Chair of Astrodynamics
School of Engineering and Design
Technical University of Munich

Real-Time Time-Optimal Reorientation of a Rigid Spacecraft

Algorithmic Design and Performance Evaluation

Leonardo Eitner, M.Sc. Candidate

Thesis for the attainment of the academic degree

Master of Science in Aerospace

at the School of Engineering and Design of the Technical University of Munich.

Examiner:
Prof. Dr. Marcello Romano

Supervisor:
M.Sc. Silvia Busi

Submitted:
Munich, 30.11.2025

I hereby declare that the work presented in this thesis is entirely the result of my own work except where otherwise indicated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced. Suggestions from the supervisors regarding language and content are excepted.

Munich, 30.11.2025 Leonardo Eitner, M.Sc. Candidate

Abstract

Rapid spacecraft reorientation is critical for emerging applications including fast low-Earth orbit communication satellites, on-orbit servicing, active debris removal and defense systems. While time-optimal attitude maneuvering has been extensively studied, existing solutions typically require significant computation time, limiting their applicability to real-time operations. This paper addresses the challenge of computing time-optimal spacecraft attitude trajectories in real-time (under 0.2 seconds) for generalized initial and final states, including orientation and angular velocities, and for spacecraft with asymmetric inertia properties. A comprehensive simulation platform was developed in C++ and CUDA using CasADi to systematically evaluate direct Optimal Control Problem (OCP) transcription methods. The work examines key algorithmic tradeoffs, including multiple shooting versus direct collocation, implicit versus explicit integration schemes, GPU versus CPU dynamics propagation, and FATROP versus IPOPT solvers. Additionally, the impact of initialization strategies is investigated, comparing fixed initial guesses with Particle Swarm Optimization (PSO) approaches. For the latter, both full control trajectory optimization and Switch Time Optimization (STO) are compared. Monte Carlo simulations are employed to assess the reliability of each configuration across diverse maneuver scenarios, with results corroborated using CGPOPS. The optimal configuration combines FATROP with a fixed deterministic initialization, multiple shooting transcription, and CPU-based fourth-order Runge-Kutta integration. For rest-to-rest maneuvers with symmetric spacecraft, this approach finds the optimal solution in over 90% of cases with computation times under 0.12 seconds. Performance degrades with increasing spacecraft asymmetry and angular velocity magnitudes, though the algorithm maintains practical utility across a wide operational envelope. This work presents a reliable real-time algorithm for time-optimal spacecraft reorientation maneuvers capable of handling arbitrary initial and final attitude states and asymmetric inertia configurations. The followed methodology provides practical guidance for implementation choices in real-time attitude control systems. Future work can extend the framework to incorporate keep-out cone constraints and trajectory path restrictions for collision avoidance and operational safety requirements.

Contents

List of Figures

List of Tables

List of Symbols

General Symbols

$\mathbf{b}$	[-]	Basis function coefficient vector
$\mathbf{e}$	[-]	Unit rotation axis vector
$\mathbf{f}$	[-]	System dynamics function
$\mathbf{f}_0$	[-]	Drift dynamics (control-free dynamics)
$\mathbf{G}$	[-]	Control influence matrix
$\mathbf{g}$	[-]	Global best position (PSO)
$\mathbf{I}_3$	[-]	$3{\times}3$ identity matrix
$\mathbf{J}$	[kg·m^2]	Inertia tensor (matrix)
$\mathbf{q}$	[-]	Unit quaternion vector
$\mathbf{q}_I$	[-]	Identity quaternion $[1\ 0\ 0\ 0]^T$
$\mathbf{R}$	[-]	Rotation matrix
$\mathbf{S}$	[-]	Switching function vector
$\mathbf{t}_i$	[s]	Switching times vector for axis i
$\mathbf{u}$	[-]	Control vector
$\mathbf{v}$	[-]	Velocity vector (PSO)
$\mathbf{x}$	[-]	State vector
$\mathbf{z}$	[-]	Position vector (PSO)
$\mathcal{B}$	[-]	Body-fixed reference frame
$\mathcal{C}$	[-]	Chebyshev polynomial
$\mathcal{I}$	[-]	Inertial reference frame
$\mathcal{J}$	[-]	Cost functional
$\mathcal{K}$	[-]	PSO iterations
$\mathcal{L}$	[-]	Lagrange interpolating polynomial
$\mathcal{N}$	[-]	PSO population size
$\mathcal{P}$	[-]	Legendre polynomial

$\mathcal{S}$	[-]	Sigmoid function
$\mathcal{T}$	[-]	Computation time
$\mathcal{U}$	[-]	Admissible control set
c	[-]	Collocation point
C_{sw}	[-]	Switch count function
D	[-]	Parameter space dimension
H	[-]	Hamiltonian function
h	[kg·m^2·s^{-1}]	Angular momentum
H_0	[-]	Control-independent Hamiltonian portion
J	[kg·m^2]	Principal moment of inertia
L	[-]	Running cost (Lagrange term)
l	[-]	Number of samples
M	[-]	Number of basis functions
m	[-]	Control vector dimension
N	[-]	Number of discretization points/intervals
n	[-]	State vector dimension
p	[-]	Polynomial degree
q	[-]	Quaternion component
r	[-]	Random number
S	[-]	Switching function component
s	[-]	Initial control direction (bang-bang)
t	[s]	Time
u	[-]	Control vector component
v	[-]	Velocity vector component (PSO)
V_{max}	[-]	Maximum velocity (PSO)
w_0	[-]	Inertia weight (PSO)
w_1	[-]	Cognitive parameter (PSO)
w_2	[-]	Social parameter (PSO)
x	[-]	State vector component
y	[-]	Number of switches per control axis
z	[-]	Position vector component (PSO)

Greek Symbols

α	[-]	Scaling parameter
β	[rad]	Rotation angle
$\boldsymbol{\gamma}$	[-]	Integrated state prediction
$\boldsymbol{\kappa}$	[-]	Runge-Kutta slope
$\boldsymbol{\lambda}$	[-]	Costate vector (adjoint variable)
$\boldsymbol{\nu}$	[-]	Lagrange multipliers for terminal constraints
$\boldsymbol{\omega}$	[N·m]	Angular velocity vector
$\boldsymbol{\Omega}$	$[\mathrm{rad} \cdot \mathrm{s}^{-1}]$	Skew-symmetric angular velocity matrix
$\boldsymbol{\Psi}$	[-]	Terminal constraint function
$\boldsymbol{\rho}$	[-]	Personal best position (PSO)
$\boldsymbol{\sigma}$	[-]	Modified Rodrigues Parameters vector
$\boldsymbol{\tau}$	[N·m]	Torque vector
$\boldsymbol{\varepsilon}$	[-]	Hermite-Simpson collocation constraint
χ	[-]	Basis function
Δ	[-]	Change
δ	[-]	Kronecker delta
ϵ	[-]	Saturation threshold
η	[-]	Number of terminal constraints
μ	[-]	Eigenvalue (complex)
ω	$[\mathrm{rad} \cdot \mathrm{s}^{-1}]$	Angular velocity vector component
Φ	[-]	Terminal cost (Mayer term)
ϕ	[rad]	Roll Euler angle
ψ	[rad]	Yaw Euler angle
Σ	[-]	Sum
τ	[N·m]	Torque vector component
θ	[rad]	Pitch Euler angle

Subscripts

$\square_0$	Initial value
$\square_{\max}$	Maximum value

$\square_{\text{min}}$	Minimum value
$\square_{\omega}$	Angular velocity component
$\square_{\text{opt}}$	Relative to an optimal value
$\square_{\text{PSO}}$	Relative to the PSO
$\square_{\text{rot}}$	Relative rotation
$\square_{\text{solver}}$	Relative to the gradient-based solver
$\square_{\text{sw}}$	Control switch-related
$\square_{\text{total}}$	Relative to a total value
$\square_{\times}$	Skew-symmetric cross-product matrix operator
$\square_{d}$	Dimension index
$\square_{f}$	Final value
$\square_{i}$	Component along axis i
$\square_{j}$	Particle/point index
$\square_{k}$	Time step/interval index
$\square_{q}$	Quaternion component

Superscripts

$\bar{\square}$	Normalization by a characteristic scale
$\square^{*}$	Optimal value
$\square^{-1}$	Inverse
$\square^{-}$	Quaternion conjugate
$\square^{\dagger}$	Pseudoinverse
$\square^{k}$	Iteration index
$\square^{T}$	Transposed
$\ddot{\square}$	Second time derivative
$\dot{\square}$	First time derivative

Special Symbols

$\cos(\cdot)$	Cosine function
$\det(\cdot)$	Matrix determinant
$\in$	Is within a specified set of values

∞	Infinity
$\mathbb{C}$	Set of complex numbers
$\mathbb{R}$	Set of real numbers
$\mathbf{X}$	Multi-dimensional variables
$\otimes$	Quaternion multiplication
$\sin(\cdot)$	Sine function
$\subset$	Is a subset of a specified set of values
$\tan(\cdot)$	Tangent function
$\mathrm{diag}(\cdot)$	Diagonal matrix
$\max(\cdot)$	Returns the maximum value
$\min(\cdot)$	Returns the minimum value
$\mathrm{sgn}(\cdot)$	Sign function
$\times$	Vector cross product

Abbrevations

AD	Automatic Differentiation
ADR	Active Debris Removal
API	Application Programming Interface
BC	Boundary Conditions
CGL	Chebyshev-Gauss-Lobatto
CPU	Central Processing Unit
DRAM	Dynamic Random Access Memory
FD	Finite Differences
GPU	Graphics Processing Unit
IP	Interior Point
KKT	Karush-Kuhn-Tucker
LEO	Low Earth Orbit
LG	Legendre-Gauss
LGR	Legendre-Gauss-Radau
LHS	Latin Hypercube Sampling
MC	Monte Carlo

MPC	Model Predictive Control
MRP	Modified Rodrigues Parameters
NLP	Non-Linear Programming
OCP	Optimal Control Problem
ODE	Ordinary Differentiable Equation
OOS	On-Orbit Servicing
PCIe	Peripheral Component Interconnect Express
PMP	Pontryagin Maximum Principle
PSO	Particle Swarm Optimization
QP	Quadratic Programming
RK4	Runge-Kutta of fourth-order
SCP	Sequential Convex Programming
SIMT	Single Instruction, Multiple Thread
SM	Streaming Multiprocessors
SO(3)	Special Orthogonal group in three-dimensional Euclidean space
SQP	Sequential Quadratic Programming
STO	Switch Time Optimization
TDP	Thermal Design Power
TPBVP	Two-Point Boundary Value Problem

1 Introduction

The precise determination and manipulation of spacecraft orientation, known as spacecraft attitude control, constitutes a fundamental enabling technology for space missions ranging from Earth observation and telecommunications to scientific exploration and defense applications. As satellite systems evolve toward increasingly agile operations, particularly in Low Earth Orbit (LEO) constellations, the demand for rapid and precise reorientation capabilities has intensified. Time-optimal attitude control, which seeks to minimize maneuver duration while respecting physical constraints on actuator torques, represents the theoretical performance limit for spacecraft agility. However, classical approaches for calculating the time-optimal trajectory, whether indirect methods based on Pontryagin's Maximum Principle (PMP) or direct methods employing pseudospectral transcription, typically require computation times incompatible with real-time onboard implementation. This thesis addresses the challenge of generating time-optimal spacecraft attitude trajectories with computation times suitable for closed-loop control, enabling autonomous spacecraft to respond dynamically to mission requirements and environmental disturbances. Through systematic investigation of algorithm optimization methods and intelligent initialization strategies, this work develops and verifies a real-time algorithm capable of computing time-optimal rest-to-rest and track-to-track maneuvers for spacecraft with arbitrary inertia properties.

1.1 Motivation

Spacecraft attitude control systems are undergoing a paradigm shift driven by applications that demand unprecedented agility and responsiveness. The proliferation of LEO satellite constellations for global communications and the development of space-based defense systems have created a critical need for rapid spacecraft reorientation capabilities. Unlike geostationary satellites that maintain relatively fixed orientations, LEO satellites require frequent attitude maneuvers to execute transitions between observation targets, whether for maintaining communication links, tracking ground targets, or executing collision avoidance maneuvers.

Additionally, the emerging fields of On-Orbit Servicing (OOS) and Active Debris Removal (ADR) missions imposes stringent requirements for agile attitude control systems, as the non-cooperative nature of client satellites and debris targets presents significant challenges. Debris objects or satellites in safe mode will be tumbling rather than maintaining stable attitudes [1], necessitating advanced control strategies capable of approaching, capturing, and stabilizing rotating targets. Underactuated spacecraft control, either by design or due to onboard failures, also represents a significant driver for attitude control innovation. Being able to compensate for the missing actuation by quickly reorienting the spacecraft can extend operational life and improve mission robustness [2], providing fault tolerance and enabling novel control architectures that reduce system complexity and mass.

The time-optimal control of spacecraft attitude has been extensively studied since the foundational work by Bilimoria and Wie [3], who demonstrated that time-optimal reorientation of asymmetric rigid bodies does not follow the intuitive eigenaxis maneuver. Subsequent research by Bai and Junkins [4] and others has refined these results, establishing theoretical frameworks for computing time-optimal trajectories. However, a persistent limitation of these approaches is their computational complexity: traditional methods based on indirect optimal control or pseudospectral transcription can require computation times ranging from several seconds to minutes [5], rendering them unsuitable for real-time onboard implementation.

This computational bottleneck creates a fundamental trade-off in spacecraft control system design. Controllers must either sacrifice optimality by using suboptimal but computationally efficient methods (e.g., eigenaxis maneuvers or proportional-derivative control) or compute optimal trajectories offline with limited adaptability to changing mission requirements or disturbances. For applications requiring rapid retargeting—such as surveillance satellites tracking multiple ground targets, communication satellites maintaining line-of-sight with mobile ground stations, or defensive systems responding to threats—neither approach is satisfactory.

1.2 Research Question

With the motivation established, the research question which this thesis seeks to answer can be stated clearly:
How can one reliably compute time-optimal spacecraft slew maneuvers in real-time?
Before answering this question it is necessary to precisely define three key terms:

1. **Real-time** refers to computation times that enable autonomous trajectory replanning within the operational timescales of spacecraft attitude determination and control systems. The specific threshold depends on the spacecraft's sensor architecture, control loop frequency, inertia properties, and available control authority. This work establishes a threshold of 200 ms based on spacecraft sensor update cycles. Star trackers—which provide absolute attitude measurements—typically operate at update rates below 5 Hz [6]. A 200 ms computation threshold ensures that trajectory generation can be run in a closed-loop at the sampling rate of a star tracker, enabling updated time-optimal trajectories with the most recent absolute attitude measurement as the initial condition. This threshold represents approximately one order of magnitude improvement over state-of-the-art methods [5].

2. **Reliability** quantifies the probability that the optimization algorithm converges to a feasible and locally optimal solution from arbitrary initial conditions within the operational envelope. High reliability is essential for autonomous onboard implementation, as convergence failures necessitate fallback to suboptimal control strategies, negating the benefits of time-optimal control. Reliability must be characterized statistically through simulations across diverse scenarios, encompassing spacecraft with varying inertia asymmetry and maneuvers involving different angular velocity magnitudes.

3. **Optimality** represents the degree to which computed solutions approximate the global time-optimal trajectory. Solution quality is verified by comparison with CGPOPS, an adaptive pseudospectral collocation framework with established accuracy for time-optimal control problems [7]. This verification ensures that computational efficiency gains do not compromise solution optimality.

Achieving a satisfactory answer to this research question requires a thorough investigation of the algorithmic design space to identify the configuration that optimally balances computational efficiency, solution reliability, and solution quality. This investigation encompasses multiple algorithmic components—transcription methods, integration schemes, Non-Linear Programming (NLP) solvers, initialization strategies, and computational architectures—each presenting distinct trade-offs that must be rigorously evaluated.

1.3 Thesis Structure

This thesis is organized into five chapters, each addressing a distinct aspect of the real-time time-optimal spacecraft attitude control problem.

Chapter 1: Introduction establishes the motivation for real-time time-optimal attitude control in autonomous spacecraft operations, defines the research objectives, articulates the key performance metrics, and outlines the contributions of this work.

Chapter 2: Theoretical Foundations reviews spacecraft attitude dynamics and optimal control theory, including attitude parameterization methods, Euler's rotational equations, and the PMP. The chapter surveys the state-of-the-art in trajectory optimization methods, covering indirect and direct approaches, problem formulation strategies, computational methods, initialization strategies, and Graphics Processing Unit (GPU) acceleration techniques for parallel computation.

Chapter 3: Methodology describes the research workflow and algorithmic framework developed in this thesis. The chapter presents the specific Optimal Control Problem (OCP) formulation employed and provides a full guide on the decisions and choices made throughout the work. It describes the software architecture and implementation details of the comprehensive simulation platform developed in C++ and CUDA. The chapter covers the integration of CasADi for automatic differentiation and NLP problem formulation, the implementation of CGPOPS for verification and benchmarking, and the GPU acceleration strategies. Computational platform specifications and software design patterns employed to ensure modularity and extensibility are documented.

Chapter 4: Results presents the outcomes of the systematic algorithmic optimization, as well as the results of the Monte Carlo (MC) simulations employed to evaluate the performance and reliability of different algorithmic configurations. The sensitivity of algorithm performance to spacecraft asymmetry and angular velocity magnitudes is analyzed. The chapter identifies the optimal configuration and characterizes its operational envelope.

Chapter 5: Conclusion summarizes the principal findings of the research, emphasizing the key algorithmic choices that enable real-time time-optimal spacecraft attitude slew maneuvers. Limitations of the current approach are discussed and future research directions are outlined. It concludes with reflections on the broader implications of this work for autonomous spacecraft systems and the potential for onboard implementation.

2 Theoretical Foundations

Time-optimal spacecraft reorientation requires the integration of rigid body dynamics, optimal control theory, and computational optimization methods. The complexity of this problem arises from several sources: nonlinear gyroscopic coupling in Euler's rotational equations, the bang-bang control structure that introduces discontinuities challenging for numerical solution, nonconvex optimization landscapes with multiple local minima, and real-time computational constraints that restrict algorithmic complexity. Developing efficient algorithms necessitates good understanding of the theoretical foundations and computational methods underlying trajectory optimization.

This chapter establishes the mathematical framework required for computing real-time time-optimal spacecraft attitude maneuvers. Section 2.1 presents spacecraft attitude dynamics and parameterization methods. Section 2.2 develops optimal control theory and specializes to time-optimal problems. Section 2.3 examines indirect and direct trajectory optimization methodologies. Section 2.4 analyzes problem formulation strategies including control parameterization and dynamics representation. Section 2.5 applies optimal control theory to spacecraft attitude maneuvers, deriving the bang-bang control structure and reviewing historical developments. Section 2.6 surveys computational infrastructure including nonlinear programming solvers, software frameworks, and derivative-free optimization methods. Section 2.7 addresses initialization strategies, GPU acceleration, and statistical reliability assessment methods.

