, and developments on new methods can be supported.
- To identify possible improvements to existing numerical methods for solution of DOPs. These include better addressing of challenging problems such as the ones with singular control, high-index DAEs, as well as large-scale problems with high number of inequality constraints implemented based on the problem specifications.
- To understand the reasons behind existing direct transcription methods' shortcomings in dealing with these challenging problems, and explore the possibility to develop new types of direct transcription methods than can address the issues directly.
- To demonstrate the benefits of new developments with established benchmarking example problems as well as challenging problems arising from aerospace applications.

1-3 Organization of the thesis

In Part I of this thesis, Chapter 1 presents the motivation and objective of this work. This is followed by Chapter 2, providing a quick introduction of dynamic optimization and an overview of existing numerical methods for the solution of DOPs, to introduce the terminologies and to provide background information that can support the discussions in later part of this thesis. At the end of this introduction part, Chapter 3 highlights the main contributions of this work.

Part II details on the numerical methods for direct transcription of DOPs, starting with Chapter 4 demonstrating the discretization process of transcribing the infinite-dimensional continuous-time DOPs into finite-dimensional DDOPs, with trajectories being approximated by continues or piecewise continues functions. Chapter 5 highlights the general inevitability of approximation errors and its implications on the way how dynamic constraints can be enforced in the formulations of DDOPs. Through a simple example, the error characteristics of different weighted residual methods are compared focusing on the shortcomings of the collocation approach. Following this, the direct collocation method is introduced in Chapter 6 explaining the necessity of a mesh refinement (MR) framework. This is followed by Chapter 7, where the concept of integrated residual method (IRM) is first introduced with insights on its connection to the direct collocation approach. Then the motivations for the development of the direct alternating integrated residuals (DAIR) scheme are given, together with discussions on the formulations and implementation strategies.

Part III discusses various aspects of solving large-scale problems efficiently. In Chapter 8, the sparsity pattern for different direct transcriptions methods are presented. This is followed by a discussion on the computation of derivative information, demonstrating why sparse finite difference would not be efficient for IRM-type of direct transcriptions methods. Chapter 9 presents a way rate constraints can be implemented in DDOPs that avoids the introduction of singular arc, and obtains faster computations at the same time. Then in Chapter 10, an external constraint handling scheme is demonstrated that can systematically exclude inactive constraints in DDOP formulations, reducing the problem size and computational time under mesh refinement frameworks. Next, Chapter 11 compares different methods for the suppression of singular arc fluctuations. Lastly, an overview of DOP toolbox ICLOCS2 is provided in Chapter 12, together with a number of software highlights and example problems.

The last part of this thesis focuses on the use of DOPs for aerospace applications. Chapter 13 includes a quick review of the history of solving DOPs for flight and orbital trajectory optimization, as well as for flight control applications in the form of MPC. Then in Chapter 14, high fidelity modelling of system dynamics for flight trajectory optimization of commercial airliners are presented, followed by discussions on different formulations of the objective, suppression of singular arcs in the solutions, and different multi-phase setups. Next, a formation flight problem is demonstrated in Chapter 15, based on a simulated scenario of joint transcontinental flights. The thesis closes with conclusions, discussions and suggestions for future work in Chapter 16.

1-4 Example problems in this thesis

Unless mentioned otherwise, all problems are transcribed using the toolbox ICLOCS2 [159] developed in this project, and solved with the interior point NLP solver IPOPT [206] with the sparse linear solver MA57 [52] to a relative convergence tolerance (`tol`) of 10^{-9}. In ICLOCS2, both state and input trajectories are continuous trajectories if the problem is formulated using a single phase, but allowed to be discontinuous with a multi-phase setup at phase interfaces.

All computational results, except ones in Chapter 9 and 10, are obtained with ICLOCS2 version 2.5 on an AMD Ryzen 5-3600 computer with 32 GB of RAM, running 64-bit Windows 10 with MATLAB version 2019a. Computational results in Chapter 9 and 10 are obtained with ICLOCS2 version 2 on an Intel i7-6700 computer with 16 GB of RAM, running 64-bit Windows 10 with MATLAB version 2017a.

To avoid confusions and excessive definition of symbols, the symbols from the original reference or the convention of the respective application fields will be used for variables in different example problems. Their definition is contained in the respective (sub)sections or chapters (for Chapter 14 and 15) where they are presented. Except for these cases, all definition of symbols for variables are consistent and valid throughout this thesis.

Chapter 2

Overview of Dynamic Optimization Problems

This chapter aims to provide the necessary background information for supporting the discussions in later part of this thesis. The characteristics of DOPs in terms of optimality conditions will be first presented and addressed in Section 2-1. This is followed by Section 2-2 with an overview on MPC, and Section 2-3 reviewing existing numerical methods and software toolboxes for the solution of DOPs.

2-1 Optimality conditions

Provided that a solution is found corresponding to an optimum point of the DOP, it is possible to analyze the properties of the solution at that point. *Optimality conditions* describe the necessary conditions that the solution must satisfy at the optimum point. In most contexts, the phrase *optimality condition* does not necessarily refer to a sufficient condition. Points that satisfy the necessary optimality conditions are often named as *candidate optimum points*, requiring additional tests to identify the actual optimality.

If the optimal solution is better than all other feasible solutions, then it is said to be a *global optimum*, otherwise it is a *local optimum*. When the term *optimality condition* is used in this thesis, unless stated otherwise, it is specifically referring to a set of necessary conditions that are sufficient to only guarantee a local optimum. For properly formulated DOPs directly according to specifications of a mission, a local optimum is sufficient for improving performance and guaranteeing constraint compliance.

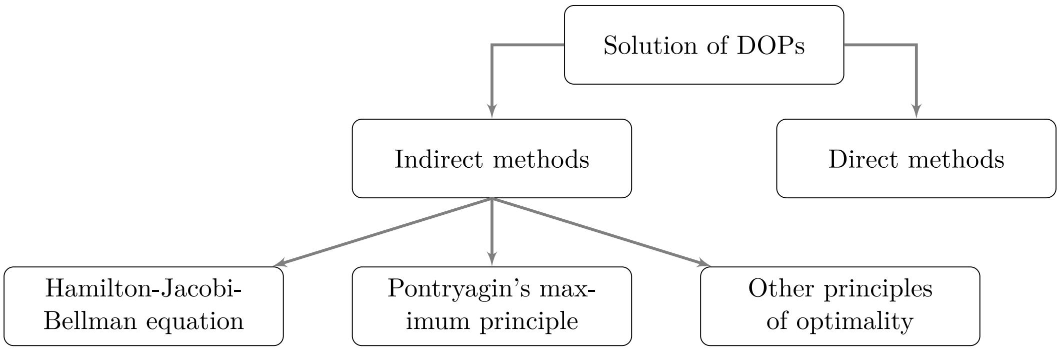

Solution of DOPs
Indirect methods
Direct methods
Hamilton-Jacobi-Bellman equation
Pontryagin's maximum principle
Other principles of optimality

Figure 2-1: Overview of the solution approach for DOPs

2-1-1 Overview

Figure 2-1 demonstrates the two main branches of methods for solving the DOPs. The *indirect methods* are often known as ‘optimise then discretise’, by first deriving the corresponding optimality conditions to the continuous DOP and then discretising the problem containing these optimality conditions. In contrast, the *direct methods* employ the philosophy of ‘discretise then optimise’, by first discretising the DOP into a finite-dimensional DDOP and then formulating the optimality conditions to the DDOP.

Details regarding direct methods will be treated later. Here the focus will be on the optimality conditions employed in indirect methods, as they directly relate to the original DOP (1-1). As shown in Figure 2-1, two commonly used optimality conditions for DOPs are the Hamilton-Jacobi-Bellman (HJB) equation and the Pontryagin's maximum principle (PMP).

The HJB equation is a nonlinear first order partial differential equation defined with the introduction of an *optimal-cost-to-go*. Solved backwards in time, the entirety of the state space is traversed using dynamic programming before arriving at an optimal solution. Hence, the HJB equation is a necessary and sufficient condition for the control to be optimal globally.

On the flip side, solving the HJB equation numerically using dynamic programming leads to long computation times and suffers from the notorious *curse of dimensionality* [14]. Another frequently used optimality condition is the PMP, a necessary condition for the solution to be at optimum; hence can only yield local optimal points. However, relatively simple computations make it possible to solve many challenging problems that would be otherwise intractable.

2-1-2 Pontryagin's maximum principle

The PMP has its root in the *calculus of variations* but may also be derived from the HJB by introducing costate (or adjoint) variables as the partial derivative of the optimal-cost-to-go function with respect to state variables, provided that the HJB equation admits a smooth optimal-cost-to-go solution [201].

Without loss of generality, consider the following ODE-constrained DOP

$$\min_{x,u,p,t_0,t_f} \Phi(x(t_0), t_0, x(t_f), t_f, p) + \int_{t_0}^{t_f} L(x(t), u(t), t, p)\ dt \tag{2-1a}$$

subject to

$$\dot{x}(t) - f(x(t), u(t), t, p) = 0,\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{2-1b}$$

$$\phi(x(t_0), t_0, x(t_f), t_f, p) = 0, \tag{2-1c}$$

To yield the PMP, first define the Hamiltonian for the DOP (1-1) as

$$\mathcal{H}(x(t), u(t), p, t, \lambda(t)) = L(x(t), u(t), t, p) + \lambda(t)^\top f(x(t), u(t), t, p) \tag{2-2}$$

with $\lambda(t) \in \mathbb{R}^n$ the costate corresponding to ODE dynamic constraints (2-1b). The PMP says that if the state and costate are optimal, the optimal control u^* minimizes the Hamiltonian, i.e.

$$u^*(t) \in \arg\min_u \mathcal{H}(x^*(t), u(t), p^*, t, \lambda^*(t)),\ \forall t \in [t_0^*, t_f^*] \text{ a.e.} \tag{2-3}$$

which leads to the first order necessary condition for obtaining the minimum of the Hamiltonian (2-2)

$$\mathcal{H}_u = \frac{\partial \mathcal{H}}{\partial u} = 0. \tag{2-4}$$

It is important to note that in the presence of inequality constraints, the solution for $u^*(t)$ changes whenever active constraints change, resulting in state dependent switches.

2-1-3 Boundary value problem

Additionally, it is possible to derive the co-state dynamics

$$\dot{\lambda}(t) = -\mathcal{H}_x = -\frac{\partial \mathcal{H}}{\partial x}, \tag{2-5}$$

as well as the transversality conditions:

$$\text{either fixed } x_0 \text{ and } t_0, \text{ or } \lambda(t_0) = -\frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)} - \nu^\top \frac{\partial \phi}{\partial \boldsymbol{x}(t_0)}, \tag{2-6a}$$

$$\text{either fixed } x_f \text{ and } t_f, \text{ or } \lambda(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} + \nu^\top \frac{\partial \phi}{\partial \boldsymbol{x}(t_f)}, \tag{2-6b}$$

$$\text{either fixed } t_0, \text{ or } \mathcal{H}(t_0) = \frac{\partial \Phi}{\partial t_0} + \nu^\top \frac{\partial \phi}{\partial t_0}, \tag{2-6c}$$

$$\text{either fixed } t_f, \text{ or } \mathcal{H}(t_f) = -\frac{\partial \Phi}{\partial t_f} - \nu^\top \frac{\partial \phi}{\partial t_f}, \tag{2-6d}$$

with $\nu \in \mathbb{R}^{n_q}$ the multiplier corresponding to boundary constraints. Then (2-1b), (2-1c), (2-4), (2-5) and (2-6b) form a two-point boundary value problem (BVP) together, characterised by partial information available at both the initial and terminal conditions. Finding solutions to the two-point BVP is key in solving DOPs.

2-1-4 Karush-Kuhn-Tucker conditions

Now the optimality conditions for the original DOP (1-1) can be derived. Without loss of generality, the formulation of the Bolza problem can be simplified by considering all ODEs in the DAE form, and all Lagrange costs in Mayer form with the total cost J, leading to

$$\min_{x,u,p,t_0,t_f} J(x(t), u(t), t, p) \tag{2-7a}$$

subject to

$$g(x(t), \dot{x}(t), u(t), t, p) = 0,\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{2-7b}$$

$$c(x(t), \dot{x}(t), u(t), t, p) \leq 0,\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{2-7c}$$

$$\phi(x(t_0), t_0, x(t_f), t_f, p) = 0, \tag{2-7d}$$

$$\tag{2-7e}$$

The Lagrangian, or Lagrange function, is defined as

$$\begin{aligned}\mathcal{L}(x(t), u(t), \lambda(t), \mu(t), t, p) = {} & J(x(t), u(t), t, p) + \lambda(t)^\top g(x(t), \dot{x}(t), u(t), t, p) \\ & + \mu(t)^\top c(x(t), \dot{x}(t), u(t), t, p) + \nu^\top \phi(x(t_0), t_0, x(t_f), t_f, p)\end{aligned} \tag{2-8}$$

with $\lambda(t) \in \mathbb{R}^{n+n_g}$ the costate corresponding to dynamic constraints (2-7b), $\mu(t) \in \mathbb{R}^{n_c}$ the Lagrange multiplier corresponding to inequality path constraints (2-7c), and $\nu \in \mathbb{R}^{n_q}$ again the multiplier corresponding to boundary constraints (2-7d).

Define z as the solution tuple consisting all decision variables for the continuous-time DOP, i.e. $z := (x, u, p, t_0, t_f)$, then the necessary condition for optimality consists of stationary conditions

$$\nabla_z \mathcal{L}(x^*(t), u^*(t), \lambda^*(t), \mu^*(t), t, p^*) = 0,\ \forall t \in [t_0^*, t_f^*] \text{ a.e.} \tag{2-9a}$$

primal feasibility

$$g(x^*(t), \dot{x}^*(t), u^*(t), t, p^*) = 0,\ \forall t \in [t_0^*, t_f^*] \text{ a.e.} \tag{2-9b}$$

$$c(x^*(t), \dot{x}^*(t), u^*(t), t, p^*) \leq 0,\ \forall t \in [t_0^*, t_f^*] \text{ a.e.} \tag{2-9c}$$

$$\phi(x^*(t_0), t_0^*, x^*(t_f), t_f^*, p^*) = 0, \tag{2-9d}$$

dual feasibility

$$\mu^*(t) \geq 0, \ \forall t \in [t_0^*, t_f^*] \text{ a.e.} \tag{2-9e}$$

and complementary slackness conditions

$$\mu^*(t)^\top c(x^*(t), \dot{x}^*(t), u^*(t), t, p^*) = 0, \ \forall t \in [t_0^*, t_f^*] \text{ a.e.} \tag{2-9f}$$

The system of equations (2-9) is commonly known as the Karush-Kuhn-Tucker (KKT) conditions [113, 121]. Similar to PMP and different from the HJB, the KKT conditions are only necessary optimality conditions. Hence, even if solution z is a KKT point and also satisfy the *second order sufficient conditions* [142] (SOSC), it can only guarantee to be a local optimum. Additional requirements includes for $g(x^*(t), \dot{x}^*(t), u^*(t), t, p^*)$ and $c(x^*(t), \dot{x}^*(t), u^*(t), t, p^*)$ to be continuously differentiable, as well as for the problem to fulfil some regularity conditions, for example, the *linear independence constraint qualification* (LICQ), requiring the gradients of the equality constraints and active inequality constraints to be linearly independent at the KKT point.

2-2 Model predictive control

MPC is a modern control design method that requires the formulation and solution of a sequence of constrained DOPs, which arise from system identification, estimation and optimal control problems. Although in conventional literature, MPC has frequently been associated with receding horizon control, quadratic regulation and discrete-time models, none of these implementation choices should be considered as the characteristics of MPC. In this work, the term MPC simply refers to a closed-loop implementation of DOP solutions.

When the DOP solutions are computed offline and implemented online via a look-up table of parameters, the method is known as *explicit MPC*. Explicit MPC is especially attractive for embedded applications of small-sized problems. Research in explicit MPC mainly focuses on methods that can reduce the memory footprint and the table look-up times [103].

In contrast, when the solutions of the DOPs are computed online as in Figure 2-2, the approach is called *implicit MPC*. Implicit MPC may be required for more complex problems, and the main challenges are related to the computational complexity for solving DOPs online.

Linear MPC describes the setup where the DDOP after transcription has a linear/quadratic objective function in terms of the decision variables, subject to linear equality and inequality constraints. As a result, the finite-dimensional DDOP is in the form of linear programming (LP) or quadratic programming (QP). In these cases, the underlying derivative information of *cost gradient* (first derivative of the objective with respect to decision variables), *constraint Jacobian* (first derivative of the constraint equations with respect to decision variables) and

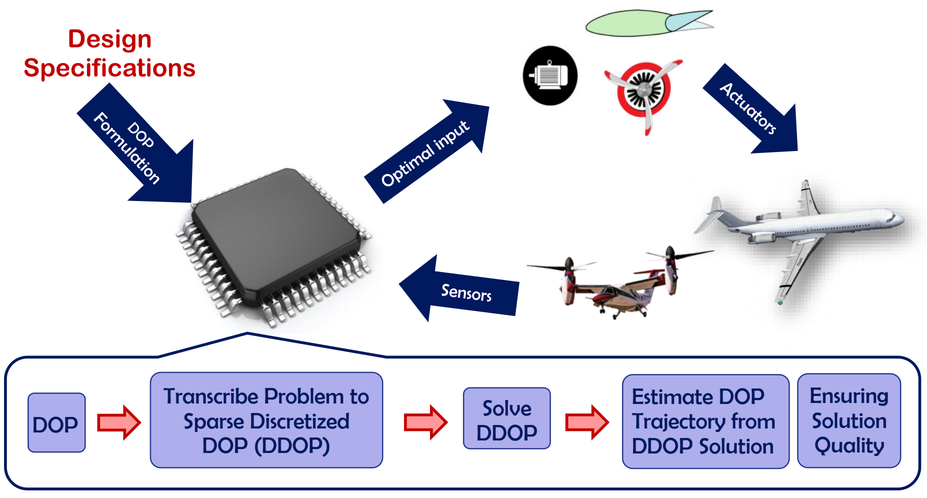

Design Specifications
DOP Formulation
Optimal input
Actuators
Sensors
DOP
Transcribe Problem to Sparse Discretized DOP (DDOP)
Solve DDOP
Estimate DOP Trajectory from DDOP Solution
Ensuring Solution Quality

Figure 2-2: Online implementation of dynamic optimization problems as implicit MPC

Lagrange Hessian (second derivative of the Lagrangian with respect to decision variables) will only need to be provided to the numerical solvers once at the beginning, allowing for more efficient solution of the problems.

For nonlinear MPC (NMPC), both the objective and the constraints can be nonlinear functions. Therefore, the DDOPs will end up to be nonlinear programming (NLP) problems. Typical methods for solving NLPs includes sequential quadratic programming (SQP) [30], interior point method (IPM) [70] and derivative-free optimization (DFO) [177].

When the objectives are formulated directly according to mission specifications in MPC, the method is commonly referred to as economic MPC (EMPC). The stability theory for EMPC and nonlinear EMPC (NEMPC) is progressing [6, 51, 81, 148] but a guarantee is not yet generally attainable. For instance, for EMPC with receding horizon control in general, no guarantee can be made in terms of stability and performance, unless sufficiently long prediction horizon is used with other assumptions on controllability and turnpike conditions satisfied [81]. Therefore, demonstrations and evaluations of stability and performance properties rely heavily on simulation results.

While the experts are focusing on necessary theoretical developments, the use of NMPC and EMPC in practice were not hindered. In contrast, the methods have gained increasing popularity in many fields of applications. A key contributor is the availability of efficient numerical methods and software tools for the solution of DOPs in the last two decades.

2-3 Overview of existing numerical methods for solving DOPs

Most practical DOPs are intractable analytically and need to be solved with numerical methods. This section provides a high-level overview of the most commonly used numerical methods for the transcription and discretization process. Also, a collection of the existing optimal control software is provided, highlighting the choice of NLP solvers and transcription methods.

2-3-1 Numerical solution of DOPs

For some nicely formulated DOPs, it is possible to follow the indirect approach to analytically construct the necessary and sufficient conditions of optimality, which may then be numerically solved. Despite advantages in solution accuracy, indirect methods require analytic expressions for the optimality conditions. The non-triviality of determining these conditions, combined with a lack of robustness w.r.t initialization of state and costate variables means that the counterpart, direct method, has instead become the de facto standard for solving practical DOPs. A detailed survey comparing various variants of direct and indirect methods is available in [168].

In direct methods, the continuous-time DOP is first transcribed into a DDOP using a discretization scheme of choice. The classical categorization of explicit and implicit schemes for solving ODEs also applies to DOP numerical solution methods. However, in DOP literature, these schemes are respectively termed as *sequential* and *simultaneous* approaches.

Time marching describes the approach where the solution at a later time-step is integrated from available solutions of the system dynamics at one or more previous time steps. Direct methods exhibiting these characteristics are referred to as sequential. In the solution process, the initial conditions, parameters and input variables are adjusted repeatedly with the help of sensitivity information until all path constraints and boundary conditions are satisfied. When the method is applied to a single global interval, it is known as the *single shooting method.* More details can be found in [17, 173].

Practical use of single shooting methods generally requires the system dynamics to be stable. Furthermore, the method can be very sensitive to numerical inaccuracies and initial guesses, leading to unstable and ill-conditioned BVPs. Consequently, the solution process tends to have a high chance of failure. A way to mitigate these shortcomings is to subdivide the global interval into multiples sub-intervals that are interconnected with appropriate continuity conditions. This method is consequently called *multiple shooting.* Since the problem now has to be solved altogether as a whole to yield a valid solution, multiple shooting is generally considered to be a hybrid between sequential and simultaneous methods.

In native simultaneous methods, both state and input variables are parameterized, and the continuous-time problem is translated into a single DDOP, instead of explicitly solving the time marching problem within each interval. This class of methods is also known as *direct transcription methods*. *Direct collocation* is a popular variant of direct transcription methods, with the dynamic equations implemented as equality constraints at some selected points in the time domain known as *collocation points*, and solved altogether with other inequality constraints and boundary conditions.

Direct multiple shooting and direct transcription have their respective advantages and disadvantages, and the choice between the two depends on the nature of the problem to be solved. Generally speaking, multiple shooting only parameterizes input variables, leading to smaller DDOPs. Also, when programming, it is easier to make the resultant code modular. These characteristics make multiple shooting favourable, particularly in implementations where the DDOP solvers are problem-size restricted.

Conversely, direct transcription often results in larger DDOPs, but with much sparser underlying system structures. The majority of DOP software developed in the past two decades were built upon direct collocation approaches. This choice over multiple shooting can be attributed to the fact that:

- with modern structure exploiting sparse solvers, a large sparse problem can have lower computational complexity than a smaller but denser one,
- direct transcription avoids the dependency on differential equation solvers as in shooting-based methods, leading to simple and stand-alone schemes,
- for dynamic models for which open-loop simulations are unsuitable (e.g. systems with unstable dynamics), multiple shooting may need a sufficiently refined grid if the time march schemes employed are sensitive to numerical inaccuracies, whereas typical direct transcription methods can often work with a coarser mesh.

These characteristics, however, are based on common implementations of the methods in different software packages. If one have full control of all aspects of the implementation, it is possible to establish mathematical equivalencies between the two methods [167].

2-3-2 Discretization methods

Numerical discretization schemes are often used in the numerical methods for the solution of DOPs. Generally speaking, they can be categorized into fixed-order h methods and variable higher-order p/hp methods.

In h methods, such as Euler, Trapezoidal, Hermite-Simpson (HS) and the Runge-Kutta (RK) family, a fixed degree polynomial is used in each mesh interval for state approximations. Improving the accuracy of an h method is achieved by placing additional mesh points during

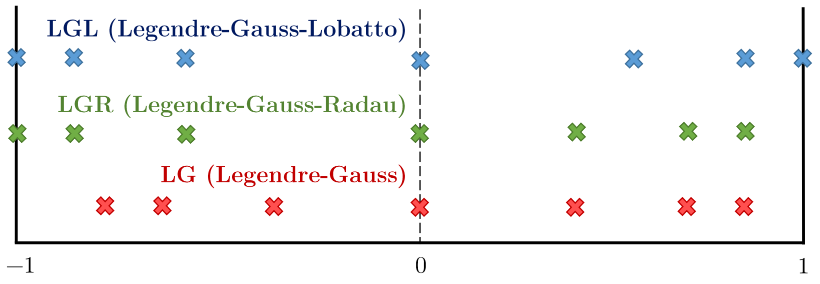


Figure 2-3: Geometric comparison of the LGL, LGR and LG collocation points

the mesh refinement process [17]. Consequently, the size of the resulting DDOP can rapidly increase.

The class of p methods, also known as pseudo-spectral methods, provides another alternative [58]. The fundamental idea is to use orthogonal polynomials (Legendre or Chebyshev) to approximate state and input trajectories in the discretization process. For smooth and well-behaved solutions, the method exhibits exponential convergence [40]. As a result, better approximations can be achieved with a relatively low degree polynomial, significantly reducing the size of the resulting DDOP.

Legendre polynomials are solutions to the Legendre differential equation defined on the interval $[-1, 1]$, with the special property that the N^{th} degree Legendre polynomial $P_N(\,\cdot\,)$ is always orthogonal to any Legendre polynomials of lower degrees. As illustrated in Figure 2-3, there are three main choices for determining polynomial data points:

- Legendre-Gauss (LG) points, being the roots of a N^{th} degree Legendre polynomial $P_N(\,\cdot\,)$, excludes both boundary points.
- Legendre-Gauss-Radau (LGR) points include -1 but do not include the end point at 1; they are the roots of the polynomial $P_N(\,\cdot\,) + P_{N-1}(\,\cdot\,)$.
- Legendre-Gauss-Lobatto (LGL) points include both boundary points -1 and 1; they are the roots of $\dot{P}_{N-1}(\,\cdot\,)$.

Efficient ways to compute LGR points and weights are provided in Appendix A, and the computation of the other two sets of points is analogous.

Depending on the nature of the DOP, choices of these collocation point locations may be particularly suitable or unsuitable. A view from [73] is that the collocation system using Gauss-Lobatto points are over-determined and should be avoided. Also, collocation at Gauss points would need one extra equation to implement the initial conditions at the endpoint

-1. However, because -1 is not a collocation point, the information for control input at the current time instance would need to be obtained via data extrapolation, potentially having an impact on the accuracy of the solution. Collocation using Gauss-Radau points avoids the above-mentioned drawback and has become the preferred choice for pseudo-spectral methods used in solving DOPs. For most problems, the need to extrapolate the values for control inputs for the last point in the interval does not pose a practical concern. On the other hand, for some finite-horizon problems [58], LGR and LG collocation methods reveal a lack of convergence as the mesh becomes finer, where LGL collocation becomes preferable.

Nevertheless, approximating discontinuities and rapid changes in the solution with a global spectral method is difficult due to Gibbs's phenomenon [165]. When the issue arises, both the convergence rate and solution accuracy can deteriorate. The development of *hp*-adaptive methods try to have the best of both worlds: the degree of the polynomial is increased in regions where the solution is expected to be smooth (hereby benefiting from exponential convergence), and mesh refinement is only conducted by adding intervals near potential discontinuities or regions of rapid solution changes [162]. As a result, the same accuracy can be obtained with much smaller DDOP sizes than the h method counterpart, resulting in potential computational advantages.

2-3-3 Solving the DDOP

The choice of the appropriate solver for the DDOP depends on the characteristics of the DOP and available computation resources. One may choose between LP, QP and NLP solvers that are derivative-free, solvers which use first-order derivative information, solvers which use first- and second-order information, and finally solvers based on first derivatives and a quasi-Newton approximation of the second derivative information (e.g. Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithms).

Generally speaking, when the solution is smooth and well behaved, solvers which use higher-order derivative information will converge in fewer iterations. However, this derivative information may need to be obtained through sparse finite differences, analytical derivatives or algorithmic differentiation packages. These can be tedious or computationally expensive. Conversely, solvers that only use first- or zeroth-order derivative information may need a larger number of iterations to converge, but the computation cost per iteration is typically significantly lower. In principle, and in practice, there is no class of DDOP algorithm that is best suited for all problem types. A proper choice should be made on a case to case basis.

2-3-4 Software for transcription and solution of DOPs

The non-trivial nature of numerically solving DOPs has prompted the creation of different software packages employing different numerical schemes. The most commonly used ones listed in Table 2-1.

It can be observed that a number of the software includes built-in NLP solvers, tailor-made for the specific transcription methods that they support. This can often lead to higher efficiency in the solution process and making the software able to handle problems of large dimensions. For example, the built-in sparse solver of SOCS is said to be capable of efficiently handling over 10000 parameters [7]. This capability is further enhanced with SOS as its limitation is said to be increased to 500000 variables together with 500000 constraints.

Nevertheless, the majority of the software directly uses off-the-shelf NLP solvers. Popular candidates include the interior point method IPOPT [205] and SNOPT [76] (an SQP solver using an active-set QP solver). In general, these third party NLP solvers tend to be less capable of handling large scale problems.

Relating the availability of solvers to the choice of discretization methods in Table 2-1, it can be seen that software that uses integrated NLP solvers are more likely to use h methods. With improved handling of large sparse systems, increasing the mesh size for temporal discretization is the preferable way to improve accuracy. In contrast, software that makes use of third party solvers tends to favour p/hp methods, benefiting a smaller problem dimension than typical h method for obtaining a solution of a similar accuracy level.

Many of these tools require immediate commitments to specific programming languages, NLP solvers and transcription schemes. In practice, however, appropriate choices would depend on the required level of accuracy, available computational time/resources and characteristic of the problem formulation (function smoothness, continuity, differentiability, etc.). Hence, it is generally difficult to know beforehand which tool will work best, but subsequent changes in the choice of DOP software increase the engineering expense and design time.

2-3-5 Numerical solution of singular control problems

When solving DOPs, a difficult situation can arise when the second partial derivative of Hamiltonian (2-2) with respect to a input $u(t)$ is singular, i.e. when $\mathcal{H}_{uu} = \frac{\partial^2 \mathcal{H}}{\partial u^2} = 0$. Hence, the optimal input trajectory can not be directly obtained from the optimality conditions. Although these cases of singular control is commonly perceived as a mathematical curiosity, it is, in fact, frequently encountered in many engineering problems: $\mathcal{H}_{uu}$ can be singular whenever $\mathcal{H}$ is a linear function of u, which can occur even for linear ODEs with no path constraints. An example problem is presented in Chapter 11.

Also as mentioned earlier, the solution structure for $u^*(t)$ changes whenever constraint activation status change, hence singular control may occur only at certain intervals of DOP solution, and these intervals are commonly referred to as the *singular arc*. Precise mathematical definitions and analysis of singular arc can be found in various literature [95, 189].

When DOPs with singular arcs are solved directly with existing numerical methods such as direct collocation, solutions trajectories with large fluctuations, also known as ringing phenomena, can occur on the singular arcs. This behavior can be attributed to a combination of factors, including:

- the time instances and conditions for the transitions towards singular arcs often can not be captured exactly by a discretization mesh, and
- to compensate for the mismatch in switch time, the solutions on the singular arc may need to be adjusted accordingly, and
- on the singular arc, solutions with negligible differences in the objective values can have drastically different characteristics, ranging from ones with no or small fluctuations that closely resembles the original continuous-time DOP solution, to ones with large fluctuations that are seemly "ugly".

As a result, the large fluctuations seen on the singular arcs of DDOP solutions can, in fact, be well-defined. However, due to larger residual errors associated with trajectories of high-frequency fluctuations, and due to actuator operational limits, solutions with large fluctuations are generally considered undesirable from an implementation point of view.

If the structures of the optimal DOP solutions are known, then higher-order time derivatives of $\mathcal{H}_u = \frac{\partial \mathcal{H}}{\partial u}$ can be computed for the singular arc to yield information regarding the optimal control input $u^*(t)$. This extra information is commonly known as *singular arc conditions*, typically in the form of extra equality constraints or boundary conditions applied specifically for that segment. Using a multi-phase problem formulation to accurately capture the switch time, and with the help of singular arc conditions, numerical solutions of DOPs without singular arc fluctuations can be obtained thereafter. However, this way of dealing with singular control requires detailed analytical derivations and can quickly become intractable for practical problems. Hence, numerical approaches that can automatically alleviate singular control issues are very attractive, and therefore are a key area of focus of this work.

Table 2-1: Various optimal control software compared

Software	**Solver**	**Programming Language**	**Discretization and Transcription**	**License Type**
SOCS [16]	Integrated dense/sparse NLP solver	FORTRAN 90	*h* methods (direct collocation)	commercial
SOS [18]	Integrated NLP solver SNLPMN, SBRNLP	Graphical user interface (GUI)	*h* methods (direct collocation)	commercial
GESOP [8]	SNOPT, SLSQP SOCS	GUI with Simulink interface	*h* methods (direct collocation)	commercial
DIRCOL [204]	SNOPT, NPSOL	FORTRAN 77 ANSI C	unknown type (direct collocation)	on request
OTIS4 [203]	SNOPT	FORTRAN 95	*h* or *p* methods (direct collocation)	on request
PSOPT [12]	IPOPT, SNOPT	C++	*h* or *p* methods (direct collocation)	open source
GPOPS [170]	SNOPT	Matlab	*p* method (direct collocation)	open source
GPOPS-II [162]	SNOPT, IPOPT KNITRO	Matlab	*hp* method (direct collocation)	commercial
JModelica [2]	Open interface (e.g. IPOPT)	Python Modelica	*h* or *p* method (direct collocation)	open source
ACADO [99]	Integrated NLP (SQP with qpOASES)	C++ with Matlab interface	*h* method (multiple shooting)	open source
ACADOS [196]	Integrated NLP (SQP type)	Low level language with interfaces	*h* method (multiple shooting)	open source
BOCOP [180]	IPOPT	GUI	*h* or *p* method (direct collocation)	open source
FALCON.m [176]	IPOPT, SNOPT	Matlab	*h*/*p* method (direct collocation)	on request
APMonitor [93]	APOPT, BPOPT IPOPT	Matlab, Python Julia, Web	*hp* method (direct collocation)	free through server
FORCES Pro [57]	Built-in Interior-Point	Matlab Python	*h* method (direct collocation)	commercial
pyomo.DAE [86]	IPOPT	Python	*hp* method (direct collocation)	open source
ADRL [75]	IPOPT, SNOPT HPIPM, Riccati-solver	C++	*h* method (multiple shooting)	open source
MUSCOD-II [120]	Built-in SQP solver	C++ with various interfaces	*h* method (multiple shooting)	commercial
DIDO [56]	Built-in solver	Matlab	*hp* method (direct collocation)	commercial
Trajectory-Optimization.jl [175]	iLQR, AL-iLQR ALTRO,IPOPT SNOPT	Julia	*h* method (indirect methods /direct collocation)	open source
NLOptControl.jl [100]	IPOPT, KNITRO	Julia	*h* or *hp* method (direct collocation)	open source
pNMPC [172]	Derivative free solver	C++ with Matlab interface	*h* method (single shooting)	not publicly available
ICLOCS [59]	IPOPT with CVODES for sensitivity analysis	Matlab	*h* method (collocation /multiple shooting)	open source

Chapter 3

Main Contributions of the Thesis

3-1 Main contributions

As discussed in Chapter 1, to maximize the benefits dynamic optimization can bring to various science and engineering fields, the corresponding DOPs should ideally be formulated directly in accordance to the specification of the original problem, i.e. to “solve a problem that needs to be solved”. However in reality, current numerical methods have their limitations in dealing with challenging problems. As a result, when solving directly using existing direct transcription methods and software toolboxes, the original DOP often have to be reformulated and ad hoc fixes have to be added, such as the inclusion of additional regularization terms in the DOP objective.

For complex engineering problems, this current practice to “solve a problem which can to be solved” can lead to significant drawbacks especially considering online implementations. For example, the DAE indexes and the existence of singular control can both change depending on constraint activation status. For a large problem with possibly hundreds or thousands of constraint equations, it is difficult to identify all challenging scenarios in the offline design phase and apply reformulations and ad hoc fixes. As a result, the development of numerical methods that are reliable and efficient for solving DOPs of different difficulty nature will be crucial; hence it is the main focus of this work.

3-1-1 New type of direct transcription method

The main contributions of this research begin with the development of **a new type of direct transcription method: the integrated residual method (IRM)**. This work is presented in Chapter 7, supported by Chapter 4, 5 and 6. By extending the least-squares methods for solving differential equations to the solution of DOPs, the new approach allows one to obtain a number of benefits over direct collocation:

- Instead of forcing the residual errors to be zero only at collocation points, IRM-type methods minimize or constrain the residual error integrated over the whole trajectory, allowing solutions of a much higher accuracy to be achieved for the same discretization mesh, compared to a collation method. The definition of accuracy for a DOP solution is discussed in Section 4-6.
- Through a detailed analysis of error characteristics, a connection between IRM-type methods and direct collocation can be established: enforcement of dynamic constraints in direct collocation can be viewed as a special case of IRM methods where the quadrature points for the numerical integration of the residual errors are selected to be the same as the collocation points. This choice, however, is insufficient to correctly reflect the actual accuracy level of the solution.
- By allowing arbitrarily small residuals to exist during the solution process, IRM-based methods have shown to have better convergence properties for dynamic constraints consists of high-index DAEs or DAEs that force the optimal solution to lay on a manifold. Direct collocation is known to struggle in both cases.
- Due to the existence of approximation error, and the multi-objective nature of IRM-type methods for the solution of DOPs, different singular arc solutions with negligible differences from the objective point-of-view can now be ranked by their error levels. Larger fluctuations in the solution generally lead to bigger errors along the trajectory when approximated by parameterized functions; therefore, a solution with the lowest singular arc fluctuations is often the most accurate solution in the IRM residual error metrics. This provides a unique opportunity for IRM-type transcription methods to automatically suppress potential singular arc fluctuations, without the need for additional treatments.
- Development of *direct alternating integrated residual* (DAIR) transcription method enables a flexible trade-off between solution accuracy and optimality, by formally treating the process of numerically solving DOPs on a given mesh as a multi-objective optimization problem.

3-1-2 Efficient solution of DOPs

In this work, a number of new methods for efficient solution of DOPs have also been proposed, suitable for different transcription methods:

- **A general approach to directly implement rate constraints on the discretization mesh for all discretization methods, and for both state and input variables.** Unlike conventional approaches that may lead to singular arcs, the solution of this on-mesh implementation has better properties. Moreover, computational speedups can be achieved by exploiting the properties of the resulting linear constraint equations. This work is presented in Chapter 9.

- **A strategy for handling inactive constraints efficiently by systematically removing the inactive and redundant constraints**. Inactive constraints do not contribute to the solution of DOPs, but increase the problem size and burden the numerical computations. The proposed method is designed to be used under a mesh refinement framework, with mild assumptions that the original problem has feasible solutions, and the initial solve of the problem is successful. The method can be tailored for numerical solvers that are sensitive to the choice of initial points in terms of feasibility. The details are presented in Chapter 10.

3-1-3 Development of a comprehensive MATLAB toolbox for fast prototyping of optimization-based control

In the course of this PhD research, ICLOCS2[*], a comprehensive MATLAB toolbox for fast prototyping of optimization-based control has been developed, presented in Chapter 12. The toolbox aims to provide the community with a freely available software that is capable of seamlessly changing between wide choice of methods at any point in the design process. The software is also accompanied with unique tools to allow the efficient and accurate solution of large problems. For example, ICLOCS2 incorporates mesh refinement schemes that also considers the reduction of constraint violation errors in-between mesh points, assuring accuracy and constraint compliance of the solution trajectories. The result is a versatile DOP package that has been seen to work on a wide array of challenging real-life and academic problems.

In a collaboration work, ICLOCS2 has successfully resolved an image quality challenge in magnetic resonance imaging (MRI), efficiently solving problems with PDE governing equations that are approximated with several thousands of ODEs, and resulting in DDOPs with more than 500000 decision variables. Additionally, ICLOCS2 has appeared in publications successfully helping researches in different fields [71, 179, 211] to address their respective challenges.

[*]available from http://www.ee.ic.ac.uk/ICLOCS/

3-1-4 Dynamic optimization for aerospace applications

With regards to aerospace applications, a number of explorations were also made in the duration of this PhD program to demonstrate the benefits of solving challenging DOPs that are formulated directly according to mission specifications. Examples include but not limited to:

- **Aircraft upset recovery:** the method have shown great potentials in yielding non-intuitive solutions to bring the aircraft from an out-of-flight-envelope status (e.g. aerodynamic stall) back to nominal flight.
- **Multiple unmanned aerial vehicle (UAV) in-range tracking:** Unlike conventional tracking formulations that aim to eliminate the deviations between the target and the chaser, tracking in-range problems only require the target to be the field-of-view of UAV's on-board camera, hence can lead to substantial savings in energy consumption and greatly increases the mission duration.
- **Joint Optimization of Transmission and Propulsion in UAV-Assisted Communication Networks:** For UAVs to off-load a fixed amount of data to a ground node at a distance, there are both the possibility to move closer to the ground station and to send a stronger signal. The energy-optimal profile would be a non-intuitive solution in-between these two extreme tactics, which can be achieved automatically using a DOP framework.

Selected to be included in this thesis are demonstrations of trajectory optimization problems for commercial flight of the future, based on concepts of continues climb and continues descend in Chapter 14, and formation flight in Chapter 15. Also, a brief review of the history of solving DOPs in aerospace applications is made in Chapter 13 to help envision the future.

3-2 List of publications

The following list of publications represent research work during the PhD. For collaborative work with authors other than the supervisor, additional details on the contributions are also provided.

Journals

1. Y. Nie and E. C. Kerrigan. Efficient implementation of rate constraints for nonlinear optimal control. *IEEE Transactions on Automatic Control*, pages 1–1, 2020. In press

2. Y. Nie and E. C. Kerrigan. Efficient and more accurate representation of solution trajectories in numerical optimal control. *IEEE Control Systems Letters*, 4(1):61–66, Jan 2020. ISSN 2475-1456. doi: 10.1109/LCSYS.2019.2921704. The contents of this paper were also selected by CDC 2019 Program Committee for presentation at the 58th Conference on Decision and Control, Nice, France, Dec 2019

3. Y. Nie and E. C. Kerrigan. External constraint handling for solving optimal control problems with simultaneous approaches and interior point methods. *IEEE Control Systems Letters*, 4(1):7–12, 2020. doi: 10.1109/LCSYS.2019.2921700. The contents of this paper were also selected by CDC 2019 Program Committee for presentation at the 58th Conference on Decision and Control, Nice, France, Dec 2019

Conference proceedings

1. Y. Nie and E. C. Kerrigan. Capturing discontinuities in optimal control problems. In *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018. doi: 10.1109/CONTROL.2018.8516770

2. Y. Nie and E. C. Kerrigan. Efficient implementation of rate constraints for nonlinear optimal control. In *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018. doi: 10.1109/CONTROL.2018.8516847

3. Y. Nie and E. C. Kerrigan. How should rate constraints be implemented in nonlinear optimal control solvers? *IFAC-PapersOnLine*, 51(20):362 – 367, 2018. ISSN 2405-8963. doi: 10.1016/j.ifacol.2018.11.060. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018

4. E. C. Kerrigan, Y. Nie, O. J. Faqir, C. Kennedy, S. A. Niederer, J. A. Solis-Lemus, P. Vincent, and S. E. Williams. Direct transcription for dynamic optimization: A tutorial and case study on dual-patient ventilation during the COVID-19 pandemic, Dec 2020. Accepted to 59th IEEE Conference on Decision and Control (CDC)
Contributions: finalizing technical details, performing numerical computations of DOPs, interpretation of the results, writing parts of manuscript.

5. Y. Nie, O. J. Faqir, and E. C. Kerrigan. ICLOCS2: Try this optimal control problem solver before you try the rest. In *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018. doi: 10.1109/CONTROL.2018.8516795
Contributions: software development, writing of the manuscript.

6. O. J. Faqir, Y. Nie, E. C. Kerrigan, and D. Gündüz. Energy-efficient communication in mobile aerial relay-assisted networks using predictive control. *IFAC-PapersOnLine*,

51(20):197 – 202, 2018. ISSN 2405-8963. doi: 10.1016/j.ifacol.2018.11.013. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018
Contributions: performing numerical computations of DOPs, interpretation of the results, finalizing the manuscript.

Poster presentation with abstract submission

1. Y. Nie and E. C. Kerrigan. ICLOCS2: Solve your optimal control problems with less pain, Aug 2018. presented at 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018

Paper under review

1. Y. Nie and E. C. Kerrigan. Solving Dynamic Optimization Problems to a Specified Accuracy: An Alternating Approach using Integrated Residuals. Submitted to *IEEE Transactions on Automatic Control*.

Working paper

1. Y. Nie, O. J. Faqir and E. C. Kerrigan. ICLOCS2: A Comprehensive MATLAB Toolbox for Fast Prototyping of Optimization Based Control.
Contributions: software development, perform numerical computations of DOPs, interpretation of the results, writing of the manuscript.

2. O. J. Faqir, D. Gündüz, E. C. Kerrigan, and Y. Nie. Joint Optimization of Transmission and Propulsion in UAV-Assisted Communication Networks.
Contributions: development of UAV model, perform numerical computations of DOPs, interpretation of the results, finalize the manuscript.

Part II

Numerical Methods for Dynamic Optimization Problems

Chapter 4

Discretization and Parameterization of Dynamic Optimization Problems[†]

The original continuous-time DOP (1-1) is an infinite-dimensional problem that innately embeds the concepts of time, space and causality. In practice, finding approximate solutions to the DOP using numerical methods requires the definition and solution a DDOP or a sequence of DDOPs through a so-called transcription process. Central to the transcription process are the discretization schemes that take the vast solution space of the infinite-dimensional (otherwise intractable) problem and define low dimensional manifolds which the system behaviour lies on.

In this chapter, typical temporal discretization approaches are discussed in Section 4-1 which could be further extended for spatial discretizations analogously. Then commonly used methods for parameterization of solution trajectories are introduced in Section 4-3 followed by a discussion on solution representation methods in Section 4-5. Lastly, error metrics for posterior error analysis are presented in Section 4-6.

4-1 Temporal discretization

For a finite horizon problem defined from t_0 to t_f, we may define accordingly K intervals $\mathbb{T}_k := [t_k, t_{k+1}]$ for all $k \in \mathbb{I}_K := \{1, \dots, K\}$ locations with $t_0 = t_1 < \cdots < t_{K+1} = t_f$, so

[†] This chapter contains materials from the following papers:
Y. Nie and E. C. Kerrigan. Solving Dynamic Optimization Problems to a Specified Accuracy: An Alternating Approach using Integrated Residuals. Under review.

that $\mathbb{T} = \bigcup_{k \in \mathbb{I}_K} \mathbb{T}_k$. These locations t_k for all $k \in \mathbb{I}_{K+1}$ are named the *major nodes*, also commonly known as *mesh nodes*. The mesh design is critically important for the solution efficiency and accuracy. Inside each interval k, additional *minor nodes* may be defined. The inclusion of these minor nodes inside each mesh interval may due to different reasons and will be discussed later.

To efficiently handle variable horizon problems, the time domain can be normalized with intervals $\mathbb{S}_k := [s_k, s_{k+1}]$ where $s_1 = 0$ and $s_{K+1} = 1$, and the relationship $t_k = t_0 + (t_f - t_0)s_k$. As the result, t_0 and t_f can be included as optimization decision variables. Using $\mathbb{S} = \bigcup_{k \in \mathbb{I}_K} \mathbb{S}_k$, t_0 and t_f together, $\mathbb{T}$ can be uniquely represented. In this way, even with fixed designs of $\mathbb{S}$, i.e. the distribution and location of mesh points inside the domain is determined a priori, the time instances corresponding to these locations can vary since t_0 and t_f can be free.

In order to simplify the notation, the explicit dependence of each time instant t_k on a given s_k, t_0 and t_f will be omitted. It should be understood throughout that the time instances, t_k, and the time differences between them, $\Delta t^{(k)} := t_{k+1} - t_k$, are actually functions of t_0 and t_f.

4-2 Approximating the DOP solution with parameterized functions

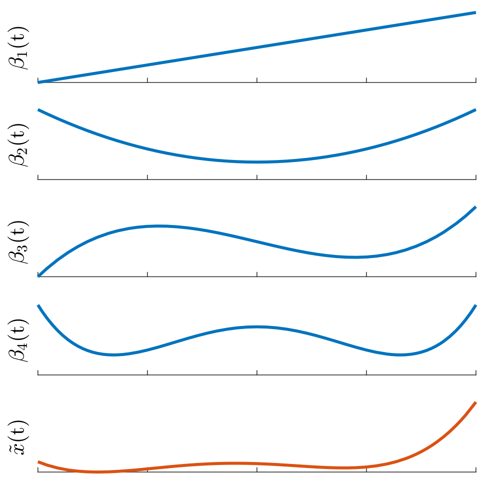
The key concept employed in the transcription process is the approximation of state, state derivatives and input trajectories with parameterized continuous or piecewise-continuous functions. The approximated trajectories will be denoted by $\tilde{x}$, $\dot{\tilde{x}}$ and $\tilde{u}$ respectively, and can be characterized by a number of discrete variables.

This process often leads to the misconception that a continuous-time DOP solution is approximated using point values. In fact, the solution trajectories are approximated with a combination of pre-specified basis functions. For example, the state trajectories x in can be approximated as

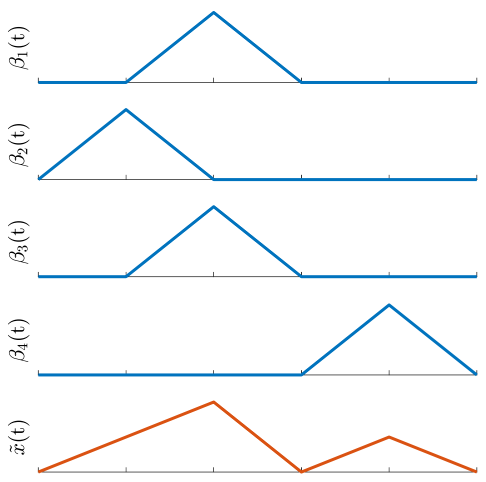
$$x(t) \approx \tilde{x}(t) := \sum_{i=1}^{N} a_i \beta_i(t), \tag{4-1}$$

with some coefficients a_i, also known as *amplitudes of the basis functions* β_i. The key to solving the solution approximation problem is to determine the discrete unknowns a_i, for all $i \in \mathbb{I}_N$. For practical problem, the unknowns are typically solved with numerical schemes hence also often called *degrees of freedom* of the problem.

The basis functions $\beta_i(\,\cdot\,)$ can either be *global* functions, which span the entire domain, or *local* functions that only have non-zero values inside specific regions. If global functions basis are used, the resulting methods are usually referred to as *spectral methods*, whereas local functions lead to *finite-element methods*. Both cases are illustrated in Figure 4-1.



(a) using global basis functions



(b) using local basis functions

Figure 4-1: Illustration for the formation of approximation function $\tilde{x}$ using basis functions

It is also possible to map global basis functions into local ones through a coordinate transformation. An interval $[t_k, t_{k+1}]$ in the global domain can be normalized and mapped to local coordinates which take value, for example, on $[-1, 1]$. Hence one could view this implementation like that, in each interval $k \in \mathbb{I}_K$, state trajectories $x^{(k)}$ are approximated as $\tilde{x}^{(k)}$ with continuity conditions specified at mesh nodes when necessary. Methods with such characteristics also belong to the class of finite-element methods.

4-3 Parameterization of state and input variables

Taking the finite element approach, the state trajectories x in each interval $k \in \mathbb{I}_K$ can be approximated as

$$x^{(k)}(t) \approx \tilde{x}^{(k)}(t) := \sum_{i=1}^{N^{(k)}} a_i^{(k)} \beta_i^{(k)}(t).$$

One common choice of such approximating functions would be polynomials, with the continuous function characterized by some polynomial coefficients. With $t \mapsto \tilde{x}^{(k)}(t)$ a polynomial, the parameterized state trajectory $t \mapsto \tilde{x}(t)$ is therefore a piecewise polynomial. Polynomial methods often use Lagrange interpolating polynomials as basis functions, because they tend to have a number of advantages from a numerical and implementation point of view, compared to using monomials or other basis functions [15]. For example, warm starting of the NLP solver is significantly faster and simpler to implement with Lagrange interpolating polynomials.

The idea is to choose parameterization methods that require $a^{(k)}$ to consist of samples of $\tilde{x}^{(k)}(\,\cdot\,)$, i.e. the trajectory $t \mapsto \tilde{x}^{(k)}(t)$ should interpolate through the components of $a^{(k)} =: (a_1^{(k)}, \dots, a_{N^{(k)}}^{(k)})$ at a given set of minor nodes τ_i for all $i \in \mathbb{I}_{N^{(k)}}$. In this case, these minor nodes are named *data points*, and the corresponding coefficients are named *parameterized states* denoted by $\chi_i^{(k)}$, i.e. $\chi_i^{(k)} = \tilde{x}^{(k)}\left(\tau_i^{(k)}\right) \in \mathbb{R}^n$ for all $i \in \mathbb{I}_{N^{(k)}}$, to distinguish this from the more general case above. Additionally, $\chi^{(k)} := [\chi_1^{(k)}, \dots, \chi_{N^{(K)}}^{(k)}]^\top \in \mathbb{R}^{N^{(k)} \times n}$ and $\chi := [\chi^{(1)}, \dots, \chi^{(K)}]^\top \in \mathbb{R}^{N \times n}$, with $N := \sum_{k=1}^{K} N^{(k)}$, are defined for brevity in later discussions. The major nodes together with the data points are often referred to as the *grid points*.

It is also possible to define a quadrature of different order for each interval, in order to numerically integrate a functional. The *quadrature points* are defined as a different choice of minor nodes inside each interval k according to the quadrature scheme, i.e. $q_i^{(k)}$, for all $i \in \mathbb{I}_{Q^{(k)}}$, where $Q^{(k)}$ is the number of quadrature points inside the interval. In this way, accurate numerical integration with higher-order quadrature rules is made possible.

For most single-phase DOPs, continuity of state trajectory is required. Continuity of the trajectories between interval k and $k+1$ can be enforced either

- implicitly by using the same decision variable for the last node of interval k and the first node of interval $k+1$, reducing the number of unknowns points from $N^{(k)}$ to $N^{(k)} - 1$, or
- explicitly by additional continuity constraints:

$$\tilde{x}^{(k)}\left(t_{k+1}\right) = \tilde{x}^{(k+1)}\left(t_{k+1}\right), \tag{4-2}$$

  which will also effectively reduce the number of degree of freedom by 1.

The parameterization of the input using the data point values $\upsilon_i^{(k)}$ can be done similarly for the approximation function $\tilde{u}^{(k)}$, where each function $\tilde{u}^{(k)}$ is continuous and differentiable. Since u can be discontinuous, therefore for methods that evaluate functions at the endpoint of the intervals (implicit Runge-Kutta, Radau or Lobatto schemes, such as the trapezoidal and Hermite-Simpson methods), $\tilde{u}^{(k)}(t_{k+1})$ will not be the same as $\tilde{u}^{(k+1)}(t_{k+1})$. Suitable choices for the functions $\tilde{u}^{(k)}$ are application-dependent. Popular choices are for each $t \mapsto \tilde{u}^{(k)}$ to be a constant or polynomial, so that the parameterized trajectory $t \mapsto \tilde{u}(t)$ is piecewise constant or (discontinuous) piecewise polynomial. The discretization and parameterization for x and u is illustrated in Figure 4-2.

In practice, some systems might require the input to be a zero-order or other hold signal. However, early on in the mesh refinement process (see Section 4-6-3 and 12-2-2), when the intervals defined by the mesh are relatively large compared to the time between control updates,

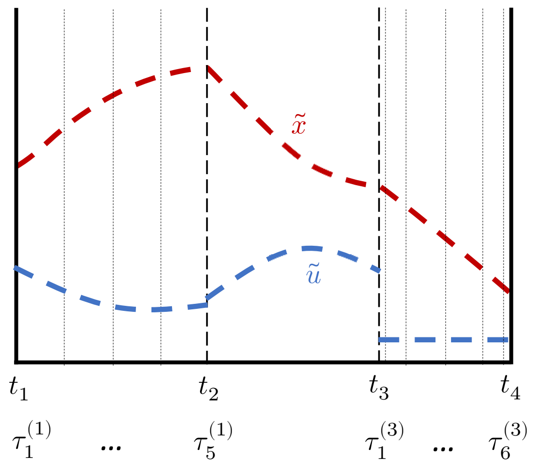

$\tilde{x}$
$\tilde{u}$
t_1
t_2
t_3
t_4
$\tau_1^{(1)}$ ... $\tau_5^{(1)}$
$\tau_1^{(3)}$ ... $\tau_6^{(3)}$

Figure 4-2: Temporal discretization and trajectory parameterization. The number and distribution of interpolation points need not be the uniform or the same in each interval. The state trajectory $\tilde{x}$ is continuous, but that the trajectory $\tilde{u}$ can be discontinuous.

Table 4-1: Order of the approximating polynomial functions (a.m.: at most)

Method	**State derivatives** ($\dot{\tilde{x}}$)	**States** ($\tilde{x}$)	**Inputs** ($\tilde{u}$)
Euler	piecewise constant	a.m. piecewise linear	same as state derivatives
Trapezoidal	a.m. piecewise linear	a.m. piecewise quadratic	
H-S	a.m. piecewise quadratic	a.m. piecewise cubic	
LGR	a.m. order $N^{(k)}$-1	a.m. order $N^{(k)}$	

it might be computationally more efficient to use a (non-constant) polynomial parameterization for υ_i, even if u is implemented in a piecewise constant manner. In these cases where both state and input trajectories inside an interval are parameterized by polynomials, the degrees of the polynomials are typically in accordance with the discretization scheme as listed in Table 4-1. Without loss of generality, the rest of this work follows this convention.

4-4 Discretized dynamic optimization problem

With the above-mentioned temporal discretization and state and input trajectory parameterization, solutions to the discretized problem, i.e. the full set of n_z optimization variables, can be denoted as $\mathcal{Z} := (\chi, \upsilon, p, t_0, t_f)$. Now the direct discretization of DOP (1-1) can be expressed as the following DDOP:

$$\min_{\chi,\upsilon,p,t_0,t_f} \Phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) + \sum_{k=1}^{K} \sum_{i=1}^{Q^{(k)}} w_i^{(k)} L\left(\tilde{x}^{(k)}\left(q_i^{(k)}\right), \tilde{u}^{(k)}\left(q_i^{(k)}\right), t_0, t_f, p\right) \tag{4-3a}$$

subject to, for all $k \in \mathbb{I}_K$,

$$\psi^{(k)}\left(\chi^{(k)}, \upsilon^{(k)}, t_0, t_f, p\right) = 0, \tag{4-3b}$$

$$\gamma^{(k)}\left(\chi^{(k)}, \upsilon^{(k)}, t_0, t_f, p\right) \leq 0, \tag{4-3c}$$

$$\phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) = 0. \tag{4-3d}$$

as well as any necessary continuity constraints in the form of (4-2). $\chi_f^{(K)}$ is the value of the state variables at terminal time t_f, and for schemes with the last node being a data point, $\chi_f^{(K)} = \chi_{N^{(K)}}^{(K)}$ holds. Otherwise $\chi_f^{(K)}$ can be transcription method depended, e.g. in LGR collocation it is $\chi_f^{(K)} = \chi_{N^{(K+1)}}^{(K)}$, with $\chi_{N^{(K+1)}}^{(K)}$ additionally added as a decision variable. The expressions for the functions $\psi^{(k)}$ and $\gamma^{(k)}$ depend on the details of the transcription method. The scalars $w_i^{(k)}, \forall i \in \mathbb{I}_{Q^{(k)}}$ are the interval-dependent quadrature weights (i.e. including the corresponding time interval $\Delta t^{(k)}$ contributions) for the numerical integration of the Lagrange cost inside the interval.

4-5 Representation of the continuous DOP solution trajectories

The discretized problem can be solved with off-the-shelf DDOP (LP/QP/NLP) solvers, outputting the discretized solution $\mathcal{Z} := (\chi, \upsilon, p, t_0, t_f)$. The approximate solution to the continuous DOP $\tilde{z}(t) := (\tilde{x}(t), \tilde{u}(t), t, p)$ can then be directly evaluated everywhere along the trajectory by substituting the values of the decision variables into the approximation functions. However, if it is challenging to evaluate the original approximation functions with good numerical accuracy at locations other than the data points, a different interpolating scheme can be used to construct an approximation of the continuous-time optimal trajectory. For example in p/hp methods, Lagrange interpolating polynomials in the barycentric form are often used to construct $\tilde{x}^{(k)}$, which has superior numerical stability than the standard Lagrange interpolating polynomials.

4-5-1 Representation through direct interpolation

The most straightforward choice is to interpolate the solution in correspondence with the discretization scheme used in the transcription process. For most commonly-used numerical schemes, the numerical differentiation formulation has an equivalent integration form, as presented in Table 4-2, where

$$\digamma_i^{(k)} := f(\chi_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p), \text{ for all } i \in \mathbb{I}_{N^{(k)}}.$$

Based on these integration schemes, direct interpolation of the sampled point solution is

Table 4-2: Typical numerical integration schemes

Method (degree $N^{(k)}$)	**Numerical Integration Scheme**
Euler (1)	$\chi_2^{(k)} = \chi_1^{(k)} + \Delta t^{(k)} \mathcal{F}_1^{(k)}$
Trapezoidal (2)	$\chi_2^{(k)} = \chi_1^{(k)} + \frac{\Delta t^{(k)}}{2}(\mathcal{F}_1^{(k)} + \mathcal{F}_2^{(k)})$
Hermite Simpson (3)	$\chi_2^{(k)} = \frac{1}{2}(\chi_2^{(k)} + \chi_1^{(k)}) + \frac{\Delta t^{(k)}}{8}(\mathcal{F}_1^{(k)} - \mathcal{F}_3^{(k)})$ $\chi_3^{(k)} = \chi_1^{(k)} + \frac{\Delta t^{(k)}}{6}(\mathcal{F}_1^{(k)} + 4\mathcal{F}_2^{(k)} + \mathcal{F}_3^{(k)})$
LGR ($N^{(k)}$)	$\mathcal{I}^{(k)} = [\mathcal{A}^{(k)}_{2:N^{(k)}+1}]^{-1}$ $\chi^{(k)}_{2:N^{(k)}+1} = \chi_1 + \frac{\Delta t^{(k)}}{2}\mathcal{I}^{(k)}\mathcal{F}^{(k)}_{1:N^{(k)}}$

possible using splines with the type and order in accordance with Table 4-1. For example, with Hermite-Simpson transcription, the trajectory of system dynamics/state derivatives inside mesh interval k using quadratic splines will be

$$\begin{aligned}\dot{\tilde{x}}^{(k)}(t) =& \mathcal{F}_1^{(k)} + \Big(-3\mathcal{F}_1^{(k)} + 4\mathcal{F}_2^{(k)} - \mathcal{F}_3^{(k)}\Big)\frac{t-t_k}{\Delta t^{(k)}} \\ &+ \Big(2\mathcal{F}_1^{(k)} - 4\mathcal{F}_2^{(k)} + 2\mathcal{F}_3^{(k)}\Big)\Big(\frac{t-t_k}{\Delta t^{(k)}}\Big)^2.\end{aligned}$$

Integrate this polynomial yields the cubic spline state trajectory,

$$\begin{aligned}\tilde{x}^{(k)}(t) =& \chi_1^{(k)} + \mathcal{F}_1^{(k)}(t-t_k) + \frac{1}{2}\Big(-3\mathcal{F}_1^{(k)} + 4\mathcal{F}_2^{(k)} - \mathcal{F}_3^{(k)}\Big)\frac{(t-t_k)^2}{\Delta t^{(k)}} \\ &+ \frac{2}{3}\Big(\mathcal{F}_1^{(k)} - 2\mathcal{F}_2^{(k)} + \mathcal{F}_3^{(k)}\Big)\frac{(t-t_k)^3}{\big(\Delta t^{(k)}\big)^2}.\end{aligned} \tag{4-4}$$

The control trajectory with quadratic splines is expressed as,

$$\begin{aligned}\tilde{u}^{(k)}(t) =& \frac{2}{\big(\Delta t^{(k)}\big)^2}(t - \frac{1}{2}t_k - \frac{1}{2}t_{k+1})(t-t_{k+1})\upsilon_1^{(k)} - \frac{4}{\big(\Delta t^{(k)}\big)^2}(t-t_k)(t-t_{k+1})\upsilon_2^{(k)} \\ &+ \frac{2}{\big(\Delta t^{(k)}\big)^2}(t-t_k)(t - \frac{1}{2}t_k - \frac{1}{2}t_{k+1})\upsilon_3^{(k)},\end{aligned}$$

for all $t \in [t_k, t_{k+1}]$. The full trajectory $\tilde{x}$, $\dot{\tilde{x}}$ and $\tilde{u}$ will therefore be expressed as a piecewise cubic and two piecewise quadratic polynomials respectively.

4-5-2 Representation through polynomial fitting

For constructing the continuous-time trajectory, one may also consider the use of other types of function approximation or polynomial fitting techniques, for example, the MATLAB built-in `pchip` (piecewise cubic Hermite interpolating polynomial) function. The choice of spline types and orders depends on the continuity of the trajectory and the required accuracy. Generally speaking, piecewise polynomials are, again, preferred to global methods for this purpose, especially considering the regions with extremely dense mesh and rapidly varying solutions. Note that `pchip` is fundamentally different from (4-4), and it uses smaller sized splines and interpolate between each grid points. For example, with HS discretization, direct interpolation would result in a single cubic function for state trajectory inside an interval, whereas `pchip` would generate two cubic functions for each of the sub-interval divided by the mid-point.

Although various options are possible, it is always a good practice to have the state derivative $\dot{\tilde{x}}$ and input $\tilde{u}$ directly fitted or interpolated, and integrate the polynomial equation to obtain a trajectory for $\tilde{x}$. This way it ensures a correct relationship between $\dot{\tilde{x}}$ and $\tilde{x}$ throughout the trajectory. With any polynomial fitting method, one must pay special attention to avoid situations where the fitting is badly conditioned. The conditioning can be improved by making use of techniques such as scaling.

4-6 Error metrics for DOP solutions

4-6-1 Errors related to dynamic constraints

Any valid solution trajectory $(t \mapsto \tilde{x},\ \tilde{u}, p, t)$ must satisfy the system dynamics (1-1b)-(1-1c) with a good level of accuracy. To measure the accuracy of the solution, one would like to compare $\tilde{x}$ with x, and $\dot{\tilde{x}}$ with $\dot{x}$; however such exact solutions are not obtainable for the majority of the practical problems, hence can only be used for benchmarking purposes. Therefore, we need appropriate error metrics that can indicate the accuracy of a solution, without knowing the accurate solution itself.

Regarding the dynamic equations, it is often regarded as a good idea in practice to compute the residuals $\varepsilon(t) \in \mathbb{R}^{n+n_g}$ defined as

$$\varepsilon(t) := \begin{bmatrix} \dot{\tilde{x}}(t) - f(\tilde{x}(t), \tilde{u}(t), t, p) \\ g(\tilde{x}(t), \dot{\tilde{x}}(t), \tilde{u}(t), t, p) \end{bmatrix}, \tag{4-5}$$

which is straightforward to compute for any DOP solution. However, as further demonstrated later in Chapter 5, the residual evaluated at a selected location of the domain is not representative of the actual accuracy of the solution. Instead, the absolute value (for a single dynamic constraint) or the norm (for multiple dynamic constraints) of the residuals integrated along certain intervals of the domain are often found to be a suitable metrics for measure of solution accuracy.

Following this idea, the most popular choices for error analysis are to integrate the absolute value of $\varepsilon(t)$ for in each interval $\mathbb{T}_k$ and for each dynamic equation $\xi = 1, \ldots, n + n_g$. With $\varepsilon_\xi^{(k)}(t) \in \mathbb{R}$ the ξ^{th} element in $\varepsilon(t) \in \mathbb{R}^{n+n_g}$ in interval k, the corresponding error can be computed as

$$\zeta_\xi^{(k)} := \int_{\mathbb{T}_k} |\varepsilon_\xi^{(k)}(t)| \, dt, \tag{4-6}$$

for all dynamic equations to obtain $\zeta^{(k)} \in \mathbb{R}^{n+n_g}$. Alternatively, a single metric can be used as

$$\eta^{(k)} := \int_{\mathbb{T}_k} \|\varepsilon^{(k)}(t)\|_2 \, dt, \tag{4-7}$$

with $\| \cdot \|_2$ the vector 2-norm. The integrals can be practically estimated by high-order quadrature. The metrics $\zeta \in \mathbb{R}^{(n+n_g)\times N}$ or $\eta \in \mathbb{R}^N$ are typically referred to as the *absolute local error*. When this error is normalized with the largest magnitudes of state and state derivatives, the error is known as the *relative local error* [17]. Other variants are also possible such as the *mean local error*, with normalization by the interval size, and *squared absolute local error*, using the square of the norm instead. For schemes using such error measures based on the evaluation of the residuals locally, a solution will be considered sufficiently accurate when the values of the error are below pre-specified tolerances for all mesh intervals.

Analogously, the integration of residual errors can be computed along the whole trajectory. For instance, the *integrated residual norm squared* (IRNS) error can be computed as

$$r(\tilde{x}, \tilde{u}, t_0, t_f, p) := \int_{t_0}^{t_f} \|\varepsilon(t)\|_2^2 \, dt. \tag{4-8}$$

which is equivalent to the sum of absolute local errors $\eta^{(k)}$ for all intervals $k \in \mathbb{I}_K$. Other variations in the definition are also possible, e.g. the *integrated residual squared* (IRS) error computed for each individual dynamic equation, and the *mean integrated residual norm squared* (MIRNS) error defined as $\frac{r}{\Delta t}$ with $\Delta t := t_f - t_0$.

4-6-2 Errors due to constraint violations

Numerical discretization also inevitably leads to possible constraint violations of the trajectories in between the grid points. For path and box constraints that are expressed semi-explicitly as (1-1d), the *absolute local constraint violation* $\epsilon(t) \in \mathbb{R}^{n_c}$ may be estimated as

$$\epsilon_l(t) := \begin{cases} 0 & \text{if } c_l(\tilde{x}(t), \dot{\tilde{x}}(t), \tilde{u}(t), p, t) \leq 0 \\ c_l(\tilde{x}(t), \dot{\tilde{x}}(t), \tilde{u}(t), p, t) & \text{if } c_l(\tilde{x}(t), \dot{\tilde{x}}(t), \tilde{u}(t), p, t) > 0 \end{cases}, \text{ for } l = 1, \ldots, n_c. \tag{4-9}$$

with c_l the l^{th} element in the equality constraints c.

4-6-3 Mesh refinement

Once the locations of errors on the mesh are obtained, appropriate modifications can be made to the discretization mesh. The DDOP can be updated and solved iteratively until a solution which fulfils all predefined error tolerances is obtained. This process is called mesh refinement, generally requiring mesh intervals, i.e. mesh points, to be added for h or hp methods, and the polynomial degree of an interval to be adjusted for p or hp methods (details can be found in [17, 129] and Section 12-2-2). The NLP problem based on the new mesh can be warm-started using the solution from the previous mesh, leading to significantly faster convergence, thus reducing the overall computation time.

Mesh refinement is an indispensable part of any direct transcription scheme that do not embed any measures to ensure solution accuracy between data points, such as direct collocation. One implementation of mesh refinement logic used in the toolbox ICLOCS2 is explained in further details in Section 12-2-2.

Chapter 5

Enforcement of Dynamic Constraints in Discretized Dynamic Optimization Problems

The key concept employed in the transcription process in Chapter 4 is the approximation of state, dynamics and input trajectories with continuous functions. It is important to realize that solutions of (1-1) can rarely be represented exactly by the approximating functions. For example, with a polynomial basis, even in the simple case where $f(x(t), u(t), p, t) = \dot{x}(t) = \alpha x(t) + u(t)$ and $u(t) = 1$ are both polynomials, the corresponding state trajectory $x(t) = x(0)e^{\alpha t} + \int_0^t e^{\alpha(t-\varsigma)}u(\varsigma)\, d\varsigma$ is clearly not a polynomial for all $\alpha \neq 0$ and approximation errors should be expected.

The general inevitability of approximation errors leads to an important implication: it is not possible for (1-1b)–(1-1c) to be satisfied everywhere along the domain for any arbitrary choice of the minor nodes. To gain better insight on how these constraints should be dealt with, a broader class of numerical methods commonly used to solve ODEs, DAEs and PDEs can be referred to, namely the Rayleigh-Ritz approach [171, Sect. 5.2–5.7] and the weighted residual methods [171, Sect. 5.8].

For the weighted residual methods, different choices of weighting functions lead to different methods, including the famous Galerkin, collocation, and least-squares methods. In this chapter, a numerical study is provided highlighting the error characteristics of each method, providing the foundations for later chapters regarding the underlying issues with direct collocation method and the motivation for the development of the integrated residual type of methods for solving DOPs.

5-1 The Rayleigh-Ritz approach

One way to solve the solution approximation problem is to define an equivalent optimization problem which minimises some measure of the solution. For instance, in finding the solution of ODEs in the strong form of $\dot{x}(t) = f(x(t))$, one could solve the following minimization problem

$$\min_{a} \int_{\mathbb{T}} \|\varepsilon(\tilde{x}(t))\|_2^2 \, dt, \tag{5-1}$$

with $a := [a_1, \ldots, a_N]^\top$ and ε the ODE residuals of the approximating function $\tilde{x}(t) := \sum_{i=1}^{N} a_i \beta_i(t)$, i.e.

$$\varepsilon(\tilde{x}(t)) = \dot{\tilde{x}}(t) - f(\tilde{x}(t)).$$

To find the extreme value (maximum or minimum) of the objective, by definition, one could require the derivatives with respect to the unknowns a_i to be zero, i.e.

$$\frac{\partial \int_{\mathbb{T}} \|\varepsilon(\tilde{x}(t))\|_2^2 \, dt}{\partial a_i} = 0, \text{ for } i \in \mathbb{I}_N. \tag{5-2}$$

The method known as the Rayleigh-Ritz approach.

5-2 Weighted residual methods

With the Rayleigh-Ritz approach, one often need to refer to calculus of variations for computing the extremal of functionals. For numerical solution of differential equations, the method of weighted residuals is more generally applicable approach than Rayleigh-Ritz method.

Instead of the strong form, with the weighted residual methods, it is also possible to define the boundary value problem of differential equations using the weak form. This approach requires that

$$\int_{\mathbb{T}} \varpi(t)\varepsilon(\tilde{x}(t))dt = 0 \tag{5-3}$$

to be hold for the entire domain $\mathbb{T}$ and for all suitable weighting functions $\varpi(t)$. By suitable, a basic requirement is that the weak form must remain integrable. Other requires may be necessary on a case to case basis, for instance, depending on the discretization scheme employed.

Since both spectral and finite-element method problems are now solved under the weak form using weighted residual methods rather than the strong form, one must show their equivalence to each other. It is straightforward to show that the residual of the exact solution to the strong form will also satisfy the weak form, with (5-3) guarantee to hold when $\varepsilon(\tilde{x}(t)) = \dot{\tilde{x}}(t) - f(\tilde{x}(t)) = 0$. The other way around requires a bit more details.

Proposition 1. *Besides the solution to the strong form of a differential equation, there does not exist other solutions for the weak form to be satisfied.*

Proof. For a solution to not satisfy the strong form, a non-zero residual $\varepsilon(\tilde{x}(t)) \neq 0$ must exist for some region of the domain $\mathbb{T}$. Since (5-3) must hold for all suitable weighting functions $\varpi(t)$, it must hold for a selection of weighting function such that it is positive over this region of $\varepsilon(\tilde{x}(t)) \neq 0$ and zero elsewhere. Computing the integral (5-3) using such a weighting function would lead to non-zero results, indicating that such a solution could not be a solution of the weak form. Hence, the solution to the weak form must be equivalent to the solution of the strong form. □

A detailed proof of the above proposition can be found in [101]. This use of such weighting functions $\varpi(t)$ (as in the proof) essentially provides a way to test the value of residuals locally. Thus, $\varpi(t)$ are also commonly referred to as *test functions* or *trial function* in the literature, with space containing the test functions known as the *test space* or *trial space*.

Using an infinite-dimensional set of weighting functions ϖ_i and a similar infinite-dimensional set of basis functions β_i, both with $i \in \mathbb{I}_N$ and $N = \infty$, the weak form (5-3) produces the exact solution to the continuous problem. In practice, a finite N will be used to yield a finite-dimensional approximation for the state trajectory x; hence one can use no more than N weighting functions ϖ_i to form a system of N equations to solve for the unknown coefficients a_i.

Different choice of test functions can lead to different variants of weighted residual methods, for instance:

- **The Bubnov-Galerkin method:** with test functions chosen to be the same as the approximations basis functions, i.e.

$$\varpi_i = \beta_i = \frac{\partial \tilde{x}}{\partial a_i}, \text{ for } i \in \mathbb{I}_N.$$

  The method is commonly referred as the Galerkin method with Bubnov omitted.

- **The collocation method:** with test functions chosen to be Dirac delta functions centered at a set of discrete points t_i, i.e.

$$\varpi_i = \delta(t - t_i), \text{ with } \int_{\mathbb{T}_i} \delta(t - t_i)\, dt = \begin{cases} 1 & \text{when } t_i \in \mathbb{T}_i \\ 0 & \text{otherwise} \end{cases}, \text{ for } i \in \mathbb{I}_N.$$

- **The least-squares method:** with test functions chosen to be the partial derivatives of the residual with respect to the unknown coefficients, i.e.

$$\varpi_i = \frac{\partial \varepsilon}{\partial a_i}, \text{for } i \in \mathbb{I}_N. \tag{5-4}$$

- **The method of moments:** with test functions chosen from the family of polynomials, i.e.
$$\varpi_i = t^i, \text{ for } i \in \mathbb{I}_N.$$

Remark 1. *When the basis functions β_i are chosen to be polynomial functions, the method of moments is identical to the Galerkin method.*

Remark 2. *The sub-domain method is often discussed in the same context, however, it is not strictly a weighted residual method as the weighting is not explicitly used. The method can be regraded as an extension to the collocation method (hence also known as sub-domain collocation method), by also forcing the residuals to be zero over various sub-domains, with weight $\varpi(t)$ set to unity, i.e.*

$$\int_{\mathbb{T}} \varpi(t)\varepsilon(\tilde{x}(t))\, dt = \sum_{i=1}^{N} \left(\int_{\mathbb{T}_i} \varepsilon(\tilde{x}(t))\, dt \right) = 0.$$

The resultant set of equations can be implemented as the equality constraints (4-3b) so that the dynamic equations (1-1b)–(1-1c) in the DOP can be approximately satisfied. However, in practice, where a finite-dimensional approximation is made, the choice of test functions will have important implications on the error characteristics of the solution. Hence, this aspect will be explored further with the following numerical study.

5-3 Comparison between different weighted residual methods – a numerical study

Comparisons between different variants of weighted residual methods has been extensively studied in the literature, both in theoretical work on numerical finite element methods [27, 28, 54, 96, 97], in practical work on computational fluid dynamics [66, 112, 166], and in structural dynamics [74], focusing on solution accuracy and computational efficiency. Here, we focus to use a numerical example to demonstrate the characteristics of these methods.

Consider the following simple problem

$$\begin{aligned} \dot{x}_1(t) =& a_c x_1(t), \\ x_1(0) =& x_{1,0}. \end{aligned}$$

The analytical solution is $x_1(t) = x_{1,0}e^{a_c t}$. The approximate solution $\tilde{x}_1$ is obtained by the above mentioned methods using polynomial basis functions with both $N = 1$ (i.e. piecewise linear) and $N = 3$ (i.e. piecewise cubic). For $N = 1$ case with collocation method, we use the standard explicit Euler rule that the collocation point is at the start-point. Comparisons were made for the absolute errors $|x_1 - \tilde{x}_1|$ and the residual errors $|\dot{\tilde{x}}_1 - a_c\tilde{x}_1|$ for an interval $[0, h]$, which is representative for a mesh interval $[t_k, t_{k+1}]$ in Chapter 4, with the following metrics

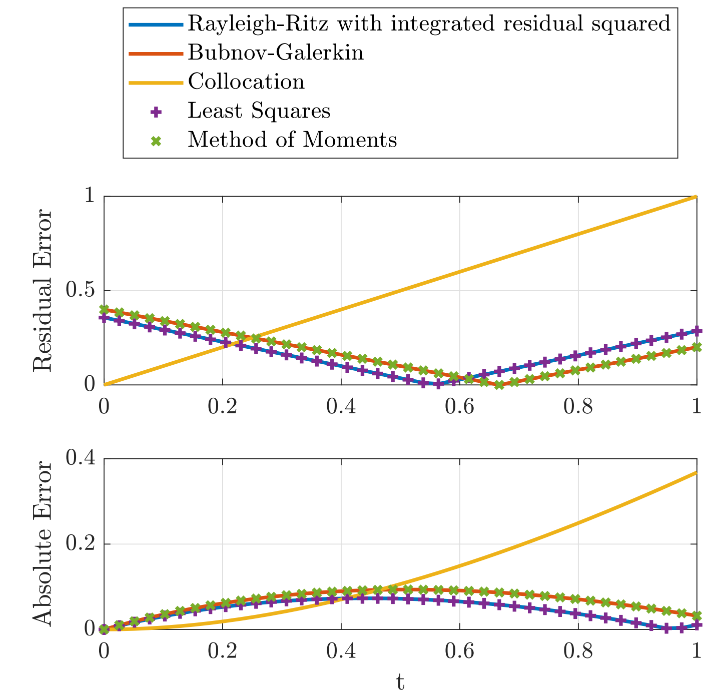
- Errors inside an interval with fixed $h = 1$ (Figure 5-1),
- Errors the end of the interval with different values of h (Figure 5-2),
- Square of the residual errors integrated along the interval with different values of h (Figure 5-3).

From Figure 5-1, large differences can be observed for the collocation scheme in comparison to the others. Firstly, the behavior of collocation scheme to force the residual error to be zero (or machine precision) can be clearly observed at $t = 0$ for the $N = 1$ case and $t = (0, 0.5, 1)$ in the case of $N = 3$. This characteristic, however, results in a larger absolute error in the approximation of x_1 with collocation. Starting with zero absolute error $x_1(0) = x_{1,0}$ at the start of the interval, the collocation scheme yields the highest absolute error at the end of the interval, in both cases of $N = 1$ and $N = 3$.

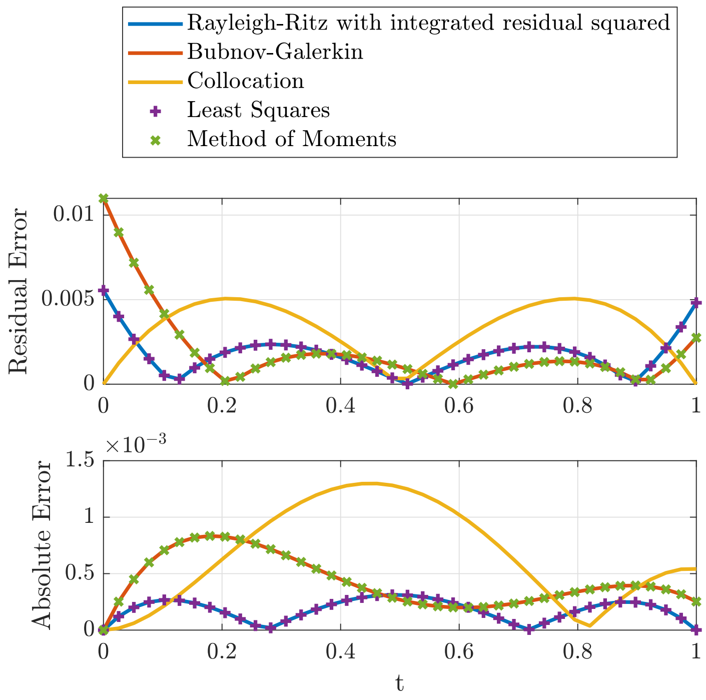
The results shown in the figure also support Remark 1, showing identical results for the Galerkin method and the method of moments when polynomial basis functions are used. Moreover, the numerical results further evidenced that the Rayleigh-Ritz method with the objective functional formulated as the integral of the residual squared (5-1) is identical to as the least-squares weighted residual method, both providing the most accurate solution at the end of the interval.

Continuing the comparison between different weighted residual methods, Figure 5-2 compared the errors at the end of the interval with different sizes of h. First focusing on the absolute error: it can be observed that the least-squares type of methods not only have the smallest error (expect for large values of h), the slope at which the absolute error reduces is also the steepest as the mesh becomes finer with decreasing h. Having such a small absolute error at the end of the interval is crucially beneficial for controlling the propagation of error in the solution trajectory.

In contrast, the residual error at a given point provides little indication of the accuracy of a solution. For $N = 3$ with the end-point being a collocation point, the residual error for collocation has shown to be negligibly small whereas the absolute error is, in fact, one of the highest. However, in the case of $N = 1$ with explicit Euler, collocation has the highest



(a) $N = 1$



(b) $N = 3$

Figure 5-1: Comparison of absolute and residual errors inside an interval $[0, 1]$ for various function approximation methods

magnitude for both the absolute and the residual errors at the end-point. This observation demonstrated that evaluating the residual error only at certain points is not sufficient to correctly reflect the level of accuracy.

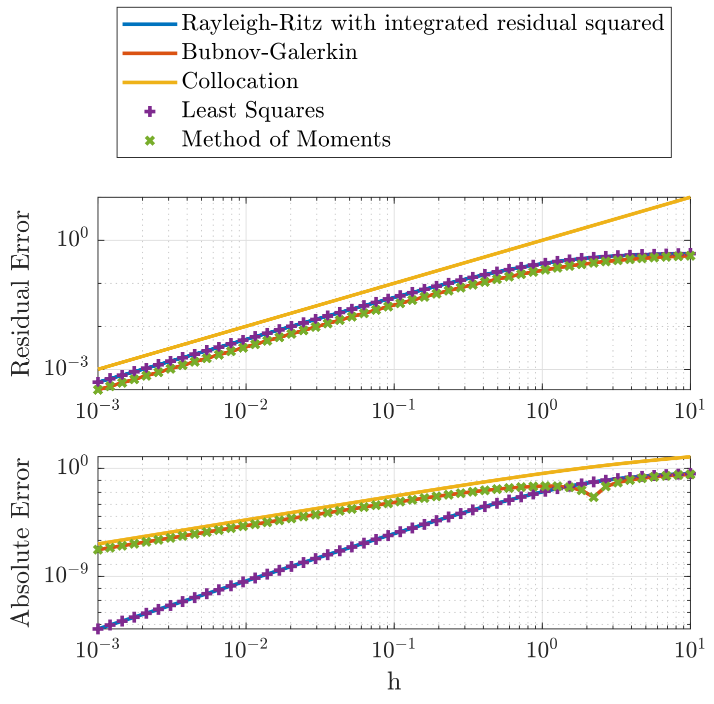
Figure 5-3 illustrates the IRNS error, i.e. $\int_0^h \varepsilon(\tilde{x}_1(t))^2 \, dt$. Here, a consistent result can be observed in both cases of $N = 1$ and $N = 3$, with the collocation method having the highest IRNS error and the least-squares class of methods (least squares and Rayleigh-Ritz with IRNS formulation) having the lowest error. It also shows that the use of residual error in the integral form is a much more suitable error metrics than only at certain points. For this simple example, increasing the basis function order to $N = 3$ would be sufficient to close the gap between the methods; however, with complex problems, the deficiency in solution accuracy for the collocation approach and the advantages in the least-squares approach may become more pronounced. This property can be consistently observed in later chapters when these methods are used for the numerical solution of DOPs.

Order of Convergence Using this opportunity, further explorations can be made regarding the order of convergence for different schemes, focusing on the order which integration error reduces with decreasing time step h. Based on available literature [83, Section II.7][82, Theorem 5.2], as h tends to zero, the error is eventually dominated by the order of the finite element polynomial, hence different methods should have equal rate.

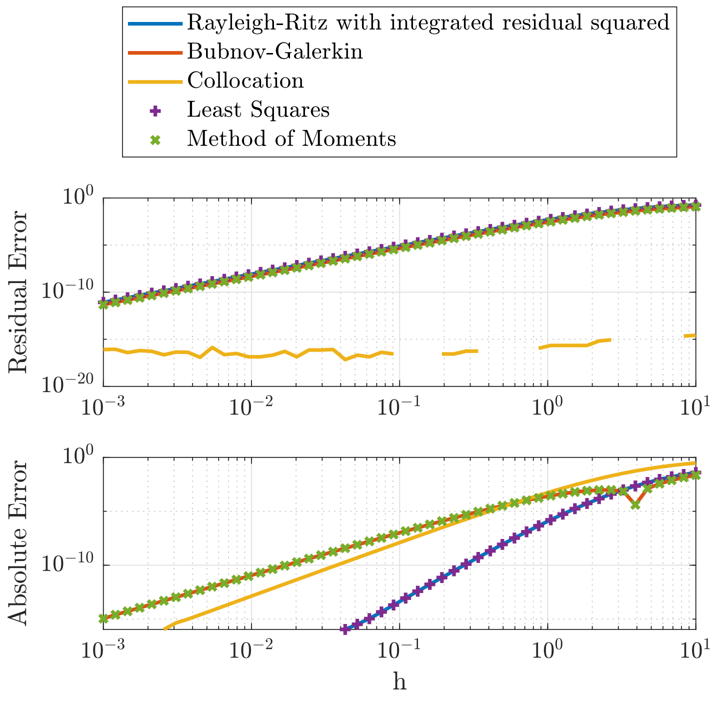
This analysis is in correspondence to the results obtained by this numerical example shown in Table 5-1, for the order of convergence in terms of the IRNS error. After removing the influence of the square operation for the integrand in (5-1), the actual rate of convergence for the integrated residual norm error is around 1 for case of $N = 1$ and 3 for case of $N = 3$, the same as the order of the finite element polynomial. Table 5-1 also shows that the rate of convergence for a different error metric can also be different. For the least-squares approach, the rate of convergence for the absolute error measured at the end of the interval is, in fact, significantly higher than the other two alternatives. Such benefit, however, can be problem dependent in practice.

Table 5-1: Approximate order of convergence for different weighted residual schemes with the example problem

N	**Method**	**IRNS error** $\int_0^h \varepsilon(\tilde{x}_1(t))^2\,dt$	**Absolute Error** $\lvert x_1(t=h) - \tilde{x}_1(t=h)\rvert$
1	Collocation	3	2
	Least-squares	3	4
	Bubnov-Galerkin	3	2
3	Collocation	6 to 7	4 to 5
	Least-squares	6 to 7	7 to 8
	Bubnov-Galerkin	6 to 7	4

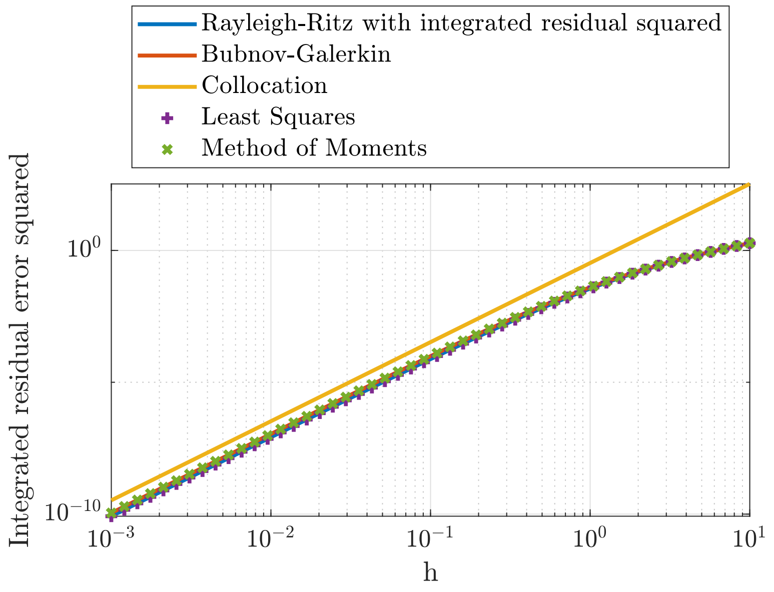


(a) $N = 1$

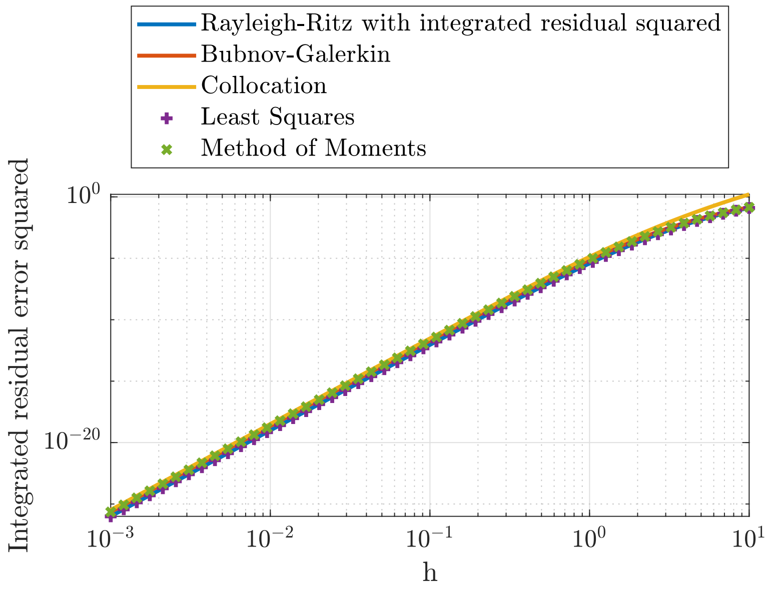


(b) $N = 3$

Figure 5-2: Comparison of absolute and residual errors at the end of an interval $[0, h]$ for different values of h and various function approximation methods. The residual error for collocation have gaps when the value is below machine precision.



(a) $N = 1$



(b) $N = 3$

Figure 5-3: Comparison of the square of the residual error integrated along the interval $[0, h]$ for different sizes of h and various function approximation methods

Chapter 6

Direct Collocation Method[†]

Direct transcription method of direct collocation is arguably the most widely used numerical method for the solution of DOPs. Details of the method are available in various literatures [17, 58, 162] hence this chapter will only provide a brief overview, with special focus on the characteristics of the direct collocation solutions.

6-1 Direct collocation transcription

For the collocation weighted residual method, the test functions are selected to be Dirac delta functions, leading to $n + n_g$ equality constraints to be applied to each of the $N^{(k)}$ data points. The Dirac delta functions possess the *isolation property*, namely that the integral of the function on an interval is zero, except the intervals that contain the center of the function, where the integral equals to 1. Therefore information needed to evaluate a constraint equation at a data point will be fully independent from information corresponding to other data points, contributing to the computational efficiency of the direct collocation method.

The other simplification commonly made in direct collocation is to also use the same data point definition for both the quadrature points in the numerical integration of the Lagrange cost and the points where path constraints (1-1d) are forced to be satisfied. As a result, the data points would be sufficient for the transcription of the problem to an DDOP, hence they are also known as the *collocation points*.

This chapter contains materials directly from the following paper:
Y. Nie and E. C. Kerrigan. Solving Dynamic Optimization Problems to a Specified Accuracy: An Alternating Approach using Integrated Residuals. Under review.

With the collocation weighted residual method, the resultant equality constraints from (5-3) with a finite-dimensional approximation is

$$\sum_{j=1}^{N^{(k)}} \mathcal{A}_{ij}^{(k)} \chi_j^{(k)} + \mathcal{D}_{ij}^{(k)} f\left(\chi_j^{(k)}, \upsilon_j^{(k)}, t_0, t_f, p\right) = 0, \tag{6-1a}$$

$$g\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) = 0, \tag{6-1b}$$

with $\dot{\chi}_i^{(k)} := \dot{x}^{(k)}\left(\tau_i^{(k)}\right) \in \mathbb{R}^n$. $\mathcal{A}^{(k)} \in \mathbb{R}^{N^{(k)} \times N^{(k)}}$ is a discretization-dependent constant matrix, where $\mathcal{A}_{ij}^{(k)}$ is element (i, j) of the matrix, and $\mathcal{D}^{(k)} \in \mathbb{R}^{N^{(k)} \times N^{(k)}}$ is a matrix containing time variables. Structures of both $\mathcal{A}^{(k)}$ and $\mathcal{D}^{(k)}$ depends on the chosen discretization scheme.

In the DDOP(4-3), (4-3b) needs to contain (6-1a)–(6-1b) for the dynamic equations to be approximately fulfilled. Also, (4-3c) is chosen such that the inequality constraints are equivalent to

$$c\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) \leq 0.$$

To sum up, the DDOP arising from direct collocation transcription of DOP (1-1) is

$$\min_{\mathcal{Z}} \Phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) + \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} w_i^{(k)} L\left(\chi_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) \tag{6-2a}$$

subject to, for $i \in \mathbb{I}_{N^{(k)}}$ and $k \in \mathbb{I}_K$,

$$\sum_{j=1}^{N^{(k)}} \mathcal{A}_{ij}^{(k)} \chi_j^{(k)} + \mathcal{D}_{ij}^{(k)} f\left(\chi_j^{(k)}, \upsilon_j^{(k)}, t_0, t_f, p\right) = 0, \tag{6-2b}$$

$$g\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) = 0, \tag{6-2c}$$

$$c\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) \leq 0, \tag{6-2d}$$

$$\phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) = 0, \tag{6-2e}$$

as well as any necessary continuity constraints.

The discretized problem (6-2) can be solved with off-the-shelf NLP solvers. In special cases where the objective (6-2a) is linear or quadratic, and all constraints (6-2b)–(6-2d) are linear, a QP solver can be used to handle the numerical solution process more efficiently. Note that for defect constraint (6-2b) to be linear for a linear function f, $\mathcal{D}$ should only contain constants thus the problem formulation must have a fixed mesh design with fixed initial and final time, i.e. t_0 and t_f are excluded from the decision variables.

6-2 Characteristics of the direct collocation solutions

In essence, direct collocation forces the residuals $\varepsilon^{(k)}$ to be zero at all collocation points. It is well-known in the field of approximation theory that if a function cannot be represented exactly by a polynomial, forcing the approximating polynomial to exactly go through some sampled data points generally results in larger errors for the function values between the data points than other methods, such as least-squares fitting. Similarly, direct collocation will generally result in large residuals in-between collocation points, regardless of the distribution and spacing of these points.

Since most direct collocation methods employ absolute or relative local error as the error metric, a mismatch arises between the error measures in the problem formulation and the error criteria for a solution to be sufficiently accurate in posterior error analysis:

- when formulating the DDOP, satisfaction of dynamic constraints are based on residuals $\underline{\text{at collocation points}}$, whereas
- during the error analysis of the solution, satisfaction of dynamic constraints are based on the norm of residuals $\underline{\text{integrated along intervals in-between collocation points}}$.

As direct consequences,

- regardless of how small the solver tolerance is, solving the DDOP once on a single given discretization mesh will provide no guarantee in terms of solution accuracy and constraint satisfaction, hence
- posterior procedures such as error analysis and mesh refinement must be considered as an indispensable part of a direct collocation method in order to ensure convergence and solution accuracy.

This is a point of caution often not realized by non-experts. The development of the integrated residual class of methods fundamentally addresses the problems arising from this error metric mismatch, by working with the residuals in integrated form also in the solution process.

Chapter 7

Direct Integrated Residual Methods[†]

In this chapter, a novel direct transcription and solution method for solving nonlinear, continuous-time DOPs is proposed. Instead of forcing the dynamic constraints to fulfill only at a selected number of points as in direct collocation, this approach minimises or constraints the squared norm of the dynamic constraint residuals integrated along the whole solution trajectories. As a result, the method can 1) obtain solutions of higher accuracy for the same mesh compared to direct collocation methods, 2) enables a flexible trade-off between solution accuracy and optimality, 3) provides reliable solutions for challenging problems, including those with singular arcs and high-index differential algebraic equations.

In this chapter, the concept of IRM is first introduced in Section 7-1, with insights on its connection to the direct collocation approach. Subsequently, in Section 7-2, the motivations for the development of the DAIR scheme is given, together with discussions on its formulation and implementation strategies. This is followed by a number of classical examples in Section 7-3, where different aspects of the method are demonstrated.

7-1 Key concepts of integrated residual methods

In the field of approximation theory, the least squares criterion is often considered as a more suitable choice than forcing the fitting error to be exactly zero only at some selected points [23].

[†]This chapter contains materials directly from the following papers:
Y. Nie and E. C. Kerrigan. Solving Dynamic Optimization Problems to a Specified Accuracy: An Alternating Approach using Integrated Residuals. Under review.
Y. Nie and E. C. Kerrigan. Efficient and more accurate representation of solution trajectories in numerical optimal control. *IEEE Control Systems Letters*, 4(1):61–66, Jan 2020 ©2019 IEEE

Before exploring the implementation of the least-squares approach for the solution of DOPs, it is worthwhile to first look at the use of such a method in solving dynamic equations in the form of ODEs and DAEs. For the reminder of this work, the MIRNS error introduced in Section 4-6 will be used as the error metric, however other variants of IRNS error may be selected also.

Following the Rayleigh-Ritz approach, finding an approximate solution of the dynamic equations is equivalent to the following optimization problem that minimizes the MIRNS error:

$$\min_{\chi,\upsilon,p,t_0,t_f} \frac{1}{\Delta t} r(\tilde{x},\tilde{u},t_0,t_f,p) \tag{7-1a}$$

subject to any continuity and boundary constraints, for example,

$$\tilde{x}^{(k)}\left(t_{k+1}\right) = \tilde{x}^{(k+1)}\left(t_{k+1}\right), \tag{7-1b}$$

$$\phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) = 0. \tag{7-1c}$$

The least squares approach as defined in the class of weighted residual methods with (5-4) is equivalent to applying the optimality condition (5-2) and obtaining a number of equality constraints to be satisfied. However, this condition is only a first order necessary condition and thus theoretically can only guarantee that the trajectory is a stationary solution in general, i.e. the trajectory could be a local maximum for the MIRNS error. In addition, using only the optimality conditions will not be able to provide indications on the magnitudes of the errors.

Hence, this work focuses on the development of methods that directly solve the optimization problem (7-1a) instead. Although the solutions obtained this way will still be local and depended on the initial guess (as the optimization problem is typically nonlinear and non-convex), it is however now possible to evaluate the error magnitudes during the solution process instead of a posteriori. Therefore, when designing numerical schemes, mechanisms can be put in place to ensure the integrated residual errors are below some specified requirements. For example, the schemes proposed in Section 7-2 make use of inequality constraints to bound the error.

In error metrics, such as r (4-8) used here, the calculation of the norm effectively introduces relative trade-offs for the accuracy between the dynamic equations. Although the original expression works well when all variables are scaled to the same numerical range, there exist situations where it may be beneficial to specify additional weighting terms with diagonal matrix $\mathcal{W} \in \mathbb{R}^{(n+n_g)\times(n+n_g)}$ for the corresponding dynamic equations. In practice, one often knows beforehand that the modeling of some relationship (e.g. between acceleration and velocity) will have a higher confidence level than the modeling of some other dynamics (e.g. relationship between gas peddle position and acceleration). In these cases, it would be preferable to formally specify what would be the desired trade-off in terms of accuracy for different dynamic equations.

7-1-1 Numerical integration with quadrature rules

For certain simple problems, r may be expressed analytically for precise computation. However, for the majority of practical problems, numerical integration with quadrature rules of sufficiently high order can be used, i.e. (7-1a) can be replaced by

$$\min_{\chi,\upsilon,p,t_0,t_f} \frac{1}{\Delta t} \sum_{k=1}^{K} \mathcal{R}\left(\chi^{(k)}, \upsilon^{(k)}, t_0, t_f, p\right) \tag{7-2a}$$

with

$$\mathcal{R}\left(\chi^{(k)}, \upsilon^{(k)}, t_0, t_f, p\right) := \sum_{i=1}^{Q^{(k)}} w_i^{(k)} \left\|\mathcal{W}\varepsilon(q_i^{(k)})\right\|_2^2 . \tag{7-2b}$$

7-1-2 Least squares method for solving the DOP

When solving the DOP using the least squares approach, instead of just solving the differential equations, it is important to address the relationship between the requirement to minimize the integrated residual (7-2a) and the desire to minimize the original objective (1-1a). With an indirect approach, the optimality system for the DOP can be formulated and subsequently solved using least-squares finite element methods [25, 26].

For direct transcription, recently proposed penalty-barrier finite element method (PBF) [149] formulates an augmented objective consisting of the original objective, the MIRNS error as a penalty term and inequality constraint violations as an integrated logarithmic barrier term. The authors were able to prove convergence of their method provided that the functions that define the problem satisfies appropriate boundedness and Lipschitz conditions.

Based on the same concept of minimizing the integrated residual, a solution representation method [156] is proposed, able to obtain solutions of much higher accuracy than collocation methods, while maintaining non-increasing objective values. This approach effectively treats computing an approximate solution of the DOP as a multi-objective optimization problem. This approach is further extended into a stand-alone scheme for the solution of DOPs, and the new method will be addressed in detail in Section 7-2. Here, the method's relationship to direct collocation and the characteristics of this class of methods will be discussed first.

7-1-3 Relationship to direct collocation

In general, IRM-type methods are considered as a different approach to collocation. Here, a different perspective can be shown, namely that direct collocation can be considered as a special case of IRM.

Proposition 2. *In the direct collocation formulation, the enforcement of dynamic constraints with* (6-1a)–(6-1b) *is equivalent to the solution of a special case of problem* (7-2)*, with the quadrature points* $q_i^{(k)}, \forall i \in \mathbb{I}_{Q^{(k)}}$ *in each interval selected to be the same as the data points* $\tau_i^{(k)}, \forall i \in \mathbb{I}_{N^{(k)}}$ *of that interval.*

Proof. For equality constraints (6-1a)–(6-1b) to be fulfilled, the residuals $\varepsilon(t)$ evaluated at collocation points $\tau_i^{(k)}$ for all $i \in \mathbb{I}_{N^{(k)}}$ and $k \in \mathbb{I}_K$ will all be zero. When the quadrature points chosen for the IRM problem (7-2) match the data points, the IRM problem will have the optimal solution with $\mathcal{R}^* = 0$, since the corresponding residuals $\varepsilon(t)$ at these data points can be forced to zero altogether. This is equivalent to enforcing (6-1a)–(6-1b). □

Hence, the enforcement of dynamic constraints in direct collocation can be interpreted as an IRM method where in (7-2) the quadrature points are chosen to be the same as the data points. The quadrature order with this choice is not sufficiently high, in general. Hence, large errors may occur between the collocation points, which will not be reflected in any convergence and error measure of the underlying NLP. In other words, successfully solving the direct collocation NLP to very small tolerances does not guarantee an accurate solution. A graphical illustration of the differences in error characteristics with sufficiently high order quadrature for IRM method is shown in Figure 7-1.

7-1-4 Improved DAE handling

For direct collocation, if DAE equations exist as part of the dynamics, in addition to fulfilling all other constraints, there may not always be sufficient remaining degrees of freedom to additionally satisfy equation (6-2c) for all $i \in \mathbb{I}_{N^{(k)}}$, causing convergence issues for the NLP solver. The opposite could happen as well, with degree of freedoms not uniquely defined by the constraints, leading to multiple or even an infinite number of solutions. In this case, significant fluctuations will occur in the obtained solution. If the original continuous-time DOP is consistent, inconsistencies as described above would be attributed to the constraint discretization process, leading to either an *over-constrained* or *under-constrained* NLP.

Direct collocation methods are known to struggle for high-index DAE systems [17], and systems with consistently over-determined constraints. In fact, without implementing the scheme with special considerations, direct collocation could also lead to undesirable numerical solution with low-index DAE systems, e.g. as demonstrated in the second example of [130, Section 6.2].

Consistently over-determined constraints describe the situation where redundant constraints are present the continuous DOP formulation, to force the solution to lie on a manifold. A

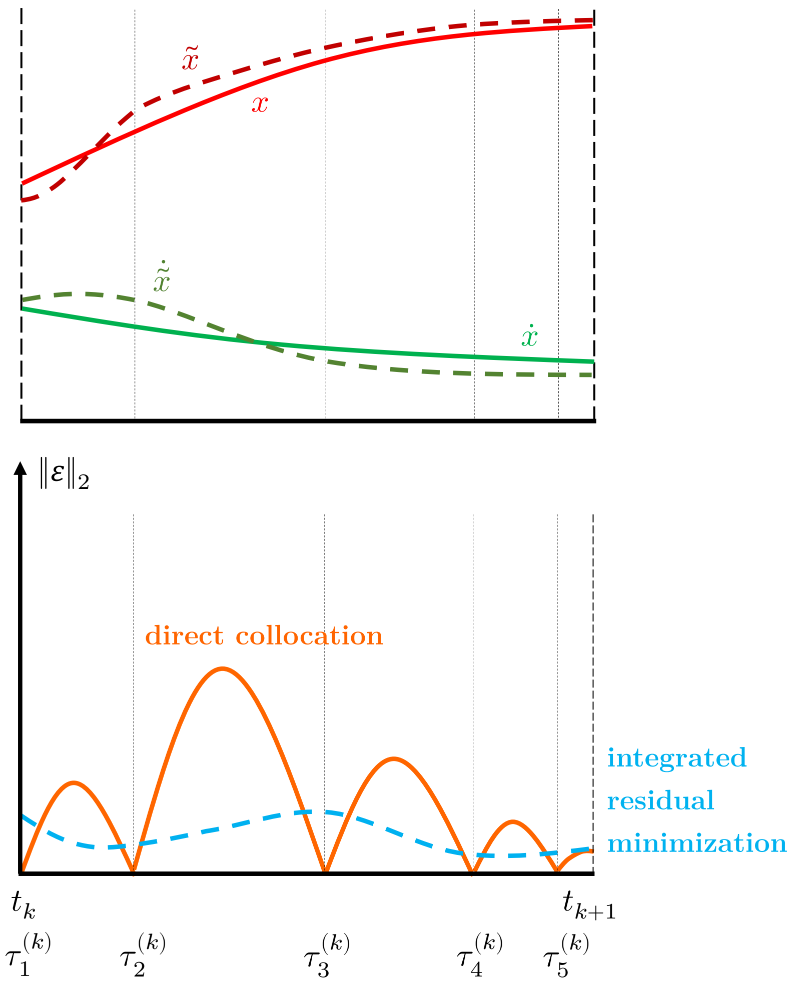


Figure 7-1: Illustration for the differences in error characteristics between a collocation scheme and a minimized integrated residual method inside a mesh interval. Collocation methods force the residual errors to zero at collocation points, but the errors between collocation points can be large. IRM-type methods minimizes some variations of the residual error integrated along the whole trajectory.

commonly encountered situation would be when the ODE equations already implicitly enforces the preservation of relevant quantities, however another DAE constraint of conservation equation is enforced at the same time but at a different integration level. For instance, in three-dimensional mechanical systems with quaternions: in addition to implicitly determined forces, an additional equation may be enforced to constraint the solutions of the quaternion states to lie on a unit sphere.

While the infinite-dimensional formulation remains consistent, the discretized problem will not necessarily be, due to differences in discretization and integration errors. With these types of problems, convergence of the NLP solver may significantly deteriorate or become unachievable. In these situations, modification to the dynamics are often made to avoid over-constrained NLPs, either by relaxing the redundant constraints or by making use of techniques such as symplectic integrators [43] and Baumgarte stabilisation [11] to ensure the preservation of the relevant quantity.

IRM-type of schemes can address these challenging cases out-of-the-box, i.e. without the need of additional care in the transcription process as in the case of direct collocation. This is because the fundamental ideas of these ad hoc fixes proposed for direct collocation are embedded in the key concepts of IRM. For instance, the proposed solution for the second example of [130, Section 6.2] with direct collocation is to control the residual error between mesh nodes by enforcing additional *residual constraints*, which are inequality constraints that use *'an error norm to control the overall error within the element for all the differential equations'*. This solution resembles many similarity to IRM, which minimizes and/or constrains the residuals integrated based on high order quadrature rules. Similarly, for cases of high-index DAEs and systems with consistently over-determined constraints, IRM based methods allows small residuals for the constraints to exist during the solution process similar to integrated penalty methods [64], hence able to automatically relax the redundant constraints and yielding better convergence properties.

7-1-5 Suppression of singular arc fluctuations

As explained in Section 4-6, in the parameterisation of DOPs, the representation of the state and input trajectories can rarely be made exact, hence approximation errors are generally unavoidable. This provides a unique opportunity for IRM-type transcription methods to automatically suppress potential singular arc fluctuations, without the need for additional treatments.

Due to the existence of approximation errors, and the multi-objective nature of IRM methods for solution of DOPs, different solution candidates on the singular arc with negligible differences from the objective point of view can now be ranked by the error. Larger fluctuations in the solution generally lead to bigger errors along the trajectory when approximated by a parameterised function, therefore a solution with the smallest fluctuations is often the most accurate solution in the IRM residual error metrics. This is the key reason behind the suppression of singular arc fluctuation with IRM-type transcriptions.

A special case would be when the choice of basis functions is capable of representing the solution trajectories exactly. Typical examples would be when system dynamics are purely integrators. In this case, it is possible for a singular arc trajectory with large fluctuations to possess no residual errors, making the IRM theoretically unable to suppress the singular arc directly.

However, the issue with fluctuating solutions being exact is only of concern for a converged optimal solution. It is unlikely that the solve of DOP will be initialized with such solution as an initial guess, hence in practice, errors in solution trajectory presented in the intermediate iterations will help IRM methods to converge to a solution with singular arc fluctuations suppressed. Therefore, in cases where trajectories being represented exactly by the parameterized function, although IRM can not guarantee the suppression of singular arc theoretically, the method can still leads to a good results in practice.

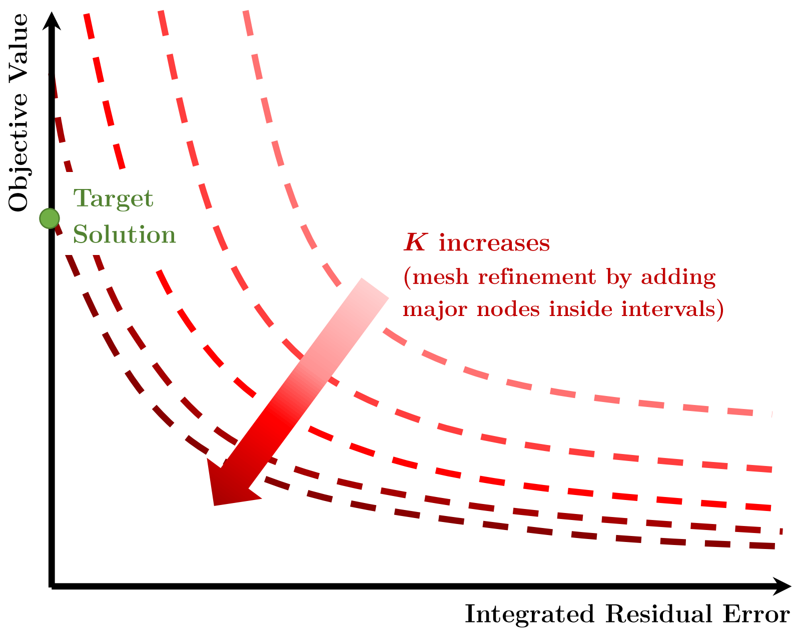
7-2 Direct alternating integrated residual method

Though PBF has a number of advantages over direct collocation in terms of solution accuracy and robust handling of some difficult problems, PBF is still a method developed focusing on off-line solution of dynamic optimization problems. Illustrated in Figure 7-2a, these types of problems often have one target solution that the DOP algorithm is searching for, namely the solution with the smallest objective value among the ones that contain the lowest possible error. In other words, one wants the convergence of the objective ($J \to J^*$) and constraint satisfaction ($r, \eta, \epsilon \to 0$) at the same time, as the discretization mesh becomes denser (K increases).

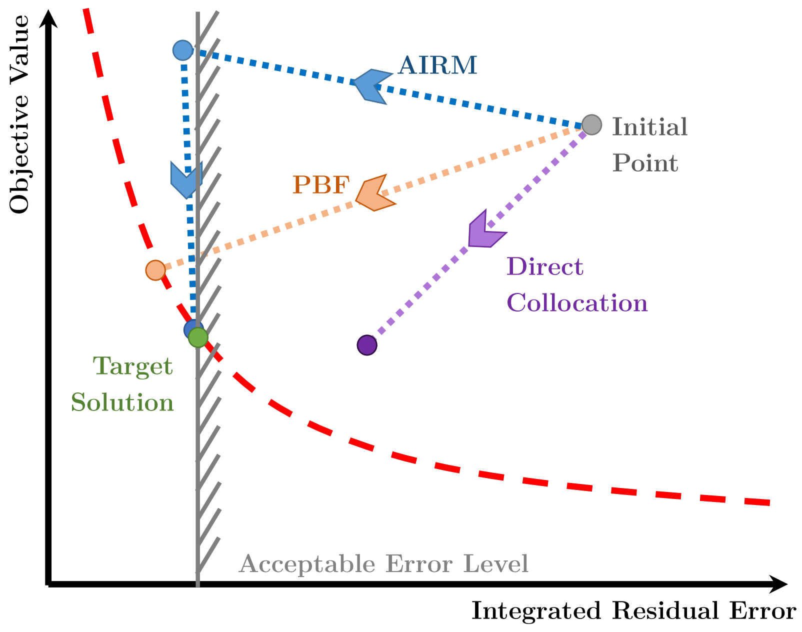
The picture is different when considering a single solve of the DOP on a given discretization mesh, especially considering on-line NMPC applications. Firstly, the nature of solving the DOP numerically will become a multi-objective problem in this case, leading to an inevitable trade-off between minimizing the objective and reducing the residual error. This often indicates that, in practice, the target solutions will not be the ones that lie on the far ends of the Pareto front.

Evaluating preferences among various solutions on the Pareto front depends on other criteria, e.g. the closed-loop performance of the NMPC controller. Details on how such a trade-off can be made are available in other work, e.g. [118]. Here we directly take the outcome of this decision-making process: an error level under which the solution accuracy can be considered acceptable. The original multi-objective optimization problem can then be translated into a single objective one, with the target solution being the one that minimizes the objective value, while satisfying the constraints concerning the acceptable error level.

Figure 7-2b illustrates the solution process of direct collocation and the PBF method for a given mesh size and discretization method. As long as the initial point and mesh design do not change, the solution obtained with direct collocation methods will not change. From earlier discussions, it can be seen that, regardless of whether this solution satisfies the acceptable error level, it is very unlikely in practice to be a solution that resides on the Pareto front. In other words, one aspect of a direct collocation solution can be improved without deteriorating


Objective Value
Target Solution
K increases
(mesh refinement by adding major nodes inside intervals)
Integrated Residual Error

(a) Trajectory optimization



(b) Single solve on fixed mesh (e.g. for NMPC)

Figure 7-2: Illustration for the differences in solving trajectory optimization problems offline and solving DOPs on a single fixed mesh

the other aspect.