2.1 Spacecraft Attitude Dynamics

The mathematical description of spacecraft rotational motion constitutes the foundation upon which attitude control algorithms are constructed. This description comprises two essential components: the dynamic equations governing the evolution of angular velocity under applied torques, and the kinematic equations relating angular velocity to the time rate of change of attitude. The former is governed by Euler's rotational equations, which express the balance of angular momentum for a rigid body in rotational motion. The latter depends critically on the choice of attitude parameterization, i.e. the mathematical representation used to describe the spacecraft's orientation in three-dimensional space.

The selection of an appropriate attitude parameterization significantly influences both the theoretical properties and computational efficiency of optimal control algorithms. Various parameterization schemes have been developed, each exhibiting distinct advantages and limitations [8]. This section presents the fundamental equations governing spacecraft attitude dynamics and examines the principal parameterization methods employed in trajectory optimization. Section 2.1.1 derives Euler's rotational equations and establishes the relationship between control torques and angular acceleration. Section 2.1.2 surveys attitude parameterization methods, analyzing their properties in the context of time-optimal control problem formulation.

2.1.1 Euler's Rotational Equations

The rotational motion of a rigid spacecraft is governed by the principle of conservation of angular momentum. For a rigid body with angular momentum vector $\mathbf{h}$, the time rate of change of angular momentum equals the applied external torque $\boldsymbol{\tau}$:

$$\left.\frac{d\mathbf{h}}{dt}\right|_{\mathcal{I}} = \boldsymbol{\tau} \tag{2.1}$$

where the derivative is taken with respect to an inertial reference frame $\mathcal{I}$. For spacecraft attitude control applications, it is advantageous to express the equations of motion in a body-fixed reference frame $\mathcal{B}$ aligned with the spacecraft's principal axes of inertia. According to the vector transport theorem, the transformation of the time derivative from the inertial frame to the body-fixed frame introduces an additional term:

$$\left.\frac{d\mathbf{h}}{dt}\right|_{\mathcal{I}} = \left.\frac{d\mathbf{h}}{dt}\right|_{\mathcal{B}} + \boldsymbol{\omega} \times \mathbf{h} \tag{2.2}$$

where $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$ represents the angular velocity vector of the body-fixed frame relative to the inertial frame, expressed in body-fixed coordinates. The angular momentum in the body frame is related to the angular velocity through the inertia tensor $\mathbf{J}$:

$$\mathbf{h} = \mathbf{J}\boldsymbol{\omega} \tag{2.3}$$

Substituting equations (2.2) and (2.3) into equation (2.1) yields:

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \boldsymbol{\tau} \tag{2.4}$$

where $\dot{\boldsymbol{\omega}}$ denotes the time derivative of angular velocity in the body frame. Equation (2.4) represents Euler's rotational equations in vector form.

For a spacecraft with principal axes aligned with the body frame, the inertia tensor is diagonal: $\mathbf{J} = \text{diag}(J_1, J_2, J_3)$, where J_1, J_2, and J_3 are the principal moments of inertia. The component form of Euler's equations is then:

$$\begin{aligned} J_1\dot{\omega}_1 + (J_3 - J_2)\omega_2\omega_3 &= \tau_1 \\ J_2\dot{\omega}_2 + (J_1 - J_3)\omega_3\omega_1 &= \tau_2 \\ J_3\dot{\omega}_3 + (J_2 - J_1)\omega_1\omega_2 &= \tau_3 \end{aligned} \tag{2.5}$$

where τ_1, τ_2, and τ_3 are the control torques applied about the principal axes. The terms involving products of angular velocity components: $(J_3 - J_2)\omega_2\omega_3$, $(J_1 - J_3)\omega_3\omega_1$, and $(J_2 - J_1)\omega_1\omega_2$, are known as gyroscopic coupling terms. These terms introduce nonlinearity into the dynamics and couple the three rotational degrees of freedom.

The magnitude of gyroscopic coupling depends on the spacecraft's inertial asymmetry. For a spherically symmetric spacecraft with $J_1 = J_2 = J_3$, the gyroscopic terms vanish entirely, and the rotational dynamics decouple into three independent scalar equations. However, most practical spacecraft exhibit asymmetric inertia distributions, resulting in significant gyroscopic coupling that fundamentally affects optimal control solutions [9].

For optimal control applications, Euler's equations are typically written in state-space form. Solving equations (2.5) for the angular accelerations yields:

$$\begin{aligned} \dot{\omega}_1 &= \frac{1}{J_1}\left[\tau_1 - (J_3 - J_2)\omega_2\omega_3\right] \\ \dot{\omega}_2 &= \frac{1}{J_2}\left[\tau_2 - (J_1 - J_3)\omega_3\omega_1\right] \\ \dot{\omega}_3 &= \frac{1}{J_3}\left[\tau_3 - (J_2 - J_1)\omega_1\omega_2\right] \end{aligned} \tag{2.6}$$

These equations establish the relationship between control torques and the resulting angular acceleration. Combined with kinematic equations that relate angular velocity to attitude, equations (2.6) forms the complete dynamical system for spacecraft attitude control.

2.1.2 Attitude Parameterization Methods

The orientation of a rigid body in three-dimensional space possesses three rotational degrees of freedom, yet various mathematical representations exist to describe this orientation. The kinematic equations relating angular velocity to attitude depend fundamentally on the chosen parameterization. This subsection examines the principal attitude parameterization schemes employed in optimal control applications, analyzing their mathematical properties and computational implications.

Euler Angles

Euler angles parameterize attitude through three successive rotations about specified axes. The most commonly adopted representation follows the 3-2-1 (yaw-pitch-roll) sequence with angles ψ (yaw), θ (pitch), and ϕ (roll). The kinematic equations relating angular velocity $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$ to Euler angle rates are:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.7}$$

While intuitive and requiring only three parameters (matching the physical degrees of freedom), this representation suffers from singularities at certain orientations where the mapping between angular velocity and the time derivative of Euler angles becomes undefined [8]. The kinematic equations (2.7) contain trigonometric functions in the denominator that approach zero when the pitch angle approaches $\pm 90°$, a configuration known as gimbal lock. At these singular orientations, the transformation matrix becomes rank-deficient, and infinitely many combinations of $\dot{\phi}$ and $\dot{\psi}$ can produce the same angular velocity. These singularities restrict the applicability of Euler angles to problems where the spacecraft orientation remains within a limited range, making them unsuitable for large-angle reorientation maneuvers [8].

Unit Quaternions

Unit quaternions provide a globally non-singular four-parameter representation of attitude. A unit quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$ (where q_0 is the scalar part and $[q_1, q_2, q_3]^T$ the vector part) satisfies the unit norm constraint $\|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$. The kinematic equation relating angular velocity to the time derivative of the quaternion is:

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \tag{2.8}$$

where $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is the skew-symmetric matrix:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \tag{2.9}$$

The quaternion kinematic equation (2.8) is linear in both $\mathbf{q}$ and $\boldsymbol{\omega}$ and remains well-defined for all orientations. However, the unit norm constraint introduces an algebraic constraint into the optimization problem. This constraint can be handled through explicit normalization at each time step, penalty methods, or direct enforcement within the NLP formulation [10]. The unit quaternion representation constitutes a double cover of the attitude space SO(3), meaning that antipodal quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same physical orientation. Despite the redundant parameterization, quaternions have become the dominant choice for spacecraft attitude control due to their singularity-free representation and computational efficiency [8].

Modified Rodrigues Parameters

MRPs constitute a three-parameter minimal representation that avoids the redundancy of quaternions. For a rotation through angle β about axis $\mathbf{e}$, the MRP vector is defined as:

$$\boldsymbol{\sigma} = \mathbf{e}\tan\left(\frac{\beta}{4}\right) \tag{2.10}$$

The kinematic equation for MRPs is:

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}\left[(1 - \|\boldsymbol{\sigma}\|^2)\mathbf{I}_3 + 2[\boldsymbol{\sigma}]_\times + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T\right]\boldsymbol{\omega} \tag{2.11}$$

where $[\boldsymbol{\sigma}]_\times$ denotes the skew-symmetric cross-product matrix, and $\mathbf{I}_3$ the 3-by-3 identity matrix. MRPs offer computational advantages for small to moderate rotations, as the kinematic equation (2.11) involves only algebraic operations without transcendental functions. However, MRPs exhibit a singularity at $\beta = \pm 360°$ (corresponding to $\|\boldsymbol{\sigma}\| \to \infty$), limiting their applicability to large-angle maneuvers [8]. Shadow sets of MRPs can be employed to extend the valid range by switching to an alternative representation when $\|\boldsymbol{\sigma}\|$ exceeds a threshold, but this introduces discontinuities that complicate gradient-based optimization.

Rotation Matrices and SO(3)

Rotation matrices provide a coordinate-free representation of attitude on the Special Orthogonal group SO(3). A rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times 3}$ satisfies the orthonormality constraints $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$ and $\det(\mathbf{R}) = 1$. The kinematic equation on SO(3) is:

$$\dot{\mathbf{R}} = \mathbf{R}[\boldsymbol{\omega}]_\times \tag{2.12}$$

Direct SO(3) parameterization avoids singularities entirely and provides a natural framework for geometric control design [11, 12]. The manifold structure of SO(3) can be preserved through Lie group integrators that maintain orthonormality without explicit constraint enforcement. However, even then, when compared to the three physical rotational degrees of freedom, the nine matrix elements still introduce six redundant terms. If orthonormality constraints are explicitly enforced within the optimization problem, its computational complexity increases beyond the other attitude parameterization strategies. Recent developments in geometric optimization and structure-preserving discretization have made SO(3) representations increasingly competitive for trajectory optimization, particularly when combined with variational integrators [12].

Selection Criteria for Optimal Control

The choice of attitude parameterization for time-optimal control involves trade-offs between singularity avoidance, computational efficiency, and constraint complexity. Table 2.1 summarizes the key characteristics of each parameterization method.

Table 2.1 Comparison of attitude parameterization methods for optimal control applications.

Method	Parameters	Singularities	Constraints
Euler Angles	3	Yes	None
Quaternions	4	No	Unit norm
MRP	3	Yes	None
SO(3)	9	No	Orthonormality (6)

For time-optimal spacecraft reorientation problems involving arbitrary large-angle maneuvers, quaternion parameterization offers the most favorable balance. The global validity without singularities enables reliable optimization across the entire attitude space, while the unit norm constraint can be efficiently managed through normalization or penalty methods. The linear kinematic equation (2.8) facilitates efficient gradient computation required by gradient-based NLP solvers. Consequently, quaternion-based formulations have become standard in modern time-optimal attitude control implementations [3, 10, 13].

2.2 Optimal Control Theory

Optimal control theory provides the mathematical framework for determining control inputs that minimize a specified performance criterion while satisfying system dynamics and constraints. For time-optimal spacecraft attitude control, the objective is to minimize the maneuver duration subject to actuator torque limitations and prescribed boundary conditions on attitude and angular velocity. The theoretical foundations trace to the calculus of variations, with Pontryagin's seminal contribution extending these principles to control-constrained problems through the introduction of costate variables (also termed adjoint variables or Lagrange multipliers) and the formulation of necessary conditions for optimality [14]. A fundamental result of PMP for time-optimal problems with bounded control is the bang-bang control structure, wherein optimal control inputs assume their extreme values and switch between these extremes a finite number of times [14]. For spacecraft attitude control with torque-limited actuators, this bang-bang structure has been extensively characterized [3, 10], and its exploitation forms the basis for efficient solution algorithms. This section presents the fundamental theory of optimal control, specializes it to the time-optimal control, and establishes the theoretical properties that inform computational solution methods.

2.2.1 Optimal Control Problem Formulation

The general OCP in Bolza form seeks to determine a control trajectory $\mathbf{u}(t)$ that minimizes a cost functional while satisfying system dynamics and constraints. The system dynamics are described by ordinary differential equations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.13}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ represents the n-dimensional state vector and $\mathbf{u}(t) \in \mathcal{U} \subset \mathbb{R}^m$ denotes the m-dimensional control vector. The admissible control set $\mathcal{U}$ encodes constraints on control values. The time interval of interest is $t \in [t_0, t_f]$, with initial state specified as $\mathbf{x}(t_0) = \mathbf{x}_0$.

Terminal state conditions may assume three forms: $\mathbf{x}(t_f) = \mathbf{x}_f$ for fixed terminal states, For constrained terminal states, the condition $\boldsymbol{\Psi}(\mathbf{x}(t_f), t_f) = \mathbf{0}$ must be satisfied, where $\boldsymbol{\Psi} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^\eta$ defines η terminal constraints. For free terminal states, $\mathbf{x}(t_f)$ remains unconstrained.

The cost functional in Bolza form combines terminal and running costs:

$$\mathcal{J} = \Phi(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \tag{2.14}$$

where $\Phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ represents the terminal cost, also called Mayer term, and $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ denotes the running cost or Lagrange term.

The optimal control $\mathbf{u}^*(t)$ minimizes the cost functional $\mathcal{J}$ subject to the dynamics (2.13), control constraints $\mathbf{u}(t) \in \mathcal{U}$, and boundary conditions. The corresponding optimal trajectory $\mathbf{x}^*(t)$ satisfies all constraints and achieves the minimum cost among all admissible solutions [15].

2.2.2 Pontryagin's Minimum Principle

With the OCP formulated it is now necessary to determinie the optimal control $\mathbf{u}^*(t)$ that minimizes the cost funtional $\mathcal{J}$. Classical calculus of variations extends naturally to problems with unconstrained controls through the Euler-Lagrange equations, but fails when control authority is bounded. The PMP, established in 1962, extends the calculus of variations to control-constrained optimization problems and provides necessary conditions for optimality [14]. The principle introduces the costate vector $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$, which represents the sensitivity of the optimal cost-to-go with respect to state perturbations [15]. The Hamiltonian function is defined as:

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) = L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{2.15}$$

The Hamiltonian quantifies the instantaneous trade-off between minimizing the running cost and progressing along the state trajectory toward the terminal target.

For an optimal solution $(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t))$, PMP requires satisfaction of the following necessary conditions:

(i) State equations:

$$\dot{\mathbf{x}}^*(t) = \frac{\partial H}{\partial \boldsymbol{\lambda}} = \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t), t) \tag{2.16}$$

(ii) Costate equations:

$$\dot{\boldsymbol{\lambda}}^*(t) = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial L}{\partial \mathbf{x}} - \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda}^*(t) \tag{2.17}$$

(iii) Optimality condition (minimum principle):

$$H(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), t) = \min_{\mathbf{u}\in\mathcal{U}} H(\mathbf{x}^*(t), \mathbf{u}, \boldsymbol{\lambda}^*(t), t) \tag{2.18}$$

(iv) Transversality conditions: For terminal constraints $\boldsymbol{\Psi}(\mathbf{x}(t_f), t_f) = \mathbf{0}$ with associated Lagrange multipliers $\boldsymbol{\nu} \in \mathbb{R}^q$, the terminal costate satisfies:

$$\boldsymbol{\lambda}^*(t_f) = \left.\frac{\partial \Phi}{\partial \mathbf{x}}\right|_{t_f} + \boldsymbol{\nu}^T \left.\frac{\partial \boldsymbol{\Psi}}{\partial \mathbf{x}}\right|_{t_f} \tag{2.19}$$

Additionally, for free terminal time, the Hamiltonian must satisfy the condition:

$$H(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), \boldsymbol{\lambda}^*(t_f), t_f) = -\left.\frac{\partial \Phi}{\partial t}\right|_{t_f} - \boldsymbol{\nu}^T \left.\frac{\partial \boldsymbol{\Psi}}{\partial t}\right|_{t_f} \tag{2.20}$$

These conditions define a two-point boundary value problem (TPBVP): the state equations integrate forward from initial conditions $\mathbf{x}(t_0) = \mathbf{x}_0$, while the costate equations integrate backward from terminal conditions determined by the transversality conditions. The optimal control at each instant is determined pointwise from the minimization condition (2.18). The computational challenge of solving the resulting TPBVP motivates the development of direct transcription methods, as discussed in section 2.3.

2.2.3 Time-Optimal Control Theory

For time-optimal control problems where the objective is to minimize maneuver duration, the cost functional specializes to $\mathcal{J} = t_f - t_0$ with terminal cost $\Phi = 0$ and running cost $L = 1$. The Hamiltonian (2.15) simplifies to:

$$H = 1 + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{2.21}$$

Correspondingly, the costate (2.17) reduces to:

$$\dot{\boldsymbol{\lambda}}^*(t) = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda}^*(t) \tag{2.22}$$

For autonomous systems where the dynamics $\mathbf{f}$ do not depend explicitly on time, and with constant running cost $L = 1$, the Hamiltonian exhibits temporal independence: $\partial H/\partial t = 0$. Along the optimal trajectory, the total time derivative of the Hamiltonian is:

$$\frac{dH}{dt} = \frac{\partial H}{\partial \mathbf{u}} \cdot \dot{\mathbf{u}} \tag{2.23}$$

If the control appears linearly in the Hamiltonian (as is the case for spacecraft dynamics, shown in section 2.5), $\partial H/\partial \mathbf{u}$ is independent of $\mathbf{u}$, and thus the Hamiltonian remains constant along continuous segments of the optimal trajectory. At switching times where the control is discontinuous, the Hamiltonian remains continuous [14]. Consequently:

$$H(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), t) = H_0 = \text{constant} \tag{2.24}$$

The transversality condition (2.20) for free terminal time with $\Phi = 0$ and terminal constraints that do not depend explicitly on time ($\partial\boldsymbol{\Psi}/\partial t = 0$) requires $H(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), \boldsymbol{\lambda}^*(t_f), t_f) = 0$. Therefore, $H = 0$ along the entire optimal trajectory:

$$1 + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*, t) = 0 \tag{2.25}$$

The bang-bang structure of time-optimal control emerges from the linear appearance of control in the Hamiltonian. For spacecraft attitude dynamics, the dynamics $\mathbf{f}$ can be partitioned such that the Hamiltonian assumes the form:

$$H = H_0(\mathbf{x}, \boldsymbol{\lambda}, t) + \sum_{i=1}^{m} S_i(t) u_i \tag{2.26}$$

where $S_i(t)$ are switching functions extracted from the costate-dynamics coupling, u_i denotes the i-th control component, and H_0 denotes the control-independent portion of the Hamiltonian. The minimization condition (2.18) requires minimizing $\sum_{i=1}^{m} S_i(t)u_i$ subject to $|u_i| \leq u_{i,\max}$. This is achieved by:

$$u_i^*(t) = \begin{cases} u_{i,\max} & \text{if } S_i(t) < 0 \\ u_{i,\min} & \text{if } S_i(t) > 0 \\ \text{undefined} & \text{if } S_i(t) = 0 \end{cases} \tag{2.27}$$

For symmetric control bounds ($u_{i,\min} = -u_{i,\max}$), this reduces to $u_i^*(t) = -u_{i,\max}\,\mathrm{sgn}(S_i(t))$, confirming the bang-bang structure wherein each control component saturates at its extreme values.

When $S_i(t) = 0$ at an isolated instant, a switching occurs. If $S_i(t) = 0$ over a finite time interval, a singular arc exists. Singular arcs require higher-order analysis through successive time derivatives of the switching function ($S_i = \dot{S}_i = \ddot{S}_i = \cdots = 0$) and satisfaction of the generalized Legendre-Clebsch condition for optimality [15]. However, for spacecraft attitude control with torque-limited actuators, singular arcs do not occur due to the linear appearance of control in the dynamics, sufficient controllability of the system, and the time-minimization objective that drives controls to their extremes [3].

2.3 Trajectory Optimization Methods

Two primary methodological frameworks for solving OCPs exist: indirect methods that apply PMP to obtain necessary conditions for optimality, and direct methods that discretize the continuous-time problem into a finite-dimensional NLP. While indirect methods provide theoretical insight into optimal solution structure through explicit computation of costate trajectories, they suffer from severe initialization sensitivity and narrow convergence basins. Direct transcription prove more robust to initialization at the cost of solution accuracy. This section examines both these approaches' relevance to real-time time-optimal control.

2.3.1 Indirect Methods

Indirect methods derive from the necessary conditions established by PMP in section 2.2.2. The approach transforms the optimal control problem into a TPBVP by introducing costate variables and applying the optimality conditions (2.16)–(2.20). The resulting TPBVP comprises forward integration of the state equations (2.16) from initial conditions $\mathbf{x}(t_0) = \mathbf{x}_0$, backward integration of the costate equations (2.17) from terminal conditions specified by the transversality conditions (2.19), and pointwise determination of optimal control through the minimization condition (2.18).

Several computational techniques exist for solving the TPBVP. Single shooting methods guess initial costate values $\boldsymbol{\lambda}(t_0)$, integrate the coupled state-costate system forward in time, and iteratively adjust $\boldsymbol{\lambda}(t_0)$ until terminal conditions are satisfied. Multiple shooting partitions the time horizon into segments, integrating each segment independently and enforcing continuity constraints at segment boundaries. Collocation methods approximate state and costate trajectories using polynomial basis functions and minimize residuals of the differential equations at collocation points. Continuation methods gradually transition from a simpler, more easily solved problem to the target problem by varying a homotopy parameter [15].

Indirect methods offer several theoretical and computational advantages. Upon convergence, they satisfy the necessary conditions for optimality exactly, achieving high solution accuracy. The formulation provides direct insight into optimal solution structure through the costate trajectories and switching functions. For problems where the bang-bang control structure is known *a priori*, the switching function framework naturally handles control discontinuities. When analytical expressions for costate derivatives are available, evaluation of the costate equation (2.17) can be computationally efficient. Furthermore, single shooting methods optimize only the initial costate values, resulting in lower-dimensional search spaces compared to discretizing entire state and control trajectories.

However, indirect methods suffer from severe practical limitations that have restricted their adoption for real-time optimal control. The TPBVP exhibits extreme sensitivity to initial guesses for the costate variables, with convergence requiring values within a typically small basin of attraction [15]. The costate variables lack direct physical interpretation, making intelligent initialization difficult. Incorporating path constraints or inequality constraints substantially complicates the necessary conditions, requiring complementarity conditions and active-set determination. Extension to multiple-phase problems with discontinuous state or control profiles increases formulation complexity. The limited robustness to problem variations necessitates re-tuning for each new configuration.

Despite their theoretical elegance and potential for high-accuracy solutions, these computational challenges and initialization sensitivities have limited the adoption of indirect methods for practical real-time systems. Indirect methods remain most appropriate for problems where solution structure is well-understood, good initialization is available through physical insight or continuation from related problems, high accuracy is paramount, and computation time is not critical.

2.3.2 Direct Transcription Methods

Direct transcription methods discretize the continuous-time optimal control problem into a finite-dimensional NLP problem. This approach relaxes the strict necessary conditions of optimality required by indirect methods, instead seeking an approximate solution to the discretized problem that can be improved through mesh refinement. The fundamental advantage of direct methods lies in their ability to leverage mature NLP solver infrastructure developed for general-purpose constrained optimization [15]. Direct transcription has become the dominant approach for real-time optimal control applications due to larger convergence basins, easier initialization, and natural accommodation of path constraints.

Direct Shooting

Direct shooting discretizes only the control trajectory, obtaining state trajectories through forward integration of the dynamics. The time interval $[t_0, t_f]$ is partitioned into a grid of N points, $\{t_0, t_1, \ldots, t_N = t_f\}$, and control values $\mathbf{u}_k$ at grid points k become optimization variables. Between grid points, the control is parameterized as piecewise constant, piecewise linear, or using higher-order polynomial basis functions. For each candidate control trajectory, the dynamics (2.13) are integrated forward from the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ to obtain the state trajectory. The objective function and constraints are then evaluated based on the resulting states.

Direct shooting offers the advantage of requiring a low amount of decision variables, as only control values are optimized while states are implicitly determined through integration. The method is straightforward to implement using standard ODE integrators. However, the approach generates dense constraint Jacobians, as each control variable influences all subsequent states through the integration process. Direct shooting provides no direct control over intermediate state values, potentially leading to large constraint violations during optimization iterations. These limitations have led to preference for alternative direct transcription methods that discretize both states and controls.

Direct Multiple Shooting

Direct multiple shooting partitions the time horizon into N segments $[t_k, t_{k+1}]$ for $k = 0, 1, \ldots, N-1$. Both state and control variables at segment boundaries become optimization variables. Within each segment, the dynamics are integrated forward from the initial state $\mathbf{x}_k$ using control $\mathbf{u}_k$ to obtain a predicted terminal state. Denoting the state obtained by integrating the dynamics from $\mathbf{x}_k$ with control $\mathbf{u}_k$ over interval $[t_k, t_{k+1}]$ as $\boldsymbol{\gamma}_k(\mathbf{x}_k, \mathbf{u}_k)$, continuity constraints enforce consistency between segments:

$$\mathbf{x}_{k+1} - \boldsymbol{\gamma}_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \quad k = 0, 1, \ldots, N-1 \tag{2.28}$$

Multiple shooting generates sparse NLP structures characterized by block-diagonal Jacobian matrices and banded Hessian matrices. Each continuity constraint (2.28) couples only adjacent segments, resulting in a block-tridiagonal structure. This sparsity can be exploited by specialized linear solvers within the NLP algorithm, substantially reducing computational cost for large-scale problems [15]. The method exhibits favorable convergence properties, as intermediate state variables can be initialized to physically reasonable values independently of the control trajectory. Multiple shooting naturally enables parallelization: segment integrations can be performed concurrently on multi-core processors or distributed computing systems. The computational efficiency for long time horizons and natural decomposition make multiple shooting well-suited for real-time applications.

Numerical Integration Schemes

The choice of integration method for propagating dynamics within shooting intervals significantly affects accuracy, stability, and computational cost. Integration schemes are broadly classified as explicit or implicit methods, each exhibiting distinct computational characteristics and stability properties relevant to spacecraft attitude dynamics.

Explicit integration methods compute the state at the next time step using only information from the current and previous time steps. The forward Euler method represents the simplest explicit scheme. For the dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ from equation (2.13), forward Euler advances the state from time t_k to $t_{k+1} = t_k + \Delta t$ using:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\, \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k) \tag{2.29}$$

where Δt denotes the time step. The method possesses first-order accuracy with local truncation error $O(\Delta t^2)$ and global error $O(\Delta t)$ [16]. Forward Euler exhibits a bounded stability region: for the scalar test equation $\dot{x} = \mu x$ with $\mu \in \mathbb{C}$, the method is absolutely stable for $|1 + \Delta t \mu| \leq 1$. This restriction limits the maximum stable step size, particularly for stiff systems characterized by widely separated timescales where eigenvalues with large negative real parts impose severe stability constraints despite the solution evolving primarily on slower timescales governed by eigenvalues closer to zero [16].

The classical fourth-order Runge-Kutta method (RK4) substantially improves accuracy through multiple function evaluations per step. The method computes four intermediate slopes and combines them to achieve fourth-order accuracy:

$$\begin{aligned}
\boldsymbol{\kappa}_1 &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k) \\
\boldsymbol{\kappa}_2 &= \mathbf{f}\left(\mathbf{x}_k + \frac{\Delta t}{2}\boldsymbol{\kappa}_1, \mathbf{u}_k, t_k + \frac{\Delta t}{2}\right) \\
\boldsymbol{\kappa}_3 &= \mathbf{f}\left(\mathbf{x}_k + \frac{\Delta t}{2}\boldsymbol{\kappa}_2, \mathbf{u}_k, t_k + \frac{\Delta t}{2}\right) \\
\boldsymbol{\kappa}_4 &= \mathbf{f}(\mathbf{x}_k + \Delta t \boldsymbol{\kappa}_3, \mathbf{u}_k, t_k + \Delta t) \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{\Delta t}{6}\left(\boldsymbol{\kappa}_1 + 2\boldsymbol{\kappa}_2 + 2\boldsymbol{\kappa}_3 + \boldsymbol{\kappa}_4\right)
\end{aligned} \tag{2.30}$$

RK4 achieves local truncation error $O(\Delta t^5)$ and global error $O(\Delta t^4)$ [16], enabling accurate integration with substantially larger step sizes than forward Euler for smooth problems. The stability region of RK4 is larger than forward Euler but remains bounded, restricting step size for stiff systems.

Explicit methods offer straightforward implementation requiring only function evaluations of $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ without solving nonlinear equations. Parallelization across multiple shooting segments is trivial, as each segment integration proceeds independently. However, explicit methods impose step size restrictions determined by stability rather than accuracy considerations.

Implicit integration methods evaluate the dynamics function at the next time step, requiring solution of a system of algebraic equations at each integration step. The backward Euler method represents the simplest implicit scheme:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\, \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_{k+1}) \tag{2.31}$$

The dependence of the right-hand side on the unknown state $\mathbf{x}_{k+1}$ necessitates solving the implicit equation:

$$\mathbf{x}_{k+1} - \mathbf{x}_k - \Delta t\, \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_{k+1}) = \mathbf{0} \tag{2.32}$$

This nonlinear system is typically solved iteratively using Newton's method or simplified Newton iterations. Despite sharing first-order accuracy with forward Euler ($O(\Delta t)$ global error), backward Euler possesses fundamentally different stability properties. The method is A-stable: for the scalar test equation $\dot{x} = \mu x$, backward Euler is absolutely stable for all $\Delta t > 0$ when $\mathrm{Re}(\mu) < 0$ [16]. This unconditional stability enables substantially larger time steps for stiff problems without numerical instability.

The computational cost per step for implicit methods exceeds that of explicit methods due to the iterative solution of equation (2.32). Each Newton iteration requires evaluation of the Jacobian matrix $\partial \mathbf{f}/\partial \mathbf{x}$ and solution of a linear system. However, the ability to use much larger step sizes can offset the per-step cost, particularly for problems where accuracy requirements permit coarse temporal resolution but stability constraints would restrict explicit methods to prohibitively small steps.

Adaptive step-size methods automatically adjust step sizes based on local error estimates, balancing accuracy and computational cost. This gain comes at the cost of higher complexity. Nevertheless, there exist options that try to reduce this issue such as the CVODES solver from the SUNDIALS library. It implements variable-order, variable-step Adams-Moulton and backward differentiation methods with automatic stiffness detection [17, 18]. CVODES provides error-controlled integration that maintains user-specified accuracy tolerances through adaptive step-size and order selection. The solver includes built-in capability for forward and adjoint sensitivity analysis [19], and GPU exploiting features [20], enabling efficient computation of gradients, required for NLP solutions, without finite-difference approximations.

Direct Collocation

Direct collocation methods discretize both state and control trajectories at collocation points and approximate trajectories using polynomial representations. The dynamics are enforced at collocation points through defect constraints that penalize deviations from the differential equation. Collocation methods differ primarily in their choice of polynomial basis functions and collocation point locations, which determine approximation accuracy, computational efficiency, and suitability for different boundary condition types.

Hermite-Simpson collocation represents a widely-used local collocation method that partitions the time interval $[t_0, t_f]$ into N subintervals $[t_k, t_{k+1}]$ for $k = 0, 1, \ldots, N-1$. Within each interval, the state trajectory is approximated by a cubic Hermite polynomial constructed from endpoint values and derivatives. The midpoint $t_{k+1/2} = (t_k + t_{k+1})/2$ serves as the collocation point where dynamics are enforced. Let $\Delta t_k = t_{k+1} - t_k$ denote the interval length. The state approximation satisfies endpoint interpolation conditions:

$$\begin{aligned} \mathbf{x}(t_k) &= \mathbf{x}_k \\ \mathbf{x}(t_{k+1}) &= \mathbf{x}_{k+1} \end{aligned} \tag{2.33}$$

and derivative conditions:

$$\begin{aligned} \dot{\mathbf{x}}(t_k) &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k) \\ \dot{\mathbf{x}}(t_{k+1}) &= \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_{k+1}) \end{aligned} \tag{2.34}$$

The cubic Hermite interpolant yields the midpoint state approximation:

$$\mathbf{x}_{k+1/2} = \frac{1}{2}(\mathbf{x}_k + \mathbf{x}_{k+1}) + \frac{\Delta t_k}{8}\left[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k) - \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_{k+1})\right] \tag{2.35}$$

The collocation constraint enforces the dynamics at the midpoint:

$$\varsigma_k = \mathbf{f}(\mathbf{x}_{k+1/2}, \mathbf{u}_{k+1/2}, t_{k+1/2}) - \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta t_k} = \mathbf{0} \tag{2.36}$$

This defect constraint (2.36) can equivalently be written using Simpson's quadrature rule for the integral of the dynamics:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\Delta t_k}{6}\left[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k) + 4\mathbf{f}(\mathbf{x}_{k+1/2}, \mathbf{u}_{k+1/2}, t_{k+1/2}) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_{k+1})\right] \tag{2.37}$$

Hermite-Simpson collocation achieves fourth-order local accuracy: the local truncation error is $O(\Delta t_k^5)$ and the global error is $O(\Delta t^4)$ where $\Delta t = \max_k \Delta t_k$ [15]. The method generates sparse constraint structures, as each defect constraint (2.36) couples only three consecutive state variables $\{\mathbf{x}_k, \mathbf{x}_{k+1/2}, \mathbf{x}_{k+1}\}$. This sparsity enables efficient computation for large-scale problems. The local nature of the approximation facilitates adaptive mesh refinement: intervals can be subdivided independently based on local error estimates without affecting the approximation quality in other regions. Hermite-Simpson collocation balances accuracy, computational efficiency, and implementation simplicity, establishing it as a popular choice for aerospace trajectory optimization [15].

Pseudospectral methods employ global polynomial approximations over the entire time interval, in contrast to the piecewise local polynomials of Hermite-Simpson collocation. The state trajectory is approximated as:

$$\mathbf{x}(t) \approx \sum_{i=0}^{N} \mathbf{x}_i \, \mathcal{L}_i(t) \tag{2.38}$$

where $\mathcal{L}_i(t)$ are Lagrange interpolating polynomials satisfying $\mathcal{L}_i(t_j) = \delta_{ij}$ (Kronecker delta), and $\{t_0, t_1, \ldots, t_N\}$ are collocation points. The choice of collocation points fundamentally determines the method's convergence properties and computational characteristics.

Pseudospectral methods achieve spectral convergence: for sufficiently smooth solutions, the approximation error decreases faster than any polynomial in the number of collocation points. Specifically, if the solution possesses a continuous derivatives, the error decreases as $O(p^{-a})$ where p is the polynomial degree [15]. This rapid convergence enables high accuracy with relatively few discretization points compared to low-order local methods.

- **Legendre-Gauss (LG):** Collocation points $\{w_1, \ldots, c_p\}$ comprise the p roots of the Legendre polynomial $\mathcal{P}_p(c)$, lying strictly interior to $[-1, 1]$. Endpoints are excluded, requiring direct specification of initial conditions and extrapolation for terminal constraints. Quadrature weights enable exact integration of polynomials up to degree $2p - 1$. Best suited for problems with free terminal states where boundary extrapolation is acceptable [15].
- **Chebyshev-Gauss-Lobatto (CGL):** Collocation points $c_k = \cos(\pi k/p)$ for $k = 0, \ldots, p$ comprise the $p + 1$ extrema of the Chebyshev polynomial $\mathcal{C}_p(c)$, including both endpoints $c_0 = 1$ and $c_p = -1$. Quadrature accuracy matches LG at degree $2p-1$. Direct enforcement of both initial and terminal boundary conditions without extrapolation makes CGL particularly suitable for problems with constrained initial and final states [15].
- **Legendre-Gauss-Radau (LGR):** Collocation points consist of $c_0 = -1$ (initial time) plus p roots of $(\mathcal{P}_p(c) + \mathcal{P}_{p+1}(c))/(1 + c)$, yielding $p+1$ points with initial endpoint included but final endpoint excluded. Direct initial condition enforcement with terminal state extrapolation. Quadrature weights provide exact integration up to degree $2p$, one degree higher than LG or CGL with equivalent interior point count. The asymmetric structure and superior quadrature accuracy make LGR advantageous for initial value problems and is employed by GPOPS-II [21] and CGPOPS [7].

Table 2.2 summarizes the key differences.

Table 2.2 Comparison of pseudospectral collocation methods for trajectory optimization.

Property	**LG**	**CGL**	**LGR**
Number of points	p	$p+1$	$p+1$
Initial point included	No	Yes	Yes
Final point included	No	Yes	No
Quadrature accuracy	$2p-1$	$2p-1$	$2p$
Initial BC enforcement	Direct	Direct	Direct
Terminal BC enforcement	Extrapolation	Direct	Extrapolation

Pseudospectral methods generate dense NLP structures due to high connectivity between variables through the polynomial basis functions. This density increases computational cost for sparse linear algebra operations within NLP solvers, but can be compensated by the lower amount of discretization points required to achieve convergence, specially for smooth, continuously differentiable optimal trajectories. However, collocation methods may experience difficulty accurately representing bang-bang controls, as the polynomial approximation poorly captures discontinuous switching behavior. Increasing polynomial order or refining the mesh near switching points can mitigate this limitation but increases problem dimensionality.

Optimization Method Selection

Direct transcription methods offer several critical advantages over indirect methods for practical trajectory optimization. The convergence basins are substantially larger, enabling successful solution from cruder initial guesses. Initialization is more intuitive, as state and control variables possess direct physical meaning—unlike costate variables in indirect methods. Linear interpolation between boundary conditions, physically motivated trajectories, or zero controls often suffice as initial guesses for moderately nonlinear problems.

Path constraints and inequality constraints are naturally incorporated as additional constraints in the NLP formulation. Multiple-phase problems with discontinuous states or controls are handled seamlessly by introducing separate variables and constraints for each phase. Direct methods exhibit robustness to problem variations: modifying spacecraft parameters, constraint bounds, or boundary conditions requires no fundamental reformulation. The availability of mature, well-tested NLP solvers such as IPOPT [22] and SNOPT [23] enables reliable solution of large-scale problems. These advantages have established direct transcription as the dominant choice for real-time optimal control implementations.