In contrast, PBF is capable of finding solutions on the Pareto front; however, controlling which solution it will terminate at would require careful selection of parameters for the penalty and barrier terms. Therefore, even if the sub-iterations can be computed efficiently, additional challenges are associated with the PBF method to converge easily to the target solution, given a specified acceptable accuracy level. The proposed direct alternating integrated residual (DAIR) method aims to address these challenges and provide a reliable and efficient approach.

7-2-1 Elementary formulations

The elementary formulations of the DAIR method consist of two problems: minimizing the MIRNS error and minimizing the objective subject to integrated residual error constraints, denoted as the *DAIR residual minimization problem* and *DAIR cost minimization problem*, respectively. The method retains the same decision variables as in (4-3), namely $\mathcal{Z} := (\chi, \upsilon, p, t_0, t_f)$, and uses the interpolation polynomial formula $\tilde{x}(\,\cdot\,)$, $\dot{\tilde{x}}(\,\cdot\,)$ and $\tilde{u}(\,\cdot\,)$ for the computation and integration of various elements of the discretized DOP.

The DAIR residual minimization problem has the formulation

$$\min_{\chi,\upsilon,p,t_0,t_f} \frac{1}{\Delta t} \sum_{k=1}^{K} \mathcal{R}\left(\chi^{(k)}, \upsilon^{(k)}, t_0, t_f, p\right) \tag{7-3a}$$

subject to, for all $i \in \mathbb{I}_{N^{(k)}}$ and $k \in \mathbb{I}_K$,

$$c\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) \leq 0, \tag{7-3b}$$

$$\phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) = 0, \tag{7-3c}$$

and optionally any continuity constraints in the form of (7-1b), as well as optionally one or more constraints from the following constraints regarding upper limits for the objective $J_c \in \mathbb{R}$:

$$\sum_{k=1}^{K} \sum_{i=1}^{Q^{(k)}} w_i^{(k)} L\left(\tilde{x}^{(k)}\left(q_i^{(k)}\right), \tilde{u}^{(k)}\left(q_i^{(k)}\right), t_0, t_f, p\right) + \Phi\left(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p\right) \leq J_c, \tag{7-3d}$$

and the mean integrated residual squared (MIRS) error for individual dynamic equations $\varrho \in \mathbb{R}_{\geq 0}^{(n+n_g)}$:

$$\sum_{k=1}^{K} \sum_{i=1}^{Q^{(k)}} \frac{w_i^{(k)}}{\Delta t} (\varepsilon_\xi^{(k)}(q_i^{(k)}))^2 \leq \varrho_\xi, \text{ for } \xi = 1, \ldots, n + n_g, \tag{7-3e}$$

with ϱ_ξ the ξ^{th} element in ϱ.

The counterpart, the DAIR cost minimization problem, is

$$\min_{\chi,\upsilon,p,t_0,t_f} \Phi(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p) + \sum_{k=1}^{K} \sum_{i=1}^{Q^{(k)}} w_i^{(k)} L\left(\tilde{x}^{(k)}\left(q_i^{(k)}\right), \tilde{u}^{(k)}\left(q_i^{(k)}\right), t_0, t_f, p\right) \tag{7-4}$$

subject to (7-3b), (7-3c), (7-3e) and optionally (7-1b), for all $i \in \mathbb{I}_{N^{(k)}}$ and $k \in \mathbb{I}_K$.

In terms of the accuracy of the solution, the above two problems can be considered as a practically balanced approach: DAIR is more reliable than direct collocation because it minimizes the MIRNS error for the dynamic equations. DAIR is easier to implement and solve than the PBF method, because DAIR avoids the need to introduce a sequence of weights for penalty and barrier terms and choosing an appropriate, tailored NLP solver; the NLPs in DAIR can be solved using most off-the-shelf solvers.

Note that inequality constraints are chosen to be enforced at polynomial data points only (similar to direct collocation), and existing constraint tightening techniques (e.g. those discussed in [68]) may be applied when necessary. This is an efficient choice for numerical computations, but is without loss of generality; the DAIR framework allows discretized inequality constraints to be enforced anywhere along the trajectory.

7-2-2 Implementation strategies

Standalone direct transcription method

Based on the elementary formulations, the DAIR framework can be implemented as a standalone method for solving the DOP numerically on a fixed discretization mesh, with one example illustrated in Figure 7-3.

The first step is to select a discretization method, design the mesh and (optionally) determine the weighting parameter for the residual norm computation. Also, the required accuracy level for each dynamic equation needs to be specified in the form of a MIRS error. The idea is to first solve the DAIR residual minimization problem to yield a solution that would determine the MIRS error upper bound for the DAIR cost minimization problem formulation.

For the DAIR residual minimization, a set of criteria can be specified in the NLP solver to terminate early once all MIRS errors are within requirement and all other constraints satisfied. This would indicate the existence of solutions for this discretization mesh that would fulfill all accuracy requirements, and the DAIR cost minimization problem can be solved subsequently with ϱ configured accordingly.

On the other hand, if the required MIRS errors are not achievable for this mesh design, early termination will not be triggered and the DAIR residual minimization problem will be fully solved. From the solution, one can extract the smallest MIRS error achievable for

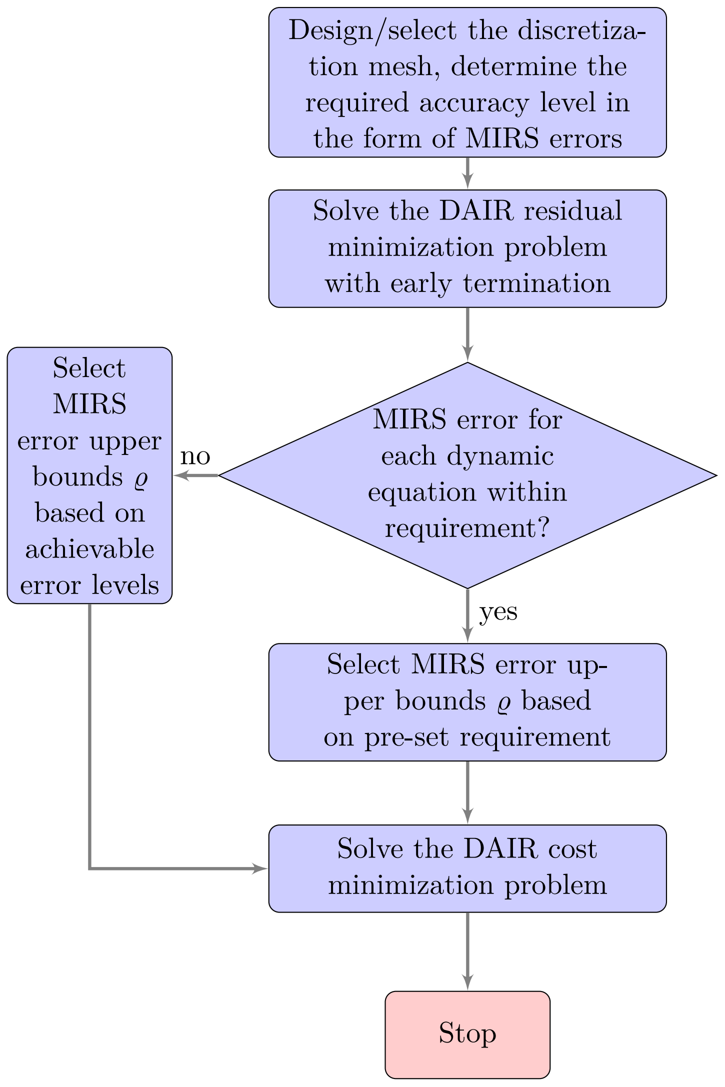

Design/select the discretization mesh, determine the required accuracy level in the form of MIRS errors
Solve the DAIR residual minimization problem with early termination
MIRS error for each dynamic equation within requirement?
no
Select MIRS error upper bounds ϱ based on achievable error levels
yes
Select MIRS error upper bounds ϱ based on pre-set requirement
Solve the DAIR cost minimization problem
Stop

Figure 7-3: Overview of DAIR scheme as a standalone method for solving DOP numerically on a given discretization mesh

any corresponding dynamic equation for which the original requirement cannot be met, and implement this achievable MIRS error to ensure the existence of feasible solutions for the DAIR cost minimization problem. In other words, the DAIR cost minimization problem is guaranteed to have at least one feasible solution, namely the solution at which the DAIR residual minimization problem terminates.

Solution representation method

In early work [156], an optimization formulation for representing continuous DOP trajectories with higher accuracy from discretized direct collocation NLP solutions is proposed. The DAIR residual minimization problem would be a more suitable candidate for the purpose of solution representation, with the following benefits:

- there is a flexible trade-off between the level of accuracy for different dynamic equations.
- DAIR allows an upper limit for both the objective and MIRS errors for individual dynamics to be set. This guarantees that the obtained trajectory will be no worse than the collocation solution in terms of optimality and accuracy.

All results shown in Section 7-3 use the DAIR formulation.

Other implementation potentials

The flexibility of the DAIR scheme could enable the formulation of various implementation procedures that are based on it, for a wide range of applications. For example, the scheme can be designed for efficient and accurate solution of dynamic optimization problems, both on-line and off-line, when used together with a suitable mesh refinement/adaptation scheme. For implementation, it is possible to either

- first solve a sequence of DAIR residual minimization problems on a sequence of refined meshes until all convergence tolerances are met regarding feasibility and accuracy, and then solve a single DAIR cost minimization problem for optimality, or
- solve the DAIR residual minimization problem and DAIR cost minimization problem in an alternating manner, converging to the target solution as the mesh becomes denser.

The DAIR scheme can potentially lead to more efficient mesh refinement procedures than that for direct collocation. This is especially beneficial for on-line NMPC implementations, where the solution accuracy cannot be ensured with a single mesh that has been designed off-line; state-of-the-art NMPC software adopt mesh refinement schemes or adaptive differential equation solvers in order to ensure that the parameterized solution trajectories are sufficiently accurate.

7-3 Example problems

To demonstrate the advantages of the IRM and DAIR transcription over direct collocation, a number of example problems are prepared focusing on different aspects. In transcription with ICLOCS2, both state and input trajectories are continuous trajectories inside a single phase. All examples and all methods are demonstrated under this configuration.

7-3-1 Goddard rocket

In the first example, different implementations of the Goddard rocket problem [17, Ex. 4.9] are compared. The DOP has the following formulation

$$\min_{h,v,m,T,t_0,t_f} -h(t_f)$$

subject to

$$\begin{aligned}
\dot{h}(t) &= v(t), \\
\dot{v}(t) &= \frac{1}{m(t)}\left(T(t) - \sigma v^2(t)\exp\left(-\frac{h}{h_0}\right)\right) - g, \\
\dot{m}(t) &= -\frac{T(t)}{c}, \\
0 \leq T(t) &\leq T_{max},
\end{aligned}$$

with state variables h the altitude, v the velocity, m the mass, and input variable T the thrust of the rocket engine. The initial and terminal conditions, as well as values of static parameters, are available in the reference hence will not be reproduced here.

The optimal solution is in the form of *bang-singular-bang* and, on the singular arc, it is known for the solution to be oscillatory when solved directly with a single phase numerical solver. Such fluctuations can be clearly seen in Figure 7-4. The conventional way of dealing with singular control problems is to introduce additional conditions once the solution structure is known. Despite yielding an accurate solution, this method, however, would normally require a multi-phase formulation support and analytical derivations of the *singular arc conditions* [17].

By taking care of the errors in solution trajectories between polynomial data points, both the proposed DAIR scheme and IRM solution representation method derived from the DAIR residual minimization problem are capable of reproducing this multi-phase solution using the original single phase formulation. This is illustrated in Figure 7-4 with two small oscillations due to approximating the discontinuous optimal input trajectory with a continuous trajectory, whereas the multi-phase setup allows for a discontinuous input.

7-3-2 High-index DAE system

To demonstrate the advantages of DAIR in dealing with high-index DAE systems, the example from [39, equation system 3] with a DOP derived from a pendulum system containing a index-3 DAE is used. The problem formulation is

$$\min_{x_1,x_2,x_3,x_4,x_5,u_1} \int_{t_0}^{t_f} cu_1(t)^2 + d\left(x_1(t) - L\sin(t+\alpha)\right)^2 + d\left(x_3(t) - L\cos(t+\alpha)\right)^2 \, dt$$

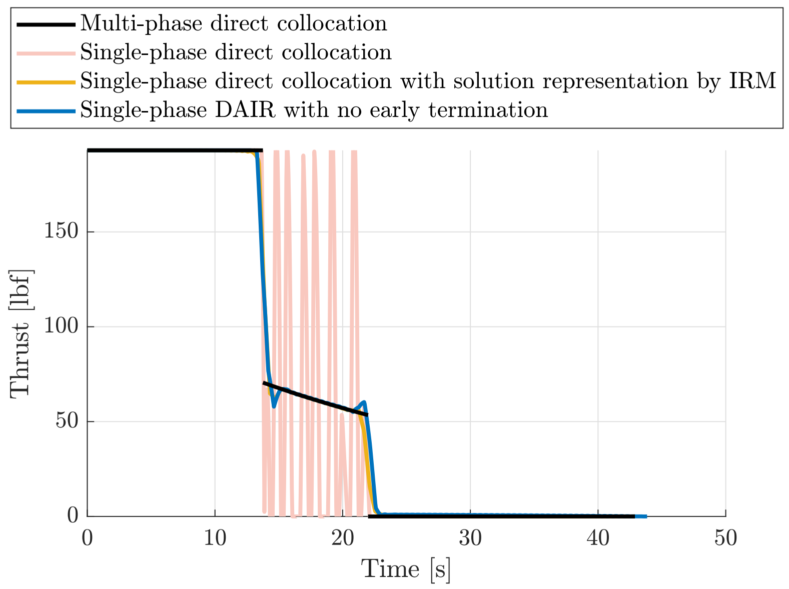


Figure 7-4: Comparison of the solutions for the Goddard rocket problem (HS/piecewise cubic parameterization with 100 major nodes)

subject to

$$
\begin{aligned}
\dot{x_1}(t) &= x_2(t), \\
\dot{x_2}(t) &= -x_5(t)x_1(t) - ax_2(t) + u_1(t)x_3(t), \\
\dot{x_3}(t) &= x_4(t), \\
\dot{x_4}(t) &= -x_5(t)x_3(t) - ax_4(t) - g - u_1(t)x_1(t), \\
0 &= x_1^2(t) + x_3^2(t) - L^2, \\
x_1(t_0) = 0, \quad x_3(t_0) &= L, \quad x_2(t_0) = 0, \quad x_4(t_0) = 0,
\end{aligned}
$$

with state variables x_1,x_2,x_3,x_4,x_5 and input u_1. $c > 0$ and $d > 0$ are weighting parameters, $L > 0$ is the radius, g is the gravitational acceleration, $a > 0$ is a friction coefficient and $\alpha > 0$ control the head start of the target.

The authors of [39] found that existing direct collocation solvers, such as GPOPS-II [162], all failed to solve the problem directly in this formulation. For these solvers to yield a solution, problem reformulation and DAE index reduction procedures are necessary.

The same behaviour with the direct collocation method can be found with ICLOCS2, shown in Figure 7-5. However, the DAIR method is able to solve the problem directly without difficulties, by minimizing the integrated residuals of the DAE system, instead of forcing the residuals to be zero at collocation points.

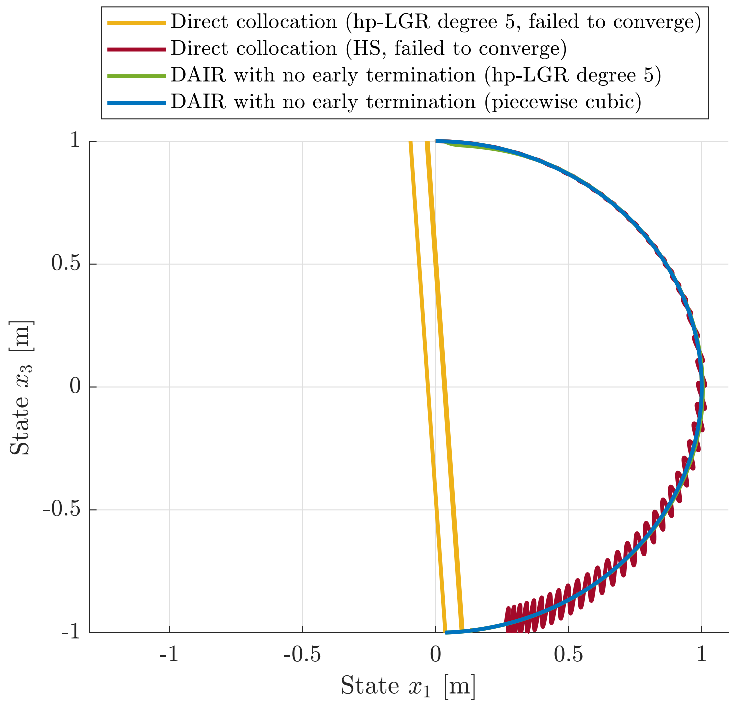


Figure 7-5: Comparison of the solutions for the high-index DAE system (with 50 major nodes)

7-3-3 Two-link robot arm

The two-link robot arm problem presented here was adapted from [132, Ex. 2, Sect. 12.4.2]. Consider a system consisting of two identical beams with the same property (mass: $m = 1\,\text{kg}$, length: $l = 1\,\text{m}$, and moment of inertia), connected at two actuated joints. The objective is to re-position a payload of mass $M = 1\,\text{kg}$ in minimum time, with the addition of a regularization term:

$$\min_{x,u,t_f} \quad t_f + 0.01 \int_0^{t_f} u_1(t)^2 + u_2(t)^2 \, dt.$$

The system dynamics can be expressed as

$$\begin{aligned}\dot{\omega}_\phi(t) =& \frac{1}{\frac{31}{36} + \frac{9}{4}\sin^2(\chi(t))}\Bigg(\sin(\chi(t))(\frac{9}{4}\cos(\chi(t))\omega_\phi^2(t)) \\ &+2\omega_\varrho^2(t) + \frac{4}{3}(u_1(t) - u_2(t)) - \frac{3}{2}\cos(\chi(t))u_2(t)\Bigg), \\ \dot{\omega}_\varrho(t) =& \frac{-1}{\frac{31}{36} + \frac{9}{4}\sin^2(\chi(t))}\Bigg(\sin(\chi(t))\frac{9}{4}\cos(\chi(t))\omega_\varrho^2(t)) \\ &+\frac{7}{2}\omega_\phi^2(t) - \frac{7}{3}u_2(t) + \frac{3}{2}\cos(\chi(t))(u_1(t) - u_2(t))\Bigg),\end{aligned}$$

$$\dot{\chi}(t) = \omega_\phi(t) - \omega_\varrho(t),$$
$$\dot{\phi}(t) = \omega_\phi(t).$$

The system has angular rates ω_ϕ, ω_ϱ, and angles ϕ, $\chi = \phi - \varrho$ as state variables, and nondimensionalized torque u_1 and u_2 as inputs. Furthermore, the variable simple bounds and boundary conditions are imposed in accordance to the reference, except that $\chi(t_f) = 0.5\,\text{rad}$, and $\phi(t_f) = 0.522\,\text{rad}$.

Figures 7-6 and 7-7 illustrate the solutions to the two-link robot arm problem problem generated with the two different solution representation methods. Presented alongside are the outcomes from the actual implementation of the resultant input trajectory on the same dynamic model, solved with a non-stiff variable-order ODE solver (MATLAB `ode113`) with a time step 100 times smaller than the discretization grid of the optimization problem. Observe that:

- Despite a very small tolerance (1×10^{-9}) and successful termination of the NLP solver, the collocation solution and interpolation of the solution exhibit large errors, leading to significant deviations to the state trajectories when the inputs are directly applied. I would like to emphasis this could be a special situation, as adding or removing one major point can actually yield better results; however, the problem is still ideal to demonstrate that for direct collocation, it is very difficult to guarantee accuracy without posterior error assessments and mesh design iterations. In contrast, only minor discrepancies can be observed for the solutions represented with IRM, on the same coarse grid with relatively low-order discretization.

- Although the constraints are implemented in the exact same way, the proposed method, to a greater extent, alleviates the issues of constraint violations inside the mesh intervals. This is because these constraint violations are often related to the large ODE defect errors in-between collocation points, which are directly dealt with by the residual minimization scheme.

7-3-4 Cart pole swing-up

Consider the cart pole swing-up problem from [114]. The problem requires movement of the cart to a specific location while making sure the pendulum attached to it achieves a vertically-up orientation at the terminal time. The problem has the position of the cart y_1 and the angle of the pendulum arm θ_1; these are state variables together with their time derivatives $\dot{y}_1$ and $\dot{\theta}_1$. The control input is $u \in [-20, 20]$ (force in Newtons). The following terminal conditions are imposed at $t_f = 2\,\text{s}$: $y_1(t_f) = 1\,\text{m}$, $\dot{y}_1(t_f) = 0\,\text{m/s}$, $\theta_1(t_f) = \pi\,\text{rad}$, and $\dot{\theta}_1(t_f) = 0\,\text{rad/s}$.

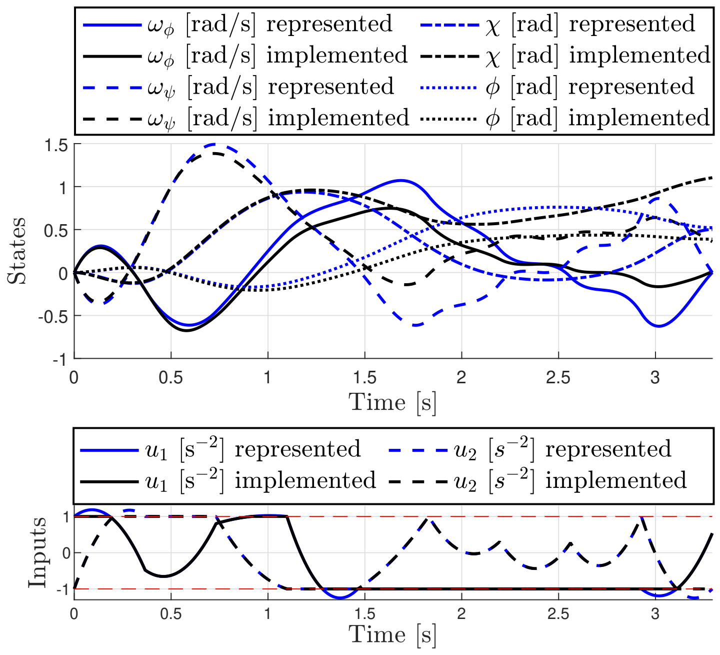


Figure 7-6: Solution to the two-link robot arm problem, direct collocation with Hermite-Simpson discretization, 10 major nodes. ©2020 IEEE

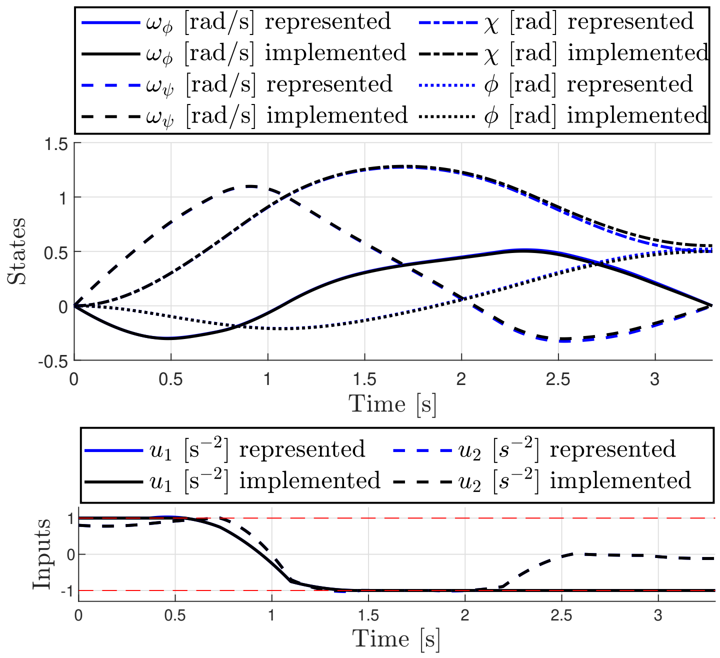


Figure 7-7: Solution to the two-link robot arm problem, solution representation with IRM, direct collocation with Hermite-Simpson discretization, 10 major nodes. ©2020 IEEE

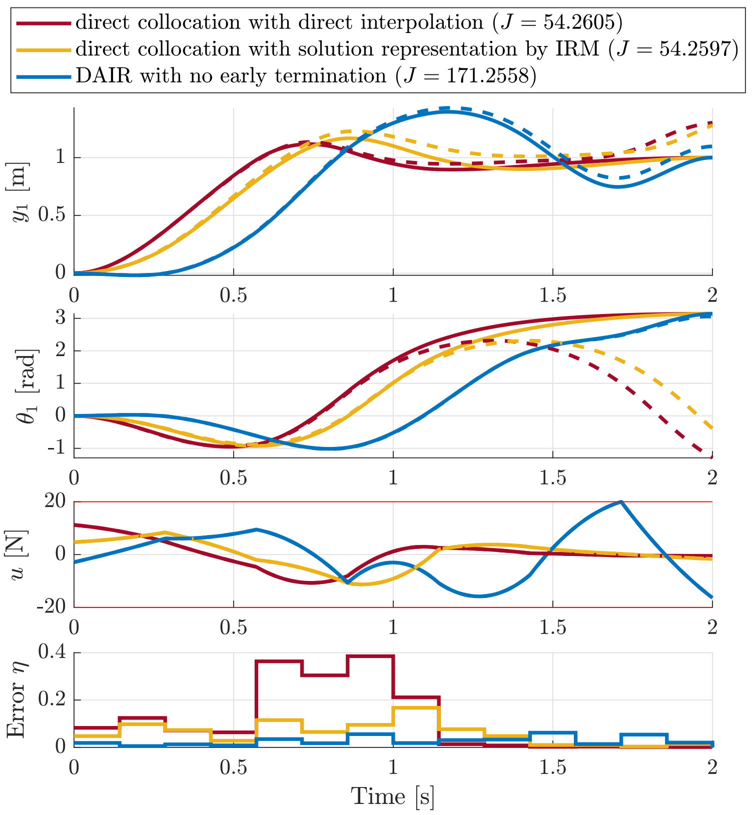


Figure 7-8: Solutions to the cart pole swing-up problem (HS/piecewise cubic parameterization with 7 mesh intervals, solid lines represent trajectories as solver output, dashed lines represent the resultant trajectory by implementing the input trajectory)

In the definition of $\mathcal{R}$, a weighting ω_ξ of 2 is selected for the dynamic equation corresponding to θ_1 while the weight for all other states remains at 1, to emphasise that the most important target for this problem is to have the pendulum up and vertical at the final time.

Figure 7-8 illustrates the comparison between various solutions solved using different methods, but on the same given and coarse discretization mesh. As seen from the figure, although the NLP problem transcribed via the direct collocation method successfully terminated with negligibly small tolerances, it becomes apparent that if the corresponding input trajectory is to be applied, the actual evolution of the system states will be very different than what was predicted by the DOP solution. Subsequently, at the final time, the state variables are at a distance far away from the terminal conditions. The last graph of Figure 7-8 illustrates the absolute local error η for each mesh interval. The discrepancies in the solution trajectories can be attributed to the large residual errors arising from trajectories between collocation points using direct interpolation.

Based on the direct collocation solution, the IRM solution representation method results

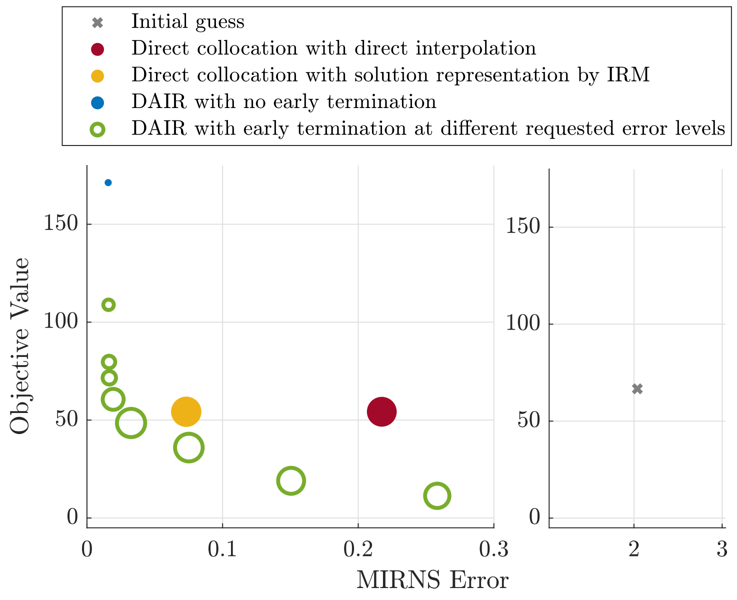


Figure 7-9: Trade-off between solution accuracy and optimality for the cart pole swing-up problem (HS/piecewise cubic parameterization with 7 mesh intervals, size of a circle is proportional to the constraint violation at $t = t_f$)

in improvements in solution accuracy (with regards to absolute local error and terminal condition violation) while maintaining the level of optimality. Nevertheless, the final position of the pendulum in both cases are at angles nowhere near the requirement to be up and vertical. In contrast, the situation can be significantly improved with the DAIR method without any early termination criteria, yielding a solution of very high accuracy considering the very coarse mesh employed. This, however, comes at a cost with a much higher objective value (indication of control effort), further emphasising the multi-objective nature of solving a dynamic optimization problem on a single discretization grid.

For further exploration, Figure 7-9 highlights the trade-off between solution accuracy and optimality. The figure shows a distinctive Pareto front formed by multiple DAIR solutions with different termination conditions depending on the requested error magnitudes. Direct collocation, on the other hand, is only capable of generating a single solution that is clearly dominated by DAIR solutions. This demonstrates the advantage of the DAIR scheme over direct collocation in terms of flexibility and Pareto optimality.

It is also important to note that due the system being open-loop unstable, a closed-loop implementation of the DOP solution will be necessary in practice. When implemented on-line as in NMPC, there will be another trade-off process between the DOP solution accuracy and closed-loop performance — the most accurate open-loop DOP solution may not always be preferred [118]. Therefore the flexibility of the DAIR scheme to reliably solve a DOP to

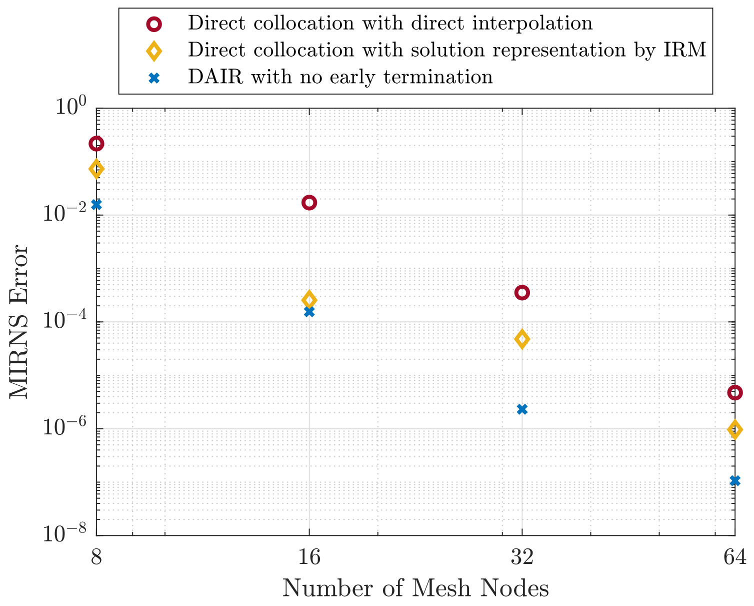


Figure 7-10: Reduction of MIRNS error as the number of mesh nodes increases (HS/piecewise cubic parameterization with equal-spaced intervals)

a specific accuracy level while ensuring Pareto optimality makes it a highly desirable method against other alternatives.

Although the MIRNS error is a good measure for solution accuracy, the MIRNS error has limitations due to the weighted norm computation, and because a practical metric for solution accuracy can be problem- and designer-dependent. For this example in particular, what really matters would be the differences between the target state values at terminal time and the ones achieved, especially concerning the up-vertical orientation of the pendulum. Figure 7-9 illustrates this aspect by making the sizes of the circles proportional to $\|y_1(t_f) - 1, 2(\theta_1(t_f) - \pi), \dot{y}_1(t_f), \dot{\theta}_1(t_f)\|_2$, a measure of terminal constraint violation. With this metric, the value corresponding to the smallest and largest circles shown in the figure is 0.46 and 11.05, respectively. By observing that the sizes of the circles are generally in correspondence with the values of the MIRNS error, it can be concluded that, for this example, the MIRNS error is a suitable metric both theoretically and practically.

Figure 7-10 illustrates the trends in the reduction of the MIRNS error as the mesh becomes denser, for some of the methods in the earlier comparisons. It can be seen that, although the gradient of the lines are similar (limited to the order of the discretization type), IRM-based approaches show a clear advantage in obtaining solutions with higher accuracy than direct collocation for the same discretization mesh.

7-4 Concluding remarks

When conventional direct transcription methods, such as direct collocation, are employed to solve nonlinear dynamic optimization problems, assurance in accuracy can only be made a posteriori through error analysis and mesh design iterations. When a given coarse mesh is used, the validity of the solution may become questionable with errors arising inside the intervals between collocation points, despite solving the nonlinear programming problem to negligibly small tolerances. Integrated residual minimization methods fundamentally address this challenge by minimizing the dynamic equation residual error integrated along the whole trajectory, with the added benefit of being capable of handling difficult problems, such as those with singular arcs and high-index DAEs.

Solving DOPs numerically is essentially a multi-objective optimization problem: for a given discretization mesh, one will inevitably face a trade-off between minimizing the objective (for optimality) and minimizing the discretization errors (for accuracy), forming a Pareto front. As demonstrated with the example problems, solutions from direct collocation with a given coarse mesh will be dominated by other solutions. In contrast, the DAIR scheme has been shown to be capable of directly obtaining a solution on the Pareto front based on the requested accuracy level.

Admittedly, the DAIR method, as well as other IRM-type direct transcription methods such as the PBF, are still in an early stage of development. Continued research will be required for them to reach the same level of maturity as direct collocation, to realise their full potentials.

Part III

Efficient and Accurate Solution of Large-scale and Challenging Problems

Chapter 8

Quick Generation of Nonlinear Programming Derivatives

The derivative-based NLP solvers require frequent computation of first and second-order derivative information. The most computationally demanding parts of derivative computation are for the constraint Jacobian and Lagrange Hessian, defined respectively as

$$\frac{\partial \mathcal{C}}{\partial \mathcal{Z}} = \begin{bmatrix} \frac{\partial \mathcal{C}_1}{\partial \mathcal{Z}_1} & \frac{\partial \mathcal{C}_1}{\partial \mathcal{Z}_2} & \cdots & \frac{\partial \mathcal{C}_1}{\partial \mathcal{Z}_{n_z}} \\ \frac{\partial \mathcal{C}_2}{\partial \mathcal{Z}_1} & \frac{\partial \mathcal{C}_2}{\partial \mathcal{Z}_2} & \cdots & \frac{\partial \mathcal{C}_2}{\partial \mathcal{Z}_{n_z}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{C}_{n_\mathcal{C}}}{\partial \mathcal{Z}_1} & \frac{\partial \mathcal{C}_{n_\mathcal{C}}}{\partial \mathcal{Z}_2} & \cdots & \frac{\partial \mathcal{C}_{n_\mathcal{C}}}{\partial \mathcal{Z}_{n_z}} \end{bmatrix},$$

$$\frac{\partial^2 \mathcal{L}}{\partial \mathcal{Z}^2} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}_1}{\partial \mathcal{Z}_1^2} & \frac{\partial^2 \mathcal{L}_1}{\partial \mathcal{Z}_1 \mathcal{Z}_2} & \cdots & \frac{\partial^2 \mathcal{L}_1}{\partial \mathcal{Z}_1 \mathcal{Z}_{n_z}} \\ \frac{\partial^2 \mathcal{L}_2}{\partial \mathcal{Z}_2 \mathcal{Z}_1} & \frac{\partial^2 \mathcal{L}_2}{\partial \mathcal{Z}_2^2} & \cdots & \frac{\partial^2 \mathcal{L}_2}{\partial \mathcal{Z}_2 \mathcal{Z}_{n_z}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{L}_{n_z}}{\partial \mathcal{Z}_{n_z} \mathcal{Z}_1} & \frac{\partial^2 \mathcal{L}_{n_z}}{\partial \mathcal{Z}_{n_z} \mathcal{Z}_2} & \cdots & \frac{\partial^2 \mathcal{L}_{n_z}}{\partial \mathcal{Z}_{n_z}^2} \end{bmatrix},$$

with $\mathcal{C}$ containing all $n_\mathcal{C} := N \times (n_g + n_c)$ discretized constraint equations, and $\mathcal{L} = J + \Omega^\top \mathcal{C}$ the Lagrangian of the NLP. $\Omega \in \mathbb{R}^{n_\mathcal{C}}$ are the Lagrange multipliers corresponding to the discretized constraints. $\mathcal{Z}_: \in \mathbb{R}^{n_z}$ is the decision variable vector (i.e. $\mathcal{Z}$ assembled in a vector) with $\mathcal{Z}_i$ the i^{th} element in $\mathcal{Z}_:$.

Fast and accurate computation of the derivative information is a crucial aspect of solving the problem efficiently. Even for linear quadratic regulator control problems, it was realized early on [210] that different grouping of variables will lead to a different structures in the resulting matrices. The same concept holds when solving NLPs: exploiting the sparsity patterns of these matrices can lead to a substantial reduction in computational complexity and storage usage. This applies both to obtaining the derivative information and for solving the NLP. In fact, sparse linear algebra has become one of the most important aspects of numerical methods for dynamic optimization, allowing efficient solution of large-scale practical problems.

In this chapter, the general sparsity patterns for various direct transcription methods will first be presented in Section 8-1. This will include the sparsity patterns for the newly proposed IRM-type methods and the sparsity patterns arising from an alternative ordering of variables for LGR collocation. Next in Section 8-2, an overview of different methods for obtaining derivative information is provided, focusing on the differences in sparse finite difference computations between direct collocation methods and IRM-type transcription methods.

8-1 Sparsity pattern for the NLP

8-1-1 Conventional h method

Literature on conventional h methods (e.g. [17]) often have the NLP decision variables arranged in an *order by stage*, or *order by time* format, i.e. having a decision vector in the form of

$$\mathcal{Z}_{:} = \begin{bmatrix} t_0 & t_f & p_1 & \dots & p_{n_p} & \chi_1^{(1)} & \upsilon_1^{(1)} & \dots & \chi_{N^{(1)}}^{(1)} & \upsilon_{N^{(1)}}^{(1)} & \dots & \chi_{N^{(K)}}^{(K)} & \upsilon_{N^{(K)}}^{(K)} \end{bmatrix}^\top .$$

The resulting sparse constraint Jacobian and Lagrange Hessian with direct collocation transcription have the structure shown in Figure 8-1, largely characterized by block structures along the diagonal, with each block representing a single mesh interval.

For (most) h methods the blocks shown in Figure 8-1 are also sparse which further increases the level of sparsity for the matrix systems. Decoupling variables may be introduced for the constraint Jacobian to transform it into a banded diagonal structure, at the cost of increasing the dimension of the NLP. This particular double-bordered band diagonal Hessian structure can be reformatted into a bordered block-diagonal matrix, also known as a block "arrowhead" matrix [150]. These special matrix system structures can be exploited by efficient linear algebra solvers, to ensure that the computational complexity for solving the system of equations scales linearly with the number of stages.

In IRM-type transcriptions, the constraint Jacobian will generally be of smaller dimensions. Depending on the respective IRM schemes, the original block structures of size $(n+n_g)N \times n_z$ corresponding the dynamic constraints, will either vanish or be replaced by residual constraints. These residual constraints have a densely populated structure of only $(n+n_g) \times n_z$

as shown in Figure 8-1b. The corresponding Lagrange Hessian structure, shown in Figure 8-2, is still valid with IRM methods; however, each block will likely to become denser.

Improving the solution accuracy for h-type discretizations through mesh refinement results in an increase in the number of discretization nodes. Therefore it will not lead to an increase in the sizes of these diagonal blocks. Instead, the number of blocks will increase. As a result, the dimension of the matrices, as well as the sparsity level, will grow quickly as the mesh becomes increasingly dense.

8-1-2 Psuedo spectral p/hp method

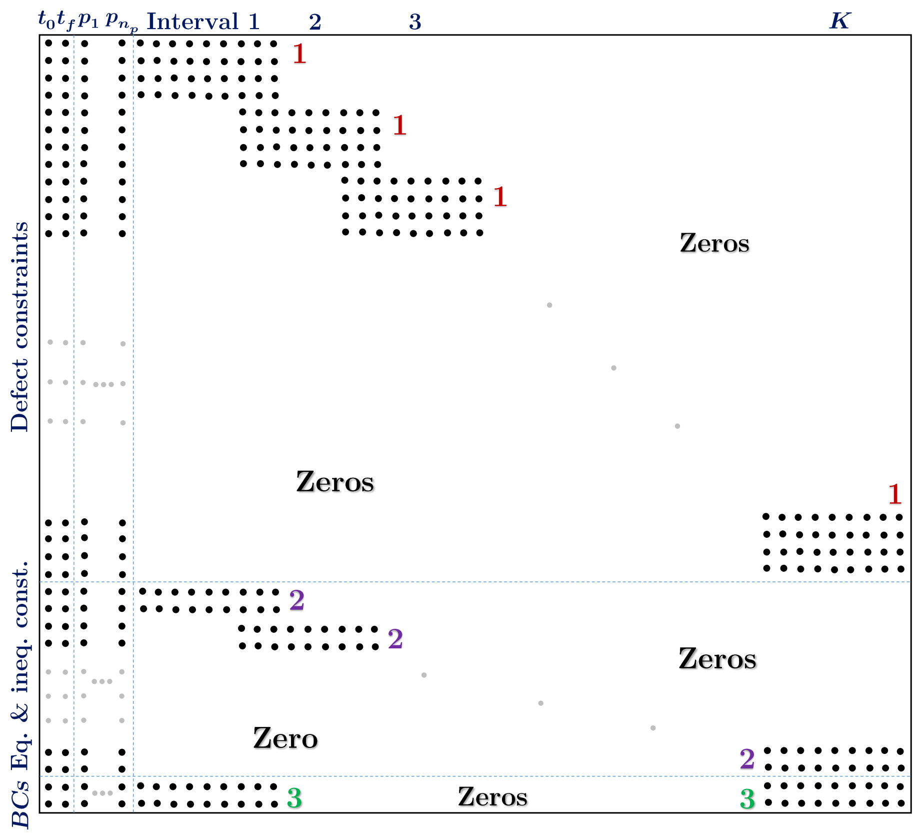
Psuedo spectral p or hp collocation does not typically use the above *order by stage* variable ordering. Rather, an *order by variables* rule is used whereby the collocation points of each state and each input variable are grouped together in an NLP decision vector. For LGR collocation with continuity enforced implicitly, the decision variables are formed as

$$\mathcal{Z}_{:} = \begin{bmatrix}\chi_1^{(1)} & \dots & \chi_{N^{(1)}}^{(1)} & \dots & \chi_{N^{(K)}+1}^{(K)} & \upsilon_1^{(1)} & \dots & \upsilon_{N^{(K)}}^{(K)} & p_1 & \dots & p_{n_p} & t_0 & \dots & t_f\end{bmatrix}^\top .$$

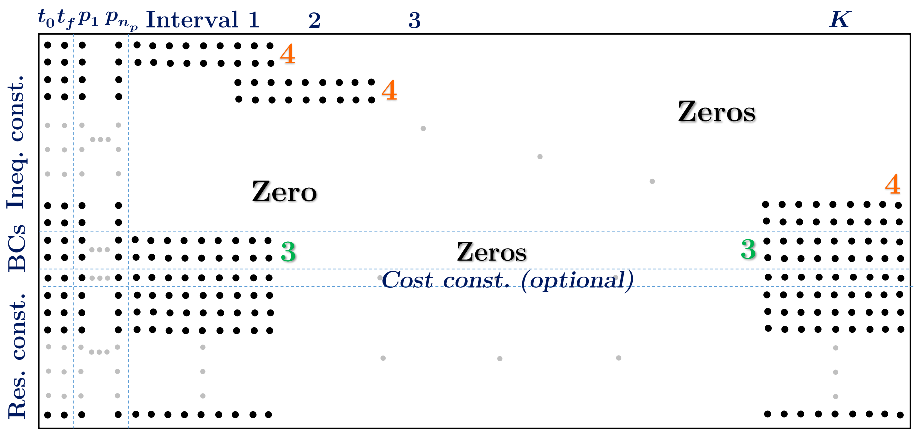
In this way, the differentiation matrix can be formulated and implemented in a natural way. The state variables at the end-point $\chi_{N^{(K)}+1}^{(K)}$ are also included as the decision variables, despite not being a collocation point for LGR collocation. Additional time variables are also added if the dynamically refined hp-adaptive mesh (see Section 12-2-3) is employed.

The differentiation matrix resulting from Radau collocation is of size $N \times (N+1)$, and is fully populated when global p methods are used (i.e. a single polynomial segment). For hp methods, multiple segments are introduced, and the structure of the differentiation matrix is as illustrated in Figure 8-3a. The domain is divided into K mesh intervals, each shown in a different colour. Each interval is represented as a block of size $N^{(k)} \times (N^{(k)}+1)$ on the diagonal of the differentiation matrix, where $N^{(k)}$ is determined by the polynomial degree of k^{th} segment. The overlapping of the last columns of the preceding blocks and the first columns of the succeeding blocks ensures the implicit satisfaction continuity conditions. The structure of the constraint Jacobian is now composed of multiple smaller (diagonal, fully populated or empty) blocks as seen in Figure 8-3b. A similar structure is observed for the Lagrange Hessian matrix, shown in Figure 8-5a.

It is notable that these structures are significantly different from that of the *order by stage* arrangement, which can be exploited by modern linear algebra solvers. Therefore, an alternative ordering of variables by time for the hp method can be made possible. The Jacobian structure now consists of K blocks on the diagonal with an offset, with the sparsity pattern shown in Figure 8-4b. The size of each diagonal block (Figure 8-4a) is determined by the number of state (n) and control (m) variables, number of path constraints (n_c) and the or-



(a) Jacobian with direct collocation



(b) Jacobian with direct IRM

Figure 8-1: Sparsity pattern of the problem when using h methods (Jacobian)

(**Block Type 1**: from ODE dynamics, not fully populated; **Block Type 2**: from equality (DAE) and inequality path constraints, not fully populated; **Block Type 3**: from boundary conditions, either zeros or endpoints); **Block Type 4**: from inequality path constraints, not fully populated;)

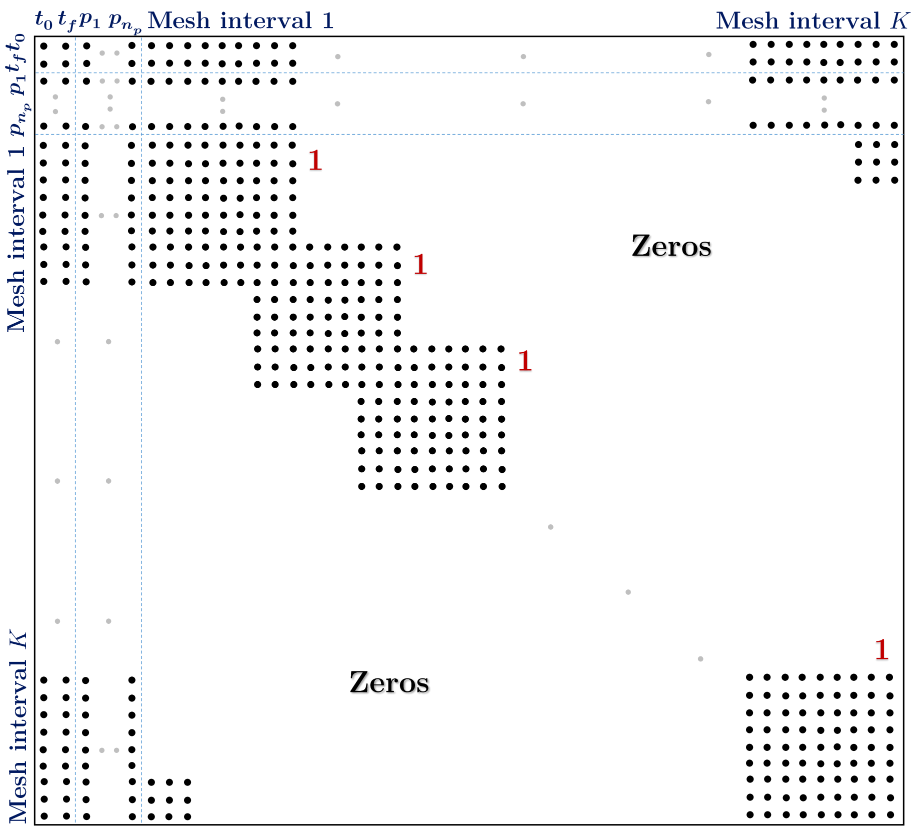


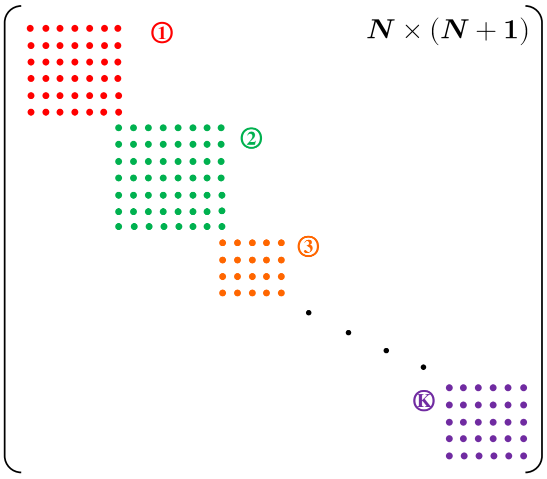
Figure 8-2: Sparsity pattern of the problem when using h methods (Hessian)

Block Type 1: with size $2(n+m) \times 2(n+m)$ for Euler and trapezoidal and $3(n+m) \times 3(n+m)$ for Hermite-Simpson discretization, not fully populated)

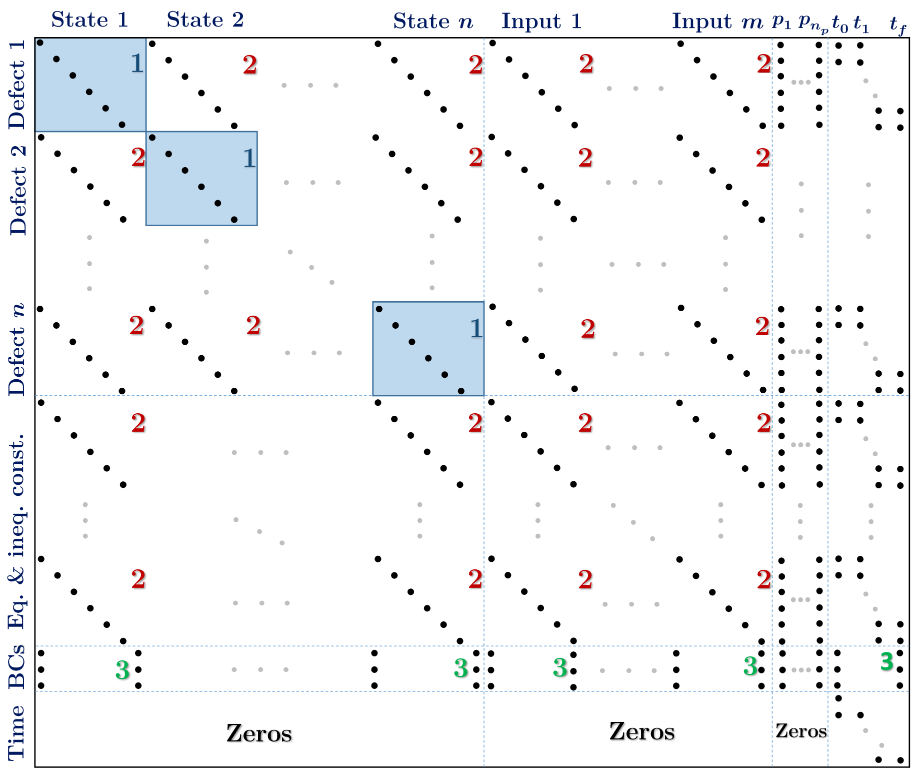
der of the polynomial (through $N^{(k)}$). The Hessian mainly consists of N sub-blocks on the diagonal, each with size $(m+n) \times (m+n)$. Time variables are populated at the right and bottom of the matrix, leading to a distinctive "arrowhead" structure.

For IRM-type transcriptions with *order by variables*, the structure of the constraint Jacobian for the *hp* method undergoes a similar transformation as for the *h* method with a significant reduction in the number of rows, as shown in Figure 8-6a. The Lagrange Hessian illustrated in Figure 8-6b, generally retains the overall structure of that for direct collocation, despite denser populated sparsity patterns inside each elementary block.

By determining and providing the above sparsity patterns, sparsity exploiting NLP solvers can be employed with linear algebra packages (e.g. IPOPT with MA57 solver) to exploit the structure of the Jacobian and Hessian. The strategy of the linear algebra solver may be different depending on whether the *order by stage* or *order by variable* formulation is used.



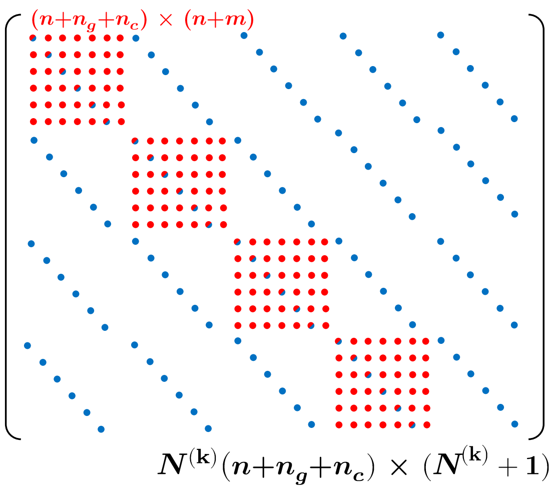
(a) Radau differentiation matrix



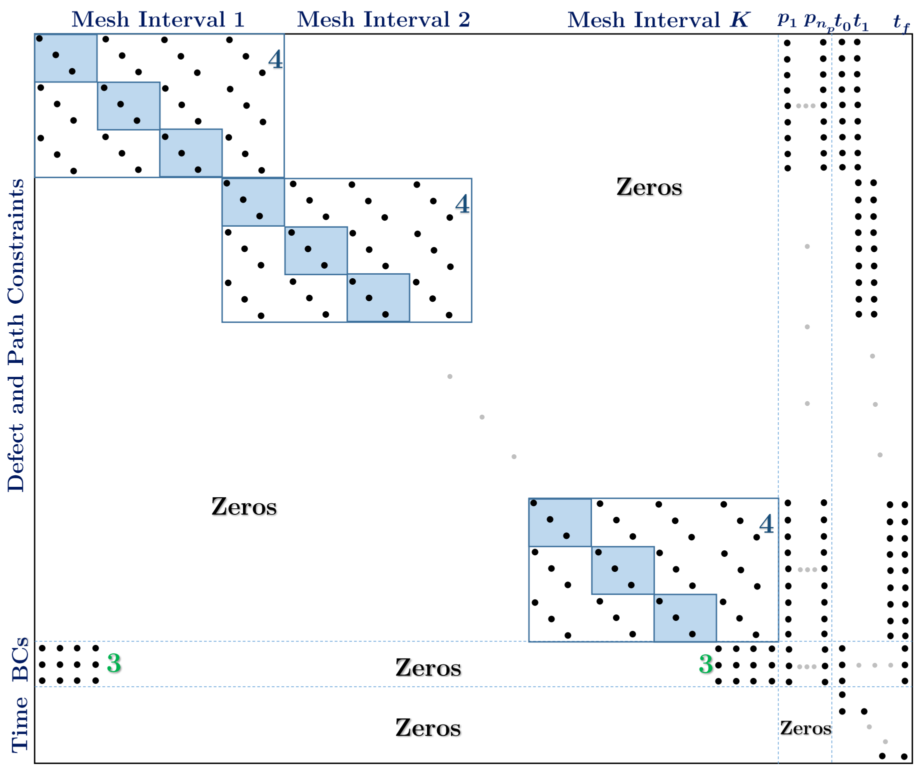
(b) Jacobian structure

Figure 8-3: Sparsity pattern for the constraint Jacobian using LGR collocation when grouped by variables

(**Block Type 1**: Radau differentiation matrix plus either zeros or diagonal; **Block Type 2**: from dynamic equations and path constraints, either zeros or diagonal; **Block Type 3**: from boundary conditions, either zeros or endpoints)


$(n+n_g+n_c) \times (n+m)$
$N^{(\mathbf{k})}(n+n_g+n_c) \times (N^{(\mathbf{k})}+1)$

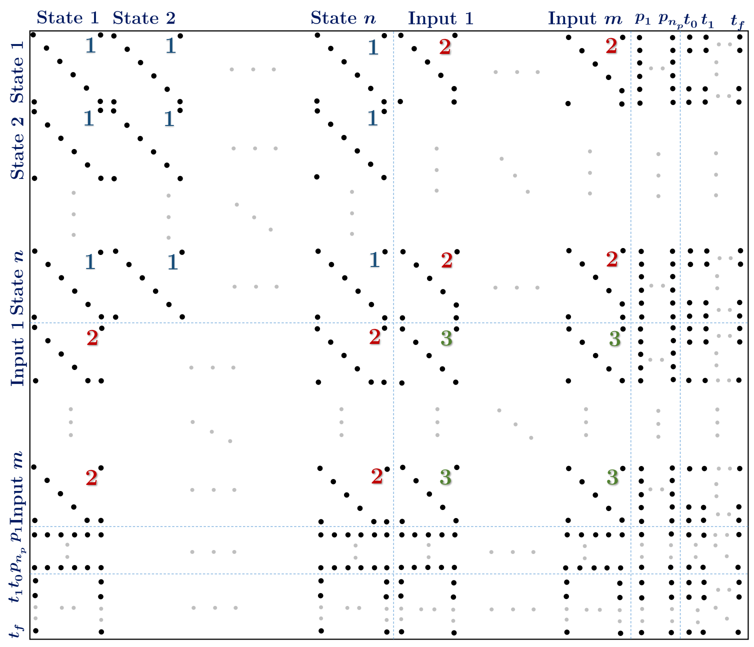
(a) Block element for LGR collocation when grouped by time



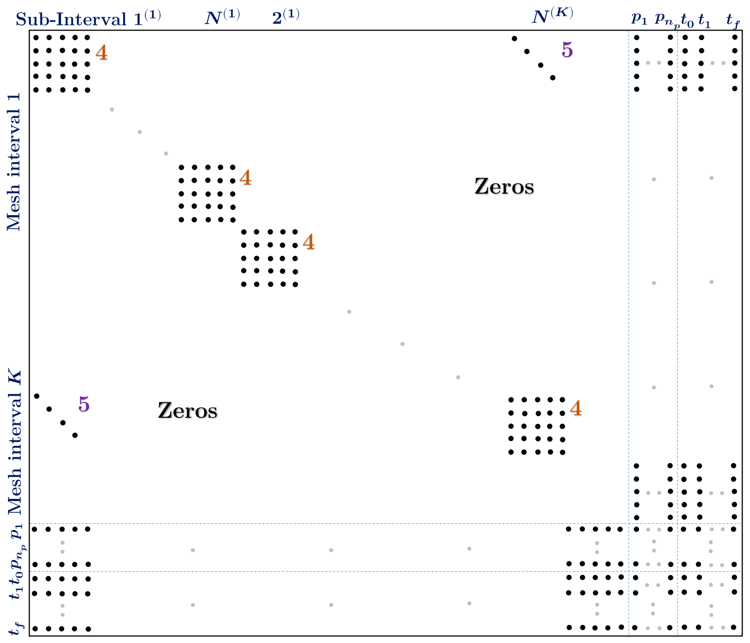
(b) Jacobian structure

Figure 8-4: Sparsity pattern for the constraint Jacobian using LGR collocation when grouped by time

(**Block Type 3**: from boundary conditions, either zeros or endpoints; **Block Type 4**: block structure as shown in figure 8-4a; blue points: entries from radau differentiation matrix; red points: entries from the dynamics and path constraints)



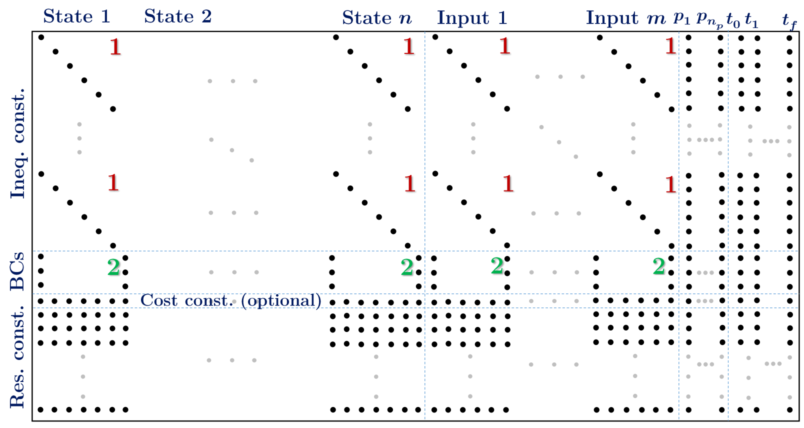
(a) Hessian structure when grouped by variables



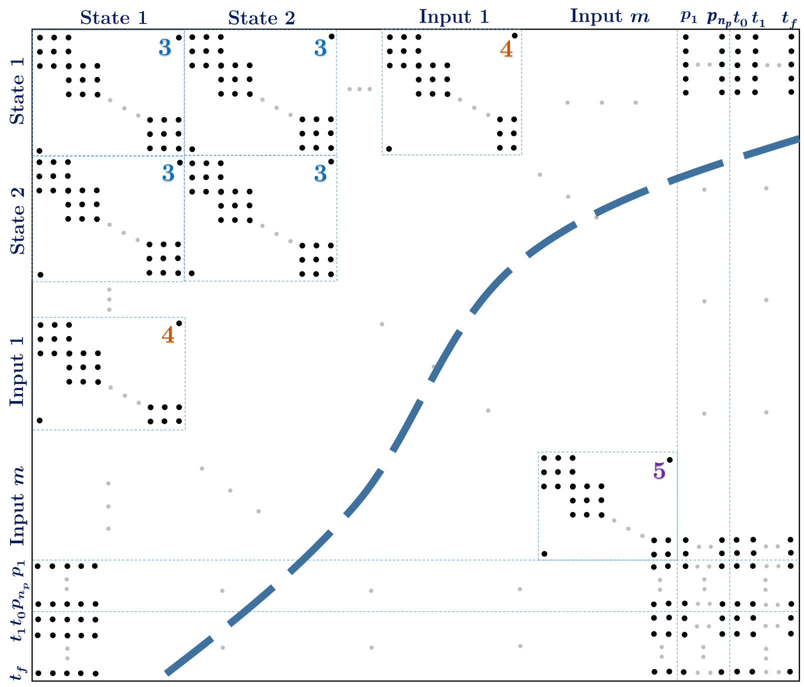
(b) Hessian structure when grouped by time

Figure 8-5: Sparsity pattern for the Lagrangian Hessian using LGR collocation

(**Block Type 1**: either zeros, or diagonal block of size $(N+1)\times(N+1)$ with corners; **Block Type 2**: either zeros, or block of $N\times(N+1)$ or $(N+1)\times N$ with corners and a diagonal of $N\times N$; **Block Type 3**: either zeros, or diagonal block of size $N\times N$ with corners; **Block Type 4**: block of size $(n+m)\times(n+m)$; **Block Type 5**: diagonal block of size $n\times n$)



(a) Jacobian structure



(b) Hessian structure

Figure 8-6: Sparsity pattern of the problem when using IRM-type methods with LGR and order by variables

(**Block Type 1**: either zeros or diagonal; **Block Type 2**: either zeros or endpoints; **Block Type 3**: of size $(N+1)\times(N+1)$ with corners, with K overlapping blocks on the diagonal, each with size $(N^{(k)}+1)\times(N^{(k)}+1)$; **Block Type 4**: of size $(N+1)\times N$ or $N\times(N+1)$ with corners, with K overlapping blocks on the diagonal, each with size $(N^{(k)}+1)\times(N^{(k)}+1)$ except the last one with $N^{(k)}\times(N^{(k)}+1)$ or $(N^{(k)}+1)\times N^{(k)}$; **Block Type 5**: of size $N\times N$ with corners, with K overlapping blocks on the diagonal, each with size $(N^{(k)}+1)\times(N^{(k)}+1)$ except the last one with $N^{(k)}\times N^{(k)}$)

8-2 Computation of NLP derivatives

The computation time and accuracy associated with the evaluation of derivative information can have a major impact on the computational efficiency of the NLP solver. Inaccurate derivative data often leads to slower convergence (or even non-convergence) of the NLP solver with higher iteration counts. On the other hand, accurate evaluation of derivative information would require more computational resources and time, making each NLP iteration more expensive. Obtaining fast and accurate solutions to the NLPs hinges on the right balance between the two criteria. A number of possible options are discussed in this section; however, the most appropriate choice is often problem dependent.