2.4 Optimization Variables

Beyond the choice of the transcription method and attitude parameterization, another strategic decision in problem formulation significantly affects computational performance and solution quality. This choice concerns which specific parameters are used as the optimization variables. In the previous sections it was established that in an OCP it is solved for the optimal control vector $\mathbf{u}_*(t)$ which minimizes a certain cost functional $\mathcal{J}$. Intuitively, said vector is normally defined as the available actuation in a system. This approach is referred to as direct dynamics formulation and is the most commonly used in the available literature. Alternatively, one can resort to inverse dynamics formulation, where system's state $\mathbf{x}(t)$ becomes the optimization variable and the inverse dynamics equations are solved to obtain the required actuation for following the obtained state trajectory. The choice between these formulations affects how constraints are enforced, how the optimization problem is structured, and the computational efficiency of gradient evaluations. This section describes these two approaches and provides the means for an informed problem-specific selection.

2.4.1 Direct Dynamics Formulation

Direct dynamics follows the conventional formulation where control inputs are specified and system states are determined through forward integration of the governing equations. The control trajectory $\mathbf{u}(t)$ serves as the decision variable vector in the optimization problem. State trajectories are obtained by integrating the dynamics constraint $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ from equation (2.13) subject to initial conditions $\mathbf{x}(t_0) = \mathbf{x}_0$. Control bounds $u_{i,\min} \leq u_i \leq u_{i,\max}$ are naturally enforced as box constraints in the NLP formulation.

However, boundary conditions on states must be explicitly enforced through equality or inequality constraints in the optimization problem. For problems with fixed terminal states $\mathbf{x}(t_f) = \mathbf{x}_f$, these constraints couple the entire trajectory through the dynamics integration. Path constraints on states, when present, similarly require explicit constraint formulation. The direct dynamics approach represents the standard formulation employed in most trajectory optimization implementations and integrates naturally with direct transcription frameworks discussed in section 2.3.2.

Within direct dynamics formulations, two fundamentally different approaches exist for parameterizing control trajectories: the more common full control trajectory optimization, where the whole control vector is used as the decision variable, allowing for full solution structure freedom, and bang-bang parameterization, know as Switch Time Optimization (STO), which exploits problems where it is known a priori that the optimal control trajectory as a bang-bang structure.

Full trajectory optimization

Full trajectory optimization treats control values at each discretization point as independent decision variables. No *a priori* structure is imposed on the control profile, granting the optimizer complete freedom to determine the control history. For a problem discretized at N points with m control inputs, the decision variable count is $O(N \times m)$. This general approach accommodates arbitrary optimal control structures, including singular arcs, continuously varying controls, and complex switching patterns.

However, full trajectory optimization requires the NLP solver to discover the bang-bang structure inherent in time-optimal problems: a discovery that may require many iterations and careful initialization. The high dimensionality increases computational expense, particularly for fine discretizations needed to resolve control discontinuities accurately. The optimizer must determine both switching times and control magnitudes, despite the knowledge from section 2.2.3 that optimal controls saturate at bounds. Grid resolution must be sufficiently fine to capture rapid switching, potentially requiring hundreds of discretization points for problems with multiple switches per axis.

Bang-bang parameterization

The STO approach exploits the theoretical result from equation (2.27) that time-optimal control with bounded authority exhibits bang-bang structure in the absence of singular arcs. Rather than optimizing control magnitudes at each time point, this approach parameterizes the control trajectory by switching times $\mathbf{t}_i = [t_i^{(1)}, t_i^{(2)}, \dots, t_i^{(y)}]$ and initial control directions $s_i \in \{-1, +1\}$ for each control channel i. Between switching times, each control component remains at its extreme value with sign reversals at each subsequent switching time. For problems with y switches per control axis, STO represents a dramatic dimensionality reduction compared to full trajectory optimization: $O(y \times m)$ versus $O(N \times m)$ where typically $y \ll N$. This reduction substantially decreases computational cost and can improve convergence reliability by eliminating the need for the solver to discover the bang-bang structure.

However, STO introduces implementation complexities beyond this dimensionality advantage. The discrete control direction variables $s_i \in \{-1, +1\}$ present incompatibility with gradient-based NLP solvers, which operate on continuous decision variables. For m independent control axes, 2^m possible initial direction combinations exist. Different approaches address this discrete optimization component:

- **Exhaustive enumeration** solves the continuous switching time problem for all 2^m sign combinations and selects the globally optimal solution, guaranteeing global optimality at multiplicative computational cost.
- **Relaxation** to continuous variables $s_i \in [-1, +1]$ followed by post-processing conversion to ± 1 enables gradient-based optimization but provides no optimality guarantees for the converted solution and can lead to solver instabilities for values close to the midpoint due to constant switching.
- **Heuristic determination** based on initial and final state analysis. May fail to identify globally optimal sign configurations [3, 24].

The switching time variables must satisfy temporal ordering constraints $t_0 < t_i^{(1)} < t_i^{(2)} < \cdots < t_i^{(y)} < t_f$ to maintain physical consistency. These $O(y)$ inequality constraints can become active during optimization, potentially complicating convergence. Constraint violations during intermediate iterations produce non-physical control histories that degrade solver performance.

Reconstruction of the continuous-time control trajectory from discrete switching times involves non-smooth operations incompatible with gradient-based optimization. The control law for each axis i is:

$$u_i(t, \mathbf{t}_i, s_i) = s_i \cdot (-1)^{C_{\text{sw}}(t, \mathbf{t}_i)} \cdot u_{i,\max} \tag{2.39}$$

where $C_{\text{sw}}(t, \mathbf{t}_i) = |\{t_i^{(j)} : t_i^{(j)} < t\}|$ counts switches occurring before time t. This operation introduces discontinuities at each switching instant, making the controls non-differentiable and violating smoothness requirements of gradient-based NLP solvers [15].

To address this non-differentiability, practical implementations often employ smoothing techniques to approximate the discontinuous switching behavior through differentiable functions like sigmoid or hyperbolic tangent functions. This regularization represents a fundamental compromise between fidelity to theoretical bang-bang structure and computational tractability [25].

The requirement for *a priori* specification of switch count remains a fundamental limitation. Insufficient switch count produces suboptimal solutions unable to reach the true optimum; excessive switch count may cause convergence failure or generate spurious switches. Iterative approaches with incrementally increasing switch counts enable empirical determination of optimal v at multiplicative computational expense.

2.4.2 Inverse Dynamics Formulation

Inverse dynamics reverses the problem structure by treating the state trajectory $\mathbf{x}(t)$ as the primary decision variable and computing required control inputs through inversion of the system dynamics. The state trajectory is parameterized using M basis function expansions such as B-splines, polynomial series, or Fourier series. Common parameterizations include:

$$\mathbf{x}(t) = \sum_{j=0}^{M} \mathbf{b}_j \chi_j(t) \tag{2.40}$$

where $\chi_j(t)$ are basis functions and $\mathbf{b}_j$ are coefficient vectors serving as optimization variables.

The time derivative of the parameterized state trajectory is computed analytically:

$$\dot{\mathbf{x}}(t) = \sum_{j=0}^{M} \mathbf{b}_j \dot{\chi}_j(t) \tag{2.41}$$

For systems where the dynamics can be written in the affine control form:

$$\dot{\mathbf{x}} = \mathbf{f}_0(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} \tag{2.42}$$

where $\mathbf{f}_0(\mathbf{x})$ represents drift dynamics and $\mathbf{G}(\mathbf{x}) \in \mathbb{R}^{n\times m}$ is the control influence matrix, the required control can be computed through algebraic inversion:

$$\mathbf{u}(t) = \mathbf{G}(\mathbf{x}(t))^{-1} \left[\dot{\mathbf{x}}(t) - \mathbf{f}_0(\mathbf{x}(t))\right] \tag{2.43}$$

provided that $\mathbf{G}(\mathbf{x})$ is full rank. For underactuated systems where $m < n$, the pseudoinverse $\mathbf{G}^\dagger$ replaces the standard inverse, though this introduces additional complexity in ensuring trajectory feasibility.

Inverse dynamics formulation provides significant advantages for boundary condition satisfaction. State boundary conditions at initial and terminal times can be enforced directly through the basis function coefficients in equation (2.40). For polynomial or spline representations, linear equality constraints on coefficients suffice to ensure $\mathbf{x}(t_0) = \mathbf{x}_0$ and $\mathbf{x}(t_f) = \mathbf{x}_f$ regardless of optimization progress [26]. This automatic satisfaction of state boundary conditions eliminates a class of potentially difficult nonlinear constraints from the NLP formulation.

Analytical differentiation of smooth basis functions provides exact expressions for state derivatives without numerical differentiation errors. Higher-order derivatives required for acceleration-dependent dynamics can be computed symbolically from the basis functions. Gradient computation for the optimization algorithm benefits from the explicit relationship between decision variables (basis coefficients) and the objective function, enabling efficient symbolic or automatic differentiation of the entire problem. Nevertheless, solution quality is dependent on how well the selected basis function approximate the real system dynamics. Additionally, inverse dynamics introduces the fundamental challenge of ensuring control feasibility. The computed controls from equation (2.43) may violate actuator constraints for arbitrary choices of trajectory parameterization. Aggressive trajectories or poorly conditioned basis function selections can result in required control magnitudes exceeding physical actuator capabilities.

For spacecraft reorientation problems, Euler's rotational equations (2.6) provide the affine control structure with control torques entering linearly and equation (2.4) defines the inverse dynamics. However, the bang-bang nature of the time-optimal solution drives the required torques to the actuator limits, necessitating careful selection of basis functions and iterative refinement to maintain control feasibility throughout the full trajectory.

2.4.3 Selection Criteria and Trade-Offs

The choice between formulation strategies depends on problem characteristics and computational requirements. For problems where the optimal switching structure is known from prior analysis or analytical results bang-bang parameterization offers substantial computational advantages. When switching structure is unknown or problem-dependent, full trajectory optimization provides greater generality at higher computational cost. The selection between direct and inverse dynamics formulations involves trade-offs between constraint handling complexity, problem dimensionality, and computational efficiency. Direct dynamics naturally handles control constraints but requires explicit state boundary condition enforcement through potentially nonlinear constraints. Inverse dynamics automatically satisfies state boundary conditions but introduces complexity in control feasibility enforcement. For problems where state boundary conditions are numerous or complex, inverse dynamics may reduce overall problem difficulty despite the control feasibility challenge. Conversely, when control bounds are tight and state boundary conditions are simple, direct dynamics often proves more efficient [26, 24].

Additional factors include available computational resources (offline optimization permits more expensive methods), required solution accuracy (high-fidelity solutions may justify increased computational cost), tolerance for implementation complexity (inverse dynamics requires careful parameterization design), and real-time performance requirements (necessitating efficient formulations and potentially structure-exploiting methods). The formulation strategy should align with the specific application requirements and available computational infrastructure, balancing optimality, reliability, and computational efficiency.

2.5 Time-Optimal Spacecraft Attitude Maneuvers

The application of optimal control theory to spacecraft reorientation problems combines the rigid body dynamics established in section 2.1 with the theoretical framework presented in sections 2.2 to 2.4. Time-optimal spacecraft attitude control has evolved from early analytical studies on simplified configurations to sophisticated numerical methods capable of handling arbitrary maneuvers in real time. The problem exhibits significant complexity arising from nonlinear gyroscopic coupling, nonconvex optimization landscapes, and the bang-bang control structure predicted by PMP. Historical developments have established analytical characterizations for specific cases while revealing fundamental limitations that necessitate numerical solution approaches. This section traces the historical evolution of time-optimal spacecraft reorientation research, presents the complete mathematical problem formulation, derives the optimal control characterization through application of PMP, and analyzes the factors contributing to problem complexity and computational difficulty.

2.5.1 Historical Development

The theoretical foundations for time-optimal spacecraft attitude control trace back to the establishment of PMP in 1962 [14], which provided the first rigorous framework for characterizing optimal control trajectories subject to bounded control authority. The principle demonstrated that for systems where control appears linearly and no singular arcs exist, optimal trajectories exhibit bang-bang structure, i.e. switching between extreme control values a finite number of times. These early theoretical developments focused on simplified spacecraft configurations to enable analytical tractability.

Initial spacecraft attitude studies in the 1960s and 1970s concentrated on idealized cases: symmetric spacecraft with three equal principal moments of inertia ($J_1 = J_2 = J_3$), rest-to-rest maneuvers with zero initial and final angular velocities ($\boldsymbol{\omega}_0 = \boldsymbol{\omega}_f = \mathbf{0}$), and systems with zero net angular momentum. These simplifications enabled analytical or semi-analytical solution techniques but provided limited insight into general spacecraft configurations. A common assumption during this period was that time-optimal maneuvers would follow eigenaxis rotations: a single-axis rotations aligned with a principal axis of inertia. This assumption appeared physically intuitive, as eigenaxis rotations avoid gyroscopic coupling and enable independent control of angular velocity about a single axis.

The 1990s witnessed key breakthroughs that fundamentally altered understanding of time-optimal spacecraft control. Bilimoria and Wie's seminal 1993 contribution [3] demonstrated that eigenaxis maneuvers are not time-optimal for spacecraft with independent three-axis control authority, $|\tau_i| \leq \tau_{i,\max}$, also known as cubically constrained control. By allowing simultaneous torque application about multiple axes, the spacecraft can exploit gyroscopic coupling to achieve faster reorientations than single-axis rotations. Their analysis of symmetric spacecraft revealed that optimal switching structures depend critically on maneuver angle magnitude. For rotations exceeding 72°, they identified a specific five-switch structure as optimal. For rotations below 72°, they initially proposed a seven-switch structure, though this was later refined by Bai and Junkins [4]. This work represented a mindset shift: rather than avoiding gyroscopic effects, optimal control actively exploits coupling dynamics to minimize maneuver time. When the total torque magnitude is constrained instead of each individual element, $||\boldsymbol{\tau}|| \leq \tau_{\max}$, known as spherically constrained control, eigenaxis rotation remains the time-optimal maneuver.

Byers extended Bilimoria and Wie's results to fully asymmetric spacecraft in 1996 [9], analyzing asymmetric spacecraft configurations where all three principal moments of inertia are distinct ($J_1 \neq J_2 \neq J_3$). Gyroscopic coupling terms in Euler's equations (2.5) introduce additional complexity, as each axis influences the dynamics of the others through products of angular velocity components. Byers characterized how inertia ratios affect both switching times and bang-bang control profiles, demonstrating that spacecraft asymmetry influences not only the optimal control structure but also the convergence properties of numerical solution methods. Highly asymmetric configurations with large disparities in principal moments exhibit more complex switching patterns and present greater challenges for gradient-based optimization algorithms.

Recent advances from the 2000s through 2010s have refined understanding of small-angle maneuvers and enabled solution of increasingly general problems. Bai and Junkins discovered in 2009 [4] that rotations below 72° admit an improved six-switch structure, contradicting the earlier seven-switch proposal and demonstrating that even well-studied problems can yield new optimal solutions through careful analysis. Concurrently, developments in direct transcription methods and nonlinear programming solvers have enabled numerical solution of time-optimal problems for arbitrary spacecraft configurations, including track-to-track maneuvers with non-zero initial and final angular velocities. Modern research emphasizes efficient algorithms suitable for onboard real-time implementation, including structure-exploiting optimization methods and intelligent initialization strategies that improve convergence reliability.

The evolution from analytical characterization of simplified cases to fully numerical approaches for general configurations reflects both increasing computational capabilities and recognition of fundamental analytical intractability for arbitrary maneuvers. Contemporary focus has shifted from theoretical analysis to practical deployment: developing algorithms that balance near-optimality with computational efficiency sufficient for real-time trajectory generation aboard resource-constrained spacecraft processors.

2.5.2 Problem Formulation

The complete optimal control problem statement for time-optimal spacecraft reorientation integrates the attitude dynamics from section 2.1 with the theoretical framework from section 2.2. When quaternions are used for attitude parametrization, the state vector looks like:

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\omega} \end{bmatrix} \in \mathbb{R}^7 \tag{2.44}$$

The control vector consists of torques applied about the principal axes:

$$\mathbf{u} = \boldsymbol{\tau} \tag{2.45}$$

The system dynamics combine the quaternion kinematics from equation (2.8) with Euler's rotational equations from equation (2.6):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{q}, \boldsymbol{\omega}, \boldsymbol{\tau}) = \begin{bmatrix} \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \\ \mathbf{J}^{-1}[\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}] \end{bmatrix} \tag{2.46}$$

Actuator limitations define the admissible control set:

$$\mathcal{U} = \{\boldsymbol{\tau} \in \mathbb{R}^3 : |\tau_i| \leq \tau_{i,\max},\ i = 1, 2, 3\} \tag{2.47}$$

where $\tau_{i,\max}$ represents the maximum torque magnitude available about the i-th principal axis. The constraints may be uniform ($\tau_{1,\max} = \tau_{2,\max} = \tau_{3,\max}$) or axis-dependent, reflecting different actuator capabilities about different axes.

Boundary conditions specify the initial and terminal spacecraft states. The initial condition prescribes both attitude and angular velocity:

$$\mathbf{x}(t_0) = \begin{bmatrix} \mathbf{q}_0 \\ \boldsymbol{\omega}_0 \end{bmatrix} \tag{2.48}$$

Similarly, the terminal condition specifies the desired final state:

$$\mathbf{x}(t_f) = \begin{bmatrix} \mathbf{q}_f \\ \boldsymbol{\omega}_f \end{bmatrix} \tag{2.49}$$

Due to the double cover property of the quaternion representation, both $\mathbf{q}_f$ and $-\mathbf{q}_f$ correspond to the same physical terminal orientation. However, these two mathematically equivalent representations yield fundamentally different optimal trajectories. If the solver does not account for this ambiguity, the optimization may converge to an undesirable equilibrium that requires rotation through angles exceeding 180°, a phenomenon known as quaternion unwinding. To ensure convergence to the time-optimal solution, the terminal quaternion must be selected such that:

$$\mathbf{q}_0^T \mathbf{q}_f \geq 0 \tag{2.50}$$

This condition guarantees that the rotation angle is less than 180°, corresponding to the shortest rotational path on SO(3). This constraint can be enforced during problem initialization by evaluating $\mathbf{q}_0^T \mathbf{q}_f$ and replacing $\mathbf{q}_f = -\mathbf{q}_f$ if the inner product is negative. This preprocessing step is computationally trivial yet essential for obtaining globally optimal time-minimal trajectories [10].

Alternatively, one can compute the relative rotation required with equation (2.51), where $\mathbf{q}_0^-$ is the quaternion conjugate of $\mathbf{q}_0$ and $\otimes$ denotes quaternion multiplication.

$$\mathbf{q}_{\text{rot}} = \mathbf{q}_f \otimes \mathbf{q}_0^- \tag{2.51}$$

This transformation approach exploits the group structure of unit quaternions. By formulating the problem in terms of the relative rotation $\mathbf{q}_{\mathrm{rot}}$, the optimization always starts from the identity quaternion $\mathbf{q}_I = [1\,0\,0\,0]^T$, which can simplify initialization and improve numerical conditioning. The trajectory is then mapped back to the physical frame via quaternion composition:

$$\mathbf{q}(t) = \mathbf{q}^*(t) \otimes \mathbf{q}_0 \tag{2.52}$$

where $\mathbf{q}^*(t)$ is the optimal quaternion trajectory in the optimization frame.