8-2-1 Supplied analytical expressions

For problems of medium to low complexity where closed-form analytical expressions of function derivatives are obtainable, directly supplying the mathematical formula can often result in fast computation and high accuracy. To avoid derivation by hand (expensive and error prone), use of symbolic differentiation toolboxes (e.g. MATLAB or Maple) is recommended.

8-2-2 Sparse finite differences

For many practical problems, the use of data-driven modelling and/or look-up table interpolation for certain parameters can lead to challenges in obtaining the analytical derivative information. A universally valid way to calculate the derivative information is to apply finite difference approximations, similar to the form

$$\left.\frac{\partial h}{\partial \mathcal{Z}_i}\right|_{\mathcal{Z}_i} = \frac{h(\mathcal{Z}_i + d) - h(\mathcal{Z}_i)}{d} + O(d), \tag{8-1a}$$

$$\left.\frac{\partial^2 h}{\partial \mathcal{Z}_i^2}\right|_{\mathcal{Z}_i} = \frac{h(\mathcal{Z}_i + d) - 2h(\mathcal{Z}_i) + h(\mathcal{Z}_i - d)}{d^2} + O(d^2), \tag{8-1b}$$

where d is a small perturbation. The computation is easy and relatively fast, but the level of accuracy depends on the choice of perturbation parameters. The approximation of the derivative information will have large round-off errors due to machine precision limits if d is too small. Conversely, large d can lead to higher truncation or linearization errors.

Computing the finite differences with (8-1) requires perturbations of all decision variables. However, if one individually perturbed each element of $\mathcal{Z}_:$, the process would be lengthy and expensive. Fortunately, using the sparsity patterns identified earlier can ensure that computations are only conducted for entries that are known to have non-zero values. Also, in many cases, multiple perturbations can be made simultaneously, without affecting the capability to independently identify the derivative information.

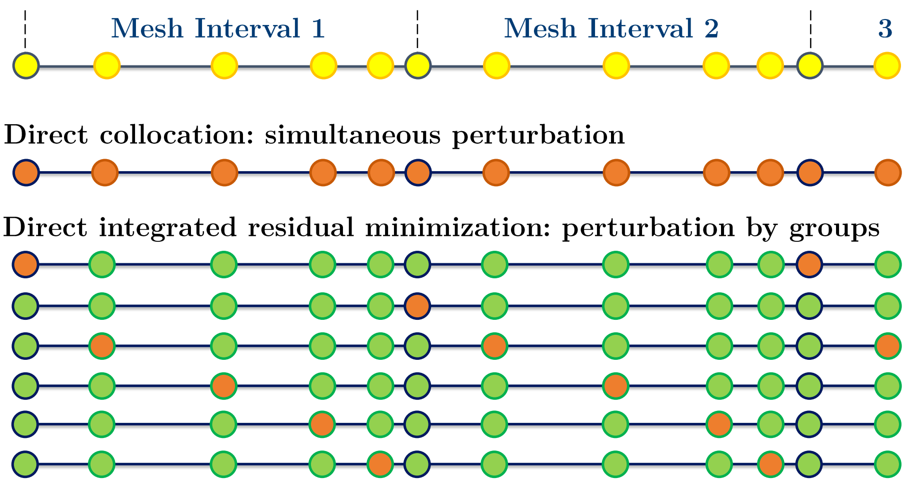

Mesh Interval 1
Mesh Interval 2
3
Direct collocation: simultaneous perturbation
Direct integrated residual minimization: perturbation by groups

Figure 8-7: Comparison of numerical finite difference computations in terms of perturbations. Orange grid points represent the position of the perturbed variables at each perturbation instance.

Direct collocation methods benefit from these simultaneous perturbations. Thanks to the isolation property, a change in the discretized state or input variable at any grid point will only affect the discretized Lagrange cost and constraint equation residuals corresponding to that particular grid point. Hence, for each state and input variable, finite difference computations can be carried out simultaneously for the full discretization grid and assembled accordingly based on the sparsity pattern of the derivative information. This approach makes the computation of sparse finite differences very efficient for direct collocation methods.

The case is different when IRM-type transcription schemes are employed. Any changes in a parameterized state or input variable at a grid point can now lead to changes in the discretized Lagrange cost and constraint equation residuals at all grid points of that mesh interval. As a result, decision variables correspond to different grid points have to be categorized into multiple groups (as demonstrated in Figure 8-7), and the computation of numerical finite differences must be carried out for each group separately. For a grid of K intervals (i.e. $K+1$ mesh nodes) and $N^{(k)}$ grid points for each interval $k \in \mathbb{I}_K$, the number of groups necessary would be $\max_{i=1}^{K} N^{(k)} + 1$. Consequently, when using sparse finite differences, computational overhead associated with IRM-type of methods can be significantly higher than that of direct collocation. Thus other options such as algorithmic differentiation would be preferred for the supply of derivative information.

8-2-3 Algorithmic differentiation

In algorithmic differentiation, the complex nonlinear equations are decomposed into several elementary operations. The derivative information of the nonlinear function can thus be obtained by differentiating each elementary operation separately, and subsequently reconstructed using the chain rule. For example, the DOP toolbox ICLOCS2 supports the use of the forward mode algorithmic differentiation toolbox Adigator [209] which is marginally more computationally expensive than finite differences, but accurate up to machine precision.

8-2-4 Fast Hessian approximations

Most NLP solvers support some form of fast Hessian update method, normally under the umbrella of Broyden-Fletcher-Goldfarb-Shanno (BFGS) or limited memory BFGS (LM-BFGS or L-BFGS) algorithms (see [38]). This fast and inaccurate approximation of the Hessian may significantly reduce the NLP computation time per iteration but can result in a larger iteration count. In practice, BFGS is found to be more efficient for regulation (stabilization/set-point tracking) type problems with short-horizon. Having accurate Hessian information can be much more beneficial for long-horizon minimum-time and minimum-energy type of problems.

Chapter 9

Efficient Implementation of Rate Constraints[†]

For many engineering problems, constraints may need to be imposed on the rate of changes for the state and input variables, to account for physical actuation limitations (e.g. the maximum rotation rate of flight control surfaces on aircraft) or to fulfill certain ride comfort requirements (e.g. the maximum longitudinal and lateral accelerations experienced by passengers).

In dynamic optimization, the underlying problem can often be formulated and implemented in a number of different ways. Under a linear framework, many implementations are computationally comparable, thus straightforward approaches are often used. For example, rate constraints on input variables are generally implemented through additional dynamic equations [135, 207], and rate constraints on state variables are commonly addressed with additional path constraints [49]. However, under a nonlinear framework, this way of implementing input rate constraints are known to result in numerical difficulties and introducing fluctuations and ringing phenomena in the solution due to singular control [17]. To improve the solution quality, additional regularization terms may be added to the DOP formulation [31]; however, this practice often leads to computational challenges by needing to solve the problem repetitively with appropriate weightings.

This chapter proposes a general approach to directly implement rate constraints on the discretization mesh. First, different approaches for implementing rate constraints are introduced and analysed in Section 9-1. This is followed by two classical examples of different complexity

[†] Most of the material presented in this chapter has been published in the following work:
Y. Nie, E.C. Kerrigan, Efficient implementation of rate constraints for nonlinear optimal control. *IEEE Transactions on Automatic Control*. 2020. In press. ©2020 IEEE

in Section 9-2, where the pros and cons of each implementation are demonstrated. Unlike conventional approaches that may lead to singular control arcs, the proposed method will not introduce singular arcs to the problem, resulting in solutions of higher accuracy. Moreover, significant computational speedups can be achieved by exploiting the properties of the resulting linear constraint equations.

9-1 Implementations of rate constraints

In many problems, constraints of the form

$$\dot{u}_L \leq \frac{du}{dt}(t) \leq \dot{u}_U, \ \forall t \in [t_0, t_f] \text{ a.e.}$$
$$\dot{x}_L \leq \frac{dx}{dt}(t) \leq \dot{x}_U, \ \forall t \in [t_0, t_f] \text{ a.e.}$$

may need to be implemented to restrict the rate of change for the state and/or input variables. Subscript $_L$ represent lower bounds and $_U$ represent the upper bounds.

9-1-1 Conventional implementation

For input variables, a common approach is to introduce u as an additional state variable, and v as the new input with a simple bound through the dynamic equation

$$\dot{u}(t) = v(t) \text{ with } \dot{u}_L \leq v(t) \leq \dot{u}_U, \ \forall t \in [t_0, t_f] \text{ a.e.} \tag{9-1}$$

For rate constraints on the state variable x, additional path constraints are needed:

$$\dot{x}_L \leq f(x(t), u(t), t, p) \leq \ \dot{x}_U, \ \forall t \in [t_0, t_f] \text{ a.e.} \tag{9-2}$$

For simplicity, (9-1) is named the *add-state* implementation, and (9-2) is named the *add-path constraint* implementation.

Unfortunately these conventional implementations exhibit many shortcomings. These are mainly:

1. The number of state variables and constraint equations are increased, resulting in a larger DDOP. In addition, the index of the DAE of the transcribed problem may also increase, leading to a problem that is often more difficult to solve numerically.
2. When (9-1) is used, singular arcs may occur and affect the solution quality. This can occur if the original control input u appears nonlinearly in the Lagrange cost or other dynamic equations, whereas the new control v appears linearly instead.

In fact, the two items are connected and demonstrated in further details in Chapter 11.

For ease of demonstration, consider the following DOP, which is simplified but still sufficiently general, with $x_1 \in \mathbb{R}$ the state variable, and $u_1 \in \mathbb{R}$ the control input. As per the conventional approach, the rate constraint on the original control input is implemented with a new control input $v_1 \in \mathbb{R}$:

$$\min_{x_1,u,v} \int_0^{t_f} g_1(x_1(t)) + g_2(x_1(t), u_1(t))dt \tag{9-3a}$$

subject to

$$\dot{x_1}(t) = g_3(x_1(t), u_1(t)),\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{9-3b}$$

$$\dot{u}_1(t) = v_1(t),\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{9-3c}$$

$$\dot{u}_{1_L} \leq v_1(t) \leq \dot{u}_{1_U},\ \forall t \in [t_0, t_f] \text{ a.e.} \tag{9-3d}$$

Here the same hypotheses as in [202] is followed: g_1, g_2 and g_3 are continuous, continuously differentiable for all $u_1 \in \mathcal{U}$, and Lipschitz in x_1. Also, the admissible control set $\mathcal{U}$ is assumed to be a bounded set in some Euclidean space.

Proposition 3. *If the DOP* (9-3) *has a linear objective and dynamics with respect to the original control input u_1, i.e. if g_2 and g_3 are both functions that only have strictly linear input (i.e. in the form of $g_\iota(x_1(t), u_1(t)) = \tilde{g}_\iota(x_1(t)) + c_{st}u_1(t)$, with c_{st} a constant), the resulting optimal control v_1^* will not contain a singular arc. However, if u_1 appears nonlinearly in the objective and/or dynamics, i.e. if g_2 and/or g_3 are arbitrary nonlinear functions, there exists problems where singular arcs will occur for some intervals of the solution.*

Proof. First, the Hamiltonian of the system can be formulated, with $\lambda(t)$ the costate of the dynamics

$$\begin{aligned} &\mathcal{H}(x_1(t), u_1(t), \lambda_{x_1}(t), \lambda_{u_1}(t), v_1(t)) := g_1(x_1(t)) \\ &\quad + g_2(x_1(t), u_1(t)) + \lambda_{x_1}(t)g_3(x_1(t), u_1(t)) + \lambda_{u_1}(t)v_1(t). \end{aligned} \tag{9-4}$$

From Pontryagin's minimum principle, if the state and costate are optimal, the optimal control v_1^* minimizes the Hamiltonian, i.e.

$$v_1^*(t) \in \arg\min_v \mathcal{H}(x_1^*(t), u_1^*(t), \lambda_{x_1}^*(t), \lambda_{u_1}^*(t), v). \tag{9-5}$$

Substituting the Hamiltonian (9-4) in (9-5) yields the optimal control

$$v_1^*(t) = \begin{cases} \dot{u}_{1_U} & \text{if } \lambda_{u_1}^*(t) < 0 \\ ? & \text{if } \lambda_{u_1}^*(t) = 0 \\ \dot{u}_{1_L} & \text{if } \lambda_{u_1}^*(t) > 0 \end{cases} \tag{9-6}$$

We first show that this implementation yields a singular arc free solutions with linear DOPs. From the first order necessary conditions for optimality we have $\dot{\lambda}_{u_1}(t) = -\frac{\partial \mathcal{H}}{\partial u_1}$. Thus, if both g_2 and g_3 only contain input terms in the form of $c_{st}u_1(t)$, $\dot{\lambda}_{u_1}(t)$ will then be constant and $\lambda_{u_1}(t)$ will be a linear straight line. Relating this to (9-6) we can see that the optimal control will exhibit bang-bang behaviour with at most one switch depending on the crossing of $\lambda_{u_1}(t)$ with the x axis. Therefore, the solution is free of singular arcs if the objective and dynamics with respect to the original control input u_1 are all linear. This is the reason why in linear optimal control problems, the conventional implementation can be used without issues.

To show that this implementation will suffer from singular arc problems when nonlinear DOPs are considered, we assume that g_2 and/or g_3 are now arbitrary nonlinear functions. Thus, $\lambda_{u_1}(t)$ can be a function of any shape and the optimal control will not be uniquely defined on intervals where $\lambda_{u_1}(t) = 0$, a.k.a. the singular arc. The problem in Section 9-2-1 is an example where such an issue arises. $\square$

For a direct collocation method to yield the correct solution for singular control problems, one might have to use a multi-phase formulation and additionally impose the singular arc condition specifically on the phases with singular control. For example, if one takes the same approach (as in the proof) for the example problem in Section 9-2-1, the condition for singular control to occur is when $\lambda_{x_2} = 0$. Repetitively taking time derivatives of this equation would yield the singular arc condition $u(t) = x_1(t)$. This approach requires analytical derivations and would become increasingly challenging for complex real-world problems.

An ad-hoc method sometimes used in practice for dealing with the singular arc is to augment the original objective with an additional regularization term (e.g. in [31]), often in the form of $\rho||v||_1$ or $\rho||v||_2^2$. With relatively large values of the penalty weight ρ, the fluctuations on the singular arc can be suppressed, but at the cost of obtaining sub-optimal trajectories. To get closer to the optimal from this point, the problem may need to be repetitively solved with the penalty weight gradually reduced.

9-1-2 On-mesh implementation

To mitigate the above-mentioned shortcomings, a method is proposed to directly impose algebraic rate constraints for input variables on the discretization grid. Based on previous

Table 9-1: Numerical differentiation schemes

Method	**No. of Data Points**	**Numerical Differentiation**
Trapezoidal (h)	2 (equal spaced)	$\varkappa'_{i,k} = \frac{\varkappa_{i+1,k}-\varkappa_{i,k}}{\Delta t^{(k)}}$
Hermite Simpson (h)	3 (equal spaced)	$\varkappa'_{i,k} = \frac{-3\varkappa_{i,k}+4\varkappa_{i+1,k}-\varkappa_{i+2,k}}{\Delta t^{(k)}}$ $\varkappa'_{i+1,k} = \frac{\varkappa_{i+2,k}-\varkappa_{i,k}}{\Delta t^{(k)}}$ $\varkappa'_{i+2,k} = \frac{\varkappa_{i,k}-4\varkappa_{i+1,k}+3\varkappa_{i+2,k}}{\Delta t^{(k)}}$
LGR (p/hp)	$N^{(k)}+1$ (LGR bases)	$\varkappa'_{1:N^{(k)}+1,k} = \frac{2}{\Delta t^{(k)}}\mathcal{A}^{(k)}_{LGR}\varkappa_{1:N^{(k)}+1,k}$

work [17], this on-mesh approach is generalized for all discretization methods (h, p and hp type), as well as for state variables.

Since the treatment for state variables x and input variables u are similar, for simplicity the variable x will be used to represent the variable on which the rate constraints are imposed. If $\varkappa_i$ represents the discretized version of x at time instance i, then the numerical differentiation of x at that grid point ($\varkappa'_{i,k}$) can be calculated using $N^{(k)}$-point finite difference approximations, with $N^{(k)}$ the number of data points in the interval k. See Table 9-1 for the formulations of some of the most commonly used discretization methods, with $\mathcal{A}^{(k)}_{LGR}$ the LGR differentiation matrix. Details on the determination of the numerical differentiation equations are available in [69].

Note that for p/hp methods, the numerical differentiation for all grid points on the polynomial are obtained altogether. It is also worth mentioning that if LGR collocation is used, the end-point value for the control ($\upsilon^{(K)}_{N+1}$) may need to be approximated.

It is then straightforward to implement the rate constraints as linear constraints

$$\dot{\mathrm{x}}_L - \varkappa'_{i,k} \leq 0 \tag{9-7a}$$

$$\varkappa'_{i,k} - \dot{\mathrm{x}}_U \leq 0 \tag{9-7b}$$

for all possible values of i. This approach will be referred to as the *on-mesh* implementation.

The on-mesh implementation of rate constraints has several benefits in comparison to the conventional add-state and add-path constraint approaches. Firstly, the solution quality in terms of singular arcs are compared. A challenge arises here, since the singular arc problem is commonly analyzed with the original DOP (1-1), but (9-7) is a discretized formulation that does not have the continuous form. Therefore, a mathematically rigorous proof that the on-

Table 9-2: Contribution to the DDOP dimensions with a K-interval mesh for different rate constraint implementations on input variables

	add-state	**on-mesh**
2 point, collocated (Trapezoidal)	$K+1$ linear equality constraints (dynamic constraints)	$2K$ linear inequality constraints (pre-computed)
3 point, collocated (H-S)	$2K+1$ linear equality constraints (dynamic constraints)	$4K$ linear inequality constraints (pre-computed)
$N^{(k)}$ point, collocated (LGR)	$N^{(k)}K$ linear equality constraints (dynamic constraints)	$2N^{(k)}K$ linear inequality constraints (pre-computed)

mesh implementation will be singular-arc free using Pontryagin's minimum principle is not yet available. However, one would observe that, without introducing a dynamic constraint in the form of $\dot{u}(t) = v(t)$, the singular control situation as described in Proposition 3 will not occur, at least not in the same way. In addition, in our computational experience, there hasn't been a single case where the on-mesh implementation causes a singular-arc free problem to become singular. In contrast, the on-mesh rate constraint method was able to convert many well-known singular control problems to be singular arc free ones, with one example demonstrated in Section 9-2-1.

Another major advantage in comparison to the conventional implementation is regarding the computational cost. For systems with nonlinear dynamics, rate constraints on state variables implemented through (9-2) will be nonlinear path constraints relating different state variables at the same time instance. Thus their Jacobian and Hessian contributions can make the solution of the DOP computationally demanding. In contrast, on-mesh implementation of rate constraints with (9-7) are linear constraints, with no contribution to the Hessian.

In addition, note that the rate constraints (9-7) only depend on the numerical differentiation schemes. Thus, once a discretization scheme for the DOP has been chosen, and the corresponding discretization mesh has been determined, the Jacobian contributions of the rate constraint equations can be pre-computed during the transcription process. Therefore, although the DDOP dimension increases more rapidly with the on-mesh implementation as shown in Table 9-2 and 9-3, the computational complexity for obtaining the derivative information with respect to the rate constraint equations is actually lower than the conventional approach. Altogether, the computational advantages can be rather significant, as demonstrated with the example problem.

A remark is appropriate when comparing the on-mesh implementation in Table 9-3 to Table 9-2: For Hermite-Simpson discretization, specifically, the increase in the size of the DDOP for

Table 9-3: Contribution to the DDOP dimensions with a K-interval mesh for different rate constraint implementations on state variables

	add-path constraint	**on-mesh**
2 point, collocated (Trapezoidal)	$2(K+1)$ nonlinear inequality constraints (path constraints)	$2K$ linear inequality constraints (pre-computed)
3 point, collocated (H-S)	$2(2K+1)$ nonlinear inequality constraints (path constraints)	$6K$ linear inequality constraints (pre-computed)
$N^{(k)}$ point, collocated (LGR)	$2N^{(k)}K$ nonlinear inequality constraints (path constraints)	$2N^{(k)}K$ linear inequality constraints (pre-computed)

implementation on input variables is less than that on state variables. This is because, when the control u is discretized as a quadratic function of time, the rate of change w.r.t. time ($\dot{u}$) is linear, thus extreme values only occur at the end-points of each interval ($\upsilon_i^{(k)}$ and $\upsilon_{i+2}^{(k)}$). In this special case only, the rate constraints relating to the middle points ($\upsilon_{i+1}^{(k)}$) can be neglected.

9-2 Example problems

The problem of singular arcs is often demonstrated with toy problems in the literature (e.g. the first example), as they are much more illustrative and free from influence of other factors. However, this common practice often results in it being neglected by engineers working on complex problems. To show that it really matters, a second real-world example is also presented here to demonstrate the acclaimed benefits of the on-mesh rate constraint implementation in terms of solution quality and computational efficiency.

9-2-1 Second order singular regulator

First, consider a simple regulator problem originally presented in [3], which is essentially the regulation control of a double integrator system, with a constraint on the acceleration.

$$\min_{x_1,x_2,u} \int_0^5 x_1^2(t) + x_2^2(t)dt$$

subject to

$$\dot{x_1}(t) = x_2(t), \quad \dot{x_2}(t) = u(t) \in [-1,1] \quad \forall t \in [0,5].$$

In this original DOP formulation, the optimal control is in the form of *bang-singular*. Figure 9-1 shows that a numerical implementation of this DOP using direct collocation would yield

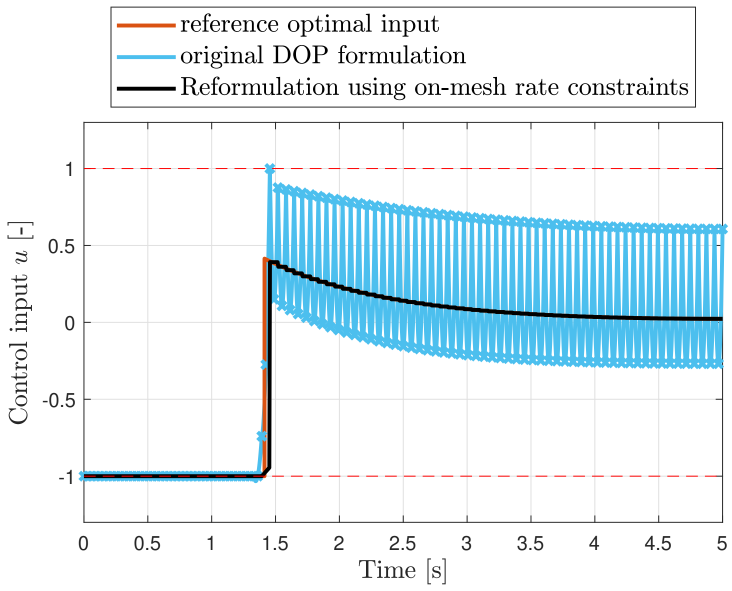


Figure 9-1: Comparison of obtained input trajectories for the second order singular regulator problem (HS with 80 mesh nodes, crosses represent the 159 collocation points)

fluctuations and ringing phenomena for solutions at collocation points, as well as for the trajectories in-between.

However, upon noticing that the second differential equation is equivalent to the add-state implementation of a rate constraint $-1 \leq \dot{x_2}(t) \leq 1$, that differential equation can be removed from the DOP and the on-mesh rate constraint method can be used instead. As illustrated in the figure, this approach successfully yields a stable and accurate solution in correspondence to the reference (analytical) optimal input trajectory. In other words, the proposed on-mesh rate constraint implementation has successfully converted the classical second order singular regulator problem into a singular-arc-free formulation.

This benefit does come with a price, however. Recall Table 4-1 regarding the order of the approximating polynomial for commonly used direct collocation schemes. By treating the state variable x_2 as an input variable instead, its approximation sees an order reduction from piece-wise cubic to piece-wise quadratic. Consequently, the trajectory of $\dot{x}_2$ will be piece-wise linear instead of piece-wise quadratic. Therefore, if solutions trajectories of higher orders are required for $\dot{x_2}$, one may have to consider increasing the order of approximating polynomials used in the direct collocation transcription and discretization process.

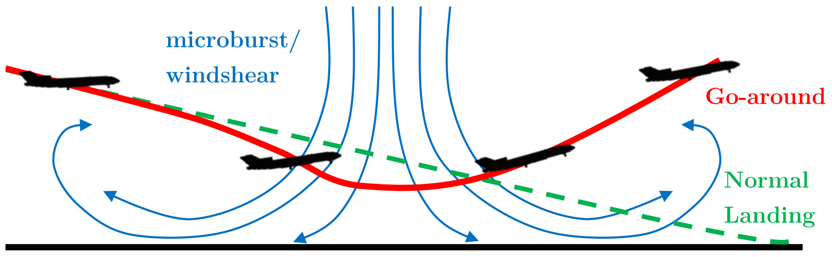


Figure 9-2: Illustration of the aircraft go-around in windshear problem.

9-2-2 Aircraft go-around in the presence of windshear

Based on the developments by [34, 35, 146], a problem is presented in [17] where the aircraft needs to stay as high above the ground as possible after encountering a severe windshear during landing. See the illustration in Figure 9-2. The implementation used here contains slight modifications to the windshear modelling, with the exponential functions approximated by piecewise polynomial functions. Note that the presence of singular arc fluctuations in this problem can also be suppressed with IRM-type of methods, see [156] for details.

The simplified dynamics of the aircraft can be described by

$$\dot{x}(t) = V(t)\cos(\gamma(t)) + w_d(t), \tag{9-9a}$$

$$\dot{h}(t) = V(t)\sin(\gamma(t)) + w_h(t), \tag{9-9b}$$

$$\begin{aligned}\dot{V}(t) =& \frac{1}{m}[T(t)\cos(\alpha(t)+\delta) - D(t)] - g\sin(\gamma(t)) \\ & - \dot{w}_d(t)\cos(\gamma(t))\sin(\gamma(t)) - \dot{w}_h(t)\sin(\gamma(t)),\end{aligned} \tag{9-9c}$$

$$\begin{aligned}\dot{\gamma}(t) =& \frac{1}{mv(t)}[T(t)\sin(\alpha(t)+\delta) + L(t)] - \frac{g\cos(\gamma(t))}{V(t)} \\ & + \frac{1}{V(t)}\dot{w}_d(t)\sin(\gamma(t)) - \frac{1}{V(t)}\dot{w}_h(t)\cos(\gamma(t)),\end{aligned} \tag{9-9d}$$

with state variables being the horizontal distance x, the altitude h, the true airspeed V, the flight path angle γ, and the input variable being the angle of attack α. Simple polynomial models are used for the maximum thrust (T_{max}), lift coefficient (C_L) and drag coefficient (C_D),

$$T_{max}(t) = a_0 + a_1 v(t) + a_2 v(t)^2,$$

$$C_L(t) = \begin{cases} c_0 + c_1\alpha(t), & \text{for } \alpha \le \alpha^*, \\ c_0 + c_1\alpha(t) + c_2(\alpha(t) - \alpha^*), & \text{for } \alpha^* \le \alpha \le \alpha_{max}, \end{cases}$$

$$C_D(t) = b_0 + b_1\alpha(t) + b_2\alpha(t)^2,$$

leading to the definition for thrust, lift and drag forces

$$\begin{aligned}
T(t) &= \begin{cases} (\beta_0 + \dot{\beta}_0 t)T_{max}(t), & \text{for } 0 \le t \le \frac{1-\beta_0}{\dot{\beta}_0}, \\ T_{max}(t), & \text{for } \frac{1-\beta_0}{\dot{\beta}_0} \le t \le t_f, \end{cases} \\
L(t) &= \frac{1}{2} C_L(t)\rho(t)Sv(t)^2, \\
D(t) &= \frac{1}{2} C_D(t)\rho(t)Sv(t)^2.
\end{aligned}$$

When comparing to classical flight mechanics equations, (9-9) clearly see the contributions of horizontal and vertical components of the wind. A simplified windshear model is used as

$$w_x(t) = \begin{cases} -50 + ax(t)^3 + bx(t)^4, & \text{for } 0 \le x(t) \le 500, \\ \frac{1}{40}(x - 2300), & \text{for } 500 \le x(t) \le 4100, \\ 50 - a(4600 - x(t))^3 - b(4600 - x(t))^4, & \text{for } 4100 \le x(t) \le 4600, \\ 50, & \text{for } 4600 \le x(t), \end{cases}$$

$$w_h(t) = \frac{h(t)}{h^*} \begin{cases} dx(t)^3 + ex(t)^4, & \text{for } 0 \le x(t) \le 500, \\ -51e^{-c(x(t)-2300)^4}, & \text{for } 500 \le x(t) \le 4100, \\ d(4600 - x(t))^3 - e(4600 - x(t))^4, & \text{for } 4100 \le x(t) \le 4600, \\ 0, & \text{for } 4600 \le x(t). \end{cases}$$

Details about the aerodynamic modeling, as well as parameter values of all relevant variables including constants a, b, c, d, e, a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2 and δ are available in the references above, thus will not be reproduced here. The following simple bounds on some of the state variables

$$\begin{aligned}
0 \le x(t) &\le 10000\,\text{ft}, & 0 \le h(t) &\le 1000\,\text{ft}, \\
0 \le V(t) &\le \infty\,\text{ft/s}, & -\infty \le \gamma(t) &\le \infty\,\text{deg}, \\
-17 \le \alpha(t) &\le 17\,\text{deg}, & -3 \le \dot{\alpha}(t) &\le 3\,\text{deg/s},
\end{aligned}$$

are imposed together with the boundary conditions

$$\begin{gathered}
x(0) = 0\,\text{ft}, \quad h(0) = 600\,\text{ft}, \quad V(0) = 239.7\,\text{ft/s}, \\
\gamma(0) = -2.25\,\text{deg}, \quad \alpha(0) = 7.35\,\text{deg}, \\
t_f = 40\,\text{s}, \quad x(t_f) = \text{free}, \quad h(t_f) = \text{free}, \\
V(t_f) = \text{free}, \quad \gamma(t_f) = 7.43\,\text{deg}, \quad \alpha(t_f) = \text{free}.
\end{gathered}$$

To avoid discontinuities and to assist convergence, a static optimization parameter h_{min}

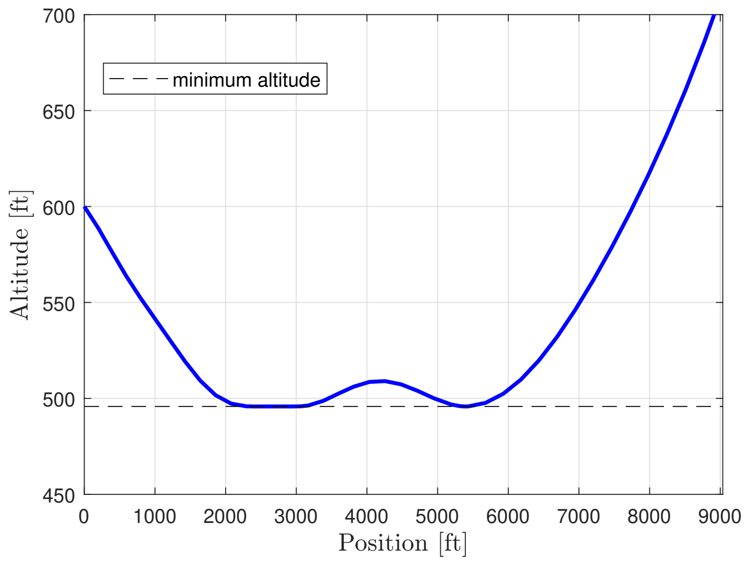


Figure 9-3: Solution to the aircraft go-around in the windshear problem, with input rate constraints

is introduced to represent the minimum altitude. The objective can then be expressed as $\Phi(x(t_0), t_0, x(t_f), t_f, p) := -h_{min}$ together with a new path constraint $h(t) \geq h_{min}$.

Figure 9-3 illustrates the solution to the problem using Hermite-Simpson discretization. All figures presented in this paper are the outcome of a mesh refinement scheme that minimizes errors to the tolerance as specified in Table 9-4.

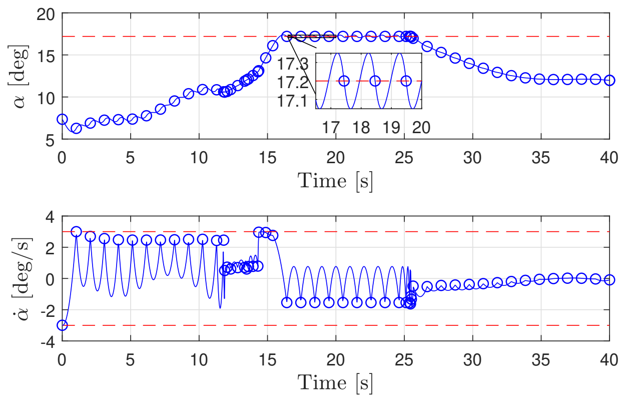
Implementation of rate constraints for input variables

Constraint $|\dot{\alpha}(t)| \leq 3$ applies directly on the rate of change for the control input α. Using the conventional approach, α can be treated as an additional state variable, and v introduced as the new control input with the dynamics

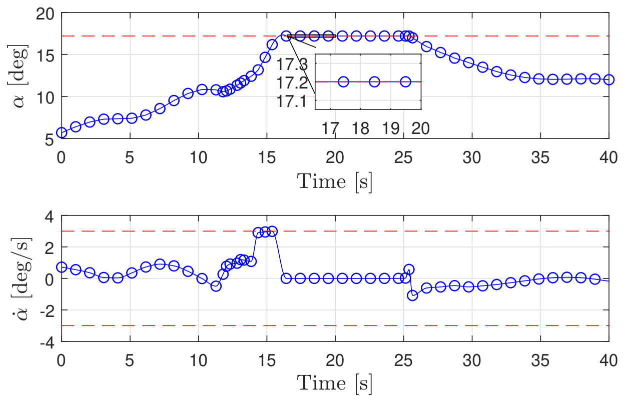
$$\dot{\alpha}(t) = v(t), \ \forall t \in [t_0, t_f] \text{ a.e.}$$

Table 9-4: Mesh refinement criteria for the aircraft go-around in the presence of windshear problem ©2020 IEEE

	x [ft]	h [ft]	v [ft/s]	γ [deg]	α [deg]	Path Constraint [m]
η_{tol}	1	0.5	0.1	0.5	0.5	-
$\varepsilon_{g_{tol}}$	1	0.5	0.1	0.5	0.5	1×10^{-5}



(a) implemented with additional state variable



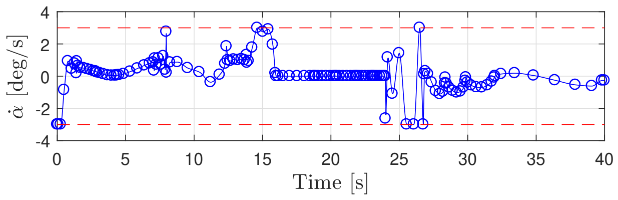
(b) direct implementation on the mesh

Figure 9-4: Control input for the solution to the aircraft go-around in the windshear problem, with different implementations for input rate constraints (H-S discretization, circles represent mesh points) ©2020 IEEE

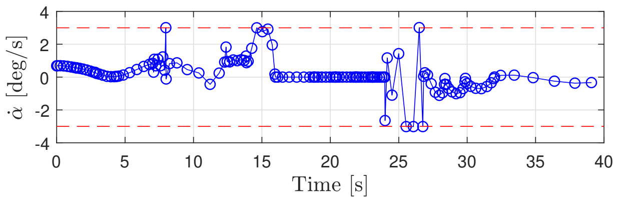
Thus the rate constraints for α can be implemented as simple bounds on v: $-3 \leq v(t) \leq 3$ deg/s.

As mentioned earlier, due to the fact that the original control input α appears nonlinearly in the system, whereas the new input v appears linearly, singular arc behaviour can occur, which is shown in Figure 9-4a, with large fluctuations in the solution. In contrast, when the rate constraints are directly implemented on the discretization mesh instead (Figure 9-4b), the optimal control input trajectory can be obtained with little ambiguity.

Comparing the solutions from the two implementations, it is interesting to observe that, although the integrated values (i.e. angle of attack) along the singular arc solution at the collocation points are generally the same, the interpolated trajectory from the add-state method is actually distorted by the fluctuations of its rate values.



(a) implemented with additional state variable



(b) direct implementation on the mesh

Figure 9-5: Control input for the solution to the aircraft go-around in the windshear problem, with different implementations for input rate constraints (LGR discretization, circles represent collocation points) ©2020 IEEE

With the LGR orthogonal collocation method, improvements are relatively minor. Because the end-point value for the control input is only approximated (not a collocation point), the errors have distortion effects on all previous points of the polynomial (Figure 9-5b). On the other hand, thanks to this extra level of continuity imposed by higher order polynomials, the problem of singular arc behaviour is far less pronounced with the conventional add-states implementation (Figure 9-5a), when compared to h type discretization methods.

From Figure 9-6 it can be seen that, for the computation time per iteration, the on-mesh implementation saw a slight advantage in comparison to the conventional approach. This is because the on-mesh implementation explicitly exploits the fact that the linear rate constraints have no contribution to the Hessian, and the contributions to the Jacobian are constants and can be pre-computed. The scale of the benefit also grows with the size of the mesh, from about 5% for a coarse mesh to around 10% for the dense mesh.

Figure 9-7 presents the computation performance of the problem when regularized with an additional $\rho||v||_2^2$ term. It can be seen that a relative large penalty weight is required to suppress the singular arc fluctuations, but with a larger ρ the result diverges quickly from the optimal. Also note that for a single solve with regularization, the norm of angle of attack rate ($||\dot{\alpha}^* - \dot{\alpha}||_2$) never reaches the accuracy level obtained by the on-mesh implementation with the same discretization mesh. Thus, to obtain a good solution, ρ needs to be gradually reduced, making the process complicated and computationally inefficient — it is also difficult to guarantee solution quality.

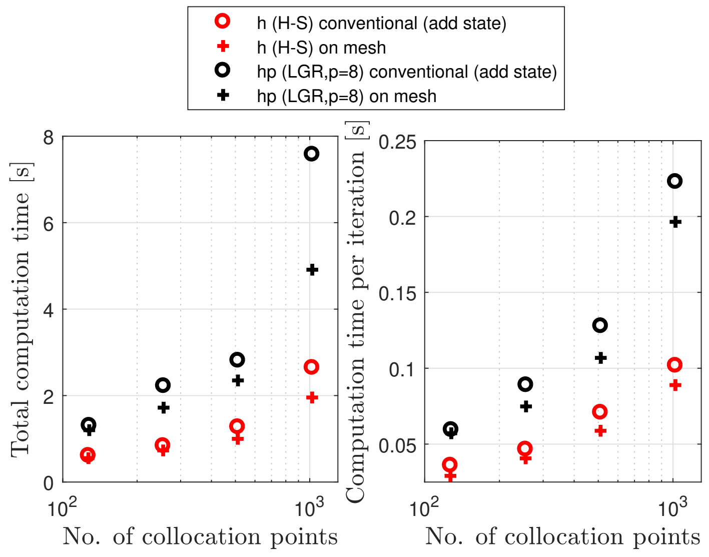


Figure 9-6: Comparison of computational performance, with input rate constraints ©2020 IEEE

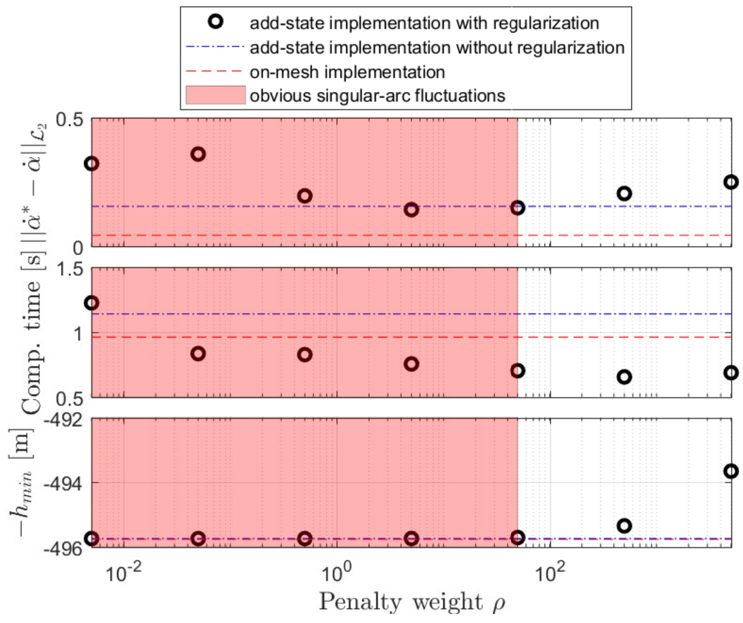


Figure 9-7: Solution of the regularized problem with different penalty weights. (H-S collocation with 79 collocation points (40 mesh nodes); reference solution $\dot{\alpha}^*$ obtained using a very dense mesh) ©2020 IEEE

Implementation of rate constraints for state variables

Additionally, a rate constraint for the velocity state is imposed as $-5 \leq \dot{v}(t) \leq 5\,\mathrm{ft/s^2}$. With this new formulation, the minimum altitude achievable is slightly lower.

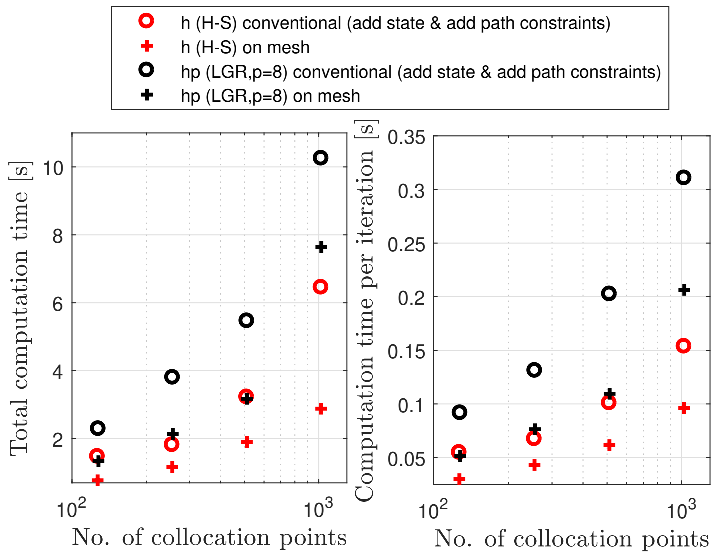


Figure 9-8: Comparison of computational performance, with state and input rate constraints ©2020 IEEE

From Figure 9-8, it is obvious that the two methods are not computationally comparable. Due to the reasons explained in the end of Section 9-1-2, although the increase in DDOP dimension is higher for the on-mesh implementation, the resulting (larger) DDOP problems with linear rate constraints are actually much easier to solve. Consequently, regardless of the discretization method, the computation time per iteration recorded for the on-mesh implementations are all significantly (more than 30%) lower than the conventional method.

Chapter 10

External Constraint Handling for Solving Dynamic Optimization Problems[†]

When formulating DOPs, it is common practice to impose a large number of constraints to ensure all mission specifications are fulfilled. However, for the solution obtained, it is often the case that only a small subset of the imposed inequality constraints will actually be active. Furthermore, even for the ones in this small subset, the duration for which each constraint is active is generally much shorter than the time dimension of the DOP. One example would be for the design of flight control systems: although all limits of the flight envelope need to be specified in the problem formulation for safety requirements, only in rare (abnormal) situations is it the case that some limits will be reached.

In numerical dynamic optimization, the DOPs are transcribed into DDOPs. The main computational overheads for solving the DDOPs are directly related to the number of decision variables and constraints. Thus there exist significant computational benefits to exclude inactive constraints in the problem formulation. One possibility is to only include the constraints that are determined to be active. Based on this, an external strategy for the handling of path constraints has been proposed in [42]. The idea is to first solve the unconstrained problem and determine which constraints are likely to be active based on constraint violations. These constraints are then added in the DOP, and the problem is repetitively solved until all original

[†]Material presented in this chapter has been published in the following work:
Y. Nie, E.C. Kerrigan, External Constraint Handling for Solving Optimal Control Problems With Simultaneous Approaches and Interior Point Methods. *IEEE Control Systems Letters.* 2019 Jun 7;4(1):7-12. ©2019 IEEE

constraints are satisfied. However, an issue arises when implementing the same idea on IPM based solvers together with mesh refinement, since good performance hinges on the initial point to be feasible, or at least close to feasible [78].

Another option is to remove constraints that are inactive. Removal of constraints for MPC has been studied to accelerate computations for linear MPC [108], tube-based robust linear MPC [107] and recently nonlinear MPC [55], with computational benefits clearly demonstrated. However, all of these implementations are based on a quadratic regulation cost, making their application specifically aimed at receding horizon control of regulation tasks.

In this chapter, an external constraint handling (ECH) strategy is introduced, tailored to IPM-based NLP solvers for solving a variety of DOPs. Implemented together with mesh refinement schemes, constraints that do not contribute to the solution are systematically removed in the problem formulation. Special attention is paid to ensure feasibility of the initial point. As a result, significant computational savings can be achieved.

Due to limitations in time and space, the proposed external constraint handling method will only be illustrated with direct collocation method. Similar benefits should be obtainable, after some adaptations, with other simultaneous methods. This might also be possible for sequential methods, such as direct single shooting.

Section 10-1 focuses on the identification of active and inactive constraints in DOP as well as in DDOP solutions. This is followed by a discussion in Section 10-2 on the criteria of ideal initial points for interior point methods. Then the proposed ECH strategy is introduced in Section 10-3. Lastly, a flight trajectory optimization example problem is presented in Section 10-4 to demonstrate the computational benefits.

10-1 Active and inactive constraints

10-1-1 Identifying active constraints in DOP

The inequality constraint (1-1d) is considered *active* if its presence influences the solution $\mathcal{Z}^* := \left(\chi^*, \upsilon^*, p^*, t_0^*, t_f^*\right)$. A constraint is *inactive* if it can be removed without affecting the solution. To clarify, consider a simplified problem:

$$y^* \in \arg\min_y \Phi(y) \quad \text{subject to} \quad c(y) \leq 0,$$

where conditions such that constraints can be determined to be active need to be identified. The most obvious criteria is when the solution y^* is at the boundary of $c(y) \leq 0$, i.e. $c(y^*) = 0$. Additionally, consider the Lagrangian $\mathcal{L} := \Phi(y) + \mu^T c(y)$ and the necessary optimality conditions (KKT conditions, see Section 2-1-4):

$$\frac{\partial \Phi(y)}{\partial y}|_{y=y^*} + \mu \frac{\partial c(y)}{\partial y}|_{y=y^*} = 0,$$
$$c(y^*) \leq 0, \qquad \mu \geq 0, \qquad \mu \circ c(y^*) = 0,$$

with $\circ$ the Hadamard product. For the last equation (the complementary slackness condition) to hold for strictly positive Lagrange multipliers ($\mu > 0$), the corresponding solution must have $c(y^*) = 0$, i.e. the constraint is active.

10-1-2 Identifying active constraints from DDOP solutions

Theoretically, the above-mentioned analysis applies only to the continuous DOP formulation (1-1). Additional challenges will arise in practice when solving the discretized problem (6-2) numerically: the NLP solver will only return the values of the discretized state χ, input υ, and Lagrange multipliers μ at grid points.

To estimate the constraint activation status in-between grid points, a criteria can be introduced based on the interpolated continuous trajectory $\tilde{z}$. By definition, inequality constraints are active if the magnitude of the differences between the actual constraint $c_l(\tilde{x}, \dot{\tilde{x}}, \tilde{u}, t_0, t_f, p)$ and the user-defined constraint bounds are zero, with c_l the l^{th} constraint in c. Due to numerical inaccuracies, however, there will always be a remainder. Thus, a constraint is considered to be potentially active if this difference is smaller than the constraint violation tolerance ϵ_{tol}.

Note that the word *potentially* is used to emphasise that, for numerical schemes under limited machine precision, no concrete determination of constraint active status can be made. On the other hand, only if the identified inactive constraints are truly inactive, then can they be removed from the DOP without affecting the solutions. Thus, it would be much more preferable to erroneously identify inactive constraints as active, than the opposite situation.

For this reason, the multiplier information is also used to enforce a larger (more conservative) selection of potentially active constraints. Here, a similar numerical challenge arises: with limited machine precision, even when the corresponding constraints are inactive, the multiplier values are rarely truly equal to zero. To identify the regions where the constraints are likely to be active, the numerical multiplier data μ is first normalized between 0 and 1 for each constraint $c_l(\chi_i, \dot{\chi}_i, \upsilon_i, t_0, t_f, p) \leq 0$. Signal processing algorithms can be used to identify different intervals where the behaviour of Lagrange multipliers have significant changes, for example using the MATLAB `findchangepts` function.

For each identified interval $\mathcal{T}_j, \forall j \in \mathbb{I}_E$, the mean value of the normalized multipliers ($\bar{\mu}_{\mathcal{T}_j}$) is calculated and compared based on the following criteria:

$$\begin{array}{ll} \text{if } \bar{\mu}_{\mathcal{T}_j} \geq \Theta & \text{constraint potentially active in interval } \mathcal{T}_j \\ \text{otherwise} & \text{constraint potentially inactive in interval } \mathcal{T}_j \end{array}$$

with Θ a threshold parameter.

To sum up, the following definition is used to determine whether the constraints are potentially active or potentially inactive at different grid points.

Definition 1. *A constraint $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) \leq 0$ is potentially active at time $\tau_i^{(k)}, \forall i \in \mathbb{I}_{N^{(k)}}$ if one of the following criteria is met:*

- *Between adjacent grid points, i.e. $t \in [\tau_{i-1}^{(k)}, \tau_{i+1}^{(k)}]$, $c_l(\tilde{x}(t), \dot{\tilde{x}}(t), \tilde{u}(t), t_0, t_f, p) \geq -\epsilon_{tol}$ holds, with $\epsilon_{tol} > 0$.*
- *$\bar{\mu}_{\mathcal{T}_j} \geq \Theta$, with $\tau_i \in \mathcal{T}_j$.*

Otherwise, a constraint is potentially inactive at time $\tau_i^{(k)}$.

In addition to identifying the time instances at which certain constraints may be potentially inactive, it is also preferable to determine the sets of constraints that never become active at all times.

Definition 2. *A set of constraints $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) \leq 0$ is potentially redundant if for all $\tau_i^{(k)}, \forall i \in \mathbb{I}_{N^{(k)}}$ and $\forall k \in \mathbb{I}_K$, the constraints $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) \leq 0$ are potentially inactive. Otherwise, this set of constraints is said to be potentially enforced.*

10-2 Initialization for interior point methods

IPM for solving NLPs were introduced in the early 1960s [61–63] and have became very popular in numerical dynamic optimization. The idea is to augment the objective function with barrier functions of constraints in order to enforce their satisfaction. Potential solutions will iterate only in the feasible region following the so-called central path, resulting in a very efficient algorithm.

Standard interior point methods are sensitive to the choice of a starting point. To ensure that the initial guess is strictly feasible with respect to constraints, various initialization methods [17, 144] have been developed and implemented in modern solvers.

To ensure reliable and efficient computation of the initialization algorithm, as well as the subsequent NLP iterations, several criteria [78] can be formulated regarding ideal initial points for IPMs. The ideal initial point should:

- satisfy or be close to primal and dual feasibility, and
- be close to the central path, and
- be as close to optimality as possible.

Because of these characteristics, external constraint handling schemes based on constraint inclusions [42], are not very suitable for IPM-based solvers theoretically. By first solving the unconstrained problem and gradually adding constraints based on the constraint violation error, the solution of previous solves will all be infeasible for the new DOP formulation, and the solution may undergo drastic changes as well.

For IPM-based NLP solvers of IPOPT, the work [42] also demonstrated with some example problems that the constraint handling scheme can lead to significant savings in computation time when using a fixed mesh design. However, our experience has shown that when used together with mesh refinement, the combination of increased problem dimensions and constraint violations often leads to a higher computational overhead for initialization, as well as higher chances for the iterations to frequently enter the slow and unreliable feasibility restoration phase.

10-3 Proposed scheme for constraint handling

Based on the criteria presented in Section 10-1-2 and the characteristics of IPMs as discussed in Section 10-2, a strategy for efficiently handling constraints in DOPs solved with IPM-based NLP solvers is proposed, with the work-flow presented in Figure 10-1. The approach is called *external*, since the modifications to the DOP are made at the mesh refinement iteration level, instead of during the NLP iterations.

The unmodified DOP is first solved on the initial coarse mesh. Even with all constraint equations included, the computation time will still be quite low at this stage due to the small problem size. Once the solution is obtained, potentially inactive constraints and potentially redundant constraint sets can be identified, based on Definitions 1 and 2, with potentially redundant constraints directly excluded from the DOP formulation. Furthermore, if the problem has a fixed terminal time, i.e. the time instance corresponding to a mesh point will not change, then potentially inactive constraints in the potentially enforced constraint sets may also be removed.

Recall that it is preferable to erroneously identify an inactive constraint as potentially active, rather than the opposite. It is therefore often a good idea in practice to enlarge the intervals with potential constraint activation by an interval of length Π in each direction, with Π either fixed or adapting during the mesh refinement process. This adaptation also guarantees the

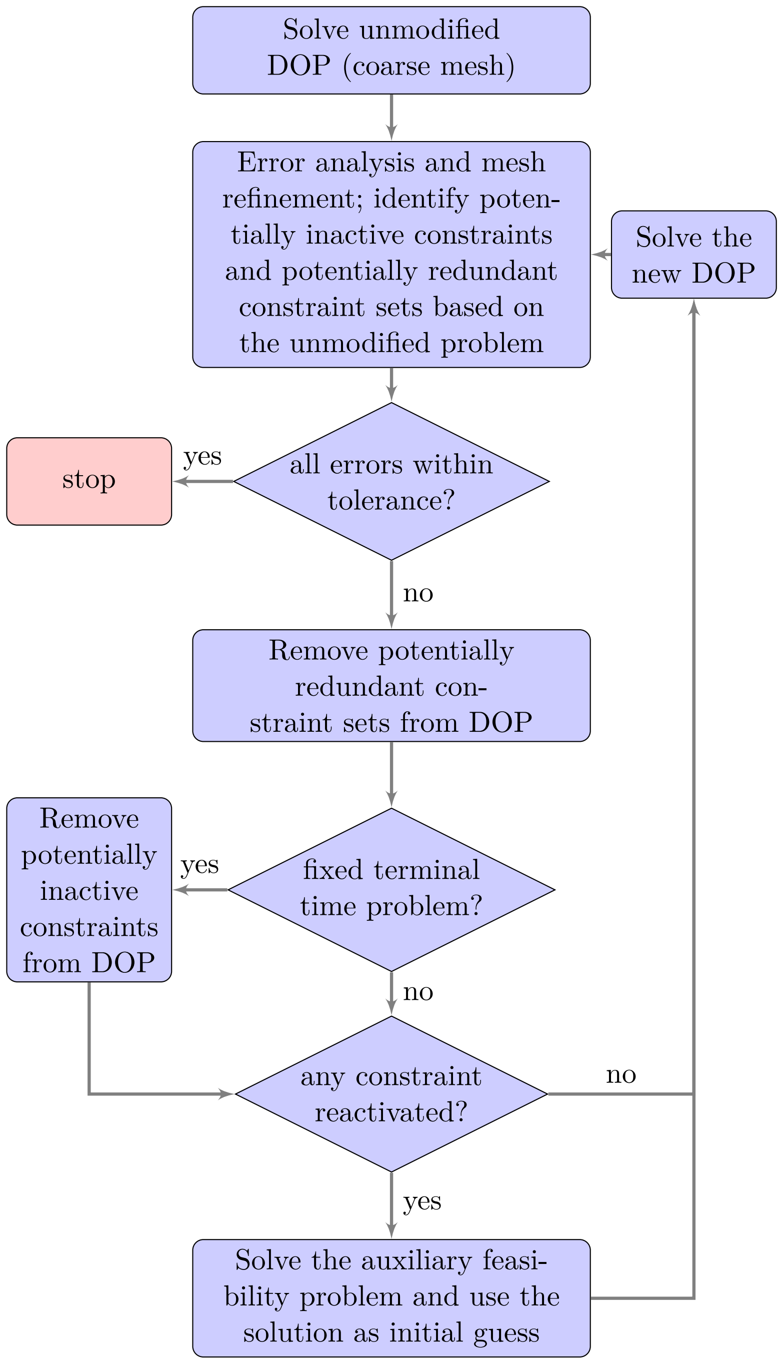

Solve unmodified DOP (coarse mesh)
Error analysis and mesh refinement; identify potentially inactive constraints and potentially redundant constraint sets based on the unmodified problem
Solve the new DOP
all errors within tolerance?
yes
stop
no
Remove potentially redundant constraint sets from DOP
fixed terminal time problem?
yes
Remove potentially inactive constraints from DOP
no
any constraint reactivated?
no
yes
Solve the auxiliary feasibility problem and use the solution as initial guess

Figure 10-1: Overview of the proposed external constraint handling scheme ©2019 IEEE

convergence of the overall scheme, i.e. in the worst case, Π can be sufficiently large to impose the constraints for the whole trajectory, with the original problem recovered.

In-between mesh refinement iterations, special attention must be made to constraints and constraint sets that were determined to be potentially inactive or redundant in the previous solves. If they never become active or enforced again, the constraint removal process may continue until mesh refinement is converged. But there will be the chance, after refining the mesh, the constraint violation error analysis dictates that certain constraints and constraint sets that have been removed earlier may become potentially active or enforced again. If this happens, they need to be included again to ensure that the solution of the modified DOP is equivalent to the unmodified problem.

Note that the previous solution will no longer be a feasible initial guess for the new problem formulation. To assist the subsequent solve of NLPs, the following auxiliary feasibility problem (AFP) can be solved before proceeding:

$$J^* := \min_{\chi,\upsilon,p,t_0,t_f} \sum_{l=1}^{n_c} \Xi_l \tag{10-1a}$$

subject to, $i \in \mathbb{I}_{N^{(k)}}$, $k \in \mathbb{I}_K$ and $l \in \mathbb{I}_{n_c}$,

$$\begin{aligned}
\sum_{j=1}^{N^{(k)}} \mathcal{A}_{ij}^{(k)} \chi_j^{(k)} + \mathcal{D}_{ij}^{(k)} f\left(\chi_j^{(k)}, \upsilon_j^{(k)}, t_0, t_f, p\right) &= 0, && \text{(10-1b)}\\
g\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) &= 0, && \text{(10-1c)}\\
c_l\left(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p\right) &\leq \Xi_l, && \text{(10-1d)}\\
\phi(\chi_1^{(1)}, t_0, \chi_f^{(K)}, t_f, p) &= 0, && \text{(10-1e)}\\
\Xi_l &\geq 0, && \text{(10-1f)}
\end{aligned}$$

with $\Xi \in \mathbb{R}^{n_c}$ slack variables and Ξ_l the l^{th} element of it. The initial guess for the AFP will be $\hat{\mathcal{Z}} := (\hat{\chi}, \hat{\upsilon}, t_0, t_f, p)$, i.e. the values of the interpolated solution $\tilde{z}$ at $\hat{N}$ grid points of the refined mesh, with $\hat{N}^{(k)}$ data points in interval k and in total $\hat{K}$ intervals. Additionally, $\dot{\hat{\chi}}$ can be obtained from $\dot{\tilde{x}}(t)$ at the grid points of the refined mesh.

10-3-1 Properties of the external constraint handling strategy

It is possible to derive proofs of feasibility and optimality invariance for the removal of constraints on a given discretization mesh. However, with the size of the mesh changing throughout the refinement process, the analysis of errors, and identification of constraint activation status (Section 10-1-2) are all subject to considerable uncertainties. When a constraint or constraint set must be included again in the problem, it will be challenging to ensure that the subsequent DOP solve can be supplied with a feasible initial guess. The introduction of the AFP is the answer to this challenge, with its solution guaranteed to be a feasible point for the corresponding original DOP.

Proposition 4. *If the original discretized DOP* (6-2) *has feasible points, then a solution to the auxiliary feasibility problem* (10-1) *will be a feasible point of* (6-2) *on the same discretization mesh, and* (10-1) *will have corresponding objective value $J^* = 0$.*

Proof. If $\mathcal{Z} := (\chi, \upsilon, t_0, t_f, p)$ is a feasible point of (6-2), then (6-2d) must hold. With $\Xi \geq 0$, the solution for the AFP (10-1) will be the situation where $\sum \Xi_l = 0$, and (6-2d) guarantees the existence of such a solution. □

Now, for the very same reason, a suitable initial guess is needed for the slack variables Ξ in the AFP. One possible way is by calculating the constraint violation errors of the interpolated solutions on the refined mesh.

Proposition 5. *Define $\hat{\Xi} \in \mathbb{R}^{\hat{N} \times n_c}$ as the absolute local constraint violation error $\epsilon(t)$ calculated at $\hat{N}$ grid points of the refined mesh, with the updated initial guess $\hat{\mathcal{Z}} := (\hat{\chi}, \hat{\upsilon}, p, t_0, t_f)$. For any set $\{\bar{\Xi} \in \mathbb{R}^{n_c} \mid \bar{\Xi}_l \geq \max_{i \in \mathbb{I}_{\hat{N}^{(k)}}, k \in \mathbb{I}_{\hat{K}}} (\hat{\Xi}_{i,l}^{(k)}), l \in \mathbb{I}_{n_c}\}$ implemented as the initial guess for Ξ, the AFP* (10-1) *will have a strictly feasible initial point with respect to the constraints* (10-1d).

Proof. $\hat{\Xi}_{i,l}^{(k)} := |\min(-c_l(\hat{\chi}_i^{(k)}, \dot{\hat{\chi}}_i^{(k)}, \hat{\upsilon}_i^{(k)}, t_0, t_f, p), 0)|$ by definition, for all $i \in \mathbb{I}_{\hat{N}^{(k)}}$, $k \in \mathbb{I}_{\hat{K}}$ and $l \in \mathbb{I}_{n_c}$,

- If $c_l(\hat{\chi}_i^{(k)}, \dot{\hat{\chi}}_i^{(k)}, \hat{\upsilon}_i^{(k)}, t_0, t_f, p) < 0$, i.e. the constraint is satisfied and the solution is not on the boundary, then $\hat{\Xi}_{i,l}^{(k)} = 0$, thus $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) < \hat{\Xi}_{i,l}^{(k)}$ holds.
- If $c_l(\hat{\chi}_i^{(k)}, \dot{\hat{\chi}}_i^{(k)}, \hat{\upsilon}_i^{(k)}, t_0, t_f, p) = 0$ (constraint satisfied and solution is on the boundary) or $c_l(\hat{\chi}_i^{(k)}, \dot{\hat{\chi}}_i^{(k)}, \hat{\upsilon}_i^{(k)}, t_0, t_f, p) > 0$ (constraint violations), then $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) = \hat{\Xi}_{i,l}^{(k)}$ holds.

Therefore, $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) \leq \hat{\Xi}_{i,l}^{(k)}$ will always be true. From the defination that $\bar{\Xi}_l \geq \max_{i \in \mathbb{I}_{\hat{N}^{(k)}}, k \in \mathbb{I}_{\hat{K}}} (\hat{\Xi}_{i,l}^{(k)})$, it can be concluded that $c_l(\chi_i^{(k)}, \dot{\chi}_i^{(k)}, \upsilon_i^{(k)}, t_0, t_f, p) \leq \bar{\Xi}_l$ holds. □

Now it can be shown that, except for the initial solve, all subsequent solves will have feasible initial guesses.

Proposition 6. *If the unmodified DOP has feasible points, and the initial solve of the discretized DOP has been successful, then all subsequent solves of DOPs and AFPs with mesh refinement schemes and the proposed external constraint handling method will have a feasible initial point with respect to the constraints* (6-2d) *or* (10-1d).

Proof. For any interpolated solution $\hat{\mathcal{Z}} := (\hat{\chi}, \hat{\upsilon}, p, t_0, t_f)$ on the new mesh, if (6-2d) is not satisfied, then the corresponding AFP will be solved and Proposition 5 ensures that AFP will have a feasible initial guess. From Proposition 4 the solution of the AFP will be a feasible initial guess for the subsequent DOP solve. □

10-3-2 A practically more efficient alternative implementation

The proposed external constraint handling scheme can guarantee feasible initial points under conditions stated in Proposition 6. Nevertheless, it may not always be efficient in practice. The frequent solve of AFPs are not only time consuming, but are often not necessary.

Recall the conditions regarding ideal initial guesses for IPM methods. It is not necessary to satisfy primal and dual feasibility — rather, one only needs to be close to fulfillment. In practice, the computational performance of a modern IPM that uses near-feasible initial guesses is very much comparable to using feasible initial points. In addition, constraint satisfaction for simple bounds can be computationally much easier to achieve by the NLP solver, thus there is no need to enforce those through the solve of an AFP.

Thus, a practically more efficient version of the external handling scheme can be formulated, by restricting the conditions for solving the AFP to the mesh refinement iteration when a potentially redundant constraint set turns into a potentially enforced constraint set.

10-4 Example problem: flight path of a commercial aircraft

To demonstrate the computational benefits of the proposed ECH scheme, a problem that is relatively large in the horizon length is shown. The task involves finding a fuel-optimal flight path of a commercial aircraft where authorities have identified five non-flight zones (NFZ) for the aircraft to avoid.

From simple flight mechanics with a flat earth assumption, both the longitudinal and lateral motion of the aircraft can be described by the dynamic equations

$$\begin{aligned}
\dot{h}(t) =& v_T(t)\sin(\gamma(t)),\\
P\dot{O}S_N(t) =& v_T(t)\cos(\gamma(t))\cos(\chi(t)),\\
P\dot{O}S_E(t) =& v_T(t)\cos(\gamma(t))\sin(\chi(t)),\\
\dot{v}_T(t) =& \frac{1}{m(t)}(T(t) - D(t) - m(t)g\sin(\gamma(t))),\\
\dot{c}(t) =& \frac{1}{m(t)v_T(t)}(L(t)\cos(\phi(t)) - m(t)g\cos(\gamma(t))),\\
\dot{\chi}(t) =& \frac{L(t)\sin(\phi(t))}{\cos(\gamma(t))m(t)v_T(t)},
\end{aligned}$$

$$\dot{m}(t) = FF(t),$$

with h the altitude, POS_N and POS_E the north and east position, v_T the true airspeed, γ the flight path angle, χ the tracking angle, and m the mass. T is the thrust force as a function of v_C, h and Γ. L and D are lift and drag force respectively, both as a function of v_T, h and α. FF is the fuel flow model, as a function of v_C, h and Γ as well.

v_C is the calibrated airspeed, which can be related to the true airspeed v_T via a conversion as follows

$$v_T(t) = \sqrt{2\xi\frac{p(t)}{\rho(t)}\left[\left[1+\frac{p_0}{p(t)}\left[(1+\frac{1}{2\xi}\frac{\rho_0}{p_0}v_C(t)^2)^{\xi}-1\right]\right]^{\frac{1}{\xi}}-1\right]},$$

with $\xi = \frac{\kappa}{\kappa-1}$, and $\kappa = 1.4$ the heat capacity ratio of the air, p and ρ the pressure and air density at flight altitude, and p_0 and ρ_0 the pressure and air density at sea level.

Additionally, $g = 9.81\,\mathrm{m/s^2}$ is gravitational acceleration. There are three control inputs, the roll angle ϕ in rad, the throttle settings Γ normalized between 0 and 1, and the angle of attack α in rad. Further details of the modelling of a Fokker 50 aircraft can be obtained from [48].

The avoidance of NFZs can be implemented with the following path constraints

$$(POS_N(t) - POS_{N_N})^2 + (POS_E(t) - POS_{E_N})^2 \geq r_N^2,$$

with POS_{N_N} and POS_{E_N} the north and east position of the center of the non-flight zones, and r_N the radius.

The problem will have the boundary cost $\Phi = -m(t_f)$ (maximize the mass at the end of the flight, with fixed $t_f = 7475\,\mathrm{s}$), subject to the dynamics and path constraints. Furthermore, variable simple bounds

$$\begin{gathered}
487.68 \leq h(t) \leq 7620\,\mathrm{m}, \quad 0 \leq POS_N(t) \leq 1.2\times10^6\,\mathrm{m}, \quad 0 \leq POS_E(t) \leq 1.2\times10^6\,\mathrm{m}, \\
100 \leq v_T(t) \leq 170.5\,\mathrm{m/s}, \quad -1.0472 \leq \gamma(t) \leq 1.0472\,\mathrm{rad}, \quad -\pi \leq \chi(t) \leq \pi\,\mathrm{rad}, \\
14000 \leq m(t) \leq 18000\,\mathrm{kg}, \quad -0.1745 \leq \alpha(t) \leq 0.1745\,\mathrm{rad}, \quad -0.7854 \leq \phi(t) \leq 0.7854\,\mathrm{rad}, \\
0 \leq \Gamma(t) \leq 1, \quad -0.0349 \leq \dot{\alpha}(t) \leq 0.0349\,\mathrm{rad/s}, \quad -0.1745 \leq \dot{\phi}(t) \leq 0.1745\,\mathrm{rad/s},
\end{gathered}$$

are imposed together with the boundary conditions,

$$\begin{gathered}
h(0) = 487.68\,\mathrm{m}, \quad POS_N(0) = 0\,\mathrm{m}, \quad POS_E(0) = 0\,\mathrm{m}, \quad v_T(0) = 100\,\mathrm{m/s}, \\
c(0) = 0.1396\,\mathrm{rad}, \quad \chi(0) = 0\,\mathrm{rad}, \quad m(0) = 18000\,\mathrm{kg}, \quad \phi(t_f) = 0\,\mathrm{rad}, \\
h(t_f) = 609.60\,\mathrm{m}, \quad POS_N(t_f) = 8\times10^5\,\mathrm{m}, \quad POS_E(t_f) = 9\times10^5\,\mathrm{m}, \\
v_T(t_f) = 100.6\,\mathrm{m/s}, \quad c(t_f) = -0.0524\,\mathrm{rad}, \quad \chi(t_f) = -2.3562\,\mathrm{rad}.
\end{gathered}$$

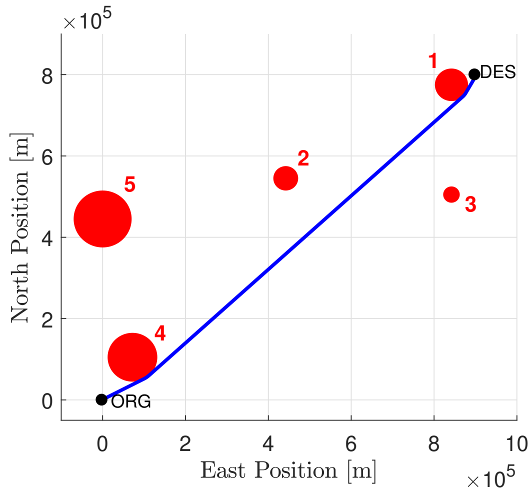


Figure 10-2: The fuel-optimal flight profile for the example problem ©2019 IEEE

Table 10-1: User-Defined Tolerances: Aircraft Flight Profile ©2019 IEEE

	η_{tol}	$\varepsilon_{g_{tol}}$		η_{tol}	$\varepsilon_{g_{tol}}$
h [m]	5	5	m [kg]	0.1	0.1
POS_N [m]	1	1	α [rad]	-	0.0087
POS_E [m]	1	1	ϕ [rad]	-	0.0087
V_T [m/s]	0.5	0.5	Γ [-]	-	0.1
γ [rad]	0.0087	0.0087	NFZ violations [m]	-	1
χ [rad]	0.0087	0.0087			

Figure 10-2 illustrates the results solved to a user-defined tolerance listed in Table 10-1. Using the proposed external constraint handling scheme, the problem is first solved with a worst-case buffer interval setting of $\Pi = 0\,\text{s}$. The history for constraint activation intervals implemented in the DOP are demonstrated in Table 10-2. It can be seen that in the initial solve (mesh refinement iteration 1), all constraint sets are enforced and all constraints are treated as potentially active. Based only on the solution from this coarse grid, the ECH method correctly identified that the constraint sets related to NFZ 2, 3 and 5 are all potentially redundant. It also determined that constraints related to NFZ 1 are only potentially active near the end of the flight, whereas for NFZ 4 they are at the beginning of the mission.

In later iterations of the mesh refinement, these intervals had only some minor adjustments. It can be seen that without implementing any buffer interval, there do exist occasions where active constraints got erroneously identified as inactive for finer meshes. However, the constraint violation error analysis in the mesh refinement process correctly identified these situations and made corrections accordingly.

Table 10-3 compares the computational performance of the standard solve, as well as solves

Table 10-2: External constraint handling history for the aircraft flight path example (t_0 =0 s, $t_f = 7475$ s, $\Pi = 0$ s) ©2019 IEEE

NFZ	Constraint Activation Intervals Implemented in the DOP [s]					
	MR Iter. 1 (K = 40)	MR Iter. 2 (K = 81)	MR Iter. 3 (K = 136)	MR Iter. 4 (K = 176)	MR Iter. 5 (K = 207)	MR Iter. 6 (K = 256)
1	$[t_0, t_f]$	[6900 7092]	[6996 7114]	[7056 7162]	[7071 7140]	[7074 7150]
2/3/5	$[t_0, t_f]$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$[t_0, t_f]$	[671 863]	[743 942]	[767 875]	[774 880]	[774 878]

Table 10-3: Computational performance for the aircraft flight path example ©2019 IEEE

	Standard Solve	With ECH ($\Pi = 0$ s)	With ECH ($\Pi = 747.5$ s)
Total Comp. Time [s]	130.54	92.14 (29% lower)	78.27 (40% lower)
mesh refinement Iterations	6	6	6
Re-comp. Time [s]	21.82	13.50 (38% lower)	10.78 (50% lower)
Fuel Used [kg]	1787.6	1787.6	1787.6

with external constraint handling using the alternative ECH implementation (allowing near-feasible initial guesses) described in Section 10-3-2, with two different buffer interval settings. With the worst-case setting of $\Pi = 0$, the total computation time saw a 29% reduction, while the number of mesh refinement iterations remained the same. Choosing a much more conservative buffer interval setting of $\Pi = 0.1(t_f - t_0) = 747.5$ s further improved this time reduction to 40%, with initial guesses being feasible for all later mesh refinement iterations.

For real-time applications, it is useful to consider the re-computation time for solving the DOP problem again with the final (refined) discretization mesh, using the obtained solutions as initial guesses. For the ECH method with $\Pi = 747.5$ s, the time taken was only half compared to the standard solve. Therefore, the benefits of the proposed scheme can be seen for both off-line and online applications.

Chapter 11

Suppression of Singular Arc Fluctuations

As illustrated in previous chapters, singular arcs can be present in the solution of DOPs. On the singular arc, the numerical solutions may have large fluctuations in the solution trajectories, leading to large residuals and constraint violation errors in-between grid points. In addition, a fluctuating input trajectory can be more prone to implementation errors when is applied. Therefore, methods that can suppress the singular arc fluctuations are particularly attractive for practical applications.

In this chapter, with the help of a simple singular control example problem, different singular arc ringing suppression approaches are demonstrated and compared, ranging from problem reformulations and different choices of transcription methods, to posterior fixes after a fluctuating solution has been obtained. Section 11-4 lists the advantages and disadvantages of each approach and provides guidelines on making the appropriate choices.

11-1 Double integrator minimum-time repositioning

To demonstrate the occurrence of singular arcs using one of the simplest problems possible, the double-integrator minimum-time repositioning example is selected. The problem has the same problem formulation as [17, Ex. 4.1], but with different initial and terminal conditions, and variable bounds. A double integrator plant can be physically interpreted as the system in figure 11-1.

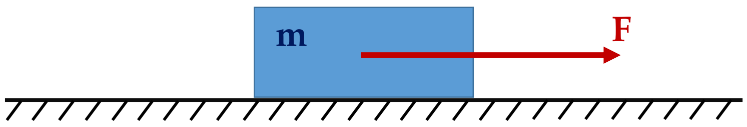


Figure 11-1: A physical interpretation of double integrator plant

by neglecting viscous friction and assuming unit mass ($m = 1$). Then the corresponding state-space formulation of the dynamics is:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} F. \tag{11-1}$$

Then the DOP to find a solution such that the object is re-positioned from $x = 0\,\mathrm{m}$ to $x = 300\,\mathrm{m}$ in minimum time will be

$$\min_{x,v,F,t_f} \quad t_f,$$

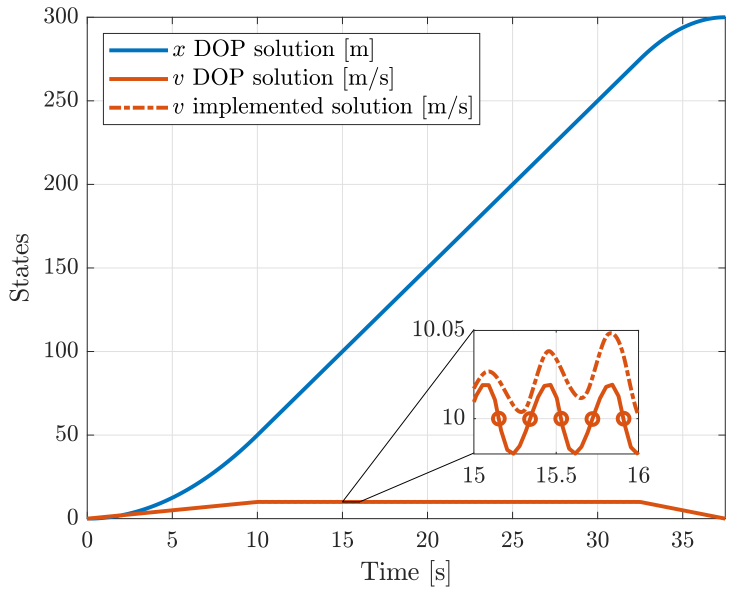
subject to dynamic constraints (11-1), variable simple bounds

$$\begin{aligned} -10 \leq x(t) \leq 300\,\mathrm{m}, &\quad -10 \leq v(t) \leq 10\,\mathrm{m/s}, \\ -2 \leq F(t) \leq 1\,\mathrm{N}, &\quad 0 \leq t_f \leq 50\,\mathrm{s}, \end{aligned}$$

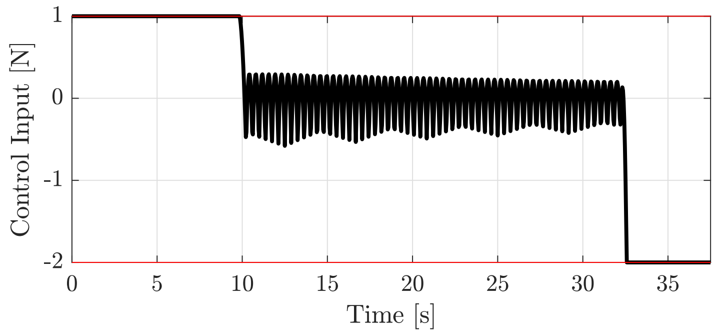
and boundary conditions

$$\begin{aligned} x(0) = 0\,\mathrm{m}, &\quad v(0) = 0\,\mathrm{m/s}, \\ x(t_f) = 300\,\mathrm{m}, &\quad v(t_f) = 0\,\mathrm{m/s}. \end{aligned}$$

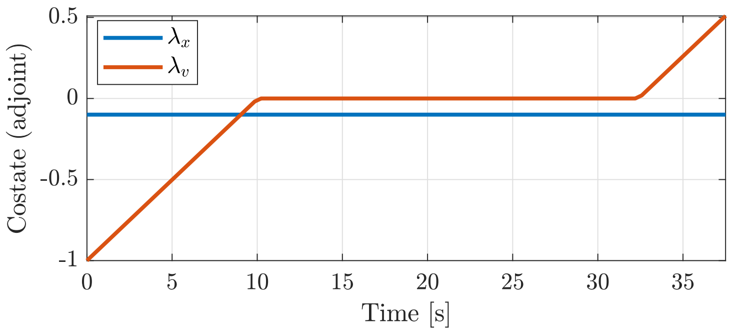
Without the simple bounds on velocity v, the problem is often known as the *bang-bang* control problem, with the optimal input trajectory contains a single switch from the largest positive force for maximum acceleration to the greatest negative force for maximum braking. However, with a tight bound in place for v, the structure of optimal input can become *bang-singular-bang*. In this case, the solution obtained directly from a direct collocation transcription will show large fluctuations on the singular arc, as demonstrated in Figure 11-2.



(a) States



(b) Input



(c) Costates

Figure 11-2: Time-optimal solution for double integrator repositioning problem (single phase direct collocation, HS discretization, 100 major nodes, circles represent collocation points)

11-2 Analysis of the problem

11-2-1 In framework of singular control

The large fluctuations in the solution trajectories can be analyzed under the framework of singular control. First, formulate the Hamiltonian of the system:

$$\mathcal{H} = \lambda_x(t)v(t) + \lambda_v(t)F(t). \tag{11-2}$$

It can be seen that the expression is not an explicit function of time, hence $\mathcal{H}$ will be a constant. Additionally, from the transversality condition (2-6d), the terminal value for Hamiltonian is known:

$$\mathcal{H}(t_f) = -\frac{\partial \Phi}{\partial t_f} = -\frac{\partial t_f}{\partial t_f} = -1.$$

Hence, combining both information, the Hamiltonian of the problem is known to be $\mathcal{H} = -1$.

From the PMP (2-3), the optimal control input will be

$$F^*(t) = \begin{cases} 1 & \text{if } \lambda_v^*(t) < 0 \\ ? & \text{if } \lambda_v^*(t) = 0 \\ -2 & \text{if } \lambda_v^*(t) > 0 \end{cases},$$

which corresponds to the observation in Figure 11-2 that as soon as λ_v equals zero, the input F starts to exhibit large fluctuations in the solution. On the singular arc with $\lambda_v(t) = 0$, the costate corresponding to x can be computed from the Hamiltonian (11-2) with

$$\lambda_x(t) = \frac{\mathcal{H} - \lambda_v(t)F(t)}{v(t)} = -\frac{1}{v(t)}.$$

substitute in $v(t) = 10$ in the equation would yield $\lambda_x(t) = -0.1$, which is in correspondence with the costate solution shown in Figure 11-2c.

Since $\frac{\partial^2 \mathcal{H}}{\partial u^2}$ is singular on the singular arc and provides no information regarding the control input, the Hamiltonian has to be differentiated respect to time:

$$\dot{\mathcal{H}} = \dot{\lambda}_x(t)v(t) + \lambda_x(t)\dot{v}(t) + \dot{\lambda}_v(t)F(t) + \lambda_v(t)\dot{F}(t). \tag{11-3}$$

From costate dynamics:

$$\dot{\lambda}_x(t) = -\frac{\partial \mathcal{H}}{\partial x} = 0,$$

and from $\mathcal{H} = -1$ and $\lambda_v(t) = 0$, it is trivial to also deduce that $\dot{\mathcal{H}} = 0$ and $\dot{\lambda}_v(t) = 0$.

Subsitituing all known information in (11-3) yield

$$\lambda_x(t)\dot{v}(t) = 0.$$

Since $\lambda_x(t) = -0.1$, the condition for the above equation to hold is therefore

$$\dot{v}(t) = 0,$$

which is the *singular arc condition* for this double integrator repositioning problem.

11-2-2 In framework of DAE constraints

Another approach to analyze the issue, inspired by [130], is to view the problem as a semi-explicit DAE system during the period of constraint activation. In other words, the original formulation with inequality constraint $v(t) \leq 10\,\mathrm{m/s}$ specified for the whole trajectory is equivalent to a problem with an equality constraint $v(t) = 10\,\mathrm{m/s}$ added in the dynamic equations only for the arc where the inequality constraint becomes active, which also effectively increases the DAE index of the system by 1.

In direct collocation transcription, when constraint $v(t) \leq 10\,\mathrm{m/s}$ is enforced for the parameterized decision variables, it is equivalent as enforcing it through a DAE dynamic constraint on the intervals where the inequality constraint becomes active, forcing fulfilment of $v(t) = 10\,\mathrm{m/s}$ at collocation points only.

In addition, since $v(t)$ is the only state variable in the ODE equations (11-1), when $v(t)$ becomes a constant with $v(t) = 10\,\mathrm{m/s}$, the ODE dynamic equations effectively becomes pure integrators. If the problem is transcribed using HS collocation, for example, then any solution with:

- cubic state trajectory that can ensure $v(t) = 10\,\mathrm{m/s}$ at the collocation point, and
- corresponding quadratic input trajectory with integral equate to zero for a mesh interval,

will be valid solutions to the DDOP. In other words, the DDOP will have an infinite number of solutions for the intervals with $v(t) = 10\,\mathrm{m/s}$, and all these solution candidates are with the same value for the objective. A graphical illustration of this behaviour is provided in Figure 11-3.

According to the requirements of the original DOP, only one solution would fully respect the constraint $v(t) \leq 10\,\mathrm{m/s}$ everywhere along the trajectory; hence the original DOP is consistent. This means that it is the transcription method of direct collocation that resulted in an under-constrained DDOP for this problem, because the constraints are only enforced at the collocation points.

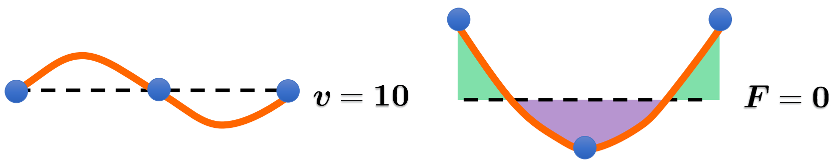


Figure 11-3: Illustration of the fluctuating solution trajectory details for the double integrator repositioning problem. The area of the green region equals to that of the purple region.

In order to address this degree of freedom mismatch, one can perform the DAE index reduction procedure and differentiate the equivalent DAE equation and yield an ODE equation $\dot{v}(t) = 0$. Additionally, there will be the *consistency condition* for $v = 10\,\mathrm{m/s}$ to be applied at any point of the singular arc. When the additional ODE equation and the consistency condition are implemented together on the segment for which the constraint is active, a direct collocation transcription will then yield the correct unique solution to the DOP.

11-3 Implementations to suppress the singular arc fluctuations

11-3-1 Multi-phase direct collocation

With the singular arc conditions (or equivalently the ODE equation after DAE index reduction), the problem can be solved with direct collocation out of the box. However, it is important to ensure that these conditions are only enforced on the singular arc, but not elsewhere, hence a multi-phase implementation would be necessary after determining the structure of the solution trajectories.

With this example problem, the solution structure is 'bang-singular-bang', hence requiring a three-phase formulation with the input variable fixed at the upper bound for the first phase, and the lower bound for the third phase. For the second phase, the extra condition $\dot{v}(t) = 0$ need to be imposed, which effectively forces the input F to be zero everywhere along the singular arc. The numerical solutions are illustrated in Figure 11-4, showing accurate solution for both state trajectory of $v(t)$ and input trajectory of $F(t)$.

11-3-2 Simultaneous regularization

To avoid the use of a multi-phase setup, a common way to deal with with singular control is to augment the original cost functional with an additional *regularization* term for the input variables u, e.g.

$$\min_{x,u,p,t_0,t_f} J(x(t), u(t), t, p) + \rho \|u(t)\|_2^2,$$

Another option is to introduce additional sets of dynamic equation $\dot{u}(t) = \nu(t)$ and regularize the new input $\nu(t)$ with

$$\min_{x,u,p,t_0,t_f} J(x(t), u(t), t, p) + \rho \|\nu(t)\|_2^2.$$

With relatively large values of the penalty weight $\rho \in \mathbb{R}$, the fluctuations on the singular arc can be suppressed, but at the cost of obtaining sub-optimal trajectories. Therefore, it is recommended that ρ is chosen such that

$$\frac{\rho \|u(t)\|^2}{|J(x(t), u(t), t, p)|} \leq \vartheta,$$

or

$$\frac{\rho \|\nu(t)\|^2}{|J(x(t), u(t), t, p)|} \leq \vartheta,$$

can be ensured throughout the solution process, with $\vartheta \in \mathbb{R}$ a small number (e.g. 0.01). In this way, the degradation in solution optimality can be controlled.

However, depending on the values for the constraint bounds, initial guess and the optimal input trajectory, it may be just impossible to identify a single suitable value for ρ. As a result, a sequence of ρ may need to be designed, so that the problem can be repetitively solved using warm-starting.

For the double integrator repositioning problem, the value for objective is about $J = 37$ and the highest magnitude for the square of input, F^2, is 4. Therefore, a choice of $\rho = 0.01$ is sufficient with the ratio ϑ being around 0.001 in the worst-case scenario. At the optimal solution, this ratio is further reduced to around 8×10^{-5}.

The obtained solution is shown in Figure 11-5. It is a sub-optimal solution with the terminal time increases from 37.5004 s to 37.5019 s, a noticeable but yet practically negligible difference. However, with this simultaneous regularization, the undesirable fluctuations can no longer be seen in the input and state trajectories. In addition, the trajectory of $v(t)$ obtained from the actual implementation of the new input trajectory corresponds much better to the DOP solution than the singular ringing case (Figure 11-2a).

11-3-3 Posterior regularization

With simultaneous regularization, it is still difficult to precisely adjust the increase in objective for the obtained solution. For this purpose, another approach based on posterior regularization is possible. The idea is to first solve the original DOP (2-7), and then use the obtained solution and corresponding nominal cost J^* to warm-start the following regularization DOP

$$\min_{x,u,p,t_0,t_f} \|u(t)\|_2^2 \tag{11-4a}$$

subject to

$$\begin{aligned}
J(x(t),u(t),t,p) \leq & J^* + \varepsilon_J,\ \forall t \in [t_0,t_f] \text{ a.e.} && \text{(11-4b)}\\
g(x(t),\dot{x}(t),u(t),t,p) = & 0,\ \forall t \in [t_0,t_f] \text{ a.e.} && \text{(11-4c)}\\
c(x(t),\dot{x}(t),u(t),t,p) \leq & 0,\ \forall t \in [t_0,t_f] \text{ a.e.} && \text{(11-4d)}\\
\phi(x(t_0),t_0,x(t_f),t_f,p) = & 0, && \text{(11-4e)}
\end{aligned}$$

with $\varepsilon_J \in \mathbb{R}$ a small positive number prescribing the acceptable increase in objective value. Theoretically, ε_J can be forced to zero; however, limitations of numerical computations often require a small positive ε_J for (11-4) to be solved successfully in practice.

For the double integrator repositioning problem, a ε_J can be chosen such that $J^* + \varepsilon_J$ will have a similar value of 37.502 as in the demonstration of simultaneous regularization. The solution obtained using posterior regularization is almost identical to that of simultaneous regularization, as shown in Figure 11-6.

11-3-4 Solution post-processing with the area rule

When singular arc fluctuations are caused by pure integrator dynamics, the behaviour illustrated in Figure 11-3 and discussed in Section 11-2-2 will occur. All possible input trajectory solutions to the under-constrained DDOP share a common characteristic: the integral values are the same inside each mesh interval for different non-unique solutions. Therefore, it is possible to simply integrate this approximated input trajectory, and construct a piece-wise constant profile that has the same integrated value. With this method, one effectively computes the area under a curve and then compute the height of a rectangle that has the same area and width; therefore, this post-processing method is named the *area rule*.

A point of caution for the area rule is that the modification should only be applied on the singular arc but nowhere else. Figure 11-7 shows the post-processing of the input trajectory of the double integrator repositioning problem using the area rule, and the simulation results of such a modified input trajectory.

It can be seen that the obtained velocity trajectory $v(t)$ from implementation contains larger deviations than the regularization formulations. This is because the regularization formulations also regularise the input at the point of discontinuity, leading to a solution with smaller errors at the very beginning of the singular arc intervals. Nevertheless, the modified input profile using the area rule does lead to the correct behaviour of constant velocity on the singular arc, and the implementation errors are noticeably smaller than that of the original singular problem (Figure 11-2).

11-3-5 Solving with ADIR transcription

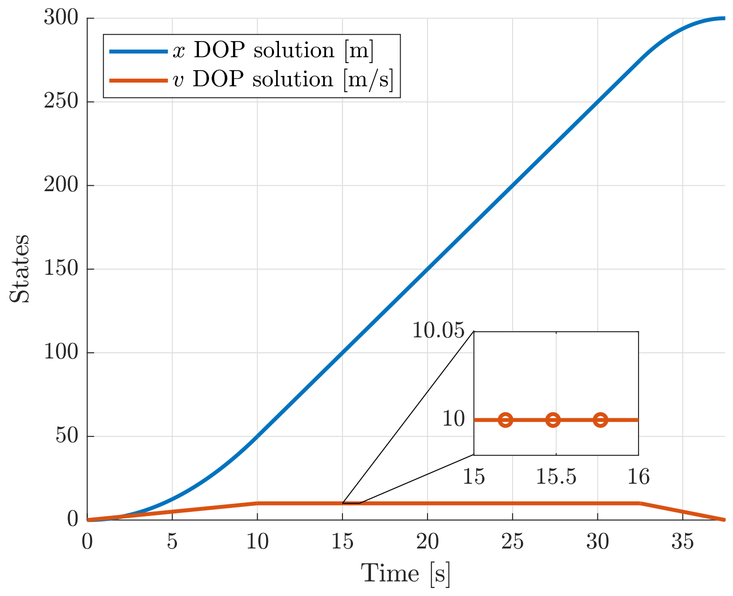
The mechanism for the IRM-type of methods to suppress singular arc fluctuations are discussed in Section 7-1-5, hence will not be repeated. The DAIR solution to the double integrator repositioning problem is presented in Figure 11-8. Since for this problem, the system effectively becomes pure integrator dynamics on the singular arc. Therefore it belongs to the special case where the approximating functions are capable of representing the solution trajectory exactly; hence IRM-type of methods are theoretically unable to suppress the singular arc directly. Nevertheless, thanks to the reasons discussed in Section 7-1-5, the DAIR transcription method is still rather successful in suppressing the ringings on the singular arc, despite some fluctuations present at the location of discontinuities.

11-4 Comparison of different methods

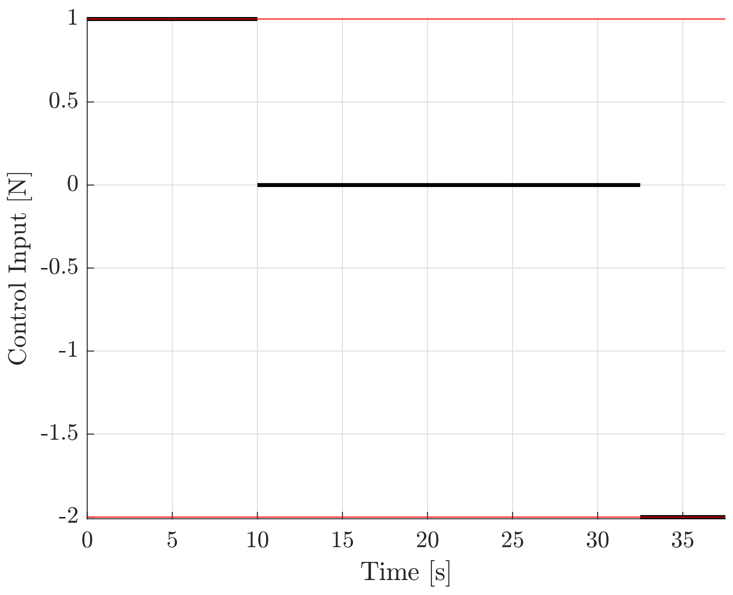
With the help from this example problem and based on past experiences, the advantages (+) and disadvantages (−) of different methods for the suppression of singular arcs can be summarized as follows:

- Multi-phase direct collocation
  - + solutions of high accuracy, generally corresponds well to the analytical solution,
  - − need to know the structure of the solution beforehand, i.e. the multi-phase setup may need to change accordingly when problem formulation slightly changes,
  - − need to derive analytically and implement the singular arc conditions for every occurrence of singular control.
- Simultaneous regularization
  - + only need to solve a single DOP once, and without any post-processing of the solution trajectory,
  - − difficult to manage the eventual trade-off between the original objective and the regularization terms, especially in large problems with many input variables,
  - − difficult to ensure the ratio between the regularization terms and the original objective is always small throughout the intermediate iterations of the solution process. If this ratio not respected, it is possible that the solver may converge to a significantly sub-optimal solution.

- Posterior regularization
  - + solution of high accuracy,
  - + designer have full control for the sub-optimality of the solution,
  - − need to setup and solve one additional DOP,
  - − due to limitations of numerical solvers in constraint handling, convergence may be affected if a small ε_J is used.
- Post-processing with area rule
  - + simple modification made to direct collocation solution,
  - + low computation cost,
  - − only suitable for singular arc fluctuations corresponding to pure integrator dynamics,
  - − unless solution is of ‘bang-singular’ type, one needs to identify the intervals of singular control and make sure the area rule is only applied on these intervals.
- Using IRM type of transcriptions
  - + only need to solve a single DOP once, and without any post-processing of the solution trajectory,
  - + solution of high accuracy,
  - − when trajectories are represented accurately by the approximating functions (e.g in case of pure integrator dynamics), suppression of singular arc fluctuations may not be achieved from a theoretical point of view.

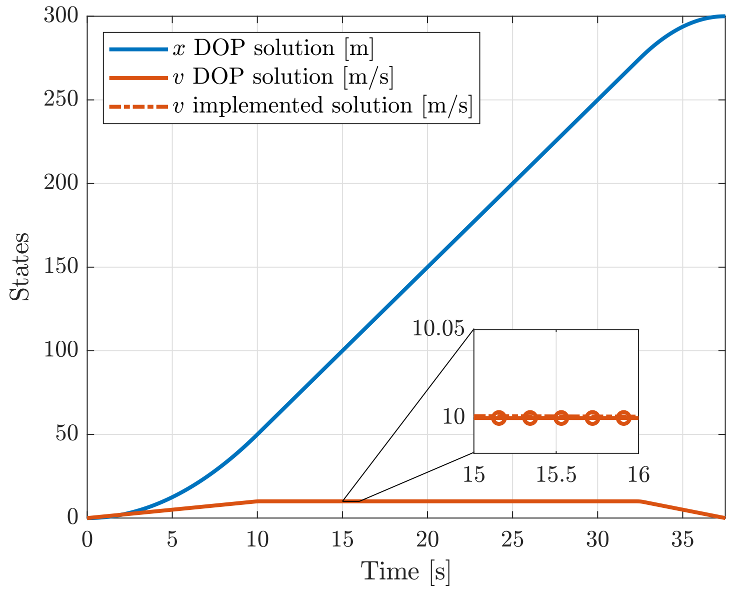


(a) States

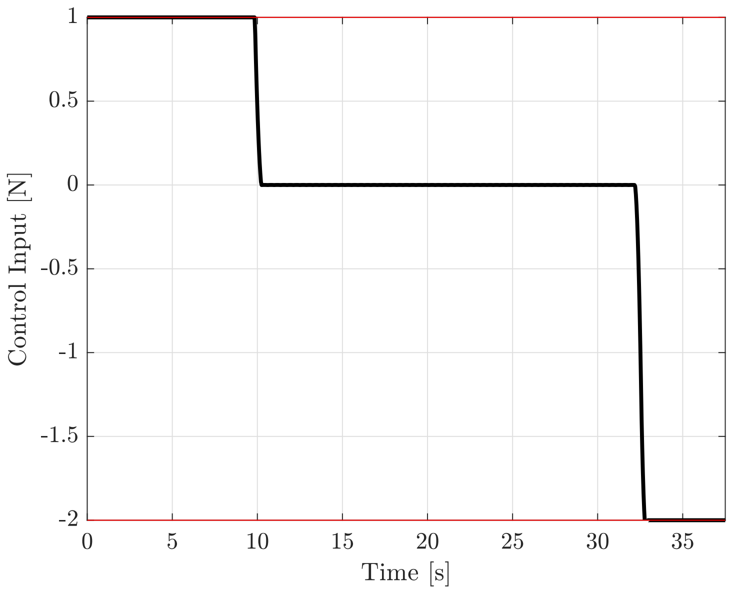


(b) Input

Figure 11-4: Time-optimal solution for double integrator repositioning problem (multi-phase direct collocation with additional singular arc conditions, HS discretization, 30 major nodes for phase 1 and 3, 40 major nodes for phase 2, circles represent collocation points)

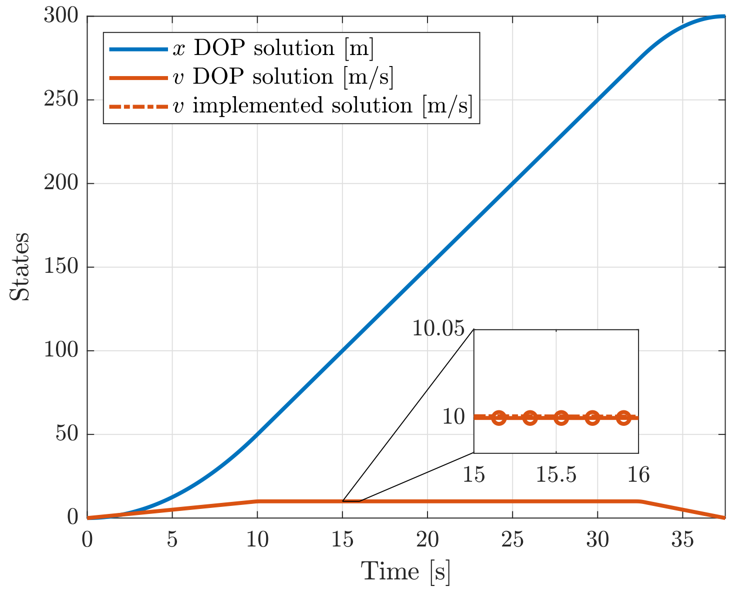


(a) States

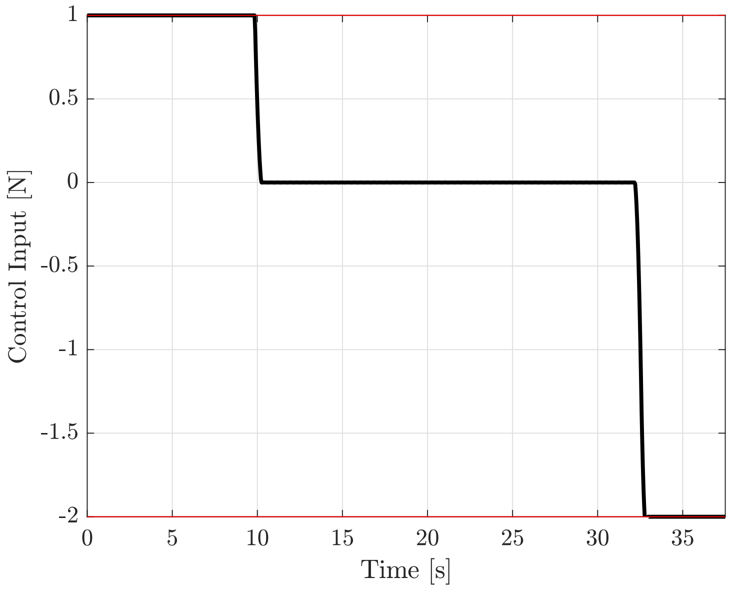


(b) Input

Figure 11-5: Time-optimal solution for double integrator repositioning problem (single phase direct collocation with simultaneous regularization, HS discretization, 100 major nodes, circles represent collocation points)

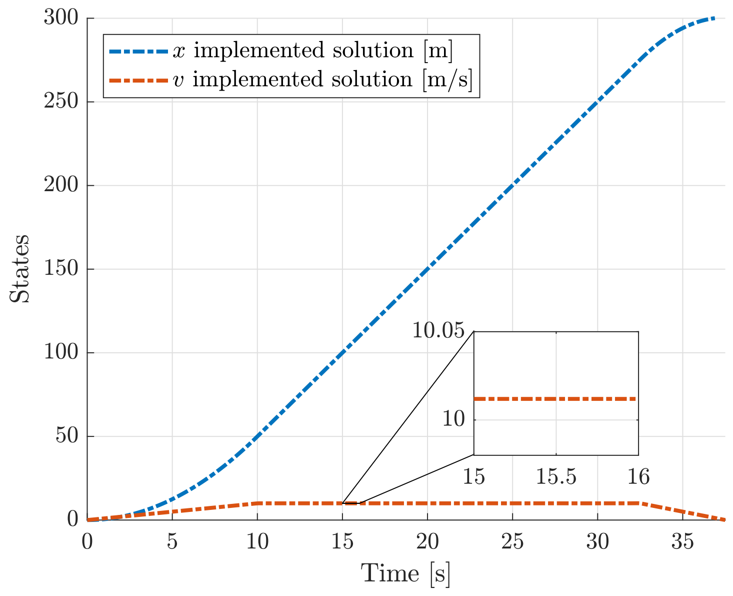


(a) States

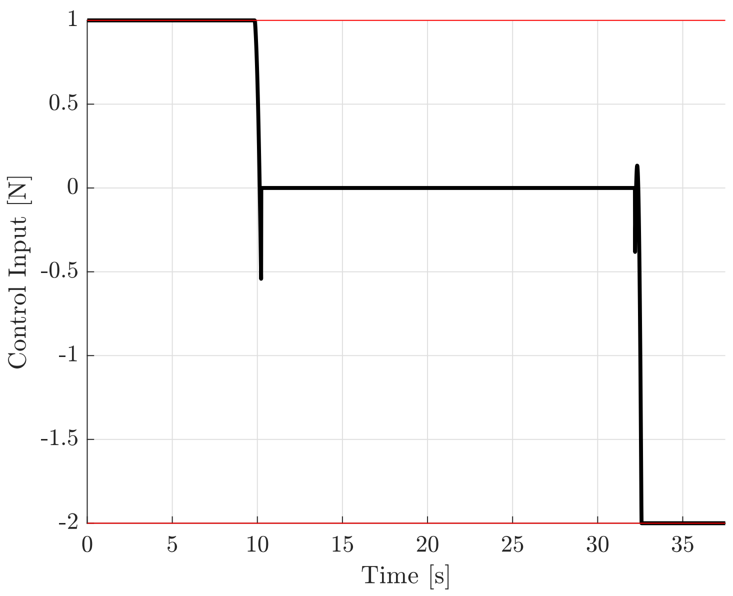


(b) Input

Figure 11-6: Time-optimal solution for double integrator repositioning problem (single phase direct collocation with posterior regularization, HS discretization, 100 major nodes, circles represent collocation points)

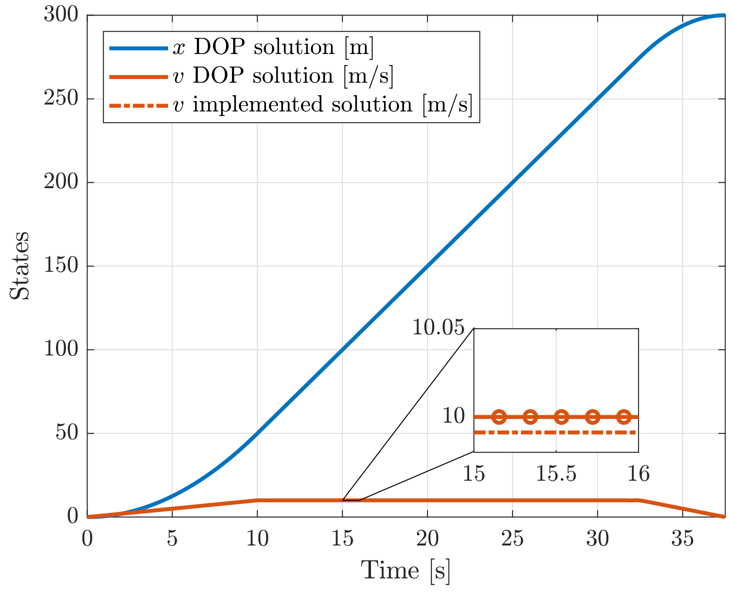


(a) States

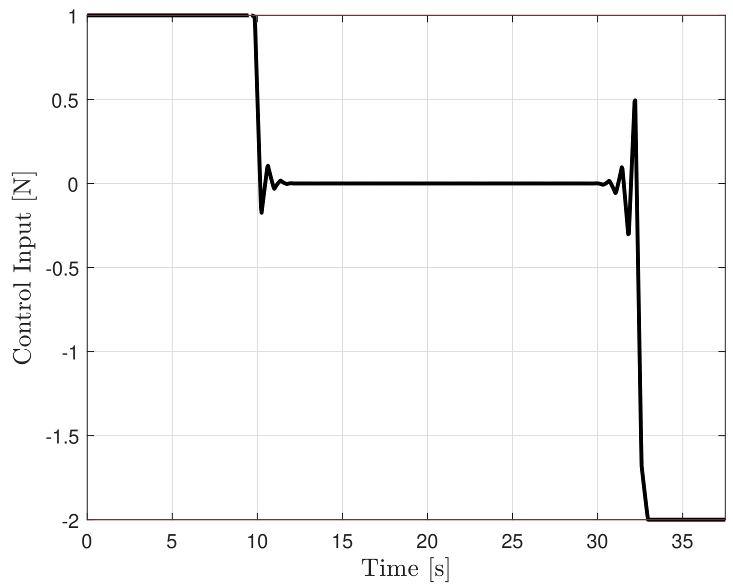


(b) Input

Figure 11-7: Time-optimal solution for double integrator repositioning problem (single phase direct collocation with modification of singular input trajectory using area rule, HS discretization, 100 major nodes)



(a) States



(b) Input

Figure 11-8: Time-optimal solution for double integrator repositioning problem (single phase DAIR, piece-wise cubic state parameterization and piece-wise quadratic input parameterization, 100 major nodes, circles represent collocation points)

Chapter 12

ICLOCS2: A Comprehensive MATLAB Toolbox for Fast Prototyping of Optimization Based Control

ICLOCS2 is the new version of ICLOCS (pronounced 'eye-clocks') and is a comprehensive software suite for solving DOPs in MATLAB and Simulink. The toolbox builds on a wide selection of numerical transcription and discretization methods, and automated tools to assist the design and implementation of DOPs. The aim is to provide the first port of call for solving DOPs of various challenging natures. The suite of solution techniques implemented in ICLOCS2 significantly increases the chance of successfully solving difficult problems to a high accuracy. On the other hand, unlike most existing DOP software that is targeting on obtaining the most accurate optimal trajectory offline, ICLOCS2 is designed with online implementation prototyping in-mind. It offers the user great flexibility in trading off the solution quality with the computational complexity. The result is a comprehensive toolbox that is capable of efficiently solving and implementing a wide variety of challenging DOPs. Consequently, users may implement the DOPs in a rather intuitive manner, making nonlinear optimal control available to a broader range of application fields.

In this chapter, the development motivation of ICLOCS2 is first presented in Section 12-1, followed by some selected software highlights in Sections 12-2 regarding multi-phase problems, mesh refinement, dynamically refined *hp*-adaptive mesh, and closed-loop simulations.

Table 12-1: ICLOCS2's solution to a list of singular control problems

	Singular Arc Ringing Mitigated	
	with On-mesh Rate Constraint Implementation	**with DAIR Transcription**
Van der Pol Oscillator [141]	No	Yes
Aly-Chan Problem [5]	No	Yes
Second Order Singular Regulator [4]	Yes	Yes
Goddard Rocket [140]	No	Yes
Aircraft Go-around in Windshear [146]	Yes	Yes

12-1 Development motivation and overview of ICLOCS2

ICLOCS2 bridges the gap between classical offline trajectory optimization and online optimization based control methods, such as NMPC. Instead of solely pursuing the most accurate open-loop result (as is done for most software in Table 2-1), ICLOCS2 has been designed to enable flexible trading-off of the solution accuracy and computational complexity. By implementing the closed-loop optimization-based control, as shown in Figure 2-2, ICLOCS2 users may also utilize the merits of feedback to resolve problems arising from external disturbances, model mismatches, as well as DOP solution errors related to discretization and round-offs. Hence, ICLOCS2 is well suited for prototyping of online implementation in the MATLAB and Simulink environments.

To enable the required level of flexibility, ICLOCS2 implements state-of-the-art transcription and discretization methods, efficient solvers and derivative information calculations. By providing access to an array of methods for solving a wider variety of optimal control problems, ICLOCS2 reduces the experience required by the software from the user. For example, when dealing of one of the most difficult classes of problems in optimal control: singular control problems, ICLOCS2 has much a higher chance to yield a solution of sufficiently good accuracy without needing an analytical expression for singular arc conditions from the user, as shown in Table 12-1. Another example is when the formulated problem belongs to simpler classes of optimization problems such as QPs and LPs. ICLOCS2 supports dedicated numerical solvers that are specialized in handling these problems efficiently, thus capable of yielding solutions faster and more reliably than generic NLP solvers.

ICLOCS2 also provides various tools which improve the algorithmic efficiency and ease of implementation, with a summary of the available methods and features shown in Appendix B.

12-2 Additional software highlights

12-2-1 Multi-phase optimal control for continuous and hybrid systems

Implementing and solving a single-phase problem can be challenging if the problem formulation contains distinctive changes at certain time instances. In these situations, one may use the multi-phase DOP implementation in ICLOCS2 to prescribe the problem before and after these discrete changes. All phases are solved together with the inclusion of linkage constraints to enforce continuity or certain jump conditions.