2.5.3 Optimal Control Characterization

The objective functional for time-optimal control minimizes the maneuver duration:

$$\mathcal{J} = t_f - t_0 \tag{2.53}$$

subject to dynamics constraint (2.46), control bounds (2.47), and boundary conditions (2.48)–(2.49).

The quaternion unit norm constraint requires special consideration within the optimization framework. Three primary approaches exist for enforcing $\|\mathbf{q}(t)\| = 1$: explicit normalization projects the quaternion onto the unit sphere after each integration step; penalty methods augment the objective function with terms penalizing norm deviations; direct enforcement incorporates $\|\mathbf{q}\| = 1$ as an algebraic constraint within the NLP formulation [10]. The choice of constraint handling method influences both computational efficiency and numerical conditioning of the optimization problem.

The problem constitutes a nonlinear, nonconvex, control-constrained optimal control problem. Nonlinearity arises from the gyroscopic coupling terms in Euler's equations and the bilinear structure of quaternion kinematics. Nonconvexity stems from the rotational manifold structure and the bang-bang control switching, leading to multiple local minima in the optimization landscape.

Application of PMP to the spacecraft attitude control problem yields the necessary conditions for time-optimal trajectories. The Hamiltonian for spacecraft attitude dynamics, following equation (2.21), takes the form:

$$H = 1 + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{q}, \boldsymbol{\omega}, \boldsymbol{\tau}) \tag{2.54}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^7$ represents the costate vector partitioned into components corresponding to attitude and angular velocity:

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_q \\ \boldsymbol{\lambda}_\omega \end{bmatrix} \tag{2.55}$$

Expanding the Hamiltonian using the spacecraft dynamics (2.46) yields:

$$H = 1 + \frac{1}{2}\boldsymbol{\lambda}_q^T \boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} + \boldsymbol{\lambda}_\omega^T \mathbf{J}^{-1}[\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}] \tag{2.56}$$

The costate differential equations follow from equation (2.17). For time-optimal problems with $L = 1$, the costate equations reduce to:

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda} \tag{2.57}$$

The Jacobian $\partial \mathbf{f}/\partial \mathbf{x}$ couples the kinematic and dynamic equations, requiring evaluation of derivatives of both quaternion kinematics and Euler's rotational equations with respect to attitude and angular velocity.

The switching function emerges from the linear appearance of control torques in the Hamiltonian. Extracting terms involving $\boldsymbol{\tau}$ from equation (2.56):

$$H = H_0(\mathbf{q}, \boldsymbol{\omega}, \boldsymbol{\lambda}_q, \boldsymbol{\lambda}_\omega) + \boldsymbol{\lambda}_\omega^T \mathbf{J}^{-1}\boldsymbol{\tau} \tag{2.58}$$

where H_0 encompasses all terms independent of control. Defining the switching function:

$$\mathbf{S}(t) = \mathbf{J}^{-1}\boldsymbol{\lambda}_\omega(t) \tag{2.59}$$

enables rewriting the Hamiltonian as:

$$H = H_0 + \mathbf{S}^T\boldsymbol{\tau} = H_0 + \sum_{i=1}^{3} S_i(t)\tau_i \tag{2.60}$$

The minimization condition (2.18) requires minimizing $\sum_{i=1}^{3} S_i(t)\tau_i$ subject to control bounds $|\tau_i| \leq \tau_{i,\max}$. Since the sum is linear in each control component, the minimum occurs at the boundary of the admissible set. For each axis independently:

$$\tau_i^*(t) = \begin{cases} -\tau_{i,\max} & \text{if } S_i(t) > 0 \\ +\tau_{i,\max} & \text{if } S_i(t) < 0 \\ \text{undefined} & \text{if } S_i(t) = 0 \end{cases}, \quad i = 1, 2, 3 \tag{2.61}$$

which can be written compactly as:

$$\tau_i^*(t) = -\tau_{i,\max} \cdot \operatorname{sgn}(S_i(t)), \quad i = 1, 2, 3 \tag{2.62}$$

This result confirms the bang-bang structure anticipated by PMP: each torque component saturates at its extreme value ($\pm\tau_{i,\max}$) with the sign determined by the corresponding switching function component.

The switching function admits physical interpretation. From equation (2.59), $\mathbf{S}(t) = \mathbf{J}^{-1}\boldsymbol{\lambda}_\omega(t)$ represents the inertia-weighted costate on angular velocity dynamics. The sign of each component $S_i(t)$ determines the optimal torque direction about the i-th principal axis: positive switching function values drive torque to its negative extreme, while negative values drive torque positive. Zero-crossings of the switching function, where $S_i(t)$ changes sign, determine switching instants at which the optimal control transitions between extreme values.

The switching function evolution is governed by the costate trajectory, which depends on the state trajectory through the costate equations (2.57). The state trajectory, in turn, depends on the applied control through the dynamics (2.46). This circular dependency, where costate affects switching function, switching function determines control, control drives state, and state influences costate, necessitates simultaneous solution of the forward state equations and backward costate equations. The resulting TPBVP couples conditions at initial time (state specified) and terminal time (costate determined by transversality conditions), presenting the fundamental computational challenge of indirect optimal control methods.

2.6 Computational Methods and Implementation

Practical implementation of direct transcription methods for spacecraft attitude control requires sophisticated computational infrastructure spanning numerical optimization algorithms, software frameworks, and initialization strategies. Many different solvers exist to solve OCP's, each adapted for specific transcription methods, problem structures and applications. This section examines the computational methods enabling efficient trajectory optimization. Section 2.6.1 analyzes gradient-based nonlinear programming solvers, distinguishing general-purpose algorithms from structure-exploiting approaches and convexification-based methods. Section 2.6.2 surveys software frameworks providing problem formulation interfaces and automatic differentiation. Section 2.6.3 examines metaheuristic algorithms for initialization and global search.

2.6.1 Nonlinear Programming Solvers

Direct transcription converts continuous-time optimal control problems into finite-dimensional NLPs characterized by large-scale sparse structure. For multiple shooting with N shooting intervals and n states, the NLP comprises $N(n+m)$ decision variables, where m denotes control dimension. The constraint Jacobian exhibits block-sparse structure: dynamics constraints couple only adjacent intervals, creating block-tridiagonal patterns in the Karush-Kuhn-Tucker (KKT) system [27]. Direct collocation generates denser constraint structures due to polynomial approximations coupling multiple collocation points within intervals [15]. Exploitation of this temporal structure differentiates general-purpose sparse NLP solvers from specialized algorithms designed for optimal control. The present section examines computational methods for trajectory optimization, distinguishing general-purpose algorithms from structure-exploiting approaches.

General-Purpose NLP Solvers

General-purpose solvers treat trajectory optimization as standard sparse NLP without exploiting temporal structure. Two algorithmic paradigms dominate: interior-point methods and sequential quadratic programming (SQP). These methods apply to arbitrary transcription schemes but achieve computational complexity scaling as $O((Nn)^3)$ due to full KKT matrix factorization [28].

- **Interior Point Methods:** IPOPT (Interior Point OPTimizer) implements a primal-dual interior-point (IP) method algorithm with filter line search [22]. The filter mechanism maintains non-dominated pairs of constraint violation and objective function values, accepting iterates that improve either metric. This approach demonstrates robust convergence across diverse problem formulations and has been extensively validated in aerospace trajectory optimization [15]. However, IPOPT factors the full KKT system as a general sparse matrix without recognizing block structure arising from temporal dynamics. For large-horizon problems, factorization becomes computationally expensive, limiting applicability to offline optimization and algorithm development.
- **Sequential Quadratic Programming:** SNOPT (Sparse Nonlinear OPTimizer) employs SQP with quasi-Newton Hessian approximation updates [23]. Each iteration solves a quadratic program (QP) approximating the Lagrangian, using gradient information to update the Hessian approximation. The quasi-Newton approach eliminates exact second-derivative computation, reducing cost when automatic differentiation is unavailable. SNOPT maintains superlinear convergence, though slower than Newton methods with exact Hessians. The solver suits preliminary design phases where derivative computation cost exceeds the cost of additional iterations [23].

Structure-Exploiting Solvers

Structure-exploiting solvers leverage block-sparse temporal structure inherent to optimal control problems, achieving computational complexity $O(Nn^3)$ linear in horizon length [27, 29]. These methods prove particularly effective for multiple shooting formulations, where dynamics constraints couple only adjacent shooting intervals, creating block-tridiagonal KKT systems. The computational advantage increases with horizon length: for $N = 50$ stages with $n = 7$ states (quaternion attitude representation), structure exploitation reduces complexity from $O((350)^3) \approx 43$ million to $O(50 \cdot 343) \approx 17$ thousand operations.

- **Riccati-Based IP:** FATROP (Fast Trajectory Optimization) implements structure-exploiting interior-point optimization using generalized Riccati recursion [27]. The method recognizes block-tridiagonal KKT structure and employs forward-backward Riccati sweeps rather than general sparse factorization. The forward pass computes Riccati matrices encoding primal-dual relationships between adjacent stages; the backward pass recursively solves for optimal state and control trajectories. FATROP handles inequality constraints through an active-set interior-point approach without introducing slack variables, maintaining problem compactness. The solver incorporates exact Hessian information via automatic differentiation, enabling quadratic convergence. Benchmarks demonstrate approximately $10\times$ speedup versus IPOPT for trajectory optimization problems, suitable for real-time applications [27].

- **Riccati-Based SQP:** The ACADOS framework paired with the HPIPM (High-Performance Interior Point Method) QP solver exploits block-tridiagonal structure via Riccati recursion [30]. The ACADOS-HPIPM combination achieves computational performance comparable to FATROP for multiple shooting problems while providing greater flexibility in transcription method selection [29]. The framework supports automatic differentiation via CasADi [31], enabling efficient derivative generation for arbitrary dynamics formulations.

Sequential Convex Programming

Sequential convex programming (SCP) provides an alternative approach to nonconvex trajectory optimization through iterative solution of convex approximations [32]. Each SCP iteration linearizes nonlinear dynamics, imposes trust region constraints to maintain approximation validity, solves the resulting convex program via second-order cone programming or QP methods, and updates the reference trajectory. SCP has seen limited application to attitude control due to quaternion constraint complexities [32].

Solver Selection Criteria

Solver selection depends on computational requirements, problem characteristics, and available infrastructure. Table 2.3 summarizes key characteristics. IPOPT provides maximum reliability for offline trajectory design and algorithm prototyping, with robust convergence across formulations but limited real-time applicability. SNOPT suits scenarios where exact Hessian computation is expensive or during preliminary design phases. Structure-exploiting solvers enable real-time implementation, but require specific sparse OCP formulations, with FATROP offering simpler integration and ACADOS providing greater algorithmic flexibility.

Table 2.3 Comparison of NLP solvers for trajectory optimization.

Solver	**Method**	**Complexity**	**Convergence**
IPOPT	IP	$O((Nn)^3)$	Quadratic
SNOPT	SQP	$O((Nn)^3)$	Superlinear
FATROP	Riccati IP	$O(Nn^3)$	Quadratic
ACADOS-HPIPM	Riccati SQP	$O(Nn^3)$	Superlinear

2.6.2 Optimal Control Software Frameworks

Software frameworks bridge domain-specific trajectory optimization concepts and low-level nonlinear programming abstractions. Key capabilities include intuitive problem specification interfaces, automatic differentiation for gradient computation, integration with multiple solvers, sparse structure recognition, and efficient function evaluation. Framework selection involves trade-offs between generality, computational efficiency, and implementation complexity.

CasADi: Symbolic Framework

CasADi (Computer Algebra System with Automatic Differentiation) provides foundational infrastructure for modern optimal control [31]. The symbolic framework constructs computational graphs representing problem functions, applies algebraic simplifications, determines sparsity patterns, and generates efficient standalone C code. CasADi supports forward and reverse mode automatic differentiation, producing exact derivatives without finite-difference errors. The framework interfaces with IPOPT, SNOPT, FATROP, and other commercial solvers through unified application programming interfaces (APIs). The Opti stack offers high-level syntax for OCP formulation [33]. CasADi's open-source availability, extensive documentation, and aerospace community adoption establish it as de facto standard infrastructure.

ACADOS: Real-Time Framework

ACADOS (Automatic Control and Dynamic Optimization Suite) targets fast embedded optimal control with emphasis on Model Predictive Control (MPC) [29]. The framework generates templated C code for embedded targets with fixed memory footprint and minimal runtime overhead. ACADOS implements real-time iteration schemes, structure-exploiting QP solvers (HPIPM, qpOASES), and custom linear algebra kernels optimized for cache performance and vectorization. Performance targets include millisecond-range computation suitable for high-rate control loops exceeding 100 Hz. Integration with CasADi enables symbolic differentiation while specialized optimizations address real-time execution requirements. Open-source availability facilitates research and commercial deployment.

GPOPS: Pseudospectral Methods

GPOPS-II (General Purpose OPtimal Control Software) implements hp-adaptive Gaussian quadrature collocation in MATLAB [21]. The framework employs LGR collocation with automatic mesh refinement adapting interval sizes (h-method) and polynomial orders (p-method) to achieve specified accuracy tolerances. Adaptive capabilities prove valuable for bang-bang problems where mesh concentration near switching times improves efficiency. GPOPS-II interfaces with IPOPT and SNOPT, supporting sparse finite differences, automatic differentiation, or user-supplied derivatives. Wide aerospace adoption has established benchmark solutions for diverse optimal control problems.

CGPOPS provides C++ implementation of similar algorithms, achieving 2-6$\times$ speedup over MATLAB through compiled language efficiency [7]. Multiple derivative methods include central finite differences, bicomplex-step differentiation eliminating subtraction error, hyper-dual differentiation for exact second derivatives, and automatic differentiation integration. Specialized mesh refinement strategies for bang-bang control include switching detection algorithms and grid concentration near discontinuities. Exact sparsity pattern identification via hyper-dual methods improves computational efficiency. Cross-platform portability and compiled performance make CGPOPS suitable for production deployment.

2.6.3 Derivative-Free Optimization

Derivative-free methods address scenarios where gradients are unavailable, unreliable, or expensive to compute. Metaheuristic algorithms employ stochastic population-based search, while direct search methods use deterministic pattern exploration. Computational cost typically exceeds gradient-based approaches by orders of magnitude, offset by parallelization potential and robustness to local minima. One of the most widely used methods is the Particle Swarm Optimization (PSO), which has gained popularity due to its simple yet effective implementation.

Particle Swarm Optimization

PSO models collective intelligence through social learning. A population of $\mathcal{N}$ particles maintains positions $\mathbf{z}_j$ and velocities $\mathbf{v}_j$ for every particle j, evolving via:

$$\begin{aligned}\mathbf{v}_j^{k+1} &= w_0\mathbf{v}_j^k + w_1 r_1(\boldsymbol{\rho}_j - \mathbf{z}_j^k) + w_2 r_2(\mathbf{g} - \mathbf{z}_j^k)\\ \mathbf{z}_j^{k+1} &= \mathbf{z}_j^k + \mathbf{v}_j^{k+1}\end{aligned} \tag{2.63}$$

where $\boldsymbol{\rho}_j$ is personal best, $\mathbf{g}$ is global best, w_0 is inertia weight, w_1, $w_2 \geq 0$ are cognitive and social parameters, and r_1, $r_2 \in [0,\, 1]$ are random numbers. Parameter tuning balances exploration and exploitation: increasing inertia increases solution space exploration, while reducing it allows for local optimum refinement. Cognitive and social parameters weight individual versus collective learning. Some variations of the traditional PSO use dynamics weights, allowing for better solution space exploration initially, but transitioning to a more precise search for the last iterations to refine the global best solution.

Applications in optimal control mostly include costate initialization for indirect methods [34] and warm-start generation for gradient-based refinement [35].

For discrete optimization problems, binary PSO extends the continuous framework to binary decision variables [36]. The velocity update in equation (2.63) remains unchanged, but positions are mapped to binary values through a sigmoid transfer function:

$$\begin{aligned}\mathcal{S}(v_{j,d}^{k+1}) &= \frac{1}{1 + e^{-\alpha v_{j,d}^{k+1}}}\\ z_{j,d}^{k+1} &= \begin{cases}1 & \text{if } r_d < \mathcal{S}(v_{j,d}^{k+1})\\ 0 & \text{otherwise}\end{cases}\end{aligned} \tag{2.64}$$

where $\alpha > 0$ is a scaling parameter controlling the sigmoid slope and $r_d \in [0,\, 1]$ is a random number for each dimension d. The parameter α determines the degree of stochasticity in position updates: larger values produce steeper sigmoid transitions, leading to more deterministic decisions, while smaller values maintain gradual transitions that preserve stochastic behavior [36].

To prevent complete loss of randomness when velocity values are big, the velocity is clamped $|v_{j,d}| \leq V_{\max}$, where $V_{\max}$ is computed to maintain the sigmoid output within bounds $[1-\epsilon,\, \epsilon]$ for a saturation threshold $\epsilon \in [0.5,\, 1.0]$:

$$V_{\max} = -\frac{1}{\alpha}\ln\left(\frac{1}{\epsilon} - 1\right) \tag{2.65}$$

Without velocity saturation, unbounded velocities can drive $\mathcal{S}(v_{j,d}) \to 0$ or $\mathcal{S}(v_{j,d}) \to 1$, eliminating the probabilistic position update and degrading search performance. For $\epsilon = 1$ the effect of bounding the velocity is nullified and the decisions are allowed to become deterministic for big velocity values. When $\epsilon = 0.5$ the velocity term loses its influence on the decision for the next position and the process becomes completely random. Binary PSO finds application in bang-bang control sign selection [24], feature selection, and scheduling problems.

Application Strategy

Although derivative-free methods comprise thousands of function evaluations, suffer from complex parameter tuning, and offer no convergence guarantees, they have found their utility in many different applications due to their global search capability and simple implementation. Additionally GPU acceleration has enabled order-of-magnitude speedups for parallel function evaluation [37, 38], making them attractive for real-time applications.

Their use in optimal control focuses primarily on initialization generation [34, 35]. A two-stage approach combines metaheuristic global search to identify promising regions with gradient-based refinement to achieve high accuracy. This hybrid strategy leverages complementary strengths: derivative-free robustness for initialization and gradient-based efficiency for convergence.

2.7 Efficiency and Reliability

The practical deployment of trajectory optimization algorithms for spacecraft attitude control requires consideration of both computational efficiency and solution reliability. Reliability, defined as the probability of convergence to a feasible and optimal solution from arbitrary initial conditions, critically depends on initialization quality for nonconvex problems. Computational efficiency determines whether algorithms can execute within real-time constraints imposed by onboard processors and operational timelines. The definition of "real-time" varies across spacecraft applications: agile imaging missions may demand sub-second computation, while station-keeping operations permit longer solution horizons. Statistical validation through MC analysis provides essential characterization of algorithm performance across the operational envelope. This section examines strategies for improving both reliability and efficiency. Section 2.7.1 analyzes initialization strategies ranging from deterministic heuristics to stochastic global search methods. Section 2.7.2 examines GPU acceleration techniques for parallel function evaluation. Section 2.7.3 presents statistical methods for quantifying algorithm reliability across problem parameter spaces.

2.7.1 Initialization Strategies

Gradient-based NLP solvers converge to local optima from initial guesses within their basins of attraction. Nonconvex trajectory optimization problems exhibit multiple local minima, with basin sizes and locations depending on problem parameters. Poor initialization leads to different failure modes such as: convergence failure when iteration limits are exceeded, generation of infeasible solutions violating dynamics or boundary conditions, and entrapment in suboptimal local minima distant from the global optimum. Initialization strategies can be divided into two categories: deterministic approaches using problem-specific knowledge, and stochastic methods employing global search algorithms.

Deterministic Initialization

Deterministic strategies construct initial guesses using available information without stochastic sampling. Some examples consist of:

- **Warm-starting from previous solutions** — exploits similarity between sequential optimization problems by initializing from previously converged trajectories. For parametric studies varying problem properties incrementally, the optimal solution changes continuously with parameters. Warm-starting from the solution at nearby parameter values typically ensures convergence with minimal iterations. The approach requires maintaining a solution library or database indexed by problem parameters. Applicability is limited to problems with similar structure and boundary conditions as novel maneuvers or significant parameter changes render previous solutions uninformative.

- **Linear interpolation between boundary conditions** — constructs state trajectories by linearly interpolating between initial and terminal states. This initialization satisfies boundary conditions by construction but ignores system dynamics, typically producing infeasible trajectories requiring significant correction during optimization. Linear interpolation may suffice for nearly linear systems or small maneuvers where dynamics approximate straight-line motion in state space, but proves inadequate for highly nonlinear trajectories.

- **Zero control initialization** — sets all control values to zero for all discretization points. This simple approach proves inadequate for control-driven systems where zero control produces trivial or infeasible trajectories. Zero control initialization may fail entirely for unstable dynamics requiring active control to maintain bounded trajectories.

- **Heuristic trajectory generation** — employs physics-based approximations tailored to problem structure. For spacecraft attitude control, eigenaxis rotation provides a physically motivated initialization. This strategy is famous for trapping solvers in non-optimal solution basins as these trajectories comply with both system dynamics and constraints and, therefore, often lay in a local minimum.

Deterministic initialization offers computational economy, deterministic reproducibility, and simple implementation. However, reliability degrades for problems with significant nonlinearity, large maneuver magnitudes, or complex constraint activity [35]. Deterministic methods prove most appropriate for well-behaved problems and small perturbations from known solutions.

Stochastic Initialization

Stochastic strategies combine derivative-free global search with gradient-based refinement, exploring multiple candidate solutions to identify promising initialization points. Metaheuristic algorithms such as PSO have demonstrated effectiveness for trajectory optimization initialization [35, 34, 24].

The two-stage approach performs global search using metaheuristics (see section 2.6.3) to generate candidate solutions, evaluates a population of trajectories across the decision variable space, selects the best candidate based on objective value and constraint satisfaction, and refines the selected solution using gradient-based NLP optimization. Stochastic initialization provides several advantages over deterministic approaches. Exploration of multiple solution candidates reduces sensitivity to single initial guess quality. Global search capability enables escape from poor local minima that trap deterministic methods. Discovery of solution structure, such as switching patterns in bang-bang control, provides physically meaningful initialization for refinement. The parallel nature of population-based algorithms enables efficient implementation on multi-core processors or GPUs [37, 38]. However, stochastic methods impose computational expense, typically requiring thousands of function evaluations before gradient-based refinement. The stochastic nature introduces solution variability between runs with different random seeds. Metaheuristic performance depends on parameter tuning, requiring problem-specific calibration. Unlike gradient-based methods, metaheuristics provide no theoretical convergence guarantees or optimality conditions.

Selection Criteria

Initialization strategy selection depends on problem characteristics and computational constraints. Problem complexity assessment guides the choice: well-behaved dynamics with smooth objective functions permit deterministic methods, while multimodal landscapes with multiple local minima require stochastic global search. Unknown solution structure, particularly switching patterns in bang-bang control, favors hybrid approaches combining stochastic structure discovery with gradient-based refinement.

Computational budget availability influences strategy selection. Tight time constraints in online optimization necessitate simple deterministic initialization, accepting reduced reliability for computational speed. Offline trajectory generation permits sophisticated stochastic methods, investing computation to achieve high reliability. The availability of prior solutions enables warm-starting when problem parameters vary continuously, eliminating the need for expensive global search.

Required reliability level determines method sophistication. Applications demanding high success rates justify stochastic or hybrid methods despite computational expense. Systems tolerating occasional failures may employ simpler deterministic methods and try alternative initializations upon convergence failure.

2.7.2 GPU Acceleration Techniques

GPUs have evolved from specialized graphics hardware to general-purpose parallel computing platforms. The CUDA (Compute Unified Device Architecture) programming model developed by

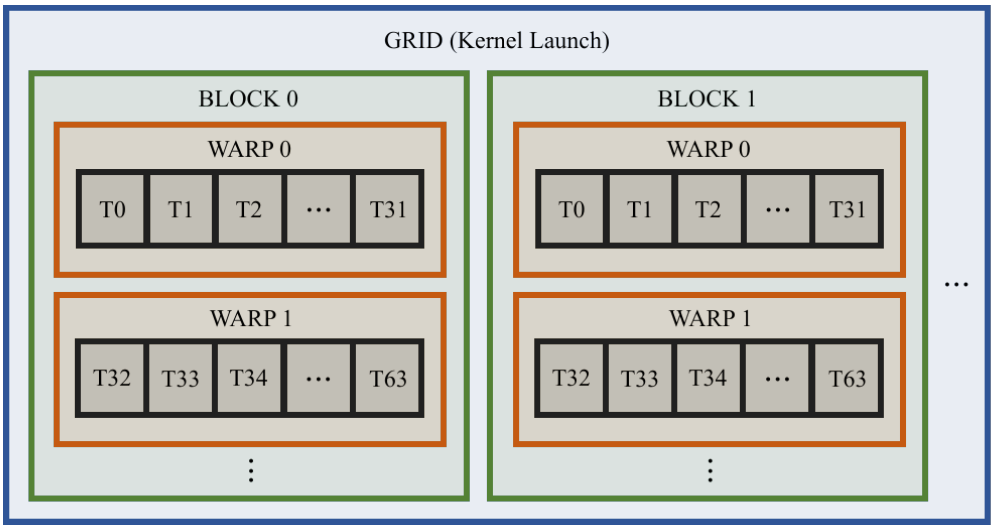


Figure 2.1 CUDA hierarchical thread organization.

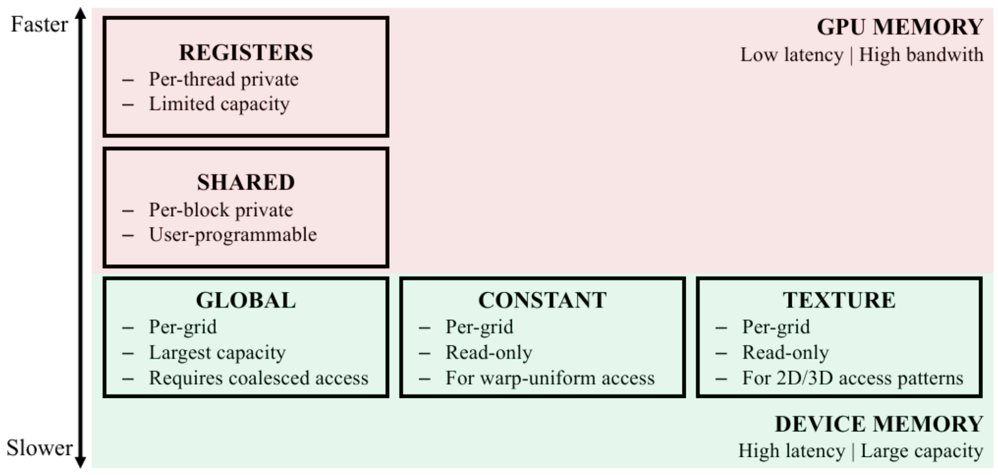


Figure 2.2 CUDA memory hierarchy from fastest to slowest.

NVIDIA provides C/C++ language extensions enabling exploitation of GPU parallelism for scientific computing [39]. Trajectory optimization exhibits parallelizable structure amenable to GPU acceleration, though sequential components within NLP solvers limit overall speedup.

CUDA Architecture and Programming Model

CUDA exposes a hierarchical thread organization comprising: threads (finest granularity execution units), blocks (thread groups executing on streaming multiprocessors (SM)), and grids (block collections comprising a kernel launch). Figure 2.1 illustrates this hierarchy, where each block contains multiple groups of 32 threads that execute instructions in lockstep (single instruction, multiple thread (SIMT) execution model) known as warps. This organizational structure enables massive parallelism while maintaining programming model simplicity.

The memory hierarchy, depicted in figure 2.2, comprises multiple levels with distinct latency and capacity characteristics. Registers provide fastest per-thread private storage with limited capacity. Shared memory offers fast per-block storage under explicit programmer control, functioning as a software-managed cache. Global memory, residing in device dynamic random access memory (DRAM), provides largest capacity with highest latency and requires coalesced access patterns for efficient bandwidth utilization. Constant and texture memory represent specialized cached device memory, optimized for warp-uniform reads and spatial locality respectively [40].

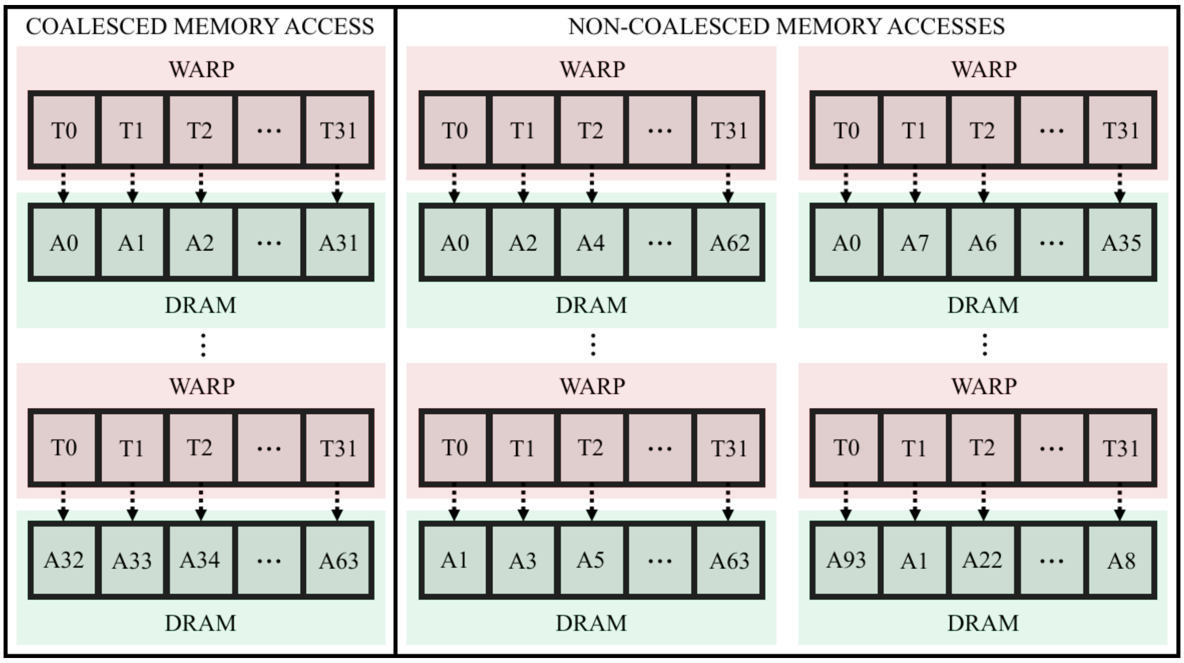


Figure 2.3 Coalesced memory access vs non-coalesced memory accesses.

Performance optimization requires exploitation of architectural characteristics. Memory coalescing through aligned, contiguous access patterns maximizes global memory bandwidth utilization. Figure 2.3 shows what this looks like. Shared memory exploitation reduces device memory traffic by enabling data reuse within thread blocks. Occupancy maximization hides memory latency through computation-memory access overlap. Thread divergence minimization maintains uniform control flow within warps, preventing serialization of divergent execution paths [40].

Well-designed computational kernels demonstrate 10–100× speedup versus sequential CPU execution [39]. However, achievable performance gains depend critically on algorithmic parallelizability, memory access patterns, and computational intensity characteristics of the specific problem domain.

GPU Acceleration in Trajectory Optimization

Direct transcription generatesa parallel structure suitable for GPU implementation. Multiple shooting with N segments enables concurrent dynamics propagation: each segment integration proceeds independently, requiring only segment-local data. Collocation methods parallelize constraint evaluation at hundreds of discretization points. Sensitivity computation for gradients and Hessians via automatic differentiation benefits from parallel evaluation of partial derivatives.

Implementation examples demonstrate the potential and limitations of GPU acceleration. CusADi provides CUDA-based symbolic automatic differentiation, generating GPU kernels for function and gradient evaluation [41]. CVODES-CUDA implements GPU-accelerated ODE integration with sensitivity analysis [20]. GPU-based interior-point methods for linear problems accelerate function evaluation but face challenges in sparse linear algebra [42].

Although function evaluation speedup has been demonstrated for dynamics propagation and constraint evaluation, overall optimization speedup remains limited by sequential components.

Limitations and Challenges

NLP solvers contain inherently sequential components resistant to parallelization. Line search algorithms backtrack sequentially, evaluating merit functions until sufficient decrease conditions are satisfied. Convergence checks require global synchronization to assess optimality conditions across all variables. KKT matrix factorization for computing Newton steps exhibits limited parallelism:

sparse direct solvers perform elimination with irregular dependency patterns poorly suited to GPU architectures.

Linear system solution presents particular challenges on GPUs. Sparse factorization involves irregular memory access patterns (indirect addressing through pointer-based data structures) and low arithmetic intensity (computation per memory access), characteristics mismatched to GPU memory hierarchy optimized for regular, compute-intensive operations. Memory bandwidth requirements can exceed available peripheral component interconnect express (PCIe) bandwidth for high-speed data transfers between CPU and GPU, with transfer latency negating computational gains for small to moderate problem sizes.

Programming complexity increases substantially for GPU implementation. Code must explicitly manage memory allocation, data transfer, kernel launch parameters, and synchronization. Debugging parallel code proves significantly more challenging than sequential programs, as race conditions and memory access violations manifest non-deterministically.

Hybrid CPU-GPU Architectures

Viable solutions employ hybrid architectures exploiting complementary strengths of CPU and GPU hardware. The CPU manages the optimization loop, performs sequential operations (line search, convergence checks), handles control-flow-intensive tasks, and executes sparse linear algebra when appropriate. The GPU performs expensive parallel function evaluations, evaluates constraints across the discretization grid, computes gradients via parallel automatic differentiation, and performs batched dynamics integration.

This division of labor can achieve superior single-thread performance and control flow handling on the CPU, while leveraging massive GPU throughput for arithmetic operations. Asynchronous computation enables overlap: while the GPU evaluates functions, the CPU can prepare the next iteration or process results from previous evaluations.

Well-designed hybrid systems achieve 2-10$\times$ speedup over CPU-only implementations [42]. The modest speedup relative to theoretical GPU peak performance reflects Amdahl's law: sequential components limit overall acceleration regardless of parallel section speedup.

2.7.3 Reliability Assessment

Algorithm reliability is the probability of successful convergence to feasible, globally optimal solutions, and requires statistical characterization across problem parameter spaces. Nonconvex optimization landscapes with multiple local minima exhibit initialization-dependent convergence behavior that defeats deterministic analysis. Gradient-based solvers demonstrate varying success rates depending on problem formulation, maneuver type, and initialization strategy [35]. Statistical validation through repeated trials with parameter variation quantifies operational envelope boundaries and identifies problem regions requiring enhanced initialization or alternative solution methods.

Practical sample size selection balances theoretical requirements with computational constraints. For a D-dimensional parameter space with l samples, the informal rule $l \approx 10D$ has been empirically validated and theoretically justified by Loeppky et al. [43] as a good starting point, though convergence studies remain advisable to verify sufficiency for specific applications.

Monte Carlo Simulation

MC simulation employs stochastic sampling to evaluate algorithm performance across parameter distributions. The methodology defines problem parameter space, specifies probability distributions for each parameter (uniform, normal, or problem-specific distributions), and performs independent random draws for each trial. Each trial solves the trajectory optimization problem from specified initialization, recording convergence success, final objective value, and computation time. Aggregating results across l trials estimates convergence probability, characterizes solution quality distribution, and identifies parameter regions with poor reliability.

MC simulation provides conceptual simplicity, general applicability across diverse problems, and well-established statistical theory for confidence interval estimation. However, convergence rate scales as $O(l^{-1/2})$: halving confidence interval width requires quadrupling sample size. Achieving tight confidence bounds on rare events (failure probabilities below 1%) necessitates tens of thousands of trials.

Latin Hypercube Sampling

Latin Hypercube Sampling (LHS) provides stratified sampling with improved space-filling properties compared to pure random sampling [44]. The method offers better parameter space coverage with fewer samples, reducing computational burden for expensive trajectory optimization evaluations.

For D-dimensional parameter space with l samples, LHS proceeds as follows. First, divide each parameter range into l equal-probability intervals. Second, sample once from each interval for every parameter dimension independently. Third, randomly permute samples across dimensions to create D-dimensional combinations. This ensures each interval is represented exactly once per dimension, avoiding clustering that afflicts pure random sampling.

Stein [45] established the theoretical foundation for LHS variance reduction, proving that the estimator variance converges as $O(l^{-1})$ when $l \gg D$, compared to $O(l^{-1/2})$ for MC simulations. The degree of variance reduction depends fundamentally on the additivity of the objective function: LHS effectively filters out additive components, with remaining variance proportional to the non-additive residual. This theoretical result demonstrates that LHS achieves asymptotically lower variance than simple random sampling for any square-integrable function, with maximum benefit for nearly-additive response surfaces.

LHS achieves reduced clustering and more uniform exploration of parameter extremes compared to traditional MC simulations, with reduced sample count for equivalent statistical confidence. Implementation complexity exceeds simple random sampling, and variance reduction effectiveness depends on response surface smoothness and additivity—assumptions potentially violated by discontinuous failure boundaries in some optimization problems.

Reliability Metrics and Analysis

Reliability assessment quantifies algorithm performance through multiple metrics beyond binary success/failure classification. Convergence rate measures the fraction of trials achieving feasible solutions satisfying optimality tolerances. Solution quality distribution characterizes objective value variation across successful trials, identifying cases where poor initialization yields suboptimal local minima. Computation time distributions inform real-time feasibility assessment. Parameter sensitivity analysis identifies configurations exhibiting degraded reliability, guiding development of specialized initialization strategies for challenging regions.

Statistical analysis of reliability data enables operational envelope characterization. Confidence intervals on convergence probability provide probabilistic guarantees for deployment. Identification of parameter boundaries where reliability degrades rapidly guides operational constraints: restricting maneuver angles, limiting initial angular velocities, or constraining spacecraft asymmetry ratios to regions with high success rates. Reliability assessment validates algorithm improvements: comparing convergence rates before and after initialization enhancement quantifies benefit with statistical rigor.