There are several different cases where this implementation can be useful:

- For problems with continuous state trajectories, multi-phase implementation can be useful when system dynamics undergo distinctive changes. In this regard, an epidemic control problem is shown, where the spreading of virus is different at different stages of the epidemic control phases.
- Another commonly encountered case is the optimal control of hybrid systems, where certain state trajectories are discontinuous and contain jumps. The reusable rocket launch example demonstrates this situation where the mass of the rocket during launch sees a discrete drop when the first stage is separated.
- Formulating the problem as multi-phase can be beneficial if the structure of the solution is known beforehand. As a result, one may not need to worry about capturing the discontinuity in the *bang-bang* control example in Section 12-2-3, and the *bang-singular-bang* control example in Section 11-1 can have the singular control mitigated by imposing additional singular arc conditions specifically for the second phase.
- Multi-phase formulation can also be useful in closed-loop implementations in the form of MPC, when distinctively different mission requirements are specified. For example, in a mission requiring a pendulum to move to the inverted position in minimum time, and then maintain that position with given error bounds using minimum energy.

Example: Epidemic control with the SEIR model

The COVID-19 epidemic has raised question on how best to control the spread of such diseases in an effective and efficient manner. Extending the classic SIR model, the SEIR model [41] is claimed to have higher fidelity for the modeling of the spreading of the virus among the susceptible population. The model can be expressed as

$$\dot{S}(t) = -\alpha S(t) - \beta \frac{S(t)I(t)}{N_{pop}}, \qquad \dot{E}(t) = -\gamma E(t) + \beta \frac{S(t)I(t)}{N_{pop}},$$

$$\dot{I}(t) = \gamma E(t) - \delta I(t), \qquad \dot{Q}(t) = \delta I(t) - \lambda(t)Q(t) - \kappa(t)Q(t),$$
$$\dot{R}(t) = \lambda(t)Q(t), \quad \dot{D}(t) = \kappa(t)Q(t), \quad \dot{P}(t) = \alpha(t)S(t),$$

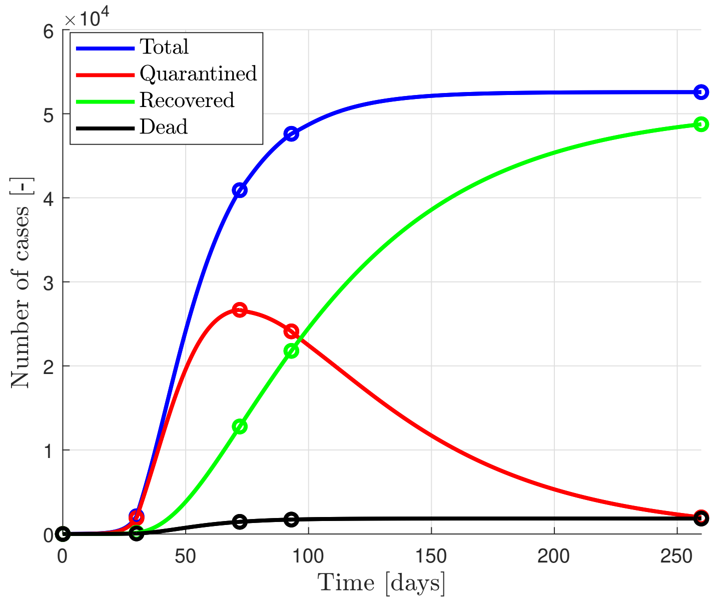
with $S(t)$ are the susceptible cases, $E(t)$ the exposed cases, $I(t)$ the infectious cases, $Q(t)$ the quarantined cases, $R(t)$ the recovered cases, $D(t)$ the dead cases and $P(t)$ the insusceptible cases. Regarding the parameters, N_{pop} is the population, α is the protection rate, β is the infection rate, γ is the inverse of the average latent time, δ is the rate at which infectious people enter in quarantine, $\lambda(t) = \lambda_0(1 - \exp(-\lambda_1 t)$ is the time-dependant cure rate, and $\kappa(t) = \kappa_0 \exp(-\kappa_1 t)$ is the time-dependant mortality rate. It is assumed that 30 % of quarantined patients will require hospital treatment.

The problem is formulated and solved as a four-phase DOP in ICLOCS2, each with different modelling parameters for the dynamics equations. The first phase is characterized as uncontrolled growth, with few cases but no measures to limit the virus spread. In the second stage, drastic measures are taken, such as social distancing and significant reductions in social activities, aiming to slow down the spread. At this stage, mass testing will also allow quick identification of infected cases; thus, the rate at which infectious people enter in quarantine is also high. This is followed by a relaxing phase where economic activities are allowed to gradually restore for a certain period until a point where pre-pandemic policy can be reinstated in phase four. A terminal constraint is in place to guarantee that at the end of phase four, there will be no more exposed and infectious cases (i.e. all remaining virus carrier are quarantined), marking the end of the epidemic. The objective of this example problem is to maximize the necessary social activity during the containment phase while ensuring the hospital bed usage are within the availability limit, commonly known as 'flattening the curve'. Figure 12-1 presents a numerical solution.

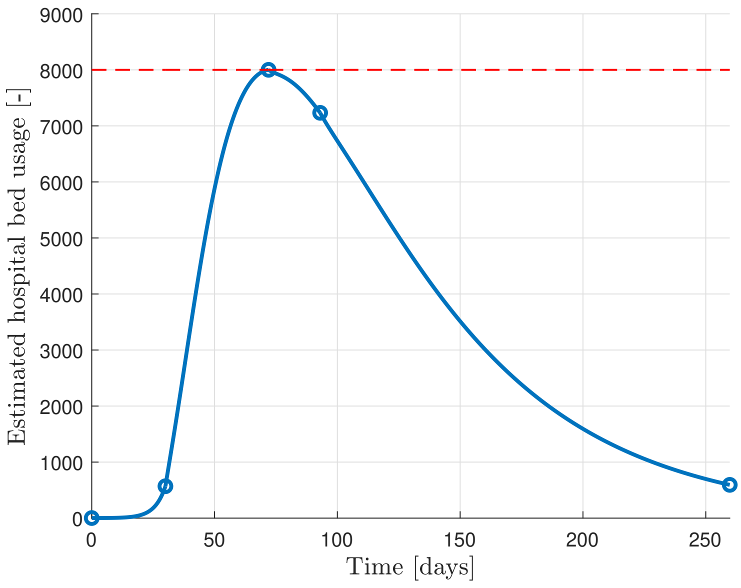
Example: Reusable rocket launch and recovery

The successful flight of SpaceX Falcon series of rocket launchers mark a new era of commercial space flight with partially or fully reusable launcher components. Ma et al. [133] presented a case study where trajectory optimization is used to generate flight trajectories for the launch and recovery of a two-stage rocket, with the first stage returning for a vertical landing at the original launch site. The objective is to maximize the remaining fuel in the second stage and the payload when they successfully enter into orbit.

Here the solution to this case study is demonstrated using ICLOCS2, but using the corrected system dynamics presented in Betts [17, Ex. 6.19], leading to different solution trajectories. In the corrected formulation, a distinction is made between the velocity in the Earth-centered inertial frame (as v in dynamic equations) and the earth relative velocity (as v_r in aerodynamic force computations). The relationship between the two is $v_r = v - \omega \times r$, with $\omega = (0, 0, \omega_E)^T$



(a) Number of cases



(b) Hospital bed usage

Figure 12-1: Solution to the multi-phase epidemic control example (nodes represent phase boundaries)

the angular velocity of the earth relative to inertial space, and r the position vector.

In the original work [133], the authors reported that conventional initialization methods would fail and designed an internal–external-growth strategy. In this strategy, several subproblems of the original trajectory optimization are first solved to obtain a good initial guess. We are happy to report that for certain discretization schemes (e.g. trapezoidal and Hermite-Simpson) with customized scaling and mesh refinement, a rudimentary two-point initial guess in ICLOCS2 is sufficient to successfully solve the problem.

Figure 12-2 presents the solution to the problem using a three-phase problem formulation. At the first phase interface at the end of the initial launch, continuity constraints are enforced for the position and velocity states. A jump condition is implemented for the mass of the first stage, and a fixed initial condition is used for the mass of the second stage and payload. After this time instance, the second stage continues to propel the payload towards its target orbit, and the first stage starts its return journey back to the launch site.

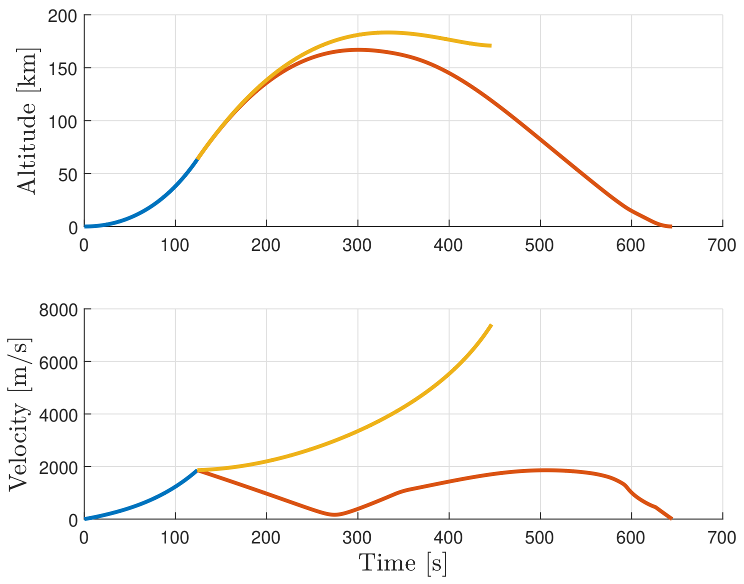
12-2-2 Mesh refinement to ensure solution accuracy and constraint compliance

Mesh refinement is essential in ensuring the accuracy of the solution and satisfaction of constraints, particularly for methods such as direct collocation that do not have error measures for the whole trajectory during the solution process. Here, the mesh refinement procedures designed for direct collocation methods in ICLOCS2 are briefly introduced.

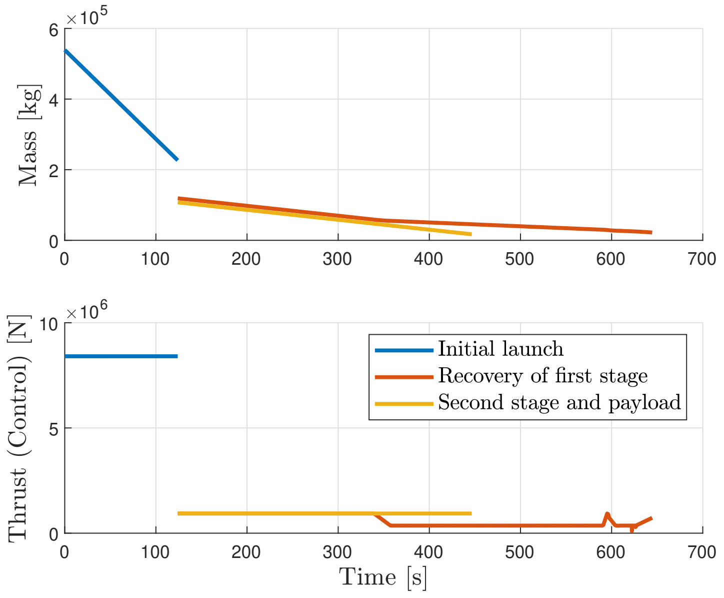
Existing DOP software with mesh refinement capabilities generally only updates the mesh based on absolute/relative local errors, but not constraint violation errors. To certain extent, the latter is, in fact, more important as it directly affects the feasibility of the solution; thus, a higher priority is given in ICLOCS2 for the reduction of the constraint violation errors.

Common approaches for mesh refinement include adding intervals and changing the polynomial degree. To date, most refinement policies in the literature employ certain form of heuristics, configured to efficiently handle a wide range of problems. This is because although a high solution accuracy is eventually required, one may also want to minimize both the accumulated computational cost of the refinement process, as well as the stand-alone computational cost associated with the final mesh. These two criteria are often seen as conflicting objectives, consequently imposing the following design requirements for a good mesh-refinement algorithm.

- The initial mesh should be coarse and discretized with a low order transcription method. If the problem is initialized with a high order dense grid instead, there will be an unnecessarily high number of grid points in some regions, adding unnecessary computations. This detrimental effect will be amplified if the scheme does not allow merging of mesh



(a) Altitude and velocity



(b) Mass and thrust

Figure 12-2: Solution to the reusable rocket launch and recovery problem

```
while not all user defined error tolerance satisfied do
    Transcribe and solve the corresponding DOP with current mesh;
    Obtain the solution ($\chi$, $\upsilon$, $p$, $t_0$, $t_f$);
    Obtain the continuous solution ($\tilde{x}(t)$, $\tilde{u}(t)$) with user's choice of representation method;
    for all states and input variables do
        Estimate the absolute local error error based on (4-6);
        Identify $M$ intervals with maximum errors not within tolerance, with $M \leq M_{max}$;
    end
    Subdivide the identified interval;
    for all inequality constraints do
        Estimate the constraint violation error based on (4-9);
        Identify peak points for constraint violations not within tolerance;
    end
    Add mesh point at identified peak locations;
    Update current mesh;
end
```

Algorithm 1: Mesh Refinement Algorithm for *h* methods in ICLOCS2

intervals and polynomial degree reduction, consequently carrying the dense initial mesh into new mesh design iterations, making both the accumulated and per-iteration computation cost very high.

- Since error estimation based on (4-5) and (4-9) are also mesh dependent, they can be very crude early in the mesh refinement process. Based on these error measures, refinement strategies that are overly aggressive may lead to high computational cost due to excessively-refined mesh. Therefore the number of refinements allowed in each iteration should be restricted, e.g. by configuring the maximum number of grid points allowed to be added (M_{max}).

Most DOP software calculates η as in (4-7), and use it as the error measure for the mesh-refinement procedure. However, in practical problems, not all dynamic equations must be resolved to the same level of accuracy. Therefore in ICLOCS2, the user can specify $\zeta_{tol} \in \mathbb{R}^{n+n_g}$ and $\epsilon_{tol} \in \mathbb{R}^{n_c}$, containing the required error threshold for each state variable and constraint. Then ICLOCS2 uses absolute local error ζ and absolute local constraint violation error ϵ as part of the termination criteria for mesh refinement iterations.

With the above-mentioned considerations, and based on ideas in Betts [17], Liu et al. [128] and Lei et al. [123], the mesh refinement procedures are designed for *h* and *hp* methods respectively, and presented in Algorithm 1 and 2.

Example: automatic car parking in minimum time

With the introduction of autonomous road vehicles, the systematic treatment of autonomous parking is a hot topic. Optimization-based control provides a framework where the vehicle

```
while not all user defined error tolerance satisfied do
    Transcribe and solve the corresponding DOP with current mesh;
    Obtain the solution ($\chi$, $\upsilon$, $p$, $t_0$, $t_f$);
    Obtain the continuous solution ($\tilde{x}(t)$, $\tilde{u}(t)$) with user's choice of representation method;
    for all states and input variables do
        Estimate the absolute local error based on (4-6);
        Identify $M$ peaks with maximum errors not within tolerance, with $M \leq M_{max}$;
        For other intervals (not containing the identified peaks), compare the error $\zeta^{(k)}$ with
          tolerance $\zeta_{tol}$.
    end
    Subdivide the intervals containing the peak;
    for other identified intervals do
        if tolerance not satisfied then
            if polynomial degree within maximum allowed degree then
                Increase the polynomial degree at most by $ceil(\log_{10}(\frac{\zeta^{(k)}}{\zeta_{tol}}))$ ;
            else
                Subdivide the interval;
            end
        else
            if polynomial degree within minimum allowed degree then
                Decrease the polynomial degree at most by $ceil(\log_{10}\sqrt{\frac{\zeta^{(k)}}{\zeta_{tol}}})$ ;
            end
        end
    end
    for all inequality constraints do
        Estimate the constraint violation error based on (4-9);
        Identify peak points for constraint violations not within tolerance;
    end
    Add mesh point at identified peak locations;
    Update current mesh;
end
```

Algorithm 2: Mesh Refinement Algorithm for *hp* methods in ICLOCS2

dynamics, physical constraints, collision-avoidance restrictions and the optimization objective can be treated altogether. In this example, the latest work by Li et al. [125] is adapted, with a few modifications to improve convergence, to demonstrate the benefit of mesh refinement w.r.t. constraint violation errors.

The rigid body dynamics of a front-wheel steering car can be described by the following kinematic equations,

$$\begin{aligned} \dot{POS}_x(t) =& v(t)\cos(\theta(t)), \\ \dot{POS}_y(t) =& v(t)\sin(\theta(t)), \\ \dot{v}(t) =& a(t), \\ \dot{a}(t) =& u_1(t), \end{aligned}$$

$$\begin{aligned}\dot{\theta}(t) =& \frac{v(t)\tan(\phi(t))}{l_a},\\ \dot{\phi}(t) =& u_2(t),\end{aligned}$$

with (POS_x, POS_y) the mid-point of the rear wheel axis of the car, v the velocity, θ the orientation angle, a the acceleration and ϕ the steering angle of the front wheels. Two control inputs, u_1 and u_2, represents the jerk and the steering rate respectively. Regarding the dimension of the car, l_a =2.5 m is the wheelbase, l_f =0.8 m the front overhang length, l_r =0.7 m the rear overhang length, and $2b$ =1.771 m the vehicle width (see Figure 12-3b).

The system is subject to simple bounds

$$\begin{aligned}-0.75 \leq a \leq 0.75\,\mathrm{m/s}^2, \qquad & -2 \leq v \leq 2\,\mathrm{m/s},\\ -33 \leq \phi \leq 33\,\mathrm{deg}, \qquad & -0.5 \leq u_1 \leq 0.5\,\mathrm{m/s}^3,\end{aligned}$$

and a path constraint on the curvature derivative

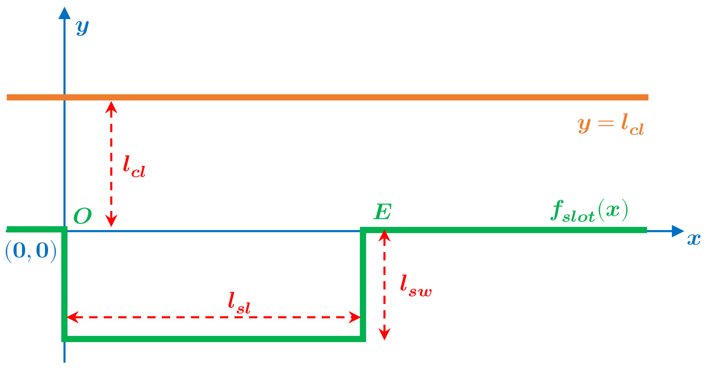
$$-0.6 \leq \frac{u_2(t)}{l_a \cos^2(\phi(t))} \leq 0.6.$$

The geometry of the parking slot is shown in Figure 12-3a, with l_{sl} =5 m its length, l_{sw} =2 m its width, and l_{cl} =3.5 m the width of the road. Condition for the vehicle not to collide with the barriers can be approximated by the requirements for the four corners of the car to fulfil the following inequality constraints

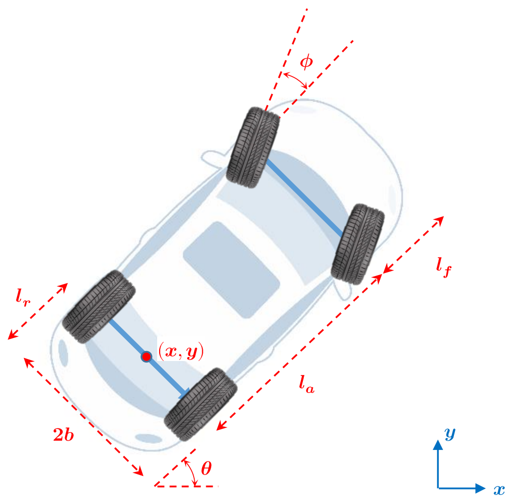
$$\begin{cases} A_y(t) \leq l_{cl} \\ B_y(t) \leq l_{cl} \\ C_y(t) \leq l_{cl} \\ D_y(t) \leq l_{cl} \end{cases} \qquad \begin{cases} A_y \leq f_{slot}(A_x) \\ B_y \leq f_{slot}(B_x) \\ C_y \leq f_{slot}(C_x) \\ D_y \leq f_{slot}(D_x) \end{cases}$$

with

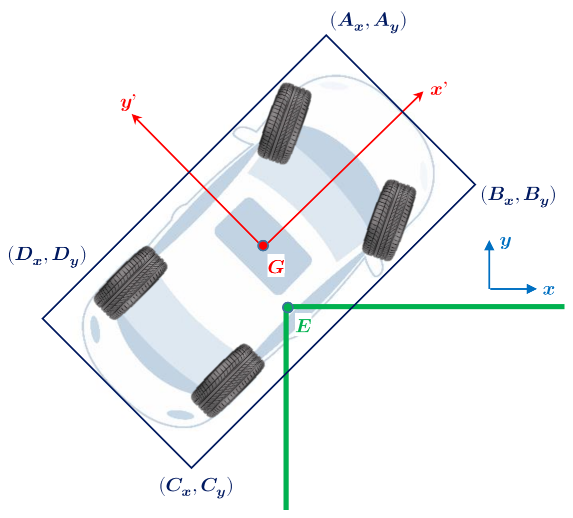
$$\begin{aligned}(A_x, A_y) = (&POS_x(t) + (l_a + l_f)\cos(\theta(t)) - b\sin(\theta(t)),\\ &POS_y(t) + (l_a + l_f)\sin(\theta(t)) + b\cos(\theta(t))),\\ (B_x, B_y) = (&POS_x(t) + (l_a + l_f)\cos(\theta(t)) + b\sin(\theta(t)),\\ &POS_y(t) + (l_a + l_f)\sin(\theta(t)) - b\cos(\theta(t))),\\ (C_x, C_y) = (&POS_x(t) - l_r\cos(\theta(t)) + b\sin(\theta(t)),\\ &POS_y(t) - l_r\sin(\theta(t)) - b\cos(\theta(t))),\\ (D_x, D_y) = (&POS_x(t) - l_r\cos(\theta(t)) - b\sin(\theta(t)),\\ &POS_y(t) - l_r\sin(\theta(t)) + b\cos(\theta(t))),\end{aligned}$$



(a) Parking Slot



(b) Vehicle Geometry



(c) Vehicle Body Coordinate

Figure 12-3: Graphical illustration of the automatic car parking in minimum time problem

$$f_{slot}(POS_x) = (-H(POS_x(t)) + H(POS_x(t) - l_{SL}))l_{SW},$$

and $H(\ \cdot\ )$ the Heaviside step function. In this example, to avoid the discontinuities introduced, *tanh* functions are implemented to approximate the parking slot geometry.

With all the formulated constraints satisfied, there is still one exception that the vehicle can hit the barrier: at two corners (point O and E, illustrate in Figure 12-3c). In order to specify additional constraints to avoid this, the expression of the corner points O and E in vehicle's body frame ($x'Gy'$) are derived as:

$$\begin{aligned}(O'_x, O'_y) =& (-POS_x(t)\cos(\theta(t)) - POS_y(t)\sin(\theta(t)) - \frac{l_a + l_f - l_r}{2}, \\ & POS_x(t)\sin(\theta(t)) - POS_y(t)\cos(\theta(t))),\end{aligned}$$

$$\begin{aligned}(E'_x, E'_y) =&(-POS_x(t)\cos(\theta(t)) - POS_y(t)\sin(\theta(t)) - \frac{l_a + l_f - l_r}{2} + l_{SL}\cos(\theta(t)),\\ &POS_x(t)\sin(\theta(t)) - POS_y(t)\cos(\theta(t)) - l_{SL}\sin(\theta(t))).\end{aligned}$$

Intuitively, as derived by Li et al. [125], the path constraints arising from this formulation are

$$\begin{aligned}|O'_x| &\geq \frac{l_f + l_a + l_r}{2}, \text{ when } |O'_y| \leq b,\\ |E'_x| &\geq \frac{l_f + l_a + l_r}{2}, \text{ when } |E'_y| \leq b.\end{aligned}$$

Unfortunately, this conditional constraint formulation will introduce convergence challenges for the optimization solver. To have a continuous function expression, the sum of the area of the triangles formed by the corner point and four corners of the car are computed. Requiring this area to be larger than the rectangular planform area of the car will ensure the constraints are satisfied. Thus,

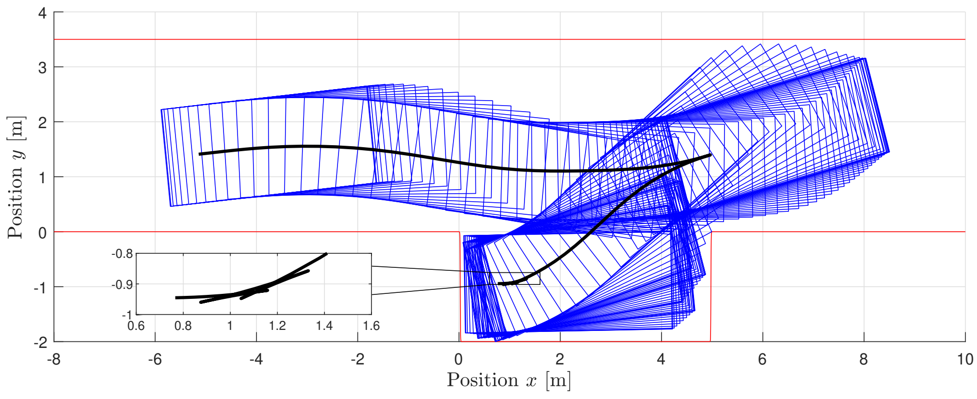
$$\begin{aligned}Area(AEB) + Area(BEC) + Area(CED) + Area(DEA) \geq Area(ABCD),\\ Area(AOB) + Area(BOC) + Area(COD) + Area(DOA) \geq Area(ABCD),\end{aligned}$$

are implemented instead to ensure that collisions not to occur. For this example problem, the initial and terminating conditions are specified as

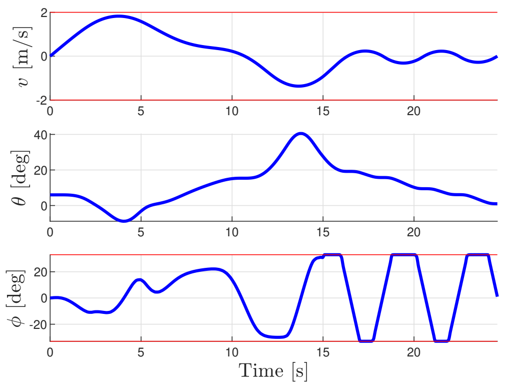
$$\begin{aligned}POS_x(0) = l_{sl} + l_r, \quad y(0) = 1.5\,\text{m}, \quad v(0) = 0\,\text{m/s}, \quad a(0) = 0\,\text{m/s}^2,\\ \theta(0) = 0\,\text{deg}, \quad \phi(0) = 0\,\text{deg}, \quad v(t_f) = 0\,\text{m/s}, \quad a(t_f) = 0\,\text{m/s},^2\\ A_y(t_f) \leq 0\,\text{m}, \quad B_y(t_f) \leq 0\,\text{m}, \quad C_y(t_f) \leq 0\,\text{m}, \quad D_y(t_f) \leq 0\,\text{m}.\end{aligned}$$

Figure 12-4 presents the solution obtained by ICLOCS2. Generally speaking, it was because a relatively dense mesh is chosen and by chance the location of mesh point is ideal, that the path of the vehicle did not lead to a collision near corner E. Figure 12-5a illustrates another potential outcome with a different configuration to the problem. It can be seen that, despite satisfying all discretized constraint equations at discretization points, the vehicle trajectory near the right corner of the parking slot is obviously not feasible due to collision.

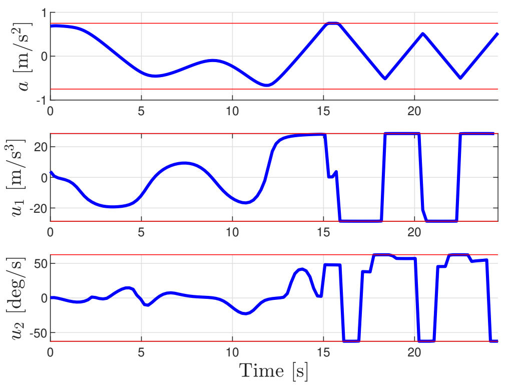
A possible way to reduce or eliminate constraint violation is via constraint tightening techniques (e.g. [67]); however, they could potentially lead to over-conservative solutions. By including the constraint violation errors in the mesh refinement procedures in ICLOCS2, additional mesh points can be added until the represented continuous trajectory satisfies all user-defined constraint violation tolerances. Figure 12-5b demonstrates that mesh refinement schemes in ICLOCS2 ensure the vehicle to successfully pass the corner.



(a) Position

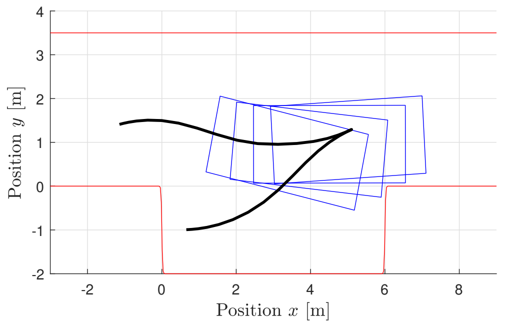


(b) v,θ and ϕ

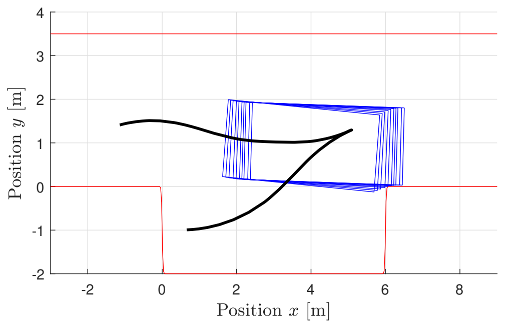


(c) a, u_1 and u_2

Figure 12-4: Solution to the automatic car parking in minimum time problem



(a) A different configuration on a coarse mesh



(b) Refinement based on constraint violation error

Figure 12-5: Illustration of possible constraint violations and effect of mesh refinement based on constraint violation errors

12-2-3 Dynamically refined hp-adaptive mesh

ICLOCS2 provides the user with an experimental implementation of dynamically refined *hp*-adaptive mesh, for which the initial and final time (t_k, t_{k+1}) of each mesh segment are also included as decision variables in the optimization problem.

This implementation followed from the observation that, in many practical problems, time-optimal solutions exhibit a discontinuity nature, often in the form of a *bang-bang* input. Therefore, the optimality is directly influenced by the approximation quality of the switching mechanism. Hence, the objective to minimize the cost and the preference to have a finer mesh at the discontinuity are aligned. As a result, the placement (location and length) of mesh intervals can be automatically adjusted during the optimization process. Example 11-1 demonstrates the benefits of the dynamically refined *hp*-adaptive mesh.

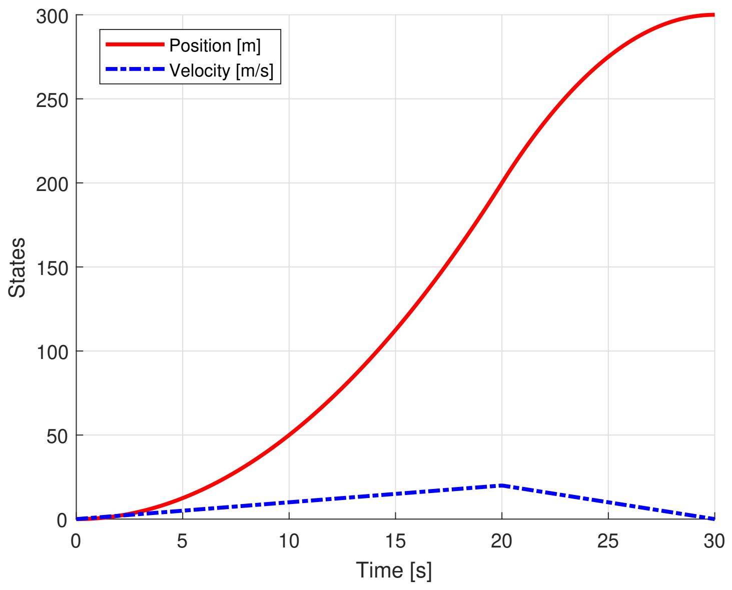
One point of caution is the influence of discretization errors. If the discretization error is allowed to be very large, the solver may be misled to a physically infeasible solution that yields a lower objective value. In the current version of ICLOCS2 with direct collocation, the user needs to configure additional constraints regarding the minimum and maximum size of each interval to ensure a good solution quality. Later, with IRM-type of direct transcription methods which can have the measure of the errors embedded in the solution process, the influence of errors can then be handled automatically for the dynamically refined *hp*-adaptive mesh.

Example: Double Integrator Minimum-time Repositioning

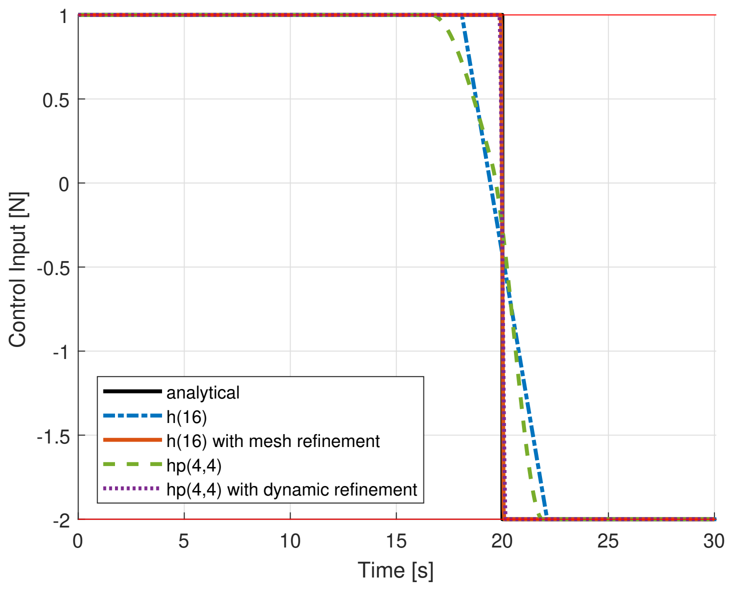
To demonstrate the benefits of the dynamically refined *hp* mesh, the same double-integrator minimum-time control problem as in Section 11-1 is used, but now with the box constraint on the velocity removed.

The solution to the problem calculated by ICLOCS2 are collected in Figure 12-6, with control switching at 20 s. It can be seen that initially with a coarse mesh, both the h (with 8 mesh nodes and 7 intervals) and hp (with 4 intervals, each polynomial degree of 4) method yield results with obvious discrepancies.

By adding additional mesh points at locations of errors, mesh refinement successfully bring the h method solution very close to the analytical result. The dynamically refined *hp*-adaptive mesh has a similar effect: enhancing the capturing of discontinuities in the optimal solution, however without the need to change the dimension of the underlying NLP problem and iteratively solve with updated mesh designs.



(a) States



(b) Input

Figure 12-6: Time-optimal solution for double integrator repositioning (bang-bang) problem

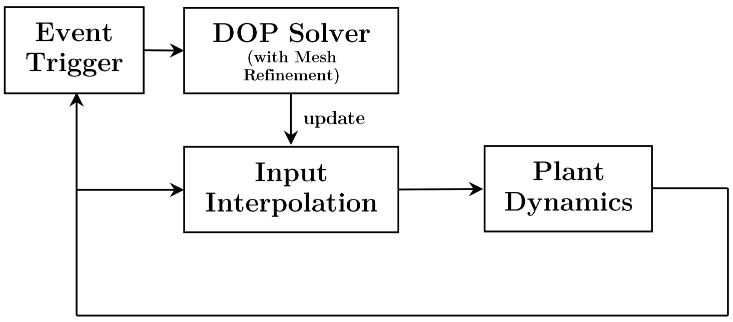
12-2-4 Closed-loop simulations

ICLOCS2 supports various types of closed-loop simulations within MATLAB/Simulink environment. Based on the nature of the closed-loop framework, the user may select the most suitable strategy for the problem formulation:

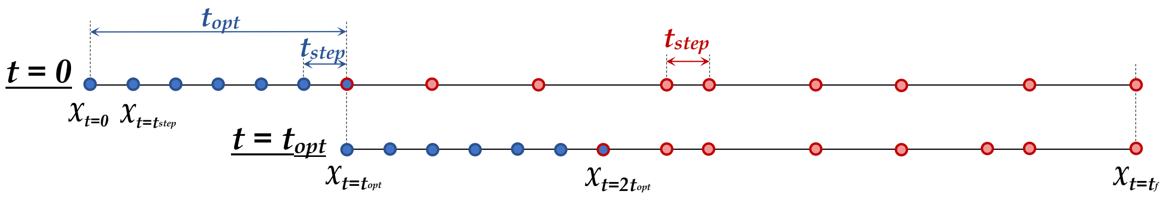
- **Receding horizon control (RHC):** Fixed (and often short) prediction and control horizon typically seen in classical MPC literature. Mostly applicable for regulation type of problems: stabilization and set-point tracking.
- **Variable horizon control (VHC):** Typically seen is the shrinking horizon control (SHC) strategy, used mainly for non-regulation type of problems (minimum-time and minimum-energy) for better stability, feasibility and optimality properties.

The user may also select betweeen different meshing techniques and control update policies:

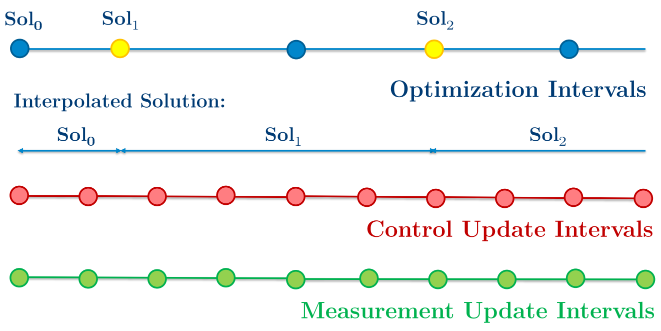
- **Uniform mesh:** The discretization mesh where DOP solutions are solved on is designed according to sampling rate or control update rate. In this way, there is an exact matching of the data on the grid points.
  - **Fixed interval update:** Based on the assumption that the computation of the DOP is shorter than each time interval, a fixed interval strategy can be used. This is similar to classical MPC, where re-computations occur at all time instances.
  - **Multi-step MPC:** With nonlinear problems, completing the computation may take longer than the sampling time. Also, for many practical problems, solving at all time instances may not be necessary. This motivates the use of multi-step MPC to resolve the DOP only at some points, based on the measurements of these time instances.
  - **Advanced multi-step MPC:** Similar to multi-step MPC, but the initial condition of the current solve is at a future time instant, estimated based on previous predictions.
- **Non-uniform mesh with interpolation** For VHC problems with minimum-time or minimum-energy objectives, the required time dimension of the problem is often large, making the use of a dense uniform mesh computationally inefficient and in some cases, prohibitive. In ICLOCS2, the user also has the possibility to use interpolated results from non-uniform mesh grids, with the accuracy and constraint satisfaction guaranteed by the mesh refinement scheme. Figure 12-7a illustrates the block diagram of such strategies.



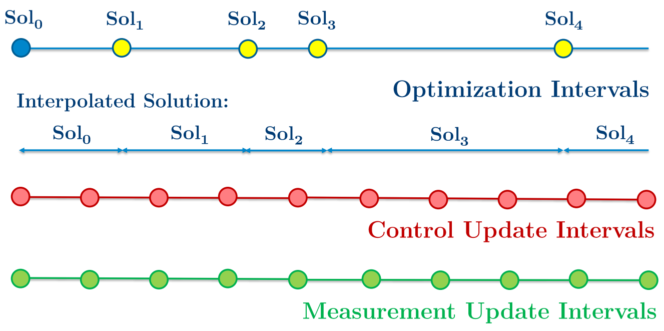
(a) Block diagram



(b) A meshing strategy with non-uniform mesh



(c) Fixed interval Update


Sol$_0$ Sol$_1$ Sol$_2$ Sol$_3$ Sol$_4$
Optimization Intervals
Interpolated Solution:
Sol$_0$ Sol$_1$ Sol$_2$ Sol$_3$ Sol$_4$
Control Update Intervals
Measurement Update Intervals

(d) Continuous Update

Figure 12-7: Closed-loop NMPC simulation with non-uniform mesh

- **Fixed interval update:** The DOP will be re-computed at a fixed interval of t_{opt}, providing updates for the input interpolation with a control rate of every t_{step} seconds. To have a better matching between the meshing points and control implementation instances, it is also possible to adapt the meshing strategy as shown in Figure 12-7b, with an equally spaced interval of length t_{step} for the first t_{opt} seconds, while the mesh for $[t_{opt},t_f]$ is determined by a mesh refinement scheme with minimum interval length equal to t_{step}.
- **Continuous update:** For an online strategy to update at fixed intervals, it becomes extremely difficult to anticipate the time at which the new solutions become available (indicated by the yellow nodes in Figure 12-7c). If the re-optimization takes too long, then the next update will be missed. On the other hand, if the solution is available much sooner than expected, there can be large duration of open-loop control. To overcome these difficulties, ICLOCS2 allows closed-loop implementations that the optimization solution is updated as soon as it becomes available (Figure 12-7d).
- **Event- and self- triggered update:** For many practical problems, neither of the above-mentioned options is preferred, as they failed to flexibly adapt to the problem nature and operational environment to efficiently utilize computing resources. ICLOCS2 allows the designer to have full control to the answer of the classical question: *"How often do one need to re-compute the control law?"*, by allowing users to specify customized events for the triggering of the re-optimizations and solution updates. For example, the events could be when
    - one of the constraints is potentially active,
    - current solution exceeds a validity period (forced re-optimization),
    - measured states deviated from the original trajectory by certain percentage.

Example: Supersonic Aircraft Minimum Time-to-climb

This example demonstrates the climbing flight path of a supersonic aircraft using minimum time. It is based on the minimum time-to-climb problem originally presented by Bryson et al. [33] and Betts [17, Ex. 6.4]. The system dynamics, as well as the propulsion and aerodynamic data, were directly taken from an equivalent implementation in metric units [163, Sec. 6.4].

From simple flight mechanics, the planar motion of the aircraft can be described by the following dynamic equations

$$\begin{aligned}
\dot{h}(t) &= v(t)\sin(\gamma(t)) \\
\dot{v}(t) &= \frac{T(t)\cos(\alpha(t)) - D(t)}{m(t)}
\end{aligned}$$

$$\dot{\gamma} = \frac{T(t)\sin(\alpha(t)) + L(t)}{m(t)v(t)} + \left(\frac{v(t)}{h(t)+R_E} - \frac{\mu}{v(t)(h(t)+R_E)^2}\right)\cos(\gamma)$$
$$\dot{m}(t) = -\frac{T(t)}{g_0 I_{sp}}$$

with h the altitude, v the velocity, γ the flight path angle, and m the mass. T is the thrust force as a function of M and h. L and D are lift and drag force respectively, both as a function of M, h and α. M is the Mach number, the variable which the aerodynamic and propulsion tabular data is based on. In terms of constants, there are $g = 9.81\,\text{m/s}^2$ the gravity acceleration, $R_E = 6378\,\text{km}$ the radius of the Earth, $\mu = 3.986 \times 10^{14}\,\text{m}^3/s^2$ the gravitational constant, and $I_{sp} = 1600\,\text{s}$ the specific impulse. α is the angle of attack serving as the control input in this problem. Please refer to the aforementioned references for more details regarding choice of aerodynamic modeling and parameter valuation.

The problem will only have a Mayer cost $\Phi = -m(t_f)$ (maximize the mass at the end of the climb), subject to the dynamics constraints. Furthermore, the simple bounds

$$0 \leq h \leq 19995 \text{ m}, \quad 5 \leq v \leq 1000 \text{ m/s}, \quad 40 \leq \gamma \leq 40 \text{ deg},$$
$$22 \leq m \leq 20410 \text{ kg}, \quad -20 \leq \alpha \leq 20 \text{ deg},$$

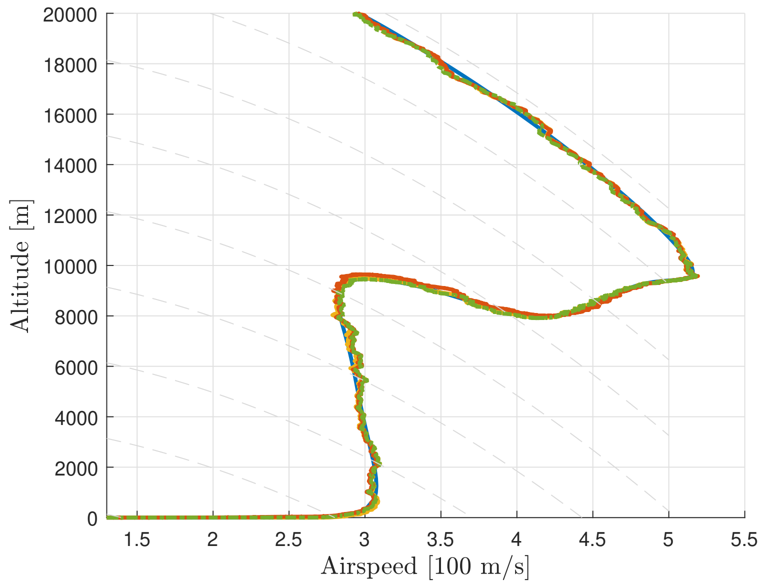
are imposed together with the boundary conditions

$$h(0) = 0 \text{ m}, \quad v(0) = 129 \text{ m/s}, \quad \gamma(0) = 0 \text{ deg}, \quad m(0) = 19050 \text{ kg},$$
$$h(t_f) = 19995 \text{ m}, \quad v(t_f) = 295 \text{ m/s}, \quad \gamma(t_f) = 0 \text{ deg}.$$

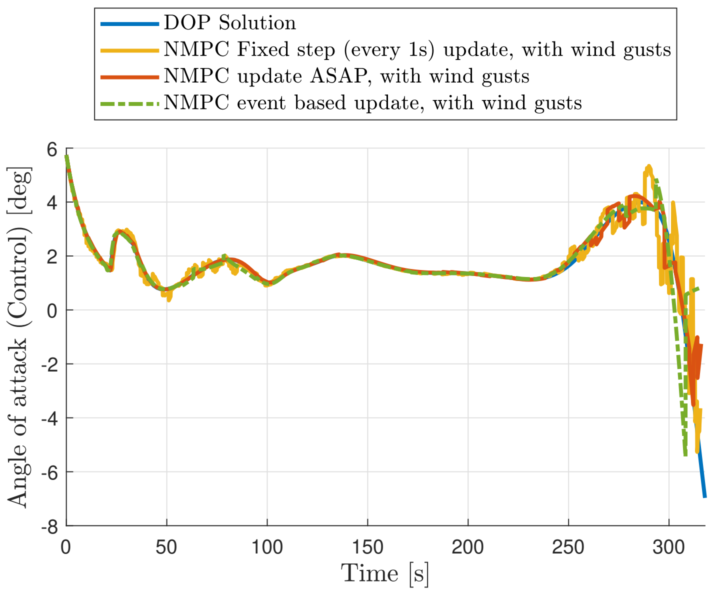
Figure 12-8 illustrates the DOP solution of the minimum time-to-climb problem, together with results from a closed-loop simulation subject to uncertainties of wind gusts (with Dryden wind turbulence model), solved to a user-defined solution accuracy requirement. It can be observed that the minimum-time climb profile for this supersonic aircraft to reach the terminal altitude of $h(t_f) = 19994.88\,\text{m}$ has a highly non-trivial solution requiring the exchange of potential energy with kinetic energy to overcome the sonic barrier efficiently. It has been verified that by focusing on a short term goal, receding horizon control (RHC) strategy with short horizons will attempt to climb continuously, thus failed to reach the final altitude due to power deficiencies. In contrast, closed-loop simulation with a shrinking horizon control (SHC) strategy and non-uniform mesh can successfully yield this non-trivial trajectory.

For the closed-loop simulation, the control update rate is assumed to be 0.1 s. Here three different re-optimization strategies are presented: update in a fixed time-step of every 1 s, continuous update as soon as each re-computation finishes, and event-triggered updates. Conditions for event-triggered updates are based on the first two criteria listed previously in Section 12-2-4, with forced re-optimization at every 15 s. Table 12-2 compares the computational performance and obtained solutions of these three closed-loop simulations.

Due to the SHC strategy, the closed-loop simulations are terminated when aircraft reaches an altitude of $h_f - 10\,\mathrm{m}$. It can be seen that for all three closed-loop simulations, the time needed to reach this attitude is lower than the DOP solution have anticipated, however at the cost of a larger γ and possible constraint violation of $\gamma(t_f) = 0°$ at $h = h(t_f)$. A fixed-step update of



(a) Trajectory



(b) Control input

Figure 12-8: Solution to the minimum energy-to-climb problem

Table 12-2: Closed-loop simulation results compared: minimum time-to-climb

	No. of Re-computations	**Average comp. time** [s]	**At $h_f - 10$ [m]**	
			t[s]	γ[°]
DOP Trajectory	-	-	316.71	3.51
NMPC fixed-step (1 s) update, with gust	315	6.21	315.60	5.65
NMPC continuous update, with gust	38	8.28	316.00	4.91
NMPC Event-Triggered, with gust	20	7.17	316.20	3.94

every 1 s results in the highest computation cost, making the simulation running shower than real-time. In contrast, the continuous update and event-triggered closed-loop implementation are simulating in real-time, being much more efficient thanks to a significantly lower number of re-computations.

Part IV

Aerospace Applications

Chapter 13

Dynamic Optimization for Aerospace Applications

Reviewing more than a century of aviation and space activities, the design and optimization of flight and orbital trajectories have played crucial roles in the success of history-making missions such as supersonic flights and landing on the moon. Looking into the future, these capabilities will also be of ever-increasing importance in the transformation towards green aviation and commercial space flights.

This chapter aims to provide a review of past developments in aerospace sciences involving the solution of DOPs, either in the form of flight and orbital trajectory optimization in Section 13-1 or flight control designs using MPC in Section 13-2. With the underlying methods and related mathematics so diverse and the fields of applications really broad, the focuses will be on a number of representative problems which reflect the progress in technology. By highlighting the efforts and contributions made by the pioneers in tacking various challenges, reflections on the current technical developments can be made regarding the potentials of new methods which are seemingly overambitious and premature today.

13-1 Flight and orbital trajectory optimization

From the early days of aviation history, engineers are in search of “best” trajectories for the motion of a vehicle in a three-dimensional space. Depending on the objective selected for the mission, the focuses of the problem can be either capability based, e.g. maximum altitude or maximum range, or based on economy and environmental impacts, e.g. minimum fuel or minimum noise footprint. Regardless of the problem design, a distinction can also be made depending on the way how DOPs are solved, either analytically or numerically.

13-1-1 Analytical approach

For simplified point-mass flight mechanics models in particular phases of the flight, the optimal flight strategies can be obtained by analytically deriving the optimality conditions in the framework of the calculus of variations. Among the developments is the famous Goddard rocket problem [77], with the aim to maximize the highest altitude reachable by a rocket using a fixed amount of propellant. Throughout history, depending on the fidelity of the modelling of the atmospheric drag, different solution structures have been identified for the optimal control input. When neglecting or considering a linear-drag (w.r.t. velocity) only [191], the solution is shown to be in the form of *bang-bang*, i.e. to exhaust all propellant with maximum thrust setting at launch and initial ascend, and then coasting to the highest point. With a quadratic drag model commonly used in subsonic flights [17, 191], the optimal solution structure changes to *bang-singular-bang* with an intermediate low thrust profile. This is also the formulation used in the demonstration in Section 7-3-1. Later, after taking into consideration the effects of wave drag encountered mainly in high subsonic and supersonic regimes, it was found that the solution structure have now changes to *bang-singular-bang-singular-bang* [192].

The evolvements in the understanding of the Goddard rocket problem highlights the pros and cons of the analytical approach: although it is capable of yielding optimal control strategies and guidance laws that are computationally trivial to implement, the analytical derivation process can be tedious, requiring substantial simplifications to be made resulting in large discrepancies between the problem formulations and the actual tasks in practice.

Even if one accepted the challenge and completes the analytical derivation without significant simplifications (i.e. the DOPs formulated directly based on the mission specifications), the solutions obtained will still become increasingly hard to straight-forwardly interpret and implement, as the fidelity and complexity of the models inside the DOPs rise. This is often a bigger concern for atmospheric flights in comparison to spacecraft orbits due to the presence of larger uncertainties in operational conditions. For example, for launch vehicle ascend problems at large, the optimal control solution is approximately piecewise linear, hence in practice implemented as a sequence of constant pitch rate commands. Another example would be the iconic supersonic aircraft minimum-time to climb problem discussed in Section 1-1, where the optimal flight profile is often approximated with several segments of guidance strategies in practice.

Other well-known aerospace problems that have been solved under the analytical approach include aircraft go-around in the presence of windshear [34, 146], minimum fuel turns at constant altitudes [199], maximum endurance and range for cruising or gliding flights [199], optimal pull-up or loop maneuvers [199], as well as spacecraft reentry [199], orbit transfer [32, 138, 199], and lunar landing [143] problems. However, with missions become more challenging and computation capabilities rapidly increasing, numerical solution of DOPs gradually becoming the mainstream of solving trajectory optimization problems.

13-1-2 Numerical solutions

In the development of numerical methods for solving DOPs, the above-mentioned problems with known analytical solutions have played important roles as benchmarking examples: aircraft go-around in the presence of windshear [35, 204], maximum endurance gliding flights [36], minimum fuel cruise [72], aircraft climb flight [17, 44], spacecraft orbit transfer [161, 213], and lunar landing [9, 92] problems.

As the numerical methods get mature, they become the backbone in the design and optimization of flight strategies. For civil aviation, frameworks were proposed to integrate the solution of DOPs into the flight management systems (FMS) for economic climb and descent [197] as well as for cruise [198]. For aerobatic flights, aggressive turn-around manoeuvres with a fixed-wing unmanned aerial vehicle (UAV) was studied [124]. For helicopters, the optimal control approach was used to design noise abatement trajectories [87]. In recent years, there has also been a growing interest to use UAVs or kites to form a so-called airborne wind energy (AWE) system, for the generation of energy from high-altitude winds. The problems were formulated as DOPs and solved numerically [98, 119] to obtain the optimal flight trajectories for maximum energy extraction.

As the inverse problem of optimal control, many estimation problems in aerospace engineering also employ the solution of DOPs, for example, for the identification of aerodynamic model parameters of UAVs in AWE systems [126, 127]. Similarly for a Boeing 777-200 commercial airliner, DOP solution are used for the inflight identification of aerodynamic coefficients to determine the drag-optimal deflection angle for wing trailing edge control surfaces [17].

In space applications, multi-phase DOP setups are used for the trajectory optimization of launch vehicles, with the capability to handle discontinues in state trajectories (e.g. mass change during separation of different rocket stages). Examples include the Delta III launch vehicle [170] and a reusable rocket launch and recovery case [133]. Strategies for attitude control and reorientation of spacecraft in orbit have been studied as well, for the NASA X-ray Timing Explorer (XTE) spacecraft [65] as well as the international space station (ISS) [13, 164]. For reentry, a numerical study for the space shuttle reentry trajectories using different criteria is made available [212].

DOP solutions have been used extensively for orbit transfers of spacecraft. Low-thrust orbit transfers to the moon [147], as well as to geosynchronous and Molniya orbits [20], have been demonstrated, with many problems characterized by very long time horizons. Additionally, the optimal transfers can be assisted by gravity, e.g. with lunar swingby [17], or the atmospheric drag with multiple passes [169].

13-2 Flight control using predictive control

Given sufficient computational resources, numerical schemes are capable of solving the majority of DOPs formulated to address certain challenges in aerospace design and engineering. Nevertheless, solving DOPs offline has its limitations and an increasing number of applications now desire the solution and implementation of DOPs online, to drastically improve the level of autonomy in mission executions. As of today, many are still skeptical regarding the reliability and real-time performance of solving nonlinear non-regulation DOPs online using NLP solvers. Admittedly, there are still many technical challenges to be resolved. However, it may be beneficial to look back into history when solving linear flight controls MPC problems in real-time was still considered to be an unachievable task, to vision what future can possibly bring.

13-2-1 Attractiveness and challenges

The popularity of MPC in process industry is largely due to its enhanced ability to control multivariable systems subject to various constraints, with a straightforward implementation. The same analogy indisputably holds for aerospace applications. Take flight control designs for example, aerodynamic limits of the angle of attack and structural integrity limits of load factors are all crucial aspects of safe flight operations that can be naturally handled by MPC with constraints. Also, the rate and saturation limits of control surface deflections can also be fully accounted for. Arguably MPC is the only control architecture that is able to handle these issues in a systematic way, while solutions of other types are mainly tackling these issues in an ad-hoc heuristic manner (e.g. pseudo control hedging for nonlinear dynamic inversion (NDI), a form of feedback linearization (FBL)).

Although it is difficult to trace back in time when exactly the research in MPC and the flight control system design have come together, many attributes it to [94], and subsequently [181]. The sabbatical visit of Jan Maciejowski to the Faculty of Aerospace Engineering of TU Delft had further accelerated this process. The lecture notes [134] has later became the foundation for his famous book *Predictive Control: with Constraints* [135]. The book has abundant aerospace examples using the linearized model of the Cessna Citation II aircraft (of TU Delft), hence shown to the world the great potential for applications of MPC in aerospace sectors. More importantly, the inherent online reconfiguration capability of MPC made its debut for aerospace applications through an example scenario of a jammed rudder, making MPC subsequently became a very popular candidate for research in fault-tolerant flight controls.

13-2-2 Fault tolerant flight control with MPC

Reconfigurable flight control has been considered as a field of application where the capabilities and flexibility of optimization-based control methods can be fully utilized and exploited [111], hence becoming a major motivation for the development of MPC for flight control applications. Main challenges for reconfigurable flight control system (RFCS) design have been identified [29, 160], and summarised [111] as :

- Strong cross couplings between modes usually appear after failure. The symmetry of the aircraft may be lost after damage to the airframe. Thus conventional simplification of decoupling longitudinal and lateral direction control might not be applicable.
- The dynamics of an aircraft can change significantly after failure. Hence, if linear control techniques are used, trim values and Jacobian linearized dynamic models used by the controller are also changing. Thus, the continuous use of a nonlinear or adaptive control algorithm is preferred.
- The system may be highly unstable, leaving very little time for control reconfiguration. This demands extremely efficient online identification and controller redesign algorithms. Pre-stabilization may be required for unstable systems to avoid numerical ill-conditioning of the prediction model.
- After an aircraft has sustained damage to a surface, its ability to produce the required control forces and moments degrades. Hence, the demands on healthy and available actuators (i.e. deflection and deflection rates) will increase. This also aggravates the control saturation problem.

The main purpose of early work on MPC for fault-tolerant flight control focuses on demonstrating the capability of MPC to naturally handle actuator failures, as well as faults that can lead to a structural change of the aircraft (e.g. damage to airframe). For example, in the safe recovery of the crashed El Al Flight 1862 with reference model following MPC control together with the availability of a fault detection and isolation (FDI) scheme [136]. Although such a scheme may not always be necessarily [195], the FDI and online system identification components are beneficial in dealing with sensor faults and improve the robustness towards model uncertainty after failure [110].