3 Methodology

This chapter establishes the methodology for developing and verifying real-time spacecraft attitude control algorithms achieving computation times below 200 milliseconds. Building upon the theoretical foundations of chapter 2, the approach follows a hierarchical framework: foundational design choices establish the problem structure; sequential algorithmic optimization identifies optimal configurations through comparative evaluation; and MC simulation quantifies reliability across the operational envelope.

Section 3.1 documents the selection of quaternion parameterization, direct dynamics formulation, and the CasADi-based computational framework. Section 3.2 compares transcription methods, integration schemes, and NLP solvers through benchmark problems. Section 3.3 defines test problem suites, performance metrics, and statistical verification protocols. Section 3.4 develops PSO-based initialization strategies to enhance convergence reliability. Section 3.5 specifies computational platform characteristics and numerical parameters enabling reproducibility. Each decision is justified through quantitative performance evaluation, ensuring the final algorithm balances computational efficiency, solution quality, and convergence reliability.

3.1 Foundational Design Choices

This section establishes fundamental design decisions that define the problem structure prior to algorithmic optimization. These choices constrain subsequent algorithmic decisions and directly influence solution feasibility, computational efficiency, and implementation complexity.

1. **Attitude Parameterization** – Quaternion parameterization is selected based on three advantages established in section 2.1.2 and table 2.1: global validity without singularities, minimal redundancy (single constraint versus six for SO(3)), and linear kinematic equations (2.8) which facilitate efficient gradient computation.

2. **Dynamics Formulation** – Direct dynamics formulation is selected because it provides natural handling of control constraints (2.47) as box constraints in the NLP, and simplified boundary condition enforcement. The inverse dynamics alternative offers automatic boundary condition satisfaction but introduces control feasibility challenges for time-optimal control where the bang-bang structure (2.62) drives controls to saturation, and additional complexity in enforcing the unit quaternion norm constraint and finding adequate basis functions.

3. **Control Parameterization** – Full trajectory optimization is selected based on the following criteria: switching structures for track-to-track maneuvers and asymmetric spacecraft remain problem-dependent despite analytical results for symmetric rest-to-rest cases; and the formulation accommodates arbitrary configurations within a unified framework. Although the decision variable count exceeds the one for bang-bang parameterization, the enhanced convergence robustness due to the the absence of discontinuities normally offsets this cost.

4. **Optimization Methodology** – Direct transcription is selected over indirect methods (section 2.3.1) due to it offering larger convergence basins enabling cruder initial guesses, intuitive initialization using physical variables rather than abstract costates, natural constraint handling, robustness to parameter variations, mature solver infrastructure, and optimal computational efficiency.

5. **Computational Framework** – CasADi provides exact derivatives via automatic differentiation eliminating finite-difference errors, automatic sparsity exploitation for block-tridiagonal structures (2.28), unified APIs for multiple NLP solvers enabling their comparison, high-level Opti syntax, and extensive aerospace validation.
6. **NLP Solver** – IPOPT is selected for initial algorithm development due to its general-purpose applicability across all transcription methods, enabling their comparison and robust filter line search convergence across parameter space. However, section 3.2.3 evaluates structure-exploiting alternatives.
7. **Verification and Benchmarking** – CGPOPS provides reference solutions via hp-adaptive LGR collocation achieving spectral convergence. Established aerospace benchmarking enables verification through literature comparison. Computation times (seconds to minutes) preclude real-time use, therefore, CGPOPS serves exclusively for verification. Results in chapter 4 are verified against CGPOPS to ensure speed optimizations preserve solution quality.

3.2 Algorithmic Optimization

This section documents the systematic selection of algorithmic components through comparative evaluation on benchmark problems. The methodology employs sequential decision-making to identify the configuration achieving optimal balance between computational efficiency and solution quality. Benchmark scenarios consist of symmetric spacecraft executing rest-to-rest maneuvers through 45°, 90° and 180° rotations, for which analytical switching structures and optimal maneuver durations have been established [3, 4]. Table 3.2 presents the dimensionless optimal durations for these canonical cases, which serve as reference solutions for verifying implementation correctness and measuring solution quality through optimality gap computation. The bounds are chosen large enough to not interfere with the optimal solution but as tight as possible to restrict solution search space and accelerate convergence.

Table 3.1 Dimensionless parameters for spacecraft attitude control optimization.

Category	Parameter	Symbol	Value
Maneuver Duration	Minimum bound	$T_{\min}$	0.0
	Maximum bound	$T_{\max}$	6.0
Symmetric Spacecraft Principal Moments of Inertia	i-axis component	$\bar{J}_i$	1.0
Angular Velocity Bounds	Maximum magnitude	$\omega_{\max}$	5.0
	Minimum magnitude	$\omega_{\min}$	-5.0
Control Torque Bounds	Maximum magnitude	$\bar{\tau}_{i,\max}$	1.0
	Minimum magnitude	$\bar{\tau}_{i,\min}$	-1.0

Note: All parameters are dimensionless. The overbar notation denotes normalization by the characteristic scales J_0 (reference inertia) and $\tau_{\max}$ (maximum control authority). Quaternion components are inherently bounded by ± 1 due to the unit norm constraint. Dimensionless time may be converted to dimensional time in seconds by multiplying by the scaling factor $\sqrt{J_0/\tau_{\max}}$. Dimensionless angular velocity must be divided by the same scaling factor to convert it into its dimensional counterpart in rad/s.

The selection process follows an iterative refinement strategy wherein earlier decisions constrain and inform subsequent choices. Transcription method selection (section 3.2.1) establishes the fundamental discretization framework, determining whether dynamics are enforced through shooting integration or collocation constraints. Integration scheme evaluation (section 3.2.2) then optimizes the dynamics propagation method within the selected transcription framework, comparing fixed-step explicit methods against adaptive implicit schemes and evaluating GPU acceleration potential.

Finally, NLP solver optimization (section 3.2.3) identifies the most efficient solver for the established problem structure, comparing general-purpose interior-point methods against structure-exploiting Riccati-based algorithms.

Each comparison employs consistent evaluation metrics: total computation time and solution quality quantified by the optimality gap relative to the analytical solutions in table 3.2.

Table 3.2 Analytical solutions for minimum-time rest-to-rest maneuvers of symmetric spacecraft.

Rotation Angle	Dimensionless Duration	Number of Switches
45°	1.7471 (1.7499)	6 (7)
90°	2.4211	5
180°	3.2431	5

3.2.1 Transcription Method Selection

The first step is comparing direct multiple shooting (section 2.3.2) with Hermite-Simpson collocation (section 2.3.2). They represent alternative frameworks for enforcing dynamics constraints in direct transcription. Multiple shooting generates block-tridiagonal constraint structures exploitable by specialized solvers and enables parallel integration across shooting intervals. Hermite-Simpson collocation requires fewer function evaluations through implicit midpoint formulation. Both achieve fourth-order accuracy $O(\Delta t^4)$: Hermite-Simpson through cubic polynomial approximation, multiple shooting when paired with RK4 integration (section 2.3.2). While Hermite-Simpson's implicit properties benefit stiff systems, spacecraft attitude dynamics with moderate inertia asymmetry exhibit negligible stiffness. The minimum amount of discretization intervals required for reliable convergence for both methods is $N = 50$, and this value is kept constant throughout the rest of the optimization.

The comparison determines that multiple shooting gives better results, leading to its adoption for all future development. These results are analysed in section 4.1.1.

3.2.2 GPU-Accelerated Dynamics Propagation

Within the multiple shooting framework, the choice of integration scheme for propagating dynamics across shooting intervals significantly is analyzed. This evaluation compares explicit fixed-step integration using classical RK4 against adaptive implicit methods implemented in CVODES, the variable-step ordinary differential equation solver from the SUNDIALS library. GPU acceleration is used to exploit the parallel structure of multiple shooting.

- **RK4** provides deterministic computational cost with four function evaluations per integration step (2.30), enabling straightforward performance prediction and memory allocation. The explicit nature requires no iterative solution of implicit equations, simplifying implementation and facilitating GPU parallelization. However, fixed step sizes may prove inefficient for trajectories with varying solution smoothness, potentially requiring very small steps across the entire interval to maintain accuracy near switching discontinuities.

- **CVODES** employs variable-order, variable-step Adams-Moulton and Backward Differentiation Formula methods with automatic stiffness detection. The adaptive step size feature provides computational efficiency for smooth trajectory segments while concentrating steps near rapid state variations. CVODES includes built-in GPU-enabled integration and sensitivity analysis capabilities that accelerate computation, potentially offsetting the overhead of adaptive step size and implicit integration.

The evaluation compares computational platforms and integration methods simultaneously. It examines GPU acceleration potential for explicit RK4 integration and a GPU CVODES implementation by comparing their performance against a CPU-run explicit RK4 integration scheme. The analysis considers total computation time including CPU-to-GPU memory transfers and kernel launch latency, as these overheads can dominate for small problems where integration time itself is negligible.

To enable initial GPU kernel implementation without requiring manual construction of exact analytical derivatives, the NLP solver employs finite-difference approximations for Jacobian evaluation and limited-memory approximations for Hessian construction. These approximations, specified through solver options, reduce implementation complexity at the expense of computational efficiency and solution accuracy when compared to exact derivative formulations. These settings are also used in the CPU implementation for fair comparison.

The analysis of the comparison results revealed that propagating the dynamics on the CPU using explicit RK4 entails the best performance/complexity ratio and is therefore selected as the numerical integration method for the final algorithm configuration. Section 4.1.2 contains the details of this analysis.

3.2.3 NLP Solver Optimization

The NLP solver selection fundamentally determines whether real-time computation times can be achieved. Here, IPOPT is compared to FATROP, which directly exploits the block-tridiagonal structure of OCPs transcribed using multiple shooting, reducing computational complexity from cubic to linear in horizon length.

The comparison employs multiple shooting with CPU-based RK4 integration. Both solvers receive identical problem formulations generated through CasADi's symbolic framework, ensuring consistency in derivative computations and constraint structures. The results presented in section 4.1.3 show that FATROP consistently performs better than IPOPT, leading to its adoption for the optimal algorithm configuration.

3.3 Algorithm Reliability Assessment

The reliability evaluation of trajectory optimization algorithms requires comprehensive testing across representative problem scenarios and statistical assessment of performance characteristics. This section establishes the verification framework employed to characterize implementation performance across the operational envelope. The methodology integrates stratified sampling techniques for efficient parameter space exploration with MC analysis for statistical reliability assessment.

3.3.1 Test Problem Suite Definition

The benchmark problem suite spans the operational envelope through variation of spacecraft configuration and maneuver characteristics. The problem difficulty is increased with every configuration, enabling assessment of algorithm robustness across configurations ranging from analytically tractable cases to computationally challenging scenarios.

Problem Configurations and Maneuver Types

Three evaluation scenarios are considered, progressing from a simpler case to increasingly complex operational conditions:

1. **Scenario A: Rest-to-rest with symmetric spacecraft** – Inertia tensor $J_1 = J_2 = J_3 = 1.0$ (dimensionless) with zero initial and final angular velocities ($\boldsymbol{\omega}_0 = \boldsymbol{\omega}_f = \mathbf{0}$). This configuration decouples Euler's rotational equations (2.5), eliminating gyroscopic coupling terms.

2. **Scenario B: Track-to-track with symmetric spacecraft** – Inertia tensor $J_1 = J_2 = J_3 = 1.0$ (dimensionless) with non-zero initial and/or final angular velocities ($\boldsymbol{\omega}_0, \boldsymbol{\omega}_f \in [\boldsymbol{\omega}_{\min}, \boldsymbol{\omega}_{\max}]$). This scenario evaluates algorithm performance under more complex boundary conditions while maintaining decoupled dynamics.

3. **Scenario C: Track-to-track with asymmetric spacecraft** – Inertia ratio $J_1 : J_2 : J_3 = 1 : 2 : 3$, representative of small satellite configurations with distinct principal axes, with non-zero boundary velocities. This configuration combines gyroscopic coupling with complex boundary conditions, representing the most challenging operational scenario.

Initialization Approach

A deterministic baseline employs warm-starting from a previously converged optimal trajectory for a rest-to-rest 90° rotation maneuver about the second principal axis (y-axis). This reference trajectory comprises complete state and control histories, as well as the time-step size and serves as the initial guess for all maneuver scenarios regardless of boundary condition similarity. This trajectory provides the solver with clues about the bang-bang behavior of the control parameters and a reasonable guess for maneuver duration. Additionally, since it is a real trajectory, all its points lie in the feasible parameter space, avoiding infeasibility problems stemming from bad initial guesses.

More accurate initialization techniques are implemented, namely PSO (subsection 2.6.3). Nevertheless, its use comes with and added complexity and computation time. This last aspect can be offset with improved gradient-based solver convergence, but its implementation is delayed until proven necessary.

3.3.2 Monte Carlo Simulation Framework

Statistical verification employs MC simulation with LHS (section 2.7.3) to characterize algorithm performance across the parameter space. This framework balances statistical rigor with computational feasibility. Boundary condition parameters are sampled from uniform distributions representing diverse operational scenarios. The sampling ranges are explained below and summarized in table 3.3.

- **Attitude quaternions:** Terminal attitudes are generated through uniform sampling of Euler angles from the interval $[-180°, 180°]$ for each of the three axis. These are subsequently converted into quaternions, ensuring their placement on the unit sphere. While this approach does not produce perfectly uniform coverage of the rotation group SO(3), as some configurations are redundant, it still provides adequate diversity for MC analysis across the operational envelope. The initial attitude is fixed to $\{0°, 0°, 0°\}$ since any pair of initial and final attitudes can be rotated to this coordinate frame for solving, and then the solution can be adjusted back to the boundary conditions interval (equations (2.51) and (2.52)).

- **Angular velocities:** Initial and terminal angular velocity components are sampled uniformly from $[-3°/\text{s}, 3°/\text{s}]$ per axis, and are normalized afterward so that $||\boldsymbol{\omega}|| \leq 3°/\text{s}$. This range encompasses typical spacecraft slew rates while remaining within typical star-sensor measurement capabilities [46].

3.3.3 Performance Metrics and Evaluation Criteria

Algorithm assessment employs quantitative metrics aligned with the research objectives established in chapter 1. Metrics span computational efficiency, solution quality, and convergence reliability dimensions.

Table 3.3 Monte Carlo simulation parameter ranges for boundary condition sampling.

Category	**Parameter**	**Range**
Initial Attitude	Euler angles (3 components)	**0**° (fixed)
	Converted to quaternion	Identity quaternion {1, 0, 0, 0}
Terminal Attitude	Euler angles (3 components)	[−180°, 180°] each
	Converted to quaternion	Unit sphere
Initial Angular Velocity	Rest-to-rest	**0**°/s (fixed)
	Per-axis components (3 axes)	[−3°/s, 3°/s] each*
Terminal Angular Velocity	Rest-to-rest	**0**°/s (fixed)
	Track-to-track (3 axes)	[−3°/s, 3°/s] each*

*Values are normalized afterward so that $||\boldsymbol{\omega}|| \leq 3°/\text{s}$.

Computational Efficiency Metrics

Total computation time $\mathcal{T}_{\text{total}}$ decomposes into initialization and solver contributions:

$$\mathcal{T}_{\text{total}} = \mathcal{T}_{\text{solver}} \tag{3.1}$$

where $\mathcal{T}_{\text{solver}}$ represents gradient-based solving time. The real-time threshold established in chapter 1 requires $\mathcal{T}_{\text{total}} < 200\,\text{ms}$ for onboard trajectory generation.

Solution Quality Metrics

Optimality gap quantifies solution quality relative to reference trajectories:

$$\Delta_{\text{opt}} = \frac{|T_{\text{computed}} - T_{\text{ref}}|}{T_{\text{ref}}} \times 100\% \tag{3.2}$$

where T_{computed} denotes the maneuver duration obtained by the evaluated algorithm and T_{ref} represents the reference optimal duration from either analytical solutions (table 3.2) or CGPOPS. Acceptable solution quality requires $\Delta_{\text{opt}} < 0.1\%$, balancing near-optimality with computational efficiency.

Convergence Reliability Metrics

Convergence rate $\mathcal{R}_{\text{conv}}$ measures the fraction of trials achieving feasible, optimal solutions:

$$\mathcal{R}_{\text{conv}} = \frac{N_{\text{success}}}{N_{\text{total}}} \times 100\% \tag{3.3}$$

where N_{success} denotes trials satisfying convergence criteria and N_{total} represents total trial count. Convergence criteria require: satisfaction of NLP optimality and constraint violations tolerances, and optimality gap $\Delta_{\text{opt}} < 0.1\%$ when reference solutions are available. High-reliability operation demands $\mathcal{R}_{\text{conv}} > 99\%$.

Table 3.4 summarizes acceptance criteria for algorithm verification.

3.4 PSO Initialization Strategy Development and Evaluation

Nonconvex trajectory optimization exhibits convergence behavior critically dependent on initialization quality, with poor initial guesses yielding convergence failure, constraint violations, or entrapment in suboptimal local minima (section 2.7.1). This section develops initialization strategies following the two-stage paradigm established in section 2.6.3: derivative-free global search via PSO

Table 3.4 Performance metric definitions and acceptance criteria for algorithm verification.

Category	**Metric**	**Symbol**	**Acceptance Criterion**
Computational Efficiency	Total computation time	$\mathcal{T}_{\text{total}}$	< 200 ms
Solution Quality	Optimality gap	Δ_{opt}	$< 0.1\%$
Convergence Reliability	Convergence rate	$\mathcal{R}_{\text{conv}}$	$> 99\%$

followed by gradient-based refinement using FATROP with multiple shooting and CPU-based RK4 integration. The evaluation compares two control parameterization approaches: full trajectory optimization versus STO, and benchmarks stochastic initialization against the previously characterized deterministic initialization method.

3.4.1 Control Parameterization Approaches

Two fundamentally different parameterization strategies are evaluated for PSO-based initialization, each representing distinct trade-offs between search space dimensionality and structural assumptions about optimal control form. All PSO implementations leverage GPU parallelization for concurrent particle evaluation, exploiting the embarrassingly parallel structure of population-based metaheuristics (section 2.7.2).

Full Control Trajectory Parameterization

Full control trajectory parameterization treats control values at all 50 discretization points as independent PSO decision variables, imposing no *a priori* structural constraints. It then uses the generated control trajectory to propagate the dynamics from the initial state, and enforces the final state constraint through a penalty on the objective function. The decision variable vector comprises the control torques $\tau_{i,k}$ for each of the 50 discretization intervals k and for each of the 3 control axis i, and the time-step Δt, yielding dimensionality $O(50\times3+1) = O(151)$. This high-dimensional search space grants complete flexibility to discover arbitrary control structures but increases computational burden and may impede convergence for metaheuristic algorithms.

Dynamic weight strategies are applied to all three weights in equation (2.63) to balance exploration in early iterations with exploitation during convergence by linearly decreasing parameters from initial to minimum values over the optimization horizon. Therefore, the PSO parameters subject to tuning comprise: inertia weight w, cognitive parameter w_1, social parameter w_2, population size n_p, iteration count $\mathcal{K}$, and dynamic weight bounds $w_{\min}$, $w_{1,\min}$, $w_{2,\min}$ for time-varying parameters.