When severe damages and failures occur, the reference model following MPC alone may not be capable of guaranteeing the stability and robustness required. Thus adapting the parameters used in the MPC algorithm, such as prediction and control horizons and/or state and input weighting may be necessary. Another alternative is to change the unachievable targets with target recalculations [111], an idea similar to pseudo control hedging.

For unanticipated fault scenarios, the data-driven subspace approach for predictive control design [109] can be applied to fault-tolerant flight control [84]. A subspace predictor for prediction of future output is recursively updated online based on new input-output data, so that the algorithm is quick in adapting changes in system conditions, implicitly addressing the issue of robustness with respect to model uncertainties [195].

Flight test of MPC based fault tolerant flight control has been successfully accomplished on the German Aerospace Center's (DLR) ATTAS aircraft using an implicit model following technique with a simulated aileron fault [47]. For this purpose, two single-input single-output (SISO) MPC controllers for the decoupled longitudinal and lateral dynamics has been implemented, together with an online FDI system. Due to the computational complexity, the flight test was conducted with a pre-defined trajectory with the invariant set for the terminal constraints calculated offline.

13-2-3 Towards real-time implementation of MPC flight control

In order to perform the required optimization problem online at each time step, considerable computational efforts are associated with finding the optimal input trajectory or sequence. To materialize the above-mentioned benefits MPC can bring, the issues associated with real-time performance must be addressed. With the rapid development of modern computing hardware as well as efficient algorithms, the computation times required for MPC are getting more tractable for systems that require faster control updates, quickly making it a viable alternative for flight control applications.

At the very early stage, researches are aware of the issues associated with ensuring stability as well as the real-time performance, and made extensive studies in these aspects. In an in-lab thrust-vectored flight experiment at Caltech, set-point tracking has been achieved by an MPC controller with computational speeds 2.5 times faster than that of the actuator dynamics [53]. In order to achieve this, a hybrid MPC and control Lyapunov function (CLF) based design was used so the controller can be stabilized without terminal constraints. As for timing, the "compute as fast as possible" strategy was adopted such that computations are running continuously and the control solution is applied as soon as it is available. This has lead to uneven time intervals typically in the range of 0.05 to 0.25 seconds.

The large operational domain of aircraft makes the traditional use of a single linearized model impossible. In order to have a scheme similar to gain scheduling for PID based flight control design, a quasi-LPV (linear parameter varying) approach can be used to schedule the parameters of a high-fidelity missile model based on actual flight conditions [102]. Later, this approach has been adapted to control a simulated nonlinear model of an F16 aircraft [116], achieving good performance while retaining modest computational complexity.

Using the same F16 setup, a comparative study was completed subsequently [117] benchmarking the performance and computational complexity of linear MPC, scheduled quasi-LPV MPC and nonlinear MPC. Linear MPC designed for a different flight condition indeed leads to steady-state errors for the disturbance rejection task, and large tracking errors appear when excursions are made in the flight envelope (demonstrated with a climb). Nonlinear MPC achieves the fastest disturbance rejection; however, due to numerical problems, peak altitude error is greater with a larger and more oscillatory control input. As for the climb task, it prioritizes velocity tracking over altitude errors and achieves the lowest cost of the objective function. In both cases, even in comparison with nonlinear technique, the scheduled MPC achieves acceptable performance. Therefore, a third scenario was tested with the scheduled MPC and demonstrated promising performance even in aggressive manoeuvres.

As for the computational time at each sampling period in the Matlab environment, a distinction was made between run-time of the optimization algorithm and other calculations. Although the average and maximum time in optimization were indistinguishable between linear and scheduled MPC, the additional computational overhead for scheduled MPC due to data interpolation results in a total execution time approximately 30 times higher. The average time spent on optimization for nonlinear MPC saw a rise of over 1000 times (for maximum value: more than 2000 times) in comparison to the linear case, making the total time needed per solve more than half a minute long on average (peak value 102.55 seconds), completely prohibiting any real-time implementations.

The idea of combining MPC and FBL to have a nonlinear flight control method that only requires the solution of linear MPC problems is not new. However already in the early work [174, 185, 193], researches found that although FBL can turn nonlinear dynamic equations into linear ones, nonlinear mappings are necessary for the simple bounds that are enforced on the original state and input variables. Since these early work is mainly focusing on improving the robustness of FBL methods, simple and intuitive approximations to the constraints were applied (e.g. methods based on [122]) to retain the form of a linear MPC. Later, more systematic and efficient treatments of this constraint mapping start to emerge, and the computational aspect of the method start to get exploited. For example, the algorithm in [106] provides an efficient way to map the input constraint for over-actuated systems ($m>n$) and to handle the control allocation. Another alternative to handle the nonlinear constraints is by using a set of dynamically generated local inner polytopic approximations [184]. In this way, the convexification of nonlinear constraints is achieved. Although the MPC problem still needs to be solved as nonlinear optimization problems, recursive feasibility and convergence can now be guaranteed for any online re-computation.

With rapid evolvement in computing hardware, and more importantly, development of new structure exploiting algorithms for the numerical solution of DOPs, the days when real-time implementation of linear flight controls MPCs are considered impossible have become the past. Computation time in the order of 10 ms per sampling interval of 0.2 s was achieved for linear MPC flight control of a Boeing 747-200 aircraft [90, 91], for both simulations on desktop computers and embedded optimization on field programmable gate arrays (FPGAs). A similar computation time has been reported for a fast MPC implementation of aircraft stall recovery guidance [182], using linearized system dynamics. Control of very flexible aircraft using successive linearized MPC has been studied [208], and have reported a computation time in the order of 0.1 s for a sampling interval requirement of 0.2 s, just by using MATLAB's built-in solver of `fmincon`. In addition to simulations, successful flight tests were achieved on a small quad-rotor UAV [10], where a low complexity algorithm was computed in real-time on an embedded hardware with the MPC controller running at 16Hz. In space applications, an article [24] revealed that the recent success in the autonomous landing of the Falcon series rockets by SpaceX was achieved through high-speed onboard convex optimization, with flight code generated by CVXGEN [139].

With the progress on advanced algorithms for nonlinear programming, especially the development of real-time iterations (RTI) [50, 80], which only requires the solution of one QP per instant of control update, the computational overheads of NMPC are also no longer intractable for flight control applications. As of today, many NMPC implementations for flight control have reported computational performances faster than real-time requirements, with the majority of them solving DOPs with the quadratic regulation cost on a relatively short horizon. For example, real-time implementation of a non-hierarchical NMPC formulation is demonstrated [79], with an execution time in the order of 30 ms, about 15 times faster than required. Simulated scenarios such as extreme manoeuvres, actuator failure and emergency obstacle avoidance have all been tested and handled well by the MPC controller.

Flight controls using NMPC with non-regulation formulations still remains computationally challenging. For example, an economic NMPC framework for aircraft deep-stall recovery [45] reported an average DOP solution time of around 2 s for a sampling time of only 50 ms. A major part of the challenge arises from the use of a much longer horizon, generally required for non-regulation formulations to demonstrate stability and good performance.

Chapter 14

Optimization of Flight Trajectory For Commercial Airliners

As discussed in Chapter 13, trajectory optimization of aerospace vehicles has played very important roles in the discovery of many non-intuitive flying strategies, often for military fighters. In civil aviation, active use of trajectory optimization is less common. This is mainly due to the fact that the most dominating phase of the flight, cruise, is considered generally as a steady phase without many manoeuvres needed. Therefore regarding the mission objective of civil airliners, i.e reaching the destination on time in the most fuel-efficient manner, the optimal trajectory is not going to differ much from the heuristic methods that are currently in use. From the point of view of a single flight, the fuel-saving potential is often neglected in favour of simplicity in flying strategies.

With passenger number in air transportation almost doubling in the past ten years [104], and with the environmental impact of the aviation industry now under the spotlight, achieving sustainable flight [85] is now considered as a system-wide challenge. Under this context, a merely 0.1% of fuel-saving which used to be considered negligible for each individual flight is now viewed as a significant figure, considering the number of flights around the world every year. As a result, attentions have been made to all technologies that can lead to potential fuel savings, with an increasing number of trajectory optimization studies for civil aircraft.

In this chapter, a relatively high fidelity model tailored for solving trajectory optimization of civil aircraft will be first presented in Section 14-1, based on various work in the literature [21, 88, 89, 188, 190]. This is followed by Section 14-2 with a continuous cruise climb problem, demonstrating the differences in the solutions obtained from different objective formulations, and discretization and transcription methods. A typical multi-phase formulation for real-world flight operations is shown in Section 14-3, which is followed by the introduction of a simplified 3-phase formulation in Section 14-4, demonstrating computational and flexibility advantages.

14-1 System dynamics modelling

Using a point mass flight mechanics model with a spherical Earth, the system dynamics for the trajectory of a flight can be described using

$$\dot{h}(t) = v_{TAS}(t)\sin(\gamma(t)), \tag{14-1a}$$

$$\dot{\vartheta}(t) = \frac{v_{TAS}(t)\cos(\gamma(t))\cos(\chi(t))}{R_E + h(t)}, \tag{14-1b}$$

$$\dot{\varpi}(t) = \frac{v_{TAS}(t)\cos(\gamma(t))\sin(\chi(t))}{(R_E + h(t))\,\vartheta(t)}, \tag{14-1c}$$

$$\dot{v}_{TAS}(t) = \frac{1}{m(t)}(T(v_{CAS}(t), h(t), \Gamma(t)) - D(v_{TAS}(t), h(t), C_L(t))) - g\sin(\gamma(t)), \tag{14-1d}$$

$$\dot{\gamma}(t) = \frac{1}{m(t)v_{TAS}(t)}(L(v_{TAS}(t), h(t), C_L(t)) - m(t)g\cos(\gamma(t))), \tag{14-1e}$$

$$\dot{m}(t) = FF(h(t), T(v_{CAS}(t), M(t), h(t), \Gamma(t))), \tag{14-1f}$$

with h the altitude, ϑ the latitude, ϖ the longitude, v_{TAS} the true airspeed, γ the flight path angle and m the mass of the aircraft, all serving as state variables. There are three input variables with C_L the lift coefficient, χ the tracking angle and Γ the throttle settings. Other parameters and variables including

- the earth radius $R_e = 6.371 \times 10^6\,\text{km}$,
- the calibrated airspeed v_{CAS}. The conversion between v_{TAS} and v_{CAS} is as follows

$$v_{TAS}(t) = \sqrt{\frac{2\kappa}{\kappa-1}\frac{p_h(t)}{\rho_h(t)}\left[\left[1 + \frac{p_{sl}}{p_h(t)}\left[\left(1 + \frac{\kappa-1}{2\kappa}\frac{\rho_{sl}}{p_{sl}}v_{CAS}(t)^2\right)^{\frac{\kappa}{\kappa-1}} - 1\right]\right]^{\frac{\kappa-1}{\kappa}} - 1\right]},$$

  with p_h and ρ_h the pressure and density at current altitude h. $p_{sl} = 101325\,\text{Pa}$ and $\rho_{sl} = 1.225\,\text{kg/m}^3$ are the air pressure and density at sea level, and $\kappa = 1.4$ is the heat capacity ratio.

- the Mach number M as a function of h and v_{TAS},
- the thrust force T as a function of h, v_{CAS} and Γ,

- the drag force D as a function of h, v_{TAS} and C_L,
- the lift force L as a function of h, v_{TAS} and C_L,
- the fuel flow model FF as a function of h and T.

In this work, the fast dynamics, such as pitch and roll attitude, are assumed to lead to negligible differences in the solutions of flight time and fuel consumption. Thus they are neglected to improve the computational efficiency by 1) reducing the number of state and input variables and 2) alleviates the resolution required to accurately solve differential equations of fast dynamics, allowing the use of a coarser mesh.

14-1-1 Lift and drag modelling

The lift force of the aircraft can be expressed as

$$L(t) = 0.5\rho_h(t)C_L(t)v_{TAS}^2(t)S_{w,ac}$$

with $S_{w,ac}$ the reference wing area of the aircraft. Similarly, the drag force is modelled as

$$D(t) = 0.5\rho_h(t)C_D(t)v_{TAS}^2(t)S_{w,ac} \tag{14-2}$$

with the drag coefficient consisting the zero-lift drag coefficient C_{D0}, the induced drag coefficient $k_{cl}C_L^2$ and an additional wave drag coefficient contribution $\Delta C_{D,wave}$ due to transonic effects at high Mach numbers, expressed as

$$\begin{aligned} C_D(t) =& C_{D0} + k_{cl}C_L^2(t) + \Delta C_{D,wave}(M(t)), \\ \Delta C_{D,wave}(t) =& \begin{cases} 0 & \text{if } M \leq \frac{M_{crit}}{\cos(\Lambda_{sw})} \\ 20(M(t) - M_{crit})^4 & \text{if } M > \frac{M_{crit}}{\cos(\Lambda_{sw})}, \end{cases} \end{aligned}$$

with M_{crit} the critical Mach number at which the airfoil will have regions reaching the speed of sound locally, and Λ_{sw} is the sweep angle of the aircraft.

14-1-2 Thrust modelling

Regarding the modeling of thrust available, a linear relationship between the thrust produced by each engine T_{eg} and the throttle setting $\Gamma(t)$ representing the percentile of the maximum thrust T_{max} is assumed, with

$$T_{eg}(t) = T_{max}(t)\Gamma(t).$$

The total thrust is simply T_{eg} times the number of engines N_{eg} for the aircraft type:

$$T(t) = T_{eg}(t)N_{eg}.$$

The maximum thrust available from the engines will vary depending on the altitude and airspeed. Here, the expressions for high, medium and low altitudes [190] are presented as

$$T_{max}(t) = \begin{cases} T_{cr,30}\left[c_{eg,5}(t)\frac{p_h(t)}{p_{30}} + \left(\frac{T_{10}(t)}{T_{cr,30}} - c_{eg,5}(t)\frac{p_h(t)}{p_{30}}\right)\right] & \text{if } h \leq 10000\,\text{ft}, \\ T_{cr,30}c_{eg,3}(t)\left(\frac{p_h(t)}{p_{30}}\right)^{c_{eg,4}(t)} & \text{if } 10000 < h \leq 30000\,\text{ft}, \\ T_{cr,30}c_{eg,1}(t)\ln\left(\frac{p_h(t)}{p_{30}}\right) + T_{cr,30}c_{eg,2}(t) & \text{if } h > 30000\,\text{ft}. \end{cases}$$

Using the static maximum thrust at sea level, T_{sl}, the maximum thrust at a cruise level of 30000 ft, $T_{cr,30}$ can be estimated to be

$$T_{cr,30} = T_{sl}\frac{p_{30}\Upsilon_{sl}}{p_{sl}\Upsilon_{30}}$$

with p_{30} the pressure at 30000 ft, and Υ_{sl} and Υ_{30} the temperature in Kelvin at sea level and 30000 ft respectively. Using this cruise thrust, the thrust at 10000 ft can be modelled as

$$T_{10}(t) = c_{eg,3}(t)T_{cr,30}\left(\frac{p_{10}}{p_{30}}\right)^{c_{eg,4}(t)}.$$

Other ingredients necessary for the computations are the coefficients $c_{eg,1}$ to $c_{eg,5}$, formulated based on the following empirical relationships:

$$\begin{aligned} c_{eg,1}(t) &= -0.4204\left(\frac{M(t)}{M_{cr}}\right) + 1.0824, \\ c_{eg,2}(t) &= \left(\frac{M(t)}{M_{cr}}\right)^{-0.11}, \\ c_{eg,3}(t) &= \left(\frac{v_{CAS}(t)}{v_{CAS,cr}}\right)^{-0.1}, \\ c_{eg,4}(t) &= -0.335\left(\frac{v_{CAS}(t)}{v_{CAS,cr}}\right) + 0.9166, \\ c_{eg,5}(t) &= -0.12043\left(\frac{v_{CAS}(t)}{v_{CAS,cr}}\right) + 0.4871, \end{aligned}$$

representing the variation of available thrust due to changes in airspeed, either expressed in terms of calibrated airspeed v_{CAS} or Mach number $M(t)$. M_{cr} is the typical cruise Mach number for the aircraft type, and $v_{CAS,cr}$ is the typical calibrated airspeed at cruise.

14-1-3 Fuel flow modelling

For computation of fuel consumption of the whole aircraft FF in kg/s, it is important to model the necessary fuel flow at different altitude and thrust levels. The modeling of fuel flow for each engine $FF_{eg}(t)$ is

$$FF_{eg}(t) = c_{ff3}\phi_{TR}^3(t) + c_{ff2}\phi_{TR}^2(t) + c_{ff1}\phi_{TR}(t) + c_{ffch}T_{eg}(t)h(t). \tag{14-3}$$

The equation consists of two parts, with the first three terms representing the fuel flow at sea level. The sea level consumption is based on a 3rd-degree polynomial fitting of the ICAO engine emission data-bank [105] with coefficients c_{ff1}, c_{ff2}, c_{ff3} and the thrust ratio ϕ_{TR}, defined as

$$\phi_{TR}(t) = \frac{T_{eg}(t)}{T_0}.$$

The last term in (14-3) accounts for the increase in fuel consumption in high altitudes, with $c_{ffch} = 5 \times 10^{-7}$ kg/s/kN/m.

14-2 Cruise flight with continuous climb

First consider the cruise phase of the flight: as the aircraft's weight reduces during the cruise, it is beneficial for reduction of fuel usage to fly at a higher altitude in order to take advantage of the lower air density. Although this is often simplified to a number of step climbs in practice and therefore replicated in flight trajectory optimization studies [21, 186], the optimal strategy in terms of fuel consumption is to consider a continuous cruise climb profile [178].

14-2-1 Formulation of the objective

With this opportunity, different objective formulations can be compared. One possibility is to design based on mission specifications: i.e to directly minimize fuel consumption. For simplicity and for this demonstration only, the aircraft's initial weight and fuel onboard are fixed; hence the problem is equivalent to maximize the final weight of the aircraft, with a Mayer cost $J = -m(t_f)$.

Solution for this formulation will be compared with typical 'control effort' formulations of Lagrange cost $J = \int_{\mathbb{T}} FF^2(t)\,dt$, $J = \int_{\mathbb{T}} T^2(t)\,dt$ and $J = \int_{\mathbb{T}} \Gamma^2(t)\,dt$, i.e. to minimize the integral of the square of fuel flow, thrust and throttle settings, respectively. For a transcontinental flight from London to Shanghai using aircraft Type 1 (see Appendix C) with an initial mass of 345840 kg, the comparison results are illustrated in Figure 14-1 and Table 14-1.

It can be observed that these different formulations have led to drastically different flight profiles, and more importantly, differences in fuel consumption. Directly minimizing the

Table 14-1: Comparison of the fuel consumption of different continuous cruise climb solutions using different objective formulations, with HS discretization using 30 major nodes

Objective Formulation	Fuel Consumption
$J = -m(t_f)$	123714.78 kg
$J = \int_{\mathbb{T}} FF(t)\,dt$	123735.77 kg (+0.017%)
$J = \int_{\mathbb{T}} FF^2(t)\,dt$	124295,77 kg (+0.47%)
$J = \int_{\mathbb{T}} T^2(t)\,dt$	125223,37 kg (+1.22%)
$J = \int_{\mathbb{T}} \Gamma^2(t)\,dt$	129365.93 kg (+4.57%)

fuel usage (i.e. reduction in weight due to fuel burn) results in the most fuel-efficient flight, whereas minimizing $J = \int_{\mathbb{T}} \Gamma^2(t)\,dt$ has led to the highest fuel consumption, with an increase of 5651.15 kg or equivalently 4.57%. This experiment highlights the issues that are associated with the classical quadratic regulation cost formulation penalizing control effort in the form of $u^2(t)$.

The solution trajectories in Figure 14-1, however, also unveils that directly minimizing the reduction in weight has resulted in the most pronounced singular arc fluctuations. The results also show that reformulating the Mayer cost into an equivalent Lagrange term $J = \int_{\mathbb{T}} FF(t)dt$ will not help. Therefore, in order to benefit from the advantages of formulating and solving the DOPs directly according to problem specifications, the issue of singular arc must be addressed.

14-2-2 Improve solution accuracy and suppress singular arc with IRM-type methods

When constant altitude and constant Mach cruise flights are considered, the system dynamics can be simplified, and in extreme cases as in [21], only the state variable of aircraft weight is used. This reduction has not only lead to lower computation complexities but also avoids the singular arc fluctuations in the minimum fuel solution, even if the the objective $J = -m(t_f)$ is directly used.

With the full dynamics considered in a continuous cruise climb setup, singular arc fluctuations occurred for all three discretization methods tested with direct collocation transcription, as shown in Figure 14-2. For the HS discretization, additional information is presented in Figure 14-3 for the implementation of the resultant input trajectory on the same model, solved with a non-stiff variable-order ODE solver (MATLAB `ode113`) with a time step of 0.1 s. Due to large errors between collocation points, significant deviations between the DOP solution and the implementation result can be observed in Figure 14-3. This result is further evidenced by the evaluation of the integrated residual squared (IRS) errors in Table 14-2.

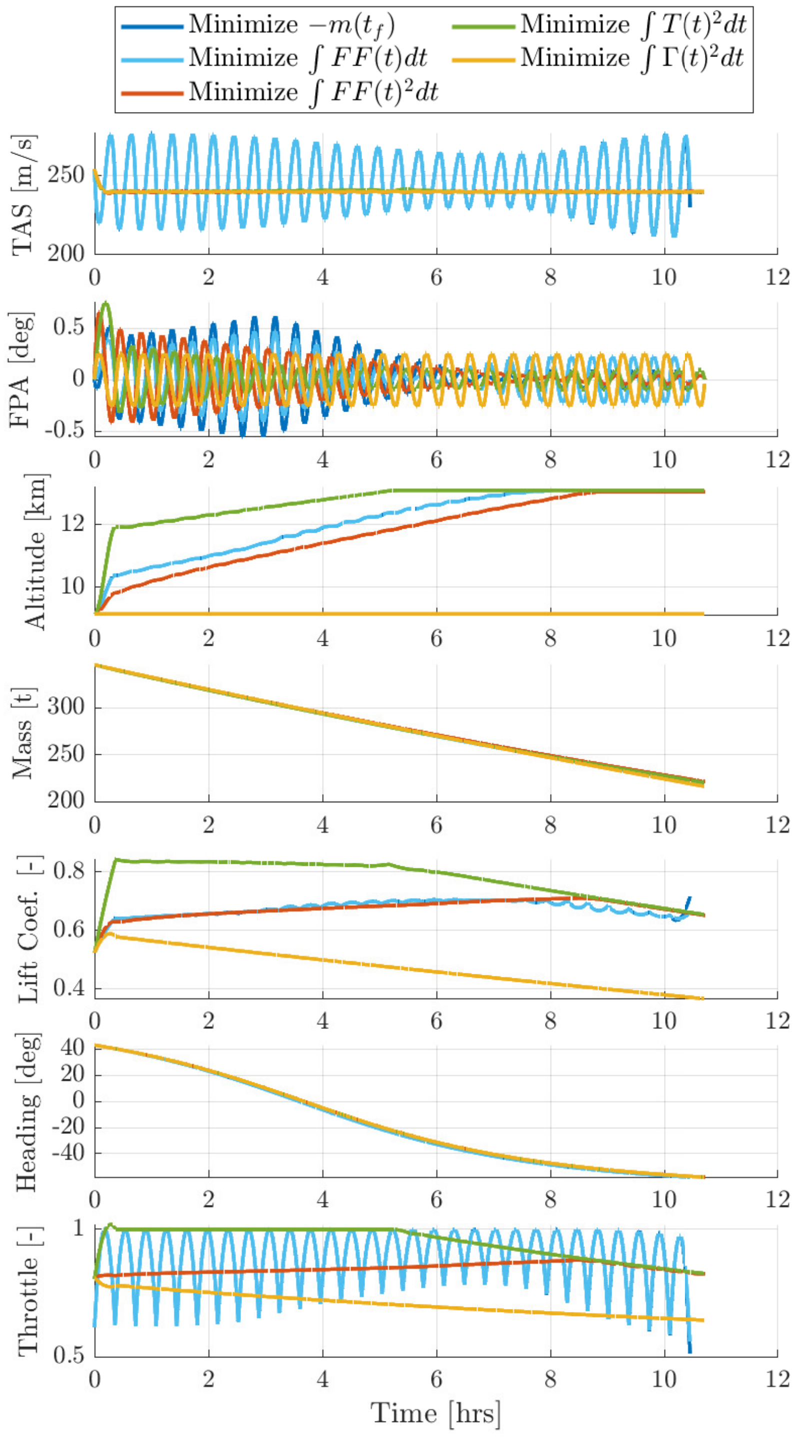


Figure 14-1: Comparison of solutions to the continuous cruise climb flight problem, with different objective formulations, direct collocation with HS discretization using 30 major nodes

Table 14-2: Comparison of the continuous cruise climb solutions with different discretization methods for direct collocation (30 major nodes)

		Trapezoidal	**Hermite-Simpson**	**hp-LGR, degree 5**
IRS	h	3.35×10^{4}	2.04×10^{-1}	1.19×10^{-2}
Error	ϑ	4.51×10^{-2}	5.67×10^{-13}	1.10×10^{-16}
of	ϖ	1.28×10^{-1}	3.93×10^{-12}	1.15×10^{-15}
ODE	v_{TAS}	3.92×10^{3}	1.09×10^{-2}	2.67×10^{-5}
Dynamic	γ	1.77×10^{2}	7.22×10^{-6}	1.02×10^{-9}
Equations	m	1.04×10^{3}	2.85×10^{-4}	1.08×10^{-5}

Table 14-3: Continuous cruise climb problem with IRM solution representation method, with HS discretization / piecewise cubic parameterization using 30 major nodes

		Direct Collocation	**Direct Collocation with Solution Representation by IRM**
IRS	h	2.04×10^{-1}	5.93×10^{-11} (-99.99%)
Error	ϑ	5.67×10^{-13}	1.36×10^{-13} (-79.06%)
of	ϖ	3.93×10^{-12}	1.22×10^{-15} (-99.97%)
ODE	v_{TAS}	1.09×10^{-2}	1.08×10^{-11} (-99.99%)
Dynamic	γ	7.22×10^{-6}	2.05×10^{-14} (-99.99%)
Equations	m	2.85×10^{-4}	9.45×10^{-07} (-99.67%)
Fuel Consumption		123714.78 kg	123714.72 kg (-0.00005.47%)

In contrast, solutions represented using IRM all have the singular arc fluctuations drastically suppressed, and more importantly, only minor discrepancies can be observed between the DOP solution and the implementation result. This observation is supported by the data in Figure 14-3, showing a negligible reduction in objective value but a substantial reduction in the IRS error for all ODE dynamic equations.

Unfortunately, due to the development of IRM-type methods being at their early stages, there is not yet a multi-phase implementation that would allow the problems in later part of this chapter as well as those in Chapter 15 to benefit from this drastic improvement in accuracy. Hence, a (potentially sub-optimal) alternative of simultaneous regularization still has to be used in later examples.

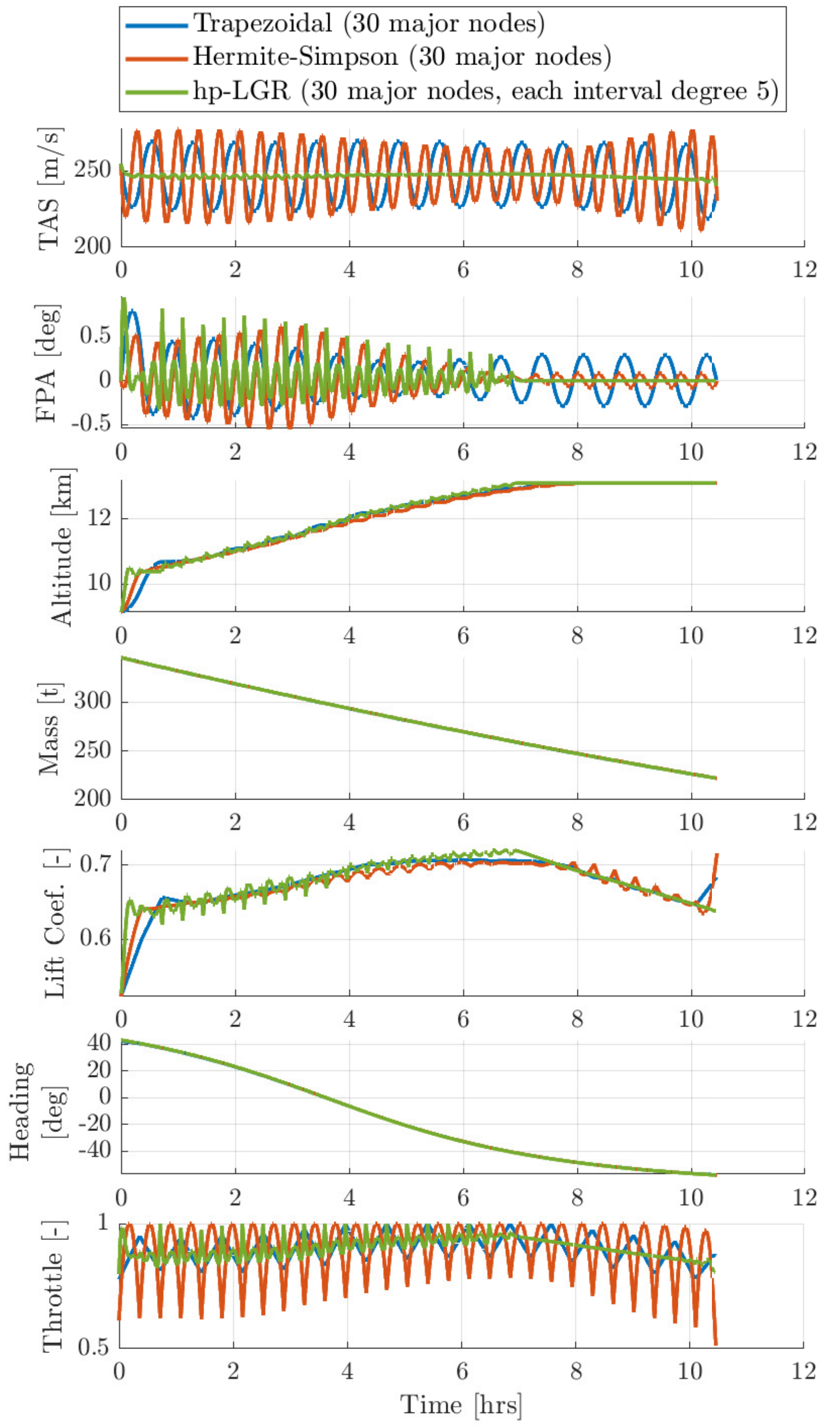


Figure 14-2: Comparison of direct collocation solutions with different discretization methods, for a continuous cruise climb flight

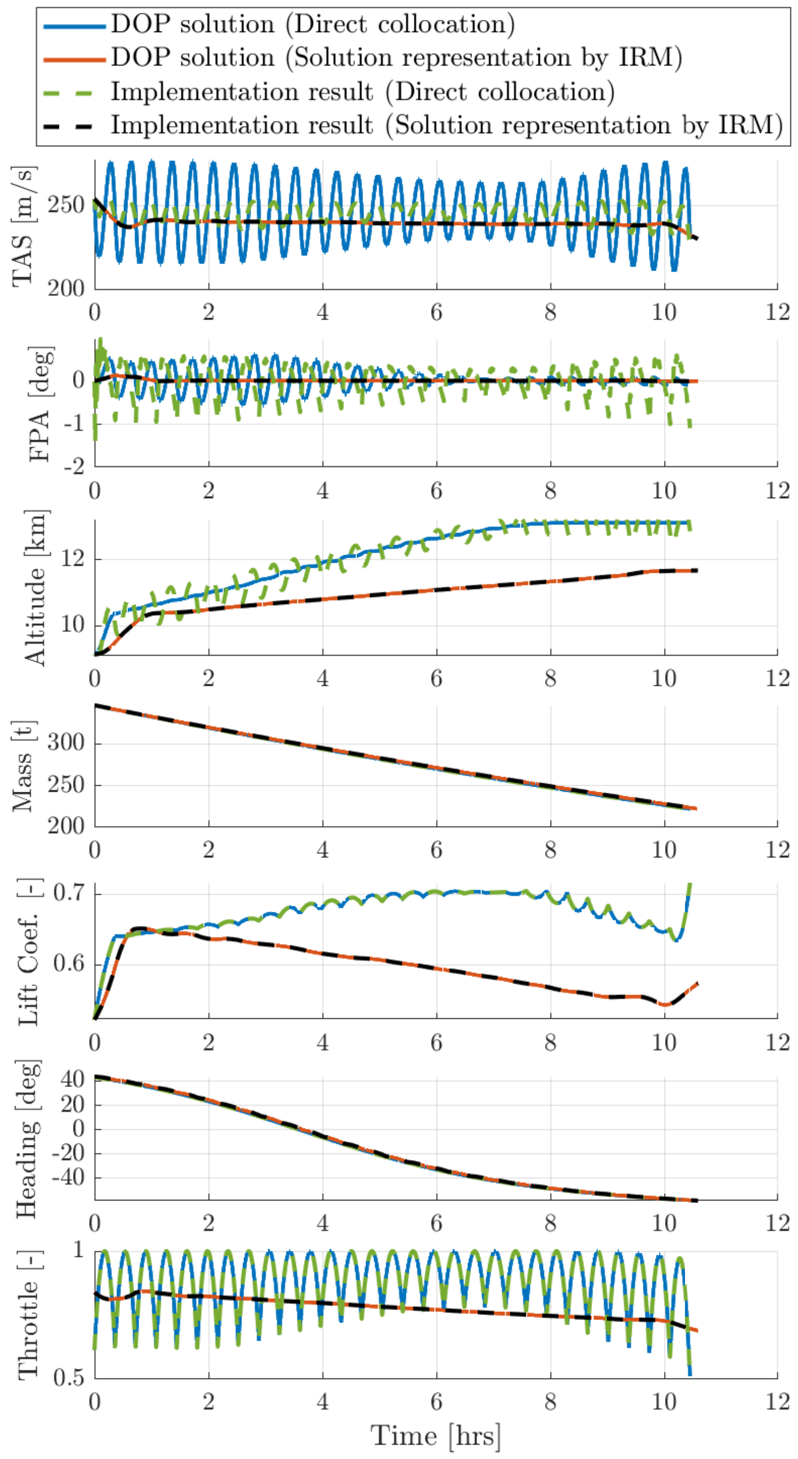


Figure 14-3: Comparison between collocation solution with solution representation by IRM, for a continuous cruise climb flight, transcribed using HS discretization with 30 major nodes. Dashed lines represent the results of an open-loop implementation of the obtained input trajectories with a simulation

14-3 Typical multi-phase formulation representing real-world flight operations

Due to restrictions in air traffic management, the flight profiles of commercial airliners follow a number of protocols. Based on these rules, optimization of commercial flight are often subdivided into a number of operation phases. For example, the Boeing 767-200ER flight from Seattle to Copenhagen [21] uses a 9-phase problem setup and the Airbus A320 flight [186] have 7 phases. Combining all different formulations in literature, and incorporating the continuous cruise climb concept, a typical real-world flight consists of the following 7 phases:

- **Phase 1:** Initial climb from acceleration altitude of 1517 ft, until reaching 10000 ft. This phase is characterized by a constant throttle setting T_s and climbing at a constant calibrated airspeed $V_{CAS} = 250$ kts. The aircraft will maintain this airspeed by adjusting its rate of climb, and for passenger comfort, the the vertical speed is constrained to be $1000 \leq \dot{h} \leq 5000$ ft/min.
- **Phase 2:** Acceleration from 10000 ft. After passing 10000 ft, the aircraft will reduce its rate of climb to $\dot{h} = 1000$ ft/min and allowing the calibrated airspeed to increase from $V_{CAS,0} = 250$ kts to $V_{CAS,f} = 314$ kts. In this short duration, the throttle setting T_s will remain to be a constant value.
- **Phase 3:** High-speed constant CAS climb. Once reaching the target calibrated airspeed $V_{CAS} = 314$ kts, the aircraft will maintain this airspeed with constant throttle setting T_s and by changing the rate of climb $1000 \leq \dot{h} \leq 5000$ ft/min. With constant CAS and the aircraft climbing, the Mach number will gradually increase. This phase finishes when the Mach number of aircraft reaches $M_f = 0.78$.
- **Phase 4:** High Mach climb, until reaching cruise. Once reaching $M_0 = 0.78$, the CAS is disregarded and the aircraft follows certain Mach number profile towards the cruise level at $h > 30000$ ft, again at a constant T_s and climb rate of $1000 \leq \dot{h} \leq 5000$ ft/min. In this formulation, we leave the options open, allowing any trajectory for the finial Mach number M_f to be between 0.78 and the maximum cruise Mach number M_{cr}.
- **Phase 5:** Cruise with continuous climb. The aircraft will cruise with Mach number $0.78 < M < M_{cr}$. Any climbs are restricted with a climb rate of $0 \leq \dot{h} \leq 1000$. In the cruise phase, the throttle setting is allowed to vary between idle and full throttle, i.e. $0.07 \leq T_s \leq 1$.
- **Phase 6:** Descend towards 10000 ft. Although conventionally the initial descend was divided into a number of segments, modern flight management systems now treat this part of flight as an integrated open descend phase where $M < M_{cr}$, $250 \leq V_{CAS} \leq$

$314\,\mathrm{kts}$ and $-1000 \leq \dot{h} \leq -5000\,\mathrm{ft/min}$. At the end of this phase with an altitude of $h_f = 10000\,\mathrm{ft}$, the calibrated airspeed must be reduced to $V_{CAS,f} = 250\,\mathrm{kts}$. This phase also employs variable throttle setting with $0.07 \leq T_s \leq 1$.

- **Phase 7:** Approaching the airport. Below 10000 ft, the aircraft will maintain an airspeed and descend further towards $h_f = 2000\,\mathrm{ft}$ at a vertical speed of $-1000 \leq \dot{h} \leq -5000\,\mathrm{ft/min}$ and a throttle setting of $0.07 \leq T_s \leq 1$.

In-between different phases, continuity constraints are imposed for h, ϑ, ϖ, v_{TAS} and m.

14-4 A simplified 3-phase formulation for commercial flights

Although the 7-phase formulation correctly represents the flight protocols under current air traffic control (ATC) rubric, a relatively large number of phases related to the climb and landing phase of the flight unnecessarily complicates the setup and solution process of the problem. To improve the efficiency in solving large scale problems with multiple aircraft, in this work, the use of a simplified 3-phase formulation is proposed. Regarding the flight time and fuel usage, this solution only results in minor differences when compared to that of the 7-phase formulation. In addition, the 3-phase formulation is also in direct correspondence to the continuous climb operations (CCO) and continuous descend operations (CDO) initiatives that are currently under extensive studies by aviation authorities. Details of the 3-phase formulation are as follows:

- **Phase 1:** Climb
  $h_0 = 1517\,\mathrm{ft}$, $h_f > 30000\,\mathrm{ft}$, $250 \leq V_{CAS} \leq 314\,\mathrm{kts}$, $M_0 \leq M \leq M_{cr}$,
  $0.78 < M_f < M_{cr}$, $1000 \leq \dot{h} \leq 5000\,\mathrm{ft/min}$, $0.07 \leq T_s \leq 1$,

- **Phase 2:** Cruise with gradual climb
  $h > 30000\,\mathrm{ft}$, $0.78 < M < M_{cr}$, $0 \leq \dot{h} \leq 1000\,\mathrm{ft/min}$, $0.07 \leq T_s \leq 1$,

- **Phase 3:** Descend
  $h_f = 2000\,\mathrm{ft}$, $250 \leq V_{CAS} \leq 314\,\mathrm{kts}$, $V_{CAS,f} = 250\,\mathrm{kts}$, $M_0 \leq M \leq M_{cr}$,
  $-1000 \leq \dot{h} \leq -5000\,\mathrm{ft/min}$, $0.07 \leq T_s \leq 1$.

It can be shown that for any feasible solution that fulfils the constraints of the 7-phase formulation, it is also a solution to the 3-phase problem. In other words, the set of feasible solution of the 7-phase problem is a subset of the feasible solution set of the 3-phase problem. The main source of differences would be for trajectories under 10000 ft, with the 7-phase formulation strictly follows the ATC limit of $v_{CAS} = 250\,\mathrm{kts}$.

Table 14-4: Comparison of obtained solutions between the problem formulations

Problem Formulation	Type	Flight Time	Fuel Consumption
7-Phase	Climb	17 min	4.8853×10^3 kg
	Cruise	10 h 02 min	1.1348×10^5 kg
	Descend	20 min	7.3800×10^2 kg
	Full Flight	10 h 39 min	1.1910×10^5 kg
3-Phase	Climb	30 min	7.7870×10^3 kg
	Cruise	9 h 46 min	1.1044×10^5 kg
	Descend	21 min	8.0944×10^2 kg
	Full Flight	10 h 37 min	1.1904×10^5 kg
Differences	Full Flight	-93 s (-0.2%)	-64.6 kg (-0.05%)

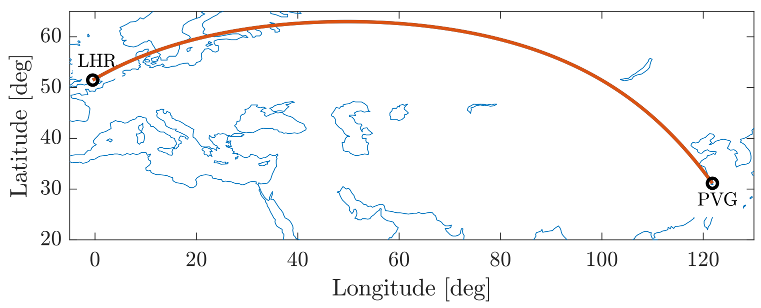
For a transcontinental flight from London to Shanghai listed as FLT1 in Table 15-1, the solutions obtained from both problem formulations are compared. The dynamic optimization problem minimizes the initial weight of the aircraft subject to all above-mentioned constraints plus a terminal constraint that at the end of the mission, the aircraft must be at its minimum allowed weight m_{MIN} defined as

$$m_{MIN} = m_{OEW} + m_{PLD} + 0.05 m_{MFC},$$

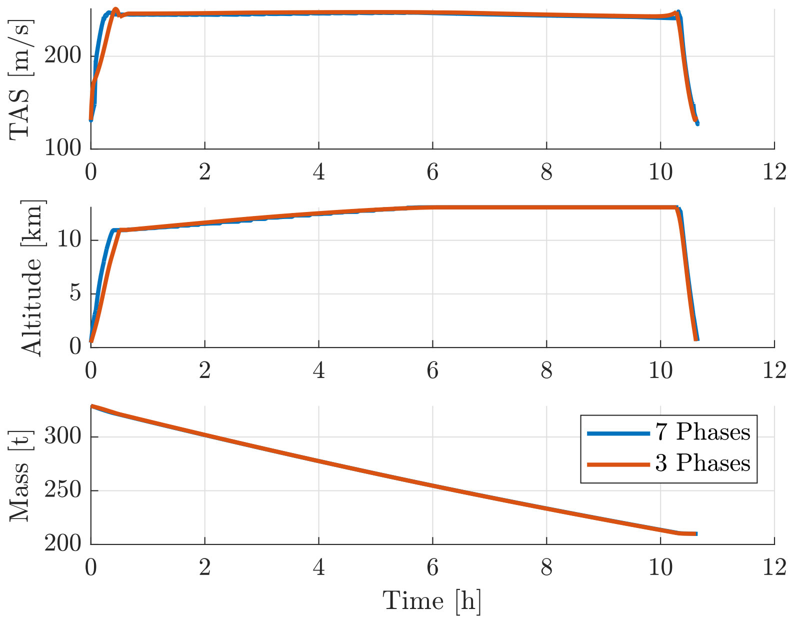
with m_{OEW} the operational empty weight (in kg), m_{PLD} the payload mass and m_{MFC} the maximum fuel capacity (in kg). The fuel-minimum solutions for the flight with both formulation are compared graphically in Figure 14-4 with details collected in Table 14-4.

It can be seen that for a flight of more than 10 hours and fuel consumption of about 119 tons, the simplified 3-phase formulation has only lead to a minor difference of 93 s in flight time and 64.5 kg in fuel usage. The main differences in state variable trajectories are related to the climb phase: while the 3-phase formulation ends up with higher airspeed below 10000 ft, the latter part of the climb is with a lower airspeed increase, ending up with a longer climb.

The increased flight time and fuel usage for the climb phase are compensated with a shorter cruise phase. Eventually, the differences between these two aspects for the complete flight can be considered as negligible. On the other hand, the computation time to reach a solution with accuracy fulfilling requirements listed in Table 14-5 is drastically different. Table 14-6 shows that the total computation time till the end of mesh refinement is 96.6% lower for the 3-phase problem formulation, comparing to its 7-phase counterpart.



(a) Flight trajectories



(b) Flight states

Figure 14-4: Comparison of different problem formulations for optimization of a commercial flight

Table 14-5: Required solution accuracy – flight of commercial airliner

	h [m]	ϑ [rad]	ϖ [rad]	v_{TAS} [m/s]	γ [rad]	m [kg]
Max. Abs. Local Error (η_{tol})	10	0.0175	0.0175	1	0.0175	1
Max. Local Constraint Violation ($\varepsilon_{g_{tol}}$)	10	0.0175	0.0175	1	0.0175	1
	C_L [-]	χ [rad]	Γ [-]	v_{CAS} [m/s]	M [-]	$\dot{h}$ [m/s]
Max. Local Constraint Violation ($\varepsilon_{g_{tol}}$)	0.1	0.0175	0.01	1	0.01	0.5

Table 14-6: Comparison of computation time between the problem formulations

Problem Formulation	**Mesh Refinement Iterations**	**Computation Time of Each Iteration** [s]	**Total Computation Time** [s]
7-Phase	4	[98.6, 302.7, 586.9, 252.9]	1241
3-Phase	2	[23.5, 18.4]	41.9
			Computation Time Reduction: 96.6%

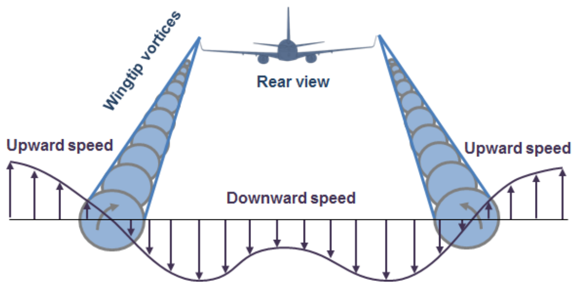
Chapter 15

Formation Flight of Aircraft

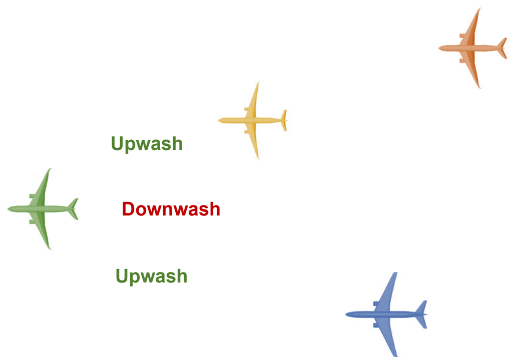
During the flight, circular patterns of rotating air will form behind the wing trailing from the tip, a phenomenon commonly known as wingtip vortices. As shown in Figure 15-1a, the main consequence of tip vortices are the generation of a downwash region, reducing the effective angle of attack and hence responsible for induced drag. Accompanying the downwash region, there will also be two upwash regions that are formed on each side of the aircraft, and travelling downstream together. A wing placed inside the upwash region of the lead aircraft will experience an increase in the effective angle of attack, hence reducing the magnitude of its own induced drag.

When birds embark on long-distance migration flight, they fly in a V-shaped formation to reduce the overall effort. This observation inspires aviation pioneers to explore the possibility for commercial airliners to fly in formation, to benefit from the reduction in drag and hence lower fuel consumption. The latest development is the fello'fly project within Airbus UpNext initiative, aiming to *'prove the technical, operational and economic viability of wake-energy retrieval for commercial aircraft'* [1]. In recent tests [22], two Airbus A350s were tested in formation flight on the transatlantic route, with a separation of just 1.5 nautical miles (around 2.8 km).

Once the concept is proven and technical challenges are resolved, schemes and tools must be developed to assist the integration of formation flying into the operational norm of commercial airlines and air traffic control management system. Recent development using multi-phase trajectory optimization [88, 89, 188] have shown to obtain promising results. Here use of ICLOCS2 in solving such a complex real-world problem is demonstrated.



(a) Tip vortices [46]



(b) Formation flight illustration

Figure 15-1: Illustration of upwash and downwash generation by aircraft tip vortices and relative positioning of aircraft in formation flight

15-1 Problem formulation

In comparison to the 3-phase flight formulation, additional phases will need to be added to accommodate the formation flight. The phase structure for each flight will become *climb, individual cruise, formation flight, individual cruise, descend*, with the formation flight phase shared with all flights. All flights have free initial and final time, with the objective to minimize the total fuel usage for all of the N_{ac} aircraft, i.e.

$$\min_{\mathcal{Z}} \sum_{i_{ac}=1}^{N_{ac}} m_{i_{ac}}(t_0) - m_{i_{ac}}(t_f).$$

with $\mathcal{Z}$ the tuple of decision variables, and subscript i_{ac} representing the aircraft index. When cruising together, the aircraft will need to fly with the same value for state variables and well as their rate of change, except the mass. Hence, only the mass of any additional aircraft $m_{i_{ac}}$ need to be added as state variables in the formation flight phase and no extra input variables needed. The inputs (C_L and Γ) of the trailing aircraft will be computed based on the flight information of the leading aircraft as follows.

First, the system dynamics for the leading aircraft can be computed based on (14-1). Then for each of the trailing aircraft, the lift force can be obtained as

$$L_{i_{ac}}(t) = \dot{\gamma}(t) m_{i_{ac}}(t) v_{TAS}(t) + m_{i_{ac}}(t) g \cos(\gamma(t)),$$

from which the lift coefficient can be obtained by

$$C_{L_{i_{ac}}}(t) = \frac{2L_{i_{ac}}(t)}{\rho_h(t) v_{TAS}^2(t) S_{w,i_{ac}}}.$$

The drag coefficient is therefore

$$C_{D_{i_{ac}}}(t) = C_{D0_{i_{ac}}} + (1 - \varkappa) k_{cl_{i_{ac}}} C_{L_{i_{ac}}}^2(t) + \Delta C_{D,wave_{i_{ac}}}(t),$$

with $\varkappa$ a reduction factor for induced drag by flying in the upwash region of the trailing vortices. In this work, a conservative reduction of $\varkappa = 0.25$ is assumed for all aircraft. In reality, aircraft behind multiple leaders are predicted to enjoy a much higher induced drag reduction, estimated to be around 50% for the third aircraft down the formation sequence [89, 188].

Next, the drag force can be computed using (14-2) for each trailing aircraft, which would lead to an expression for the require thrust

$$T(t) = \dot{v}_{TAS}(t) m_{i_{ac}}(t) + D(t) + m_{i_{ac}}(t) g \sin(\gamma(t)).$$

With the thrust known, the corresponding fuel flow can be computed for each aircraft and implemented as the dynamic equation for the additional state variables of $m_{i_{ac}}$. Moreover, constraints regarding the maximum thrust available T_{max} at different altitude–airspeed combinations and regarding the maximum lift coefficient need to be implemented for compliance with flight performance limits.

15-2 DOP solution of a simulated scenario

To demonstrate the dynamic optimization of an aircraft formation flight problem, a scenario with three transcontinental flights with different origins and destinations is designed, with details collected in Table 15-1. Information regarding the aircraft performance parameters of each aircraft type is presented in Table C-1 in Appendix C. From the nominal direct flight, the most efficient flight in terms of fuel usage is identified to be FLT3 with aircraft Type 3. Hence this flight is selected to be the leading aircraft in the formation flight.

Table 15-1: Details of the three flights in the simulated scenario

Flight ID	**FLT1**	**FLT2**	**FLT3**
Aircraft	Type 1	Type 2	Type 3
Origin Destination	London (LHR) Shanghai (PVG)	Paris (CDG) Wuhan (WUH)	Amsterdam (AMS) Beijing (PKX)
Nominal Flight Time	11 h 20 min	11 h 10 min	9 h 35 min
Payload m_{PLD} [kg]	33000	20000	28000
Minimum Mass m_{MIN} [kg]	209865	147150	175926.15

The main challenges associated with solving the problem numerically are related to

- very long horizon more than 11 hours, while system dynamics specified in seconds, and
- large number of phases: 13 phases are needed for the 3 aircraft formation flight problem, and
- large differences in variables numerical range, from altitudes of several thousands meters, to angles of a few radius.

The problem is transcribed using direct collocation and Hermite-Simpson discretization, with a total number of 290 node points for the initial mesh. The solution obtained at the end of the mesh refinement process are illustrated in Table 15-2 and Figure 15-2—15-5.

After taking off from respective origin airports, the formation of aircraft is formed near the northern Germany–Netherlands border and ends near the Mongolia-China border. The formation flight duration is about 8 h and 13 min, consists of a gradual climb and a constant altitude cruise at the maximum service ceiling of aircraft Type 2 used in FLT2. After the formation flight phase, FLT1 and FLT3 climb to higher altitudes before arriving at their respective destinations.

Due to longer flight routes, slight increases in the flying time for all three flights are reported, around 10 to 20 minutes. Despite a 2% penalty in fuel consumption for the leading aircraft FLT2, the total fuel consumption combining all three flights have seen a 4.31% reduction which equates to 11402 kg of jet fuel.

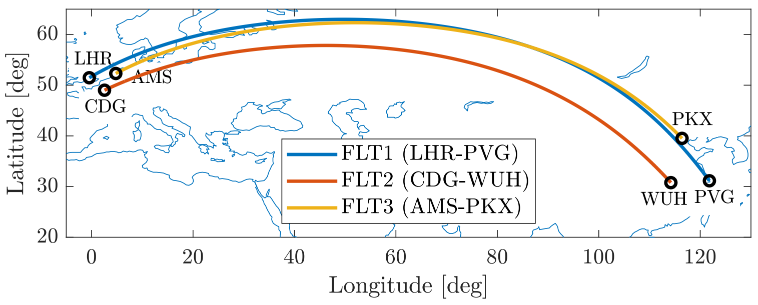
Considering the relatively conservative fuel-saving parameters used, and the number of flights around the world each year, the total savings in fuel consumption and reduction in carbon footprint could be enormous numbers for the aviation industry as a whole.

In practice, the aircraft in the group could take the lead position in turns so that the maximum range for each one of them can be boosted. This would enable ultra-long-haul flights that

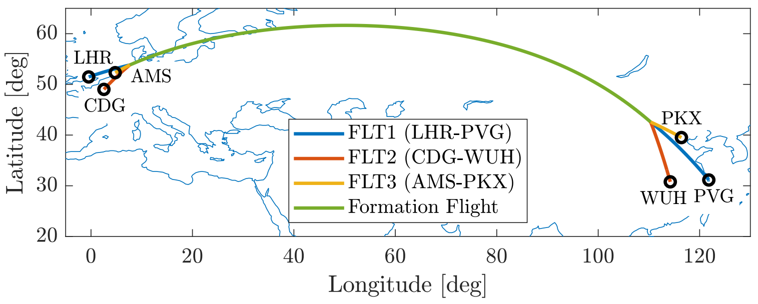
Table 15-2: Comparison of flight time and fuel usage between individual direct flights and formation flight

Flight ID	Route Type	Flight Time	Fuel Consumption
FLT1	Direct	10 h 37 min	1.1904×10^5 kg
	Formation	10 h 58 min (+3%)	1.1072×10^5 kg (-7%)
FLT2	Direct	10 h 34 min	7.5126×10^4 kg
	Formation	10 h 47 min (+4%)	7.0877×10^4 kg (-6%)
FLT3	Direct	9 h 13 min	7.0099×10^4 kg
	Formation	9 h 25 min (+2%)	7.1266×10^4 kg (+2%)
		Total Fuel Savings:	1.1402×10^4 kg (-4.31%)

are currently unachievable economically by today's aircraft, such as the London to Sydney direct route. In short, many trendy and challenging problems in aerospace engineering can be naturally formulated and solved as DOPs, providing non-trivial and efficient solutions to help shape aviation of the future.

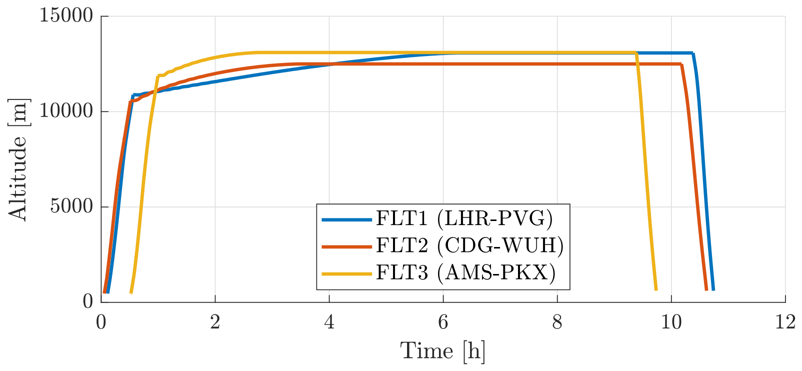


(a) Direct flight

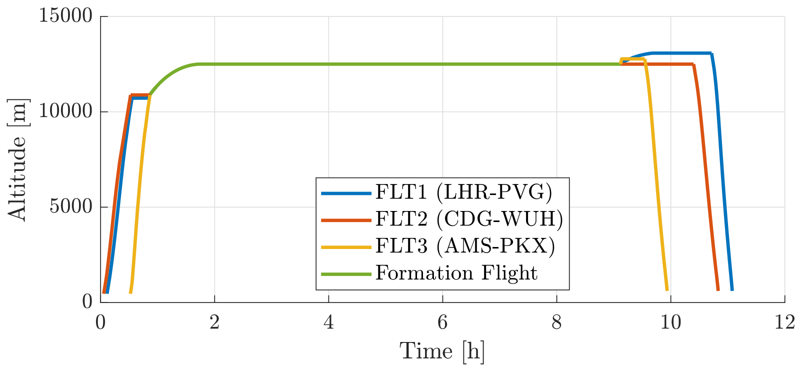


(b) Formation flight

Figure 15-2: Comparison of flight trajectories between individual direct flights and formation flight

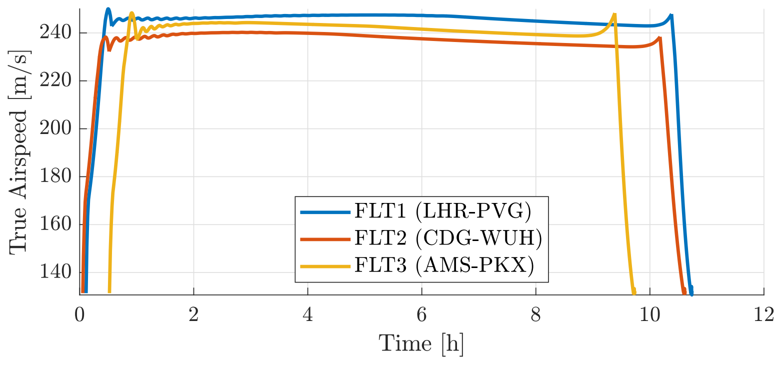


(a) Direct flight

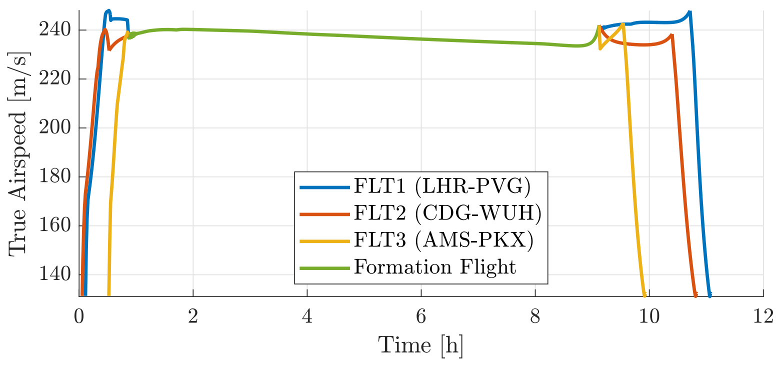


(b) Formation flight

Figure 15-3: Comparison of altitude between individual direct flights and formation flight



(a) Direct flight



(b) Formation flight

Figure 15-4: Comparison of true airspeed between individual direct flights and formation flight

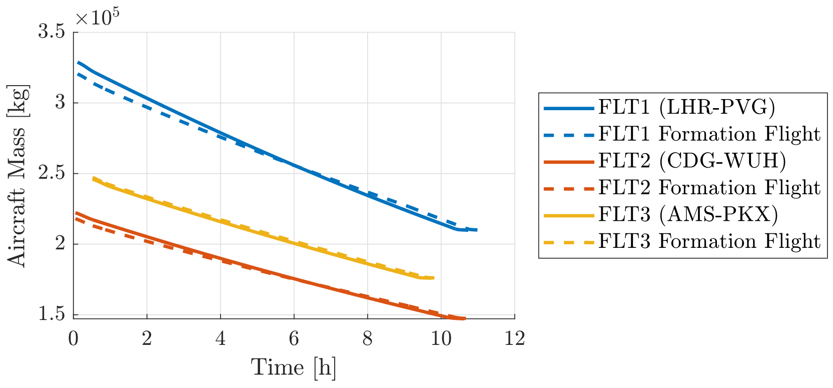


Figure 15-5: Comparison of aircraft mass between individual direct flights and formation flight

Part V

Conclusions and recommendations

Chapter 16

Conclusions and Recommendations

In this work, a number of explorations have been made aiming to numerically solve dynamic optimization problems (DOPs) of different challenging nature with improved accuracy, reliability and efficiency. In the following sections, detailed concluding remarks will be given for the main findings, accompanied by recommendations and suggestions for future work.

16-1 Direct transcription methods based on integrated residuals

Detailed analyses of the direct collocation error characteristics have lead to the investigation of a wider range of numerical methods that are traditionally used in the solution of differential equations. The developments have resulted in a new class of direct methods based on integrated residual minimization (IRM), as well as a new direct transcription method named direct alternating integrated residual (DAIR) for the solution of DOPs.

16-1-1 Concluding remarks

- **Direct collocation transcription using a given discretization mesh provides no guarantee in solution accuracy and constraint satisfaction.**

  For direct collocation, accuracy can only be ensured a posteriori through error analysis and mesh design iterations. When a given coarse mesh is used, for example as commonly found in nonlinear model predictive control (NMPC), the validity of the solution may become questionable with errors arising inside the intervals in-between collocation points, despite solving the nonlinear programming problem (NLP) to negligibly small tolerances. Therefore, a mesh refinement framework must be considered as an indispensable part of a direct collocation method in order to ensure convergence and solution accuracy.

- **In numerical solution of DOPs, approximation errors are generally inevitable hence least-squares criteria may be more suitable.**

  Solutions of the continuous-time DOPs can rarely be represented exactly by the approximating function employed in numerical schemes. According to established theories on function approximations, the least-squares criterion is often considered as a more suitable choice than forcing the fitting error to be exactly zero only at some selected points. Hence, for a given mesh, the least-squares approach of IRM can yield solutions of higher accuracy than direct collocation.

- **Solving the residual minimization problem directly may bring benefits in comparison to the use of necessary optimality conditions.**

  The least-squares approach, as defined in the class of weighted residual methods, is equivalent to applying the optimality conditions and obtaining a number of equality constraints to be satisfied. However, this condition is only necessary and thus theoretically can only guarantee that the trajectory is a stationary solution in general, i.e. the trajectory could be a local maximum for the respective error metric used. In addition, using only the optimality conditions will not be able to provide indications on the magnitudes of the errors. Methods that directly solve the integrated residual minimization problem would have the additional benefit that the evaluation of error magnitudes can be integrated into the solution process instead of a posteriori.

- **Solving DOPs numerically on a given mesh can be considered as a multi-objective optimization problem.**

  For a given discretization mesh, one will inevitably face a trade-off between minimizing the objective for optimality and minimizing the discretization errors for accuracy, forming a Pareto front. Solutions from direct collocation with a given coarse mesh will likely be dominated by other solutions. In contrast, schemes based on IRM have been shown to be capable of directly obtaining a solution on the Pareto front. Moreover, for a given mesh, the proposed direct alternating integrated residuals (DAIR) transcription method is able to directly solve problems to a specified accuracy level, by taking advantage of the multi-objective nature of the optimization problem.

- **IRM-type of methods are more reliable in solving challenging problems, including those with singular arcs and high-index differential algebraic equations.**

For direct collocation, if differential-algebraic equations (DAEs) exist as part of the system dynamics, without special considerations to ensure consistency, the transcription process can lead to over-constrained or under-constrained NLP. This issue could lead to convergence difficulties or singular-arc type of fluctuations in the obtained solution. By allowing arbitrarily small residuals for the constraints to exist during the solution process similar to penalty methods, IRM-based transcriptions can have better convergence properties in cases of high-index DAEs and DAEs that force the solution to lie on a manifold. Also, since the errors can now be ranked for different solution candidates on the singular arc that are indistinguishable from the objective point of view, IRM type of methods are capable of suppressing singular arc fluctuations without the need for additional treatments.

16-1-2 Suggestions for future work

- **Continued efforts required for the IRM-type of methods to mature.**

  Admittedly, the method is still at its very early stage of development. For instance, a detailed exploration of the problem structure and sparsity patterns are needed to improve the computational efficiency of such a scheme. In this work, through elementary analyses of problem structures and derivative information computations, an observation was made that classical finite difference computations required by IRM-type of methods may not be as efficient as that for direct collocation. Hence, efficient implementations of other derivative computation methods such as algorithmic differentiation may be necessary. Continued research on the IRM methods as well as associated tools will be required in order to realise its full potential and to reach the same level of maturity as direct collocation methods.

- **Numerical solutions of DOPs can be further exploited beyond direct collocation and direct multiple shooting.**

  With decades of development, direct methods of direct collocation and direct multiple shooting have almost become the norm for solving practical DOPs, shadowing the indirect approach. However, out in the academic field, especially in relation to numerical solutions of differential equations, many alternatives exist that can be utilized for the enforcement of dynamic equations. Exploiting these methods may lead to developments of new numerical methods for the solution of DOPs with different solution characteristics and computational efficiency.

16-2 Methods to allow efficient computation of large problems

Besides focusing on developments that could become popular approaches for the solution of DOPs in the future, this work also proposes a number of methods that would allow efficient and accurate computation of large and challenging problems, valid also for current transcription methods.

16-2-1 Concluding remarks

- **Directly implement rate constraints on the discretization mesh appears to be an attractive alternative to the conventional intuitive approach.**

  In contrast to conventional approaches, the proposed method is numerically verified not to introduce singular control arcs. Additionally, this on-mesh implementation replaces the additional dynamic equations and nonlinear path constraints in conventional implementations with linear rate equations. Thus, there is no contribution to the Hessian and the contribution to the Jacobian can be pre-computed, enabling faster iterations. Based on observations in Section 9-2-1, the scale of reduction in computation time seems to grow quite quickly with the increase in mesh size (number of collocation points), making the method especially suitable for the solution of large scale problems. A possible downside of the proposed scheme is that the method cannot be directly implemented in most of the existing DOP packages through their current interfaces.

- **Inactive constraints burden the computations but do not contribute to the solution, hence excluding them can lead to benefits.**

  A strategy has been developed to systematically identify and handle inactive constraints and redundant constraint sets for numerically solving DOPs together with mesh refinement schemes. Unlike previous work that would always result in infeasible initial guesses for intermediate steps, the proposed scheme is capable of providing guarantees on the feasibility of initial points in mesh refinement iterations. The method only requires some mild conditions — the original DOP to have feasible points, and the initial solve of the discretized DOP to be successful, making it particularly suitable for DOP toolboxes that utilize interior point method (IPM)-based NLP solvers in lowering the computational cost.

- **Suppression of singular arc fluctuations is crucial for practical engineering problems.**

  The issues of singular arcs are often demonstrated with toy problems in the literature, resulting in them being neglected by engineers working on complex problems. In fact, singular control can easily arise from a wide range of problems ranging from the re-positioning of a double-integrator system, to the cruise flight of an aircraft as demonstrated in this thesis. Since singular arc often only occurs on some parts of the DOP solution trajectories and can be affected by many factors such as the activation status of some constraints, it may not always be possible to address the singular arc fluctuations a posteriori as an afterthought. In this work, different methods that are capable of suppressing the singular arc fluctuations are studied and compared. More importantly, implementations that would avoid the introduction of singular control in certain scenarios are identified to prevent the issue in the first place.

- **For DOP software toolboxes, it is beneficial to provide a wide range of methods under a unified interface.**

  For the solution of DOPs in practice, appropriate choices of transcription scheme, discretization method and the numerical solver all depend on the required level of accuracy, available computational time and resources, and characteristic of the problem formulation (e.g. function smoothness, continuity, differentiability). To date, there is not a single method that would work well for all challenges of different natures. Also, among the ones that are available, it is difficult to know beforehand which choice will work best.

  Many of existing DOP solution software packages require immediate commitments to specific programming languages and numerical schemes. If considered unsuitable later, subsequent changes can lead to an increase in the engineering expense and design time. By providing toolboxes that have access to an array of methods for solving a wider variety of DOPs under a unified interface, the experience required by the user can be reduced making the method available to a wider range of applications.

16-3 DOPs for aerospace applications

By revisiting the history when the solution of DOPs made crucial contributions to flight and orbital trajectory optimizations and mission designs, as well as by reviewing the progress made in model predictive control (MPC) for flight control applications, the future potential for online real-time implementation of nonlinear and non-regulation MPC can be visioned. In this work, a brief study into flight trajectory optimization with continuous climb and descend is provided, followed by a quick investigation into future formation flight of commercial airliners.

16-3-1 Concluding remarks

- **Solution of DOPs directly based on mission specifications can result in unique capabilities and potentials in shaping aviation of the future.**

Solving nonlinear DOPs formulated directly based on mission specifications can yield nontrivial and non-intuitive solutions subject to operational and safety constraints, making it an ideal candidate addressing many aerospace-related challenges. Besides the formation flight problem demonstrated in this thesis, many other challenges were identified and investigated in the course of this research project, including aircraft upset recovery, multiple unmanned aerial vehicles (UAVs) in-range tracking, joint optimization of transmission and propulsion in UAV-assisted communication networks. By formulating and solving the problems as DOPs, solutions that significantly outperform heuristic methods can be obtained to help shape the future of aerospace engineering.

- **Methods that can provide efficient and reliable solve of DOPs of different challenging nature would be required to address aerospace challenges.**

  Shown with a relatively simple continuous cruise climb problem, when solved directly according to mission specifications, the current method of direct collocation is not capable of yielding a solution of high accuracy without singular arc fluctuations. Therefore, alternative methods that can provide efficient and reliable solutions of DOPs of different challenging nature would be required. IRM-methods have shown to be an option with great potentials.

16-3-2 Suggestions for future work

- **In future developments, successfully demonstrating the capabilities and the benefits of solving nonlinear DOPs is as important as finding proves for stability and convergence.**

  Admittedly, the aviation sector at the current moment still maintains a skeptical view for solving and implementing nonlinear and non-regulation (i.e. minimum energy or minimum time) DOP solutions online, largely due to the associated computational complexity and lack of sufficiently general proofs regarding the convergence of numerical solvers and stability in closed-loop implementations.

  It is crucially important to first able to successfully demonstrate the capabilities and the benefits such new technology can bring, which is still at its very early stage of development. If such benefits do not exist, or it is only marginal compared to currently used proven technology, the development may not yield sufficient impact. This is also the reason why the decision was made early for this work to focus on the non-regulation type of problems, as experiences have shown nonlinear MPC for the regulation type of tasks (stabilization and set-point tracking) often only offer marginal benefits in comparison to the linear MPC, while at the same time losing many of the theoretical proofs on properties such as stability.

Once the technical challenges have been dealt with, and a good performance is seen for the most majority of problems tested, then it would be easier to address the concerns and develop formal proofs on conditions of stability. This is not only thanks to the insights gained by analyzing the implementations that were successful, but also because methods that have shown to work well in practice will attract more experts to join in to tackle the challenges with stronger motivations.

The unique advantages of MPC in fault-tolerant flight controls attracted lots of attentions very early on in the development process. These attentions are helpful later in bring the technology to a mature state, allowing real-time computations on embedded platforms. Similarly, the popularity of artificial intelligence and machine learning are also greatly due to the substantial benefits that are obtainable for a wide range of applications, using intuitive implementations. Therefore, demonstrating unique capabilities and the benefits of solving nonlinear DOPs formulated directly based on mission specifications will be crucial for the future popularity and success of this technology.

Appendices

Appendix A

Legendre Polynomials and Legendre-Gauss-Radau Points

The Legendre polynomials are the solutions to the Legendre's differential equation

$$\frac{d}{dt}\left[(1-t^2)\frac{d}{dt}P_N(t)\right] + N(N+1)P_N(t) = 0, \tag{A-1}$$

with $P_N(t)$ the Legendre polynomials of N^{th} degree. The solutions are given as

$$\begin{aligned} P_0(t) &= 1, \\ P_1(t) &= t, \end{aligned}$$

and subsequently following the recurrence relationship

$$(N+1)P_{N+1}(t) = (2N+1)tP_N(t) - NP_{N-1}(t). \tag{A-2}$$

A-1 Computation of LGR points

The Legendre-Gauss-Radau (LGR) points are the roots of the polynomial ($P_N(t)+P_{N-1}(t) = 0$), defined only on the interval [-1,1), excluding the end point at 1. Directly solving for the LGR points as a root finding problem is computationally intensive especially for polynomials of higher orders.

To demonstrate the procedures for a numerically more efficient calculation, first reorder the recurrence relationship in the descending order for N, and note the following trend:

$$tP_N(t) = \frac{N+1}{2N+1}P_{N+1}(t) + \frac{N}{2N+1}P_{N-1}(t), \tag{A-3}$$

$$tP_{N-1}(t) = \frac{N}{2N-1}P_N(t) + \frac{N-1}{2N-1}P_{N-2}(t), \tag{A-4}$$

$$tP_{N-2}(t) = \frac{N-1}{2N-3}P_{N-1}(t) + \frac{N-2}{2N-3}P_{N-3}(t), \tag{A-5}$$

$$tP_{N-3}(t) = \frac{N-2}{2N-5}P_{N-2}(t) + \frac{N-3}{2N-5}P_{N-4}(t), \tag{A-6}$$

Thus we can rewrite the recurrence in matrix format

$$t\begin{bmatrix} P_1+P_0 \\ P_2+P_1 \\ \vdots \\ P_{N-2}+P_{N-3} \\ P_{N-1}+P_{N-2} \end{bmatrix} = \begin{bmatrix} a_0 & c_0 & 0 & \dots & 0 \\ b_1 & a_1 & c_1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & \dots & b_{N-3} & a_{N-3} & c_{N-3} \\ 0 & 0 & \dots & b_{N-2} & a_{N-2} \end{bmatrix} \begin{bmatrix} P_1+P_0 \\ P_2+P_1 \\ \vdots \\ P_{N-2}+P_{N-3} \\ P_{N-1}+P_{N-2} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ d_{N-2}(P_N+P_{N-1}) \end{bmatrix}, \tag{A-7}$$

or

$$tP = TP + d_{N-2}(P_N + P_{N-1})e_{N-2}, \tag{A-8}$$

with for example

$$\begin{aligned} d_{N-2} &= \frac{N}{2N-1}, \quad a_{N-2} = \frac{1}{(2N-3)(2N-1)}, \quad b_{N-2} = \frac{N-2}{2N-3}, \\ c_{N-3} &= \frac{N-1}{2N-3}, \quad a_{N-3} = \frac{1}{(2N-5)(2N-3)}, \quad b_{N-3} = \frac{N-3}{2N-5}. \end{aligned} \tag{A-9}$$

Since LGR points are the roots of $P_N(t) + P_{N-1}(t) = 0$, the last vector becomes zero and the matrix system becomes effectively $tP = TP$, meaning t is equal to the eigenvalues of matrix T. With numerical computation of eigenvalues for symmetric matrices being more efficient, and the eigenvalues are always preserved by a similarity transformation, it is possible to symmetrize the matrix T as

$$J = DTD^{-1} = \begin{bmatrix} a_0 & \bar{b}_1 & 0 & \dots & 0 \\ \bar{b}_1 & a_1 & \bar{b}_2 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & \dots & \bar{b}_{N-3} & a_{N-3} & \bar{b}_{N-2} \\ 0 & 0 & \dots & \bar{b}_{N-2} & a_{N-2} \end{bmatrix}, \tag{A-10}$$

with, for example,

$$\bar{b}_{N-2} = \frac{\sqrt{(N-2)(N-1)}}{2N-3}. \tag{A-11}$$

The LGR points t can therefore be obtained by the eigenvalue computation of J, avoiding the complication of root finding.

A-2 Computation of LGR weights

For estimation of integrals using Radau quadrature in the form of

$$\int_{-1}^{1} f(t)dt \approx \sum_{k=1}^{N} w_k f(t_k), \tag{A-12}$$

the LGR weights are calculated as

$$\begin{aligned} w_1 &= \frac{2}{N^2}, \\ w_i &= \frac{1}{(1-t_i)\dot{P}_{N-1}^2(t_i)} \quad , \text{ for } i = 2, \dots, N, \end{aligned} \tag{A-13}$$

or equivalently [183] by

$$w_i = \frac{1}{N^2}\frac{1-t_i}{P_{N-1}^2(t_i)} \quad , \text{ for } i = 1, \dots, N. \tag{A-14}$$

Appendix B

List of Methods Available in ICLOCS2

Table B-1: Methods available in ICLOCS2

Transcription Method	Direct collocation	Direct integrated residual minimization (experimental)* (penalty barrier method & alternating method)		
Discretization Method	Euler	Trapezoidal	Hermite-Simpson	Legendre-Gauss-Radau*
Solver Supported	IPOPT [205]	fmincon [137]	WORHP* [37]	OSQP* (for QP/LP) [187]
Jacobian Computations	User defined	Sparse finite differences	Algorithmic differentiation (forward) with Adigator*[209]	
Hessian Computations	User defined	Sparse finite differences	Algorithmic differentiation with Adigator*	Limited memory BFGS with IPOPT
Solution Representation	User defined*	In accordance with discretization scheme	Using the method of integrated residual minimization*[156]	
Meshing Strategy	Fixed mesh	Automatic mesh refinement*		Flexible *hp* mesh*
Extra Functionalities	Automatic scaling*	Automatic handling of regularization parameters*	Improved DAE support*	Implementation simulations*
	Closed-loop Simulations*	Automatic selection of discretization method*	Support for multi-phase and hybrid-dynamics problems*	
	Efficient implementation of rate constraints*[157]		External constraint handling* [155]	

* New functionalities compared to the previous version [59].

Appendix C

Aircraft Performance Parameters in Formation Flight Example Problem

Table C-1: Aircraft performance parameters

Parameters		**Aircraft Type 1**	**Aircraft Type 2**	**Aircraft Type 3**
Aerodynamics & Flight Parameters	$S_{w,ac}$ [m^2]	436.8	361.6	442
	C_{D0} [-]	0.037	0.029	0.029
	k_{cl} [-]	0.046	0.044	0.041
	M_{cr} [-]	0.84	0.82	0.85
	$V_{CAS,cr}$ [m/s]	148	129	131
	M_{crit} [-]	0.646	0.642	0.646
	Λ_{sw} [°]	31.6	30	31.9
	h_{ceil} [m]	13100	12500	13100
Mass Parameters	m_{MTOW} [kg]	351500	230000	280000
	m_{OEW} [kg]	167800	120200	142400
	m_{MLW} [kg]	251300	182000	205000
	M_{MFC} [kg]	181300	139000	110523
Engine Parameters	T_0 [N]	513900	230000	379000
	c_{ff1} [-]	4.38063	2.77093	3.20896
	c_{ff2} [-]	-2.47085	-1.04299	-1.87934
	c_{ff3} [-]	2.79032	1.26557	1.49836

Bibliography

[1] Airbus. Airbus upnext - innovation ecosystem, Jul 2020. URL https://www.airbus.com/innovation/innovation-ecosystem/airbus-upnext.html.

[2] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit. Modeling and optimization with optimica and jmodelica. org-languages and tools for solving large-scale dynamic optimization problems. *Computers & Chemical Engineering*, 34(11): 1737–1749, 2010.

[3] G. M. Aly. The computation of optimal singular control. *International Journal of Control*, 28(5):681–688, 1978.

[4] G. M. Aly. The computation of optimal singular control. *International Journal of Control*, 28(5):681–688, 1978. doi: 10.1080/00207177808922489. URL https://doi.org/10.1080/00207177808922489.

[5] G. M. Aly and W. C. Chan. Application of a modified quasilinearization technique to totally singular optimal control problems. *International Journal of Control*, 17(4): 809–815, 1973. doi: 10.1080/00207177308932423. URL https://doi.org/10.1080/00207177308932423.

[6] D. Angeli, R. Amrit, and J. B. Rawlings. On average performance and stability of economic model predictive control. *IEEE Transactions on Automatic Control*, 57(7): 1615–1626, 2012.

[7] Astos Solutions. Astos solutions - products. https://www.astos.de/products, Feb 2017.

[8] Astos Solutions GmbH. Gesop - graphical environment for simulation and optimization. https://www.astos.de/products/gesop/details, 2017. URL https://www.astos.de/products/gesop/details. Retrieved Jan 5, 2017.

[9] B. Açıkmeşe, J. M. Carson, and L. Blackmore. Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem. *IEEE Transactions on Control Systems Technology*, 21(6):2104–2113, 2013.

[10] M. Bangura and R. Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.

[11] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, 1(1):1–16, 1972.

[12] V. M. Becerra. Solving complex optimal control problems at no cost with psopt. In *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pages 1391–1396. IEEE, 2010.

[13] N. S. Bedrossian, S. Bhatt, W. Kang, and I. M. Ross. Zero-propellant maneuver guidance. *IEEE Control Systems Magazine*, 29(5):53–73, 2009.

[14] R. Bellman and R. Karush. *Dynamic Programming: A Bibliography of Theory and Application*. Memorandum (Rand Corporation). Rand Corporation, 1964.

[15] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004. doi: 10.1137/S0036144502417715. URL https://doi.org/10.1137/S0036144502417715.

[16] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[17] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming: Second Edition*. Advances in Design and Control. Society for Industrial and Applied Mathematics, 2010. ISBN 9780898718577.

[18] J. T. Betts. *Sparse optimization suite (SOS)*. Applied Mathematical Analysis, LLC., 2013.

[19] J. T. Betts. A collection of optimal control test problems, 2015.

[20] J. T. Betts. Optimal low–thrust orbit transfers with eclipsing. *Optimal Control Applications and Methods*, 36(2):218–240, 2015.

[21] J. T. Betts and E. J. Cramer. Application of direct transcription to commercial aircraft trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 18(1):151–159, 1995.

[22] H. Bewicke. Airbus tests a350 bird-like formation flights to save fuel, Mar 2020. URL https://simpleflying.com/airbus-tests-formation-flights/.

[23] R. T. Birge and J. W. Weinberg. Least-squares' fitting of data by means of polynomials. *Reviews of Modern Physics*, 19(4):298–360, 1947. ISSN 0034-6861.

[24] L. Blackmore. Autonomous precision landing of space rockets. *Winter Bridge on Frontiers of Engineering*, 4(46), Dec 2016.

[25] P. Bochev and M. Gunzburger. Least-squares finite-element methods for optimization and control problems for the stokes equations. *Computers & Mathematics with Applications*, 48(7):1035 – 1057, 2004. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2004.10.004.

[26] P. Bochev and M. D. Gunzburger. Least-squares finite element methods for optimality systems arising in optimization and control problems. *SIAM journal on numerical analysis*, 43(6):2517–2543, 2006.

[27] P. B. Bochev and M. D. Gunzburger. Finite element methods of least-squares type. *SIAM review*, 40(4):789–837, 1998.

[28] P. B. Bochev and M. D. Gunzburger. *Least-squares finite element methods*. Springer, 2006.

[29] M. Bodson. Identification with modeling uncertainty and reconfigurable control. In *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, pages 2242–2247. IEEE, 1993.

[30] P. T. Boggs and J. W. Tolle. Sequential quadratic programming for large-scale nonlinear optimization. *Journal of Computational and Applied Mathematics*, 124(1):123 – 137, 2000. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(00)00429-5. URL http://www.sciencedirect.com/science/article/pii/S0377042700004295. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[31] F. Bonnans, P. Martinon, and E. Trélat. Singular arcs in the generalized goddard's problem. *Journal of Optimization Theory and Applications*, 139(2):439–461, Nov 2008. ISSN 1573-2878. doi: 10.1007/s10957-008-9387-1. URL https://doi.org/10.1007/s10957-008-9387-1.

[32] A. E. Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[33] A. E. Bryson, Jr, M. N. Desai, and W. C. Hoffman. Energy-state approximation in performance optimization of supersonicaircraft. *Journal of Aircraft*, 6(6):481–488, 1969.

[34] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 1: Necessary conditions. *Journal of Optimization Theory and Applications*, 70(1):1–23, 1991.

[35] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 2: Multiple shooting and homotopy. *Journal of Optimization Theory and Applications*, 70(2):223–254, 1991.

[36] R. Bulirsch, E. Nerz, H. J. Pesch, and O. von Stryk. Combining direct and indirect methods in optimal control: Range maximization of a hang glider. In *Optimal control*, pages 273–288. Springer, 1993.

[37] C. Büskens and D. Wassel. The esa nlp solver worhp. In G. Fasano and J. D. Pintér, editors, *Modeling and Optimization in Space Engineering*, volume 73, pages 85–110. Springer New York, 2013. doi: 10.1007/978-1-4614-4469-5_4. URL http://dx.doi.org/10.1007/978-1-4614-4469-5_4.

[38] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

[39] S. Campbell and P. Kunkel. Solving higher index dae optimal control problems. *Numerical Algebra, Control & Optimization*, 6(4):447, 2016.

[40] C. Canuto, M. Y. Hussaini, A. M. Quarteroni, A. Thomas Jr, et al. *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.

[41] E. Cheynet. Generalized seir epidemic model (fitting and computation). https://www.github.com/ECheynet/SEIR, 2020. URL https://www.github.com/ECheynet/SEIR. Retrieved April 19, 2020.

[42] H. Chung, E. Polak, and S. Sastry. An external active-set strategy for solving optimal control problems. *IEEE Transactions on Automatic Control*, 54(5):1129–1133, 2009.

[43] M. Chyba, E. Hairer, and G. Vilmart. The role of symplectic integrators in optimal control. *Optimal control applications and methods*, 30(4):367–382, 2009.

[44] O. Cots, J. Gergaud, and D. Goubinat. Direct and indirect methods in optimal control with state constraints and the climbing trajectory of an aircraft. *Optimal Control Applications and Methods*, 39(1):281–301, 2018.

[45] T. Cunis, D. Liao-McPherson, J. Condomines, L. Burlion, and I. Kolmanovsky. Economic model-predictive control strategies for aircraft deep-stall recovery with stability guarantees′. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 157–162, 2019.

[46] Ddeakpeti. Lifting-line model of a finite-span wing (wingtip vortices), Jun 2015. URL http://petersengineering.blogspot.com/2015/06/.

[47] F. de Almeida and D. Leißling. Fault-tolerant model predictive control with flight test results on attas. In *AIAA Guidance, Navigation, and Control Conference*, page 5621, 2009.

[48] *Performance Model Fokker 50*. Delft University of Technology, 2010.

[49] L. Deori, S. Garatti, and M. Prandini. A model predictive control approach to aircraft motion control. In *2015 American Control Conference (ACC)*, pages 2299–2304. IEEE, 2015.

[50] M. Diehl, H. G. Bock, and J. P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on control and optimization*, 43(5):1714–1736, 2005.

[51] M. Diehl, R. Amrit, and J. B. Rawlings. A lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control*, 56(3):703–707, 2011.

[52] I. S. Duff. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):118–144, 2004.

[53] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray. Model predictive control of a thrust-vectored flight control experiment. *IFAC Proceedings Volumes*, 35(1):355–360, 2002.

[54] W. R. Dyksen, E. N. Houstis, R. E. Lynch, and J. R. Rice. The performance of the collocation and galerkin methods with hermite bi-cubics. *SIAM Journal on Numerical Analysis*, 21(4):695–715, 1984. ISSN 00361429. URL http://www.jstor.org/stable/2157003.

[55] R. Dyrska, , and M. Mönnigmann. Accelerating nonlinear model predictive control by constraint removal. In *Proc. 6th IFAC Conference on Nonlinear Model Predictive Control*, 2018.

[56] Elissar Global. Dido optimal control software, 2020. URL http://www.elissarglobal.com/industry/products/software-3/. Lastchecked: 2020-05-12.

[57] Embotech. Forces pro code generator, 2017. URL https://www.embotech.com/FORCES-Pro.

[58] F. Fahroo and I. M. Ross. Advances in pseudospectral methods for optimal control. In *AIAA guidance, navigation and control conference and exhibit*, page 7309, 2008.

[59] P. Falugi, E. C. Kerrigan, and E. Van Wyk. *Imperial college london optimal control software user guide (ICLOCS)*. Department of Electrical and Electronic Engineering, Imperial College London, London, England, UK, 2010.

[60] O. J. Faqir, Y. Nie, E. C. Kerrigan, and D. Gündüz. Energy-efficient communication in mobile aerial relay-assisted networks using predictive control. *IFAC-PapersOnLine*, 51(20):197 – 202, 2018. ISSN 2405-8963. doi: 10.1016/j.ifacol.2018.11.013. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[61] A. V. Fiacco and G. P. McCormick. Programming under nonlinear constraints by unconstrained minimization: a primal-dual method. Technical report, Research Analysis Corp Mclean VA, 1963.

[62] A. V. Fiacco and G. P. McCormick. The sequential unconstrained minimization technique for nonlinear programing, a primal-dual method. *Management Science*, 10(2):360–366, 1964.

[63] A. V. Fiacco and G. P. McCormick. Extensions of SUMT for nonlinear programming: equality constraints and extrapolation. *Management Science*, 12(11):816–828, 1966.

[64] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1990. ISBN 9781611971316. URL https://books.google.co.uk/books?id=icWjpwgigkAC.

[65] A. Fleming, P. Sekhavat, and I. Ross. Minimum-time reorientation of an asymmetric rigid body. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7012, 2008.

[66] C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics: Specific Techniques for Different Flow Categories*. Springer Series in Computational Physics. Springer Berlin Heidelberg, 2012. ISBN 9783642970719. URL https://books.google.co.uk/books?id=rcT3CAAAQBAJ.

[67] F. A. C. C. Fontes and L. T. Paiva. Guaranteed constraint satisfaction in continuous-time control problems. *IEEE Control Systems Letters*, 3(1):13–18, 2019.

[68] F. A. C. C. Fontes and L. T. Paiva. Guaranteed constraint satisfaction in continuous-time control problems. *IEEE Control Systems Letters*, 3(1):13–18, Jan 2019. ISSN 2475-1456. doi: 10.1109/LCSYS.2018.2849853.

[69] B. Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

[70] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM review*, 44(4):525–597, 2002.

[71] L. Gao, B. Shi, M. Kleeberger, and J. Fottner. Optimal control of the hydraulic actuated boom system based on port-hamiltonian formulation. In *12th International Fluid Power Conference*, pages 489–498, 2020.

[72] J. García-Heras, M. Soler, and F. J. Sáez. A comparison of optimal control methods for minimum fuel cruise at constant altitude and course with fixed arrival time. *Procedia Engineering*, 80:231 – 244, 2014. ISSN 1877-7058. doi: https://doi.org/10.1016/j.proeng.2014.09.083. URL http://www.sciencedirect.com/science/article/pii/S187770581401176X. 3rd International Symposium on Aircraft Airworthiness (ISAA 2013).

[73] D. Garg, M. A. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington. An overview of three pseudospectral methods for the numerical solution of optimal control problems. *Advances in the Astronautical Sciences*, 135(1):475–487, 2009.

[74] A. A. Gholampour and M. Ghassemieh. Nonlinear structural dynamics analysis using weighted residual integration. *Mechanics of Advanced Materials and Structures*, 20(3):199–216, 2013. doi: 10.1080/15376494.2011.584146. URL https://doi.org/10.1080/15376494.2011.584146.

[75] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli. The Control Toolbox - an open-source C++ library for robotics, optimal and model predictive control. https://adrlab.bitbucket.io/ct, 2018. arXiv:1801.04290 [cs.RO].

[76] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[77] R. H. Goddard. A method of reaching extreme altitudes, 1920.

[78] J. Gondzio. Crash start of interior point methods. *European Journal of Operational Research*, 255(1):308–314, 2016.

[79] S. Gros, R. Quirynen, and M. Diehl. Aircraft control based on fast non-linear mpc & multiple-shooting. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 1142–1147. IEEE, 2012.

[80] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl. From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1):62–80, 2020.

[81] L. Grüne. Economic receding horizon control without terminal constraints. *Automatica*, 49(3):725–734, 2013.

[82] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2010. ISBN 9783642052217.

[83] E. Hairer, S. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2013. ISBN 9783662126073.

[84] R. Hallouzi and M. Verhaegen. Fault-tolerant subspace predictive control applied to a boeing 747 model. *Journal of guidance, control, and dynamics*, 31(4):873–883, 2008.

[85] A. W. A. Hammad, D. Rey, A. Bu-Qammaz, H. Grzybowska, and A. Akbarnezhad. Mathematical optimization in enhancing the sustainability of aircraft trajectory: A review. *International Journal of Sustainable Transportation*, 14(6):413–436, 2020.

[86] W. E. Hart, C. Laird, J. P. Watson, and D. L. Woodruff. *Pyomo – Optimization Modeling in Python*. Springer Optimization and Its Applications. Springer New York, 2012. ISBN 9781461432258. URL https://books.google.co.uk/books?id=CW19kgEACAAJ.

[87] S. Hartjes and H. G. Visser. Optimal control approach to helicopter noise abatement trajectories in nonstandard atmospheric conditions. *Journal of Aircraft*, 56(1):43–52, 2019.

[88] S. Hartjes, M. E. G. van Hellenberg Hubar, and H. G. Visser. Multiple-phase trajectory optimization for formation flight in civil aviation. *CEAS Aeronautical Journal*, 10(2): 453–462, 2019.

[89] S. Hartjes, H. G. Visser, and M. E. G. van Hellenberg Hubar. Trajectory optimization of extended formation flights for commercial aviation. *Aerospace*, 6(9):100, 2019.

[90] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides. Predictive control of a boeing 747 aircraft using an fpga. *IFAC Proceedings Volumes*, 45(17):80 – 85, 2012. ISSN 1474-6670. doi: https://doi.org/10.3182/20120823-5-NL-3013.00016. URL http://www.sciencedirect.com/science/article/pii/S1474667016314318. 4th IFAC Conference on Nonlinear Model Predictive Control.

[91] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides. Predictive control using an fpga with application to aircraft control. *IEEE Transactions on Control Systems Technology*, 22(3):1006–1017, 2013.

[92] A. M. Hawkins. Constrained trajectory optimization of a soft lunar landing from a parking orbit. Master's thesis, Massachusetts Institute of Technology, 2005.

[93] J. D. Hedengren, R. A. Shishavan, K. M. Powell, and T. F. Edgar. Nonlinear modeling, estimation and predictive control in APMonitor. *Computers & Chemical Engineering*, 70:133 – 148, 2014. ISSN 0098-1354. doi: http://dx.doi.org/10.1016/j.compchemeng.2014.04.013. URL http://www.sciencedirect.com/science/article/pii/S0098135414001306. Manfred Morari Special Issue.

[94] S. Heise and J. Maciejowski. Model predictive control of a supermaneuverable aircraft. In *Guidance, Navigation, and Control Conference*, page 3768, 1996.

[95] H. Hermes and G. Haynes. On the nonlinear control problem with control appearing linearly. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(2):85–108, 1963.

[96] T. R. Hopkins and R. Wait. A comparison of numerical methods for the solution of quasilinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 9(2):181 – 190, 1976. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(76)90060-8. URL http://www.sciencedirect.com/science/article/pii/0045782576900608.

[97] T. R. Hopkins and R. Wait. A comparison of galerkin, collocation and the method of lines for partial differential equations. *International Journal for Numerical Methods in Engineering*, 12(7):1081–1107, 1978.

[98] G. Horn, S. Gros, and M. Diehl. Numerical trajectory optimization for airborne wind energy systems described by high fidelity aircraft models. In *Airborne wind energy*, pages 205–218. Springer, 2013.

[99] B. Houska, H. J. Ferreau, and M. Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[100] Huckleberry Febbo. Nloptcontrol.jl, 2017. URL https://github.com/JuliaMPC/NLOptControl.jl. Retrieved 2017-06-17.

[101] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Civil and Mechanical Engineering. Dover Publications, 2003. ISBN 9780486411811. URL https://books.google.co.uk/books?id=E9IoAwAAQBAJ.

[102] M. Huzmezan and J. M. Maciejowski. Reconfiguration and scheduling in flight using quasi-lpv high-fidelity models and mbpc control. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, volume 6, pages 3649–3653 vol.6, Jun 1998. doi: 10.1109/ACC.1998.703294.

[103] D. Ingole, M. Kvasnica, H. De Silva, and J. Gustafson. Reducing memory footprints in explicit model predictive control using universal numbers. *IFAC-PapersOnLine*, 50(1):11595 – 11600, 2017. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2017.08.1518. URL http://www.sciencedirect.com/science/article/pii/S2405896317321018. 20th IFAC World Congress.

[104] International Civil Aviation Organization. The world of air transport in 2018, 2018. URL https://www.icao.int/annual-report-2018/Pages/the-world-of-air-transport-in-2018.aspx.

[105] International Civil Aviation Organization. Icao aircraft engine emissions databank, Mar 2020. URL https://www.easa.europa.eu/domains/environment/icao-aircraft-engine-emissions-databank.

[106] D. Joosten, T. Boom, and T. Lombaerts. Computationally efficient use of mpc and dynamic inversion for reconfigurable flight control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7431, 2008.

[107] M. Jost, G. Pannocchia, and M. Mönnigmann. Accelerating tube-based model predictive control by constraint removal. In *54th IEEE Conference on Decision and Control (CDC)*, pages 3651–3656, 2015.

[108] M. Jost, G. Pannocchia, and M. Mönnigmann. Constraint removal in linear mpc: An improved criterion and complexity analysis. In *European Control Conference (ECC)*, pages 752–757. IEEE, 2016.

[109] R. Kadali, B. Huang, and A. Rossiter. A data driven subspace approach to predictive controller design. *Control engineering practice*, 11(3):261–278, 2003.

[110] M. M. Kale and A. J. Chipperfield. Reconfigurable flight control strategies using model predictive control. In *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*, pages 43–48. IEEE, 2002.

[111] M. M. Kale and A. J. Chipperfield. Stabilized mpc formulations for robust reconfigurable flight control. *Control Engineering Practice*, 13(6):771–788, 2005.

[112] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics: Second Edition*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2005. ISBN 9780198528692. URL https://books.google.co.uk/books?id=zioTDAAAQBAJ.

[113] W. Karush. *Minima of functions of several variables with inequalities as side conditions*. PhD thesis, Thesis (S.M.)–University of Chicago, Department of Mathematics, December 1939., 1939.

[114] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[115] E. C. Kerrigan, Y. Nie, O. J. Faqir, C. Kennedy, S. A. Niederer, J. A. Solis-Lemus, P. Vincent, and S. E. Williams. Direct transcription for dynamic optimization: A tutorial and case study on dual-patient ventilation during the COVID-19 pandemic, Dec 2020. Accepted to 59th IEEE Conference on Decision and Control (CDC).

[116] T. Keviczky and G. Balas. Software enabled flight control using receding horizon techniques. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 5671, 2003.

[117] T. Keviczky and G. J. Balas. Receding horizon control of an F-16 aircraft: A comparative study. *Control Engineering Practice*, 14(9):1023–1033, 2006.

[118] B. Khusainov, E. C. Kerrigan, and G. A. Constantinides. Automatic software and computing hardware codesign for predictive control. *IEEE Transactions on Control Systems Technology*, 27(5):2295–2304, Sep. 2019. ISSN 2374-0159. doi: 10.1109/TCST.2018.2855666.

[119] A. T. Klesh and P. T. Kabamba. Solar-powered aircraft: Energy-optimal path planning and perpetual endurance. *Journal of guidance, control, and dynamics*, 32(4):1320–1329, 2009.

[120] P. Kuhl, J. Ferreau, J. Albersmeyer, C. Kirches, L. Wirsching, S. Sager, A. Potschka, G. Schulz, M. Diehl, D. Leineweber, and A. Schafer. Muscod-ii users' manual, June 2007.

[121] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200500249.

[122] M. J. Kurtz and M. A. Henson. Input-output linearizing control of constrained nonlinear processes. *Journal of Process Control*, 7(1):3–17, 1997.

[123] H. Lei, T. Liu, D. Li, J. Ye, and L. Shao. Adaptive mesh iteration method for trajectory optimization based on hermite-pseudospectral direct transcription. *Mathematical Problems in Engineering*, 2017, 2017.

[124] J. M. Levin, M. Nahon, and A. A. Paranjape. Aggressive turn-around manoeuvres with an agile fixed-wing uav. *IFAC-PapersOnLine*, 49(17):242 – 247, 2016. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2016.09.042. URL http://www.sciencedirect.com/science/article/pii/S2405896316315142. 20th IFAC Symposium on Automatic Control in AerospaceACA 2016.

[125] B. Li, K. Wang, and Z. Shao. Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3263–3274, 2016.

[126] G. Licitra, A. Bürger, P. Williams, R. Ruiterkamp, and M. Diehl. Optimal input design for autonomous aircraft. *Control Engineering Practice*, 77:15–27, 2018.

[127] G. Licitra, A. Bürger, P. Williams, R. Ruiterkamp, and M. Diehl. Aerodynamic model identification of an autonomous aircraft for airborne wind energy. *Optimal Control Applications and Methods*, 40(3):422–447, 2019.

[128] F. Liu, W. W. Hager, and A. V. Rao. Adaptive mesh refinement method for optimal control using nonsmoothness detection and mesh size reduction. *Journal of the Franklin Institute*, 352(10):4081–4106, 2015.

[129] F. Liu, W. W. Hager, and A. V. Rao. Adaptive mesh refinement method for optimal control using decay rates of legendre polynomial coefficients. *IEEE Transactions on Control Systems Technology*, 26(4):1475–1483, 2017.

[130] J. S. Logsdon and L. T. Biegler. Accurate solution of differential-algebraic optimization problems. *Industrial & engineering chemistry research*, 28(11):1628–1639, 1989.

[131] K. J. Lush. *A Review of the Problem of Choosing a Climb Technique, with Proposals from a New Climb Technique for High Performance Aircraft.* [Aeronautical Research Council. Reports and Memoranda. no. 2557.]. His Maj. Stat. Office, 1951. URL https://books.google.co.uk/books?id=etbptgAACAAJ.

[132] R. Luus. *Iterative Dynamic Programming*. Monographs and Surveys in Pure and Applied Mathematics. Taylor & Francis, 2000. ISBN 9781584881483.

[133] L. Ma, K. Wang, Z. Shao, Z. Song, and L. T. Biegler. Direct trajectory optimization framework for vertical takeoff and vertical landing reusable rockets: case study of two-stage rockets. *Engineering Optimization*, 51(4):627–645, 2019. doi: 10.1080/0305215X.2018.1472774. URL https://doi.org/10.1080/0305215X.2018.1472774.

[134] J. M. Maciejowski. *Predictive control: A lecture course given in the Aerospace Engineering Faculty TU Delft*. Series 03: Control and Simulation 05. Delft University Press, 1998. ISBN 9040717141.

[135] J. M. Maciejowski. *Predictive Control: With Constraints*. Pearson Education. Prentice Hall, 2002. ISBN 9780201398236.

[136] J. M. Maciejowski and C. Jones. MPC fault-tolerant flight control case study: Flight 1862. In *Proceedings of the International Federation of Automatic Control on Safeprocess Sympoisum*, pages 119–124, 2003.

[137] MathWorks. Find minimum of constrained nonlinear multivariable function - matlab fmincon, 2017. URL https://uk.mathworks.com/help/optim/ug/fmincon.html.

[138] Y. Matogawa. Optimum low thrust transfer to geosynchronous orbit. *Acta Astronautica*, 10(7):467 – 478, 1983. ISSN 0094-5765. doi: https://doi.org/10.1016/0094-5765(83)90019-X. URL http://www.sciencedirect.com/science/article/pii/009457658390019X.

[139] J. Mattingley and S. Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[140] H. Maurer. Numerical solution of singular control problems using multiple shooting techniques. *Journal of Optimization Theory and Applications*, 18(2):235–257, Feb 1976. ISSN 1573-2878. doi: 10.1007/BF00935706. URL https://doi.org/10.1007/BF00935706.

[141] H. Maurer. Theory and applications of bang-bang and singular control problems, 2007. URL http://www.cmap.polytechnique.fr/~bonnans/www-commands/Conferences/ceaedfinria-optcont/SLIDES/Maurer_INRIA07_Bang.pdf.

[142] G. P. McCormick. Second order conditions for constrained minima. *SIAM Journal on Applied Mathematics*, 15(3):641–652, 1967. ISSN 00361399. URL http://www.jstor.org/stable/2946203.

[143] J. Meditch. On the problem of optimal thrust programming for a lunar soft landing. *IEEE Transactions on Automatic Control*, 9(4):477–484, 1964.

[144] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

[145] S. R. Merritt, E. M. Cliff, and H. J. Kelley. Energy-modelled climb and climb-dash—the kaiser technique. *Automatica*, 21(3):319–321, 1985.

[146] A. Miele, T. Wang, H. Wang, and W. W. Melvin. Optimal penetration landing trajectories in the presence of windshear. *Journal of Optimization Theory and Applications*, 57(1):1–40, 1988.

[147] G. Mingotti, F. Topputo, and F. Bernelli-Zazzera. Low-energy, low-thrust transfers to the moon. *Celestial Mechanics and Dynamical Astronomy*, 105:61–74, 11 2009. doi: 10.1007/s10569-009-9220-7.

[148] M. A. Müller, D. Angeli, and F. Allgöwer. On necessity and robustness of dissipativity in economic model predictive control. *IEEE Transactions on Automatic Control*, 60(6): 1671–1676, 2015.

[149] M. P. Neuenhofen and E. C. Kerrigan. Dynamic optimization with convergence guarantees. *arXiv preprint arXiv:1810.04059*, 2018.

[150] B. L. Nicholson, W. Wan, S. Kameswaran, and L. T. Biegler. Parallel cyclic reduction strategies for linear systems that arise in dynamic optimization problems. *Computational Optimization and Applications*, 70(2):321–350, 2018. doi: 10.1007/s10589-018-0001-7. URL https://doi.org/10.1007/s10589-018-0001-7.

[151] Y. Nie and E. C. Kerrigan. Capturing discontinuities in optimal control problems. In *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018. doi: 10.1109/CONTROL.2018.8516770.

[152] Y. Nie and E. C. Kerrigan. Efficient implementation of rate constraints for nonlinear optimal control. In *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018. doi: 10.1109/CONTROL.2018.8516847.

[153] Y. Nie and E. C. Kerrigan. ICLOCS2: Solve your optimal control problems with less pain, Aug 2018. presented at 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[154] Y. Nie and E. C. Kerrigan. How should rate constraints be implemented in nonlinear optimal control solvers? *IFAC-PapersOnLine*, 51(20):362 – 367, 2018. ISSN 2405-8963. doi: 10.1016/j.ifacol.2018.11.060. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[155] Y. Nie and E. C. Kerrigan. External constraint handling for solving optimal control problems with simultaneous approaches and interior point methods. *IEEE Control Systems Letters*, 4(1):7–12, 2020. doi: 10.1109/LCSYS.2019.2921700. The contents of this paper were also selected by CDC 2019 Program Committee for presentation at the 58th Conference on Decision and Control, Nice, France, Dec 2019.

[156] Y. Nie and E. C. Kerrigan. Efficient and more accurate representation of solution trajectories in numerical optimal control. *IEEE Control Systems Letters*, 4(1):61–66, Jan 2020. ISSN 2475-1456. doi: 10.1109/LCSYS.2019.2921704. The contents of this paper were also selected by CDC 2019 Program Committee for presentation at the 58th Conference on Decision and Control, Nice, France, Dec 2019.

[157] Y. Nie and E. C. Kerrigan. Efficient implementation of rate constraints for nonlinear optimal control. *IEEE Transactions on Automatic Control*, pages 1–1, 2020. In press.

[158] Y. Nie, O. J. Faqir, and E. C. Kerrigan. ICLOCS2: Try this optimal control problem solver before you try the rest. In *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018. doi: 10.1109/CONTROL.2018.8516795.

[159] Y. Nie, O. J. Faqir, and E. C. Kerrigan. ICLOCS2: Solve your optimal control problems with less pain. In *Proc. 6th IFAC Conference on Nonlinear Model Predictive Control*, 2018.

[160] M. Pachter, P. R. Chandler, and M. Mears. Reconfigurable tracking control with saturation. *Journal of Guidance, Control, and Dynamics*, 18(5):1016–1022, 1995.

[161] M. A. Patterson and A. V. Rao. Exploiting sparsity in direct collocation pseudospectral methods for solving optimal control problems. *Journal of Spacecraft and Rockets*, 49 (2):354–377, 2012.

[162] M. A. Patterson and A. V. Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[163] M. A. Patterson and A. V. Rao. *User's Manual, GPOPS-II: A General Purpose MATLAB Software for Solving Multiple-Phase Optimal Control Problems*, 2.3 edition, Dec 2016. URL http://www.gpops2.com/resources/gpops2UsersGuide.pdf.

[164] J. A. Pietz. Pseudospectral collocation methods for the direct transcription of optimal control problems. Technical report, RICE UNIV HOUSTON TX, 2003.

[165] J. Piotrowska, J. M. Miller, and E. Schnetter. Spectral methods in the presence of discontinuities. *Journal of Computational Physics*, 390:527 – 547, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.03.048. URL http://www.sciencedirect.com/science/article/pii/S0021999119302311.

[166] M. M. J. Proot and M. I. Gerrtisma. Least-squares spectral elements applied to the stokes problem. *Journal of Computational Physics*, 181(2):454 – 477, 2002. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.2002.7137. URL http://www.sciencedirect.com/science/article/pii/S0021999102971378.

[167] R. Quirynen, S. Gros, B. Houska, and M. Diehl. Lifted collocation integrators for direct optimal control in acado toolkit. *Mathematical Programming Computation*, 9(4): 527–571, 2017.

[168] A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[169] A. V. Rao, S. Tang, and W. P. Hallman. Numerical optimization study of multiple-pass aeroassisted orbital transfer. *Optimal Control Applications and Methods*, 23(4):215–238, 2002.

[170] A. V. Rao, D. A. Benson, C. Darby, M. A. Patterson, C. Francolin, I. Sanders, and G. T. Huntington. Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):22, 2010.

[171] S. S. Rao. *The Finite Element Method in Engineering*. Elsevier Science, 2010. ISBN 9780080952048. URL https://books.google.co.uk/books?id=9PwTNzyySzwC.

[172] K. M. M. Rathai, M. Alamir, and O. Sename. pNMPC - A Code Generation Software Tool for Implementation of Derivative Free Parameterized NMPC Scheme for Embedded Control Systems. working paper or preprint, June 2020. URL https://hal.archives-ouvertes.fr/hal-02882652.

[173] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Santa Barbara CA, USA, 2nd edition, 2019.

[174] J. J. Recasens, Q. P. Chu, and J. A. Mulder. Robust model predictive control of a feedback linearized system for a lifting-body re-entry vehicle. In *AIAA Guidance, Navigation, and Control Conference*, volume 4, pages 2952–2984, 2005.

[175] REx Lab. *TrajectoryOptimization.jl Documentation*. Stanford University, 2019. URL https://roboticexplorationlab.github.io/TrajectoryOptimization.jl/dev/.

[176] M. Rieck, M. Bittner, B. Grüter, and J. Diepolder. Falcon.m user guide, 2016. URL www.falcon-m.com.

[177] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3): 1247–1293, 2013.

[178] J. Rosenow, H. Fricke, and M. Schultz. Air traffic simulation with 4d multi-criteria optimized trajectories. In *2017 Winter Simulation Conference (WSC)*, pages 2589–2600. IEEE, 2017.

[179] M. Rushdi, A. Hussein, T. N. Dief, S. Yoshida, and R. Schmehl. Simulation of the transition phase for an optimally-controlled tethered vtol rigid aircraft for airbornewind energy generation. In *AIAA Scitech 2020 Forum*, page 1243, 2020.

[180] I. Saclay. Bocop: an open source toolbox for optimal control. http://bocop.org, 2017.

[181] G. Schräm, R. A. J. de Vries, E. Cevaal, and T. J. J. van den Boom. Predictive control applied to a civil aircraft benchmark problem. In *1997 European Control Conference (ECC)*, pages 1297–1302, July 1997.

[182] S. Schuet, T. Lombaerts, J. Kaneshige, K. H. Shish, and V. Stepanyan. Stall recovery guidance using fast model predictive control. In *AIAA Guidance, Navigation, and Control Conference*, page 1513, 2017.

[183] J. Shen, T. Tang, and L. L. Wang. *Spectral Methods: Algorithms, Analysis and Applications*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2011. ISBN 9783540710417.

[184] D. Simon, J. Lofberg, and T. Glad. Nonlinear model predictive control using feedback linearization and local inner convex constraint approximations. In *Control Conference (ECC), 2013 European*, pages 2056–2061. IEEE, 2013.

[185] S. N. Singh, M. Steinberg, and R. D. DiGirolamo. Nonlinear predictive control of feedback linearizable systems and flight control system design. *Journal of Guidance, Control, and Dynamics*, 18(5):1023–1028, 1995.

[186] M. Soler, A. Olivares, and E. Staffetti. Hybrid optimal control approach to commercial aircraft trajectory planning. *Journal of Guidance, Control, and Dynamics*, 33(3):985–991, 2010.

[187] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 2020. doi: 10.1007/s12532-020-00179-2. URL https://doi.org/10.1007/s12532-020-00179-2.

[188] G. Stolwijk. A hybrid optimal control approach to multi-aircraft formation flying. Master's thesis, https://repository.tudelft.nl/islandora/object/uuid:f35d7060-8c87-47e6-a555-1bd8b1d0373c?collection=education, Nov 2017. URL https://repository.tudelft.nl/islandora/object/uuid:f35d7060-8c87-47e6-a555-1bd8b1d0373c?collection=education.

[189] T. A. Straeter. Singular extremaloids in optimal control theory and the calculus of variations. Technical report, National Aeronautics AND Space Administration, 1970.

[190] J. Sun. *Open Aircraft Performance Modeling: Based on an Analysis of Aircraft Surveillance Data*. PhD thesis, Delft University of Technology, 06 2019.

[191] H. S. Tsien and R. C. Evans. Optimum thrust programming for a sounding rocket. *Journal of the American Rocket Society*, 21(5):99–107, 1951.

[192] P. Tsiotras and H. J. Kelley. Drag-law effects in the goddard problem. *Automatica*, 27 (3):481–490, 1991.

[193] E. R. Van Oort, Q. Chu, J. A. Mulder, and T. J. J. van Den Boom. Robust model predictive control of a feedback linearized nonlinear F-16/MATV aircraft model. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 2148–2174. AIAA, 2006.

[194] G. N. Vanderplaats. *Multidicipline Design Optimization*. VR&D, 2007. ISBN 978-0944956045.

[195] M. Verhaegen, S. Kanev, R. Hallouzi, C. Jones, J. M. Maciejowski, and H. Smail. Fault tolerant flight control-a survey. In *Fault Tolerant Flight Control*, pages 47–89. Springer, 2010.

[196] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl. acados: a modular open-source framework for fast embedded optimal control. *arXiv preprint arXiv:1910.13753*, 2019.

[197] J. Villarroel and L. Rodrigues. An optimal control framework for the climb and descent economy modes of flight management systems. *IEEE Transactions on Aerospace and Electronic Systems*, 52(3):1227–1240, 2016.

[198] J. Villarroel and L. Rodrigues. Optimal control framework for cruise economy mode of flight management systems. *Journal of Guidance, Control, and Dynamics*, 39(5): 1022–1033, 2016.

[199] N. Vinh. *Optimal Trajectories in Atmospheric Flight*. Studies in Astronautics. Elsevier Science, 1981. ISBN 9780444601452.

[200] N. Vinh. *Optimal Trajectories in Atmospheric Flight*. Studies in Astronautics. Elsevier Science, 2012. ISBN 9780444601452. URL https://books.google.co.uk/books?id=70oHPG49hhEC.

[201] R. B. Vinter. Is the costate variable the state derivative of the value function? In *1986 25th IEEE Conference on Decision and Control*, pages 1988–1989, 1986.

[202] R. B. Vinter. *Optimal Control and Pontryagin's Maximum Principle*, pages 1–9. Springer London, London, 2013. ISBN 978-1-4471-5102-9. doi: 10.1007/978-1-4471-5102-9_200-1. URL https://doi.org/10.1007/978-1-4471-5102-9_200-1.

[203] W. G. Vlases, S. W. Paris, R. M. Lajoie, P. J. Martens, and C. R. Hargraves. *Optimal Trajectories By Implicit Simulation, Version 2. 0 User's Manual*. Boeing Aerospace and Electronics, Seattle, WA, 1990.

[204] O. von Stryk. *User's Guide for DIRCOL (Version 2.1): A Direct Collocation Method for the Numerical Solution of Optimal Control Problems*. Technische Universität München, 2000.

[205] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y. URL https://doi.org/10.1007/s10107-004-0559-y.

[206] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[207] L. Wang. *Model Predictive Control System Design and Implementation Using MATLAB®*. Advances in Industrial Control. Springer London, 2009. ISBN 9781848823303.

[208] Y. Wang, A. Wynn, and R. Palacios. Model-predictive control of flexible aircraft dynamics using nonlinear reduced-order models. In *57th aiaa/asce/ahs/asc structures, structural dynamics, and materials conference*, page 0711, 2016.

[209] M. J. Weinstein and A. V. Rao. Algorithm 984: Adigator, a toolbox for the algorithmic differentiation of mathematical functions in matlab using source transformation via operator overloading. *ACM Transactions on Mathematical Software (TOMS)*, 44(2):21, 2017.

[210] S. J. Wright. Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, 77(1):161–187, 1993.

[211] D. Wu, F. Wolter, B. Wang, and L. Xie. Searching for the shortest path to the point of voltage collapse on the algebraic manifold, 2020.

[212] K. Zondervan, T. Bauer, J. T. Betts, and W. Huffman. Solving the optimal control problem using a nonlinear programming technique. iii-optimal shuttle reentry trajectories. In *Astrodynamics Conference*, page 2039, 1984.

[213] K. P. Zondervan. *Optimal low thrust, three burn orbit transfers with large plane changes.* PhD thesis, California Institute of Technology, 1983.