STO Parameterization

STO exploits the bang-bang control structure established in section 2.2.3 and equations (2.27) and (2.62) by parameterizing trajectories through switching times n_i and initial control directions s_i for each of the 3 axis i. From the available analytical solutions, the maximum number of control switches per axis $n_{i,max}$ is 3 [3]. However, the total maximum number of control switches $n_{\max}$ for all 3 axis in an optimal trajectory is 6 (table 3.2). The dimensionality of the problem is then reduced to $O(3 + 6 + 1) = O(10)$, substantially decreasing search space volume when compared to the $O(151)$ dimensionality of the previous method and improving metaheuristic convergence characteristics discussed in section 2.4.

Binary PSO addresses the discrete control direction variables through sigmoid transfer functions (equation (2.64)), introducing additional tuning parameters beyond those in full parameterization: sigmoid scaling parameter α controlling transition steepness and velocity saturation threshold $\epsilon \in [0.5, 1.0]$ maintaining stochastic behavior (equation (2.65)). The continuous switching time variables employ standard PSO updates with temporal ordering constraints $t_0 < t_i^{(1)} < \cdots < t_i^{(y)} < t_f$ enforced by reordering the values at each iteration.

3.4.2 Parameter Tuning and Optimization

Particle Swarm Optimization performance depends critically on parameter selection (section 2.6.3). Systematic tuning via LHS identifies configurations balancing convergence reliability with computational efficiency. The tuning methodology evaluates PSO performance when coupled with gradient-based refinement, reflecting the two-stage approach established in section 3.4.

Tuning Methodology

Parameter tuning employs nested optimization: an outer LHS loop samples PSO parameter combinations while an inner evaluation assesses performance on representative combination of rest-to-rest maneuvers. The methodology proceeds as follows:

1. Generate 1000 PSO parameter combinations via LHS across specified parameter ranges (table 3.5).
2. For each parameter combination, execute 1000 independent trials with LHS-sampled initial and final boundary conditions.
3. Record convergence rate $\mathcal{R}_{\text{conv}}$ and mean computation time $\mathcal{T}_{\text{total}}$ for each parameter combination.
4. Identify configurations that minimize computation time and maximize convergence rate.

Parameter Space

For full trajectory parameterization (section 3.4.1), the parameter space comprises:

- **Population size** ($\mathcal{N}$): The population size is constrained to multiples of 640, corresponding to the number of available GPU cores, to maximize parallel efficiency. The upper bound is set at 5120 particles to ensure PSO computation time remains below 100 ms, allocating sufficient computational budget for subsequent gradient-based refinement within the 200 ms real-time constraint: $\mathcal{N} \in \{640, 1280, 1920, 2560, 3200, 3840, 4480, 5120\}$.
- **Iteration count** ($\mathcal{K}$): The maximum iteration count is capped at 750 to maintain total PSO computation time below 100 ms across the population size range, preserving adequate time allocation for solver convergence: $\mathcal{K} \in [10, 750]$.
- **Exploration parameters** (w_0, w_1, w_2): Initial values for inertia weight, cognitive parameter, and social parameter are sampled from $[0.1, 10.0]$. While typical PSO implementations employ values in the range $[0.5, 3.0]$, the extended upper bound enables exploration of aggressive parameter settings potentially beneficial for high-dimensional trajectory optimization problems: $w_0, w_1, w_2 \in [0.1, 10.0]$.
- **Minimum exploration parameters** ($w_{0,\text{min}}$, $w_{1,\text{min}}$, $w_{2,\text{min}}$): Terminal values for dynamic weight scheduling are sampled as fractions of their corresponding initial values to ensure monotonic decrease throughout optimization. Sampling $\zeta_0, \zeta_1, \zeta_2 \in [0.0, 1.0]$ yields minimum parameters $w_{0,\text{min}} = \zeta_0 \cdot w_0$, $w_{1,\text{min}} = \zeta_1 \cdot w_1$, $w_{2,\text{min}} = \zeta_2 \cdot w_2$, satisfying the constraint $w_{0,\text{min}} \leq w_0$, $w_{1,\text{min}} \leq w_1$, $w_{2,\text{min}} \leq w_2$ by construction.

For STO parameterization employing binary PSO (section 3.4.1), additional parameters include:

- **Sigmoid scaling** (α): The scaling parameter controlling sigmoid function steepness in the binary position update (equation (2.64)) is sampled from the same extended range as continuous parameters to enable exploration of both gradual and sharp probabilistic transitions: $\alpha \in [0.1, 10.0]$.
- **Sigmoid saturation threshold** (ϵ): The saturation threshold determining maximum sigmoid output bounds (equation (2.65)) is sampled from $[0.5, 1.0]$, where $\epsilon = 0.5$ enforces complete randomization and $\epsilon = 1.0$ permits deterministic position updates for large velocity magnitudes, as established in section 2.6.3.

Table 3.5 summarizes parameter ranges.

Table 3.5 PSO parameter ranges for LHS-based tuning.

Category	**Parameter**	**Range**
Swarm Configuration	Population size	$[640, 5120]^*$
	Iteration count	$[10, 750]$
Inertia Weight	Initial weight	$[0.1, 10.0]$
	Minimum weight	$[0.0\%, 1.0\%]$
Cognitive Parameter	Initial weight	$[0.1, 10.0]$
	Minimum weight	$[0.0\%, 1.0\%]$
Social Parameter	Initial weight	$[0.1, 10.0]$
	Minimum weight	$[0.0\%, 1.0\%]$
Binary PSO (STO only)	Sigmoid scaling	$[0.1, 10.0]$
	Sigmoid saturation	$[0.5, 1.0]$

*Values are sampled in multiples of 640 (number of available GPU cores).
Actual allowed values: $\{640, 1280, 1920, 2560, 3200, 3840, 4480, 5120\}$

3.4.3 Performance Evaluation

Once both PSO approaches are correctly tuned, their performance is compared to the same gradient-based algorithm using a fixed previously-converged optimal trajectory as the initial guess. The methodology described in section 3.4 is followed again and the same metrics are used to evaluate the new algorithms' performance. The computational efficiency metric (3.4) is now:

$$\mathcal{T}_{\text{total}} = \mathcal{T}_{\text{solver}} + \mathcal{T}_{\text{PSO}} \tag{3.4}$$

where $\mathcal{T}_{\text{PSO}}$ is the time spent on the PSO.

The results of this analysis determine the most reliable algorithm configuration developed throughout this paper and can be found in section 4.4.

3.5 Implementation Details

This section documents the computational platform, software architecture, and parameter values employed in the present study. Complete specification of the implementation environment enables reproducibility of results and facilitates comparative evaluation against alternative approaches. The computational infrastructure comprises consumer-grade hardware, demonstrating the feasibility of real-time trajectory optimization without requiring specialized aerospace computing resources.

3.5.1 Computational Platform Specifications

All computational experiments were executed on a machine equipped with an Intel Core i7-8750H processor and NVIDIA GeForce GTX 1050 Mobile graphics accelerator. This hardware configuration represents a mid-range computational platform from 2018.

Hardware Configuration

The computational platform comprises two primary processing units whose specifications are detailed in table 3.6.

Table 3.6 Computational platform hardware specifications.

Component	Specification	Value
CPU	Model	Intel Core i7-8750H
	Architecture	Coffee Lake (14 nm)
	Physical cores	6
	Logical threads	12
	Base frequency	2.20 GHz
	Maximum frequency	4.10 GHz
	L3 cache	9 MB
	TDP	45 W
GPU	Model	NVIDIA GeForce GTX 1050 Mobile
	Architecture	Pascal (GP107M)
	CUDA cores	640
	SM count	5
	Base clock	1354 MHz
	Boost clock	1493 MHz
	Memory capacity	4 GB GDDR5
	Memory bandwidth	112 GB/s
	Compute capability	6.1
	TDP	75 W

Software Environment

The software stack comprises open-source tools and libraries selected for numerical robustness, community support, and cross-platform portability. Table 3.7 enumerates version-specific identifiers enabling exact reproduction of the computational environment.

3.5.2 Problem-Specific Parameter Values

This subsection documents numerical parameter values employed throughout the study, organized by functional category. Unless otherwise stated, parameters remained constant across all computational experiments to ensure consistency in comparative evaluations.

Numerical Precision

Floating-point precision is selected according to computational requirements and hardware characteristics. All CPU-based optimization computations employ double-precision (64-bit) arithmetic to maintain numerical accuracy throughout the NLP solution process. The PSO initialization

Table 3.7 Software environment version specifications.

Category	**Component**	**Version**
Operating System	Ubuntu Linux	24.04 LTS
Compiler	GCC/g++	13.3.0
	C++ Standard	C++17
GPU Computing	CUDA Toolkit	12.8.93
Libraries	CasADi	3.7.0+
	SUNDIALS	6.7.0
	IPOPT	3.14.18
	FATROP	1.0.0
	CGPOPS	1.0
Build System	CMake	3.28.3

algorithms are implemented using single-precision (32-bit) arithmetic on the GPU, yielding computational performance improvements of approximately 1.5× relative to double-precision implementations. Single-precision arithmetic is not evaluated for GPU dynamics propagation since the resulting numerical accuracy proves insufficient when compared with the accuracy required by the NLP solvers.

NLP Solver Configuration

IPOPT and FATROP are used in their default configuration with CasADi, with the exception of the parameters summarized in table 3.8.

Table 3.8 NLP solver configuration parameters.

Solver	**Parameter**	**Value**	**Description**
IPOPT	`tol`	10^{-7}	Convergence tolerance
	`max_iter`	1000	Maximum iterations
	`linear_solver`	MUMPS	Sparse direct linear solver
	`mu_strategy`	adaptive	Barrier parameter update strategy
	`warm_start_init_point`	yes	Enable warm-start initialization
FATROP	`tol`	10^{-7}	Convergence tolerance
	`max_iter`	1000	Maximum iterations
	`linear_solver`	BLASFEO	Linear algebra backend

CGPOPS Verification Parameters

Reference solution generation through CGPOPS employs the mesh refinement and accuracy parameters listed in table 3.9. These settings ensure reference solution accuracy exceeds test algorithm precision, enabling reliable verification. CGPOPS uses the LGR pseudospectral collocation method (section 2.3.2) to transcribe the OCP.

3.6 Chapter Summary

This section summarizes the methodology adopted for this thesis with the help of diagrams. Diagram 3.1 depicts the foundational work described in section 3.1 that leads to the development of the

Table 3.9 CGPOPS configuration parameters.

Parameter	**Value**	**Description**
	Derivative Computation and Scaling	
`derivativeSupplier`	1	Bicomplex-step differentiation
`scaled`	1	Automatic variable scaling
	Initial Mesh Configuration	
`numintervals`	50	Initial mesh intervals per phase
`initcolpts`	5	Initial collocation points per interval
	Adaptive Mesh Refinement	
`meshRefineType`	1	hp-Patterson-Rao refinement
`meshTol`	10^{-7}	Mesh convergence tolerance
`maxMeshIter`	5	Maximum refinement iterations
`minColPts`	4	Minimum collocation points per interval
`maxColPts`	10	Maximum collocation points per interval
	NLP Solver Configuration	
`runIPOPTFlag`	1	Enable IPOPT solver
`NLPtol`	10^{-7}	NLP convergence tolerance
`NLPmaxiter`	1000	Maximum NLP iterations
`linear_solver`	MUMPS	IPOPT linear solver backend
	Control Structure Detection	
`useLTIHDD`	1	Bang-bang control detection

baseline algorithm. This is then sequentially optimized as shown in diagram 3.2 and described in section 3.2. The optimized algorithm resilience to changing boundary conditions is tested through a MC simulation (diagram 3.3). This scheme is followed once for rest-to-rest maneuvers and a second time for track-to-track maneuvers, both using symmetric spacecraft inertial parameters. It is followed a third and last time for track-to-track maneuvers, this time using an asymmetric spacecraft. The exact parameters used and boundary conditions bounds can be found in section 3.3. Diagram 3.4 represents the methodology used to develop and tune the PSO initialization strategies, and section 3.4 provides a more in-depth description. Finally, diagram 3.5 shows how the final combinations of PSO and optimized solver are evaluated for their performance in the same three scenarios described previously.

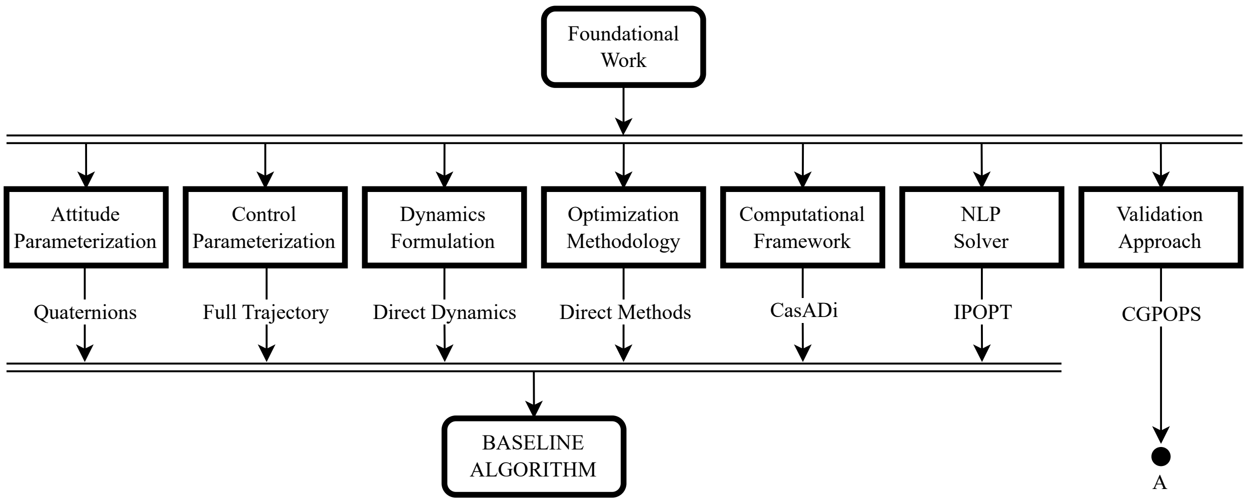


Figure 3.1 Foundational work and baseline algorithm implementation diagram.

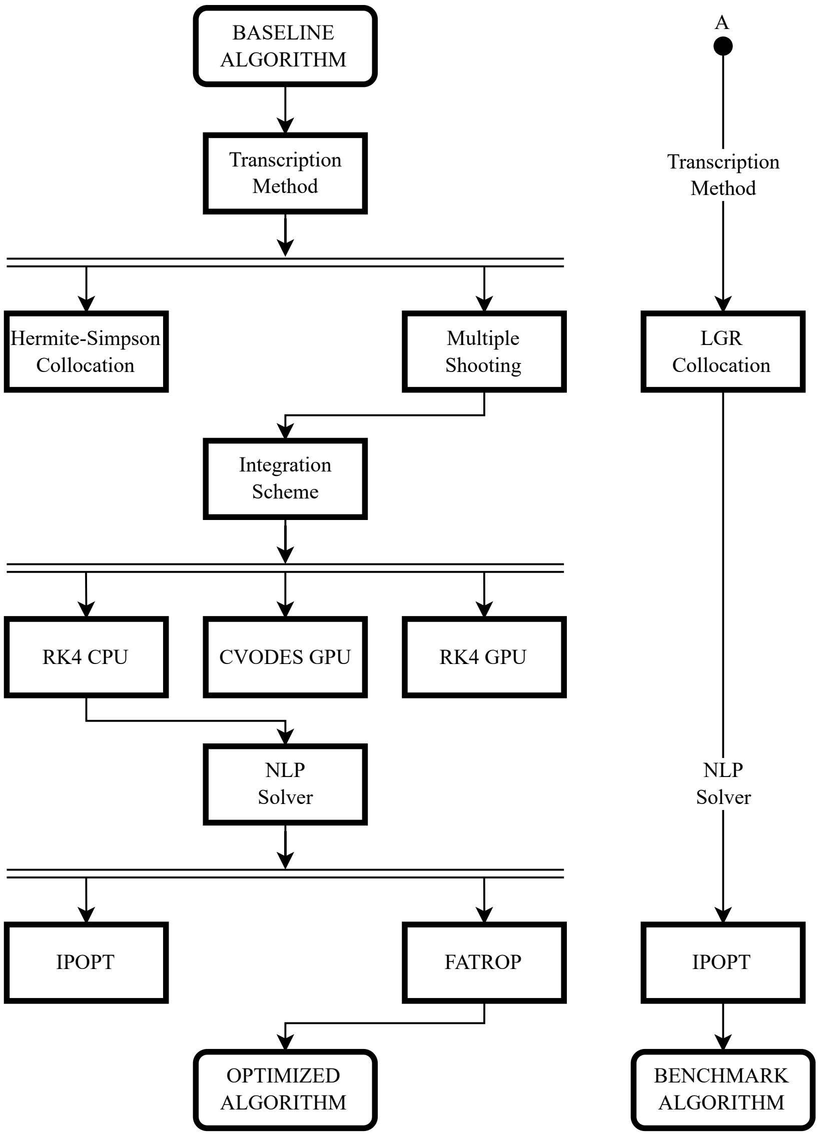


Figure 3.2 Optimized and benchmark algorithm development diagram.

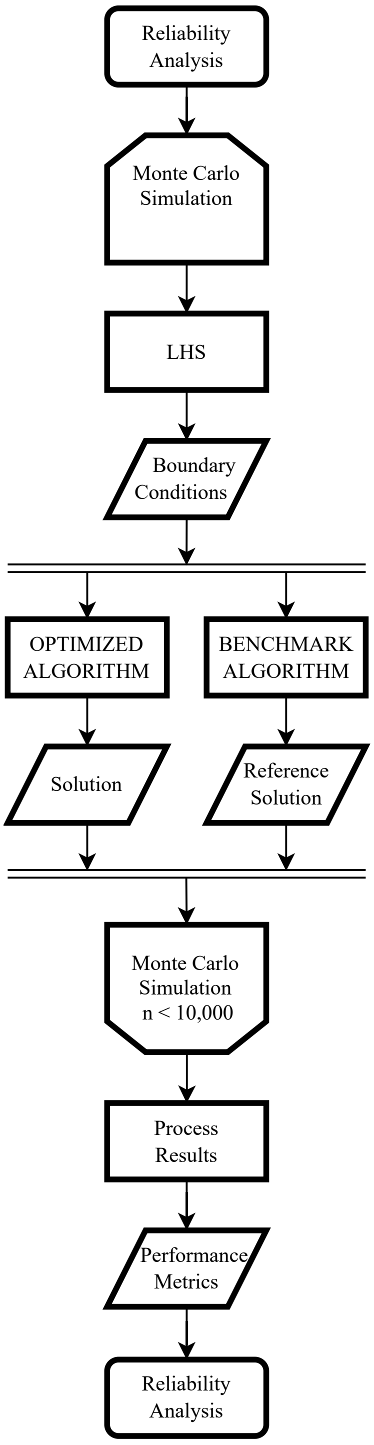


Figure 3.3 Optimized algorithm reliability analysis diagram.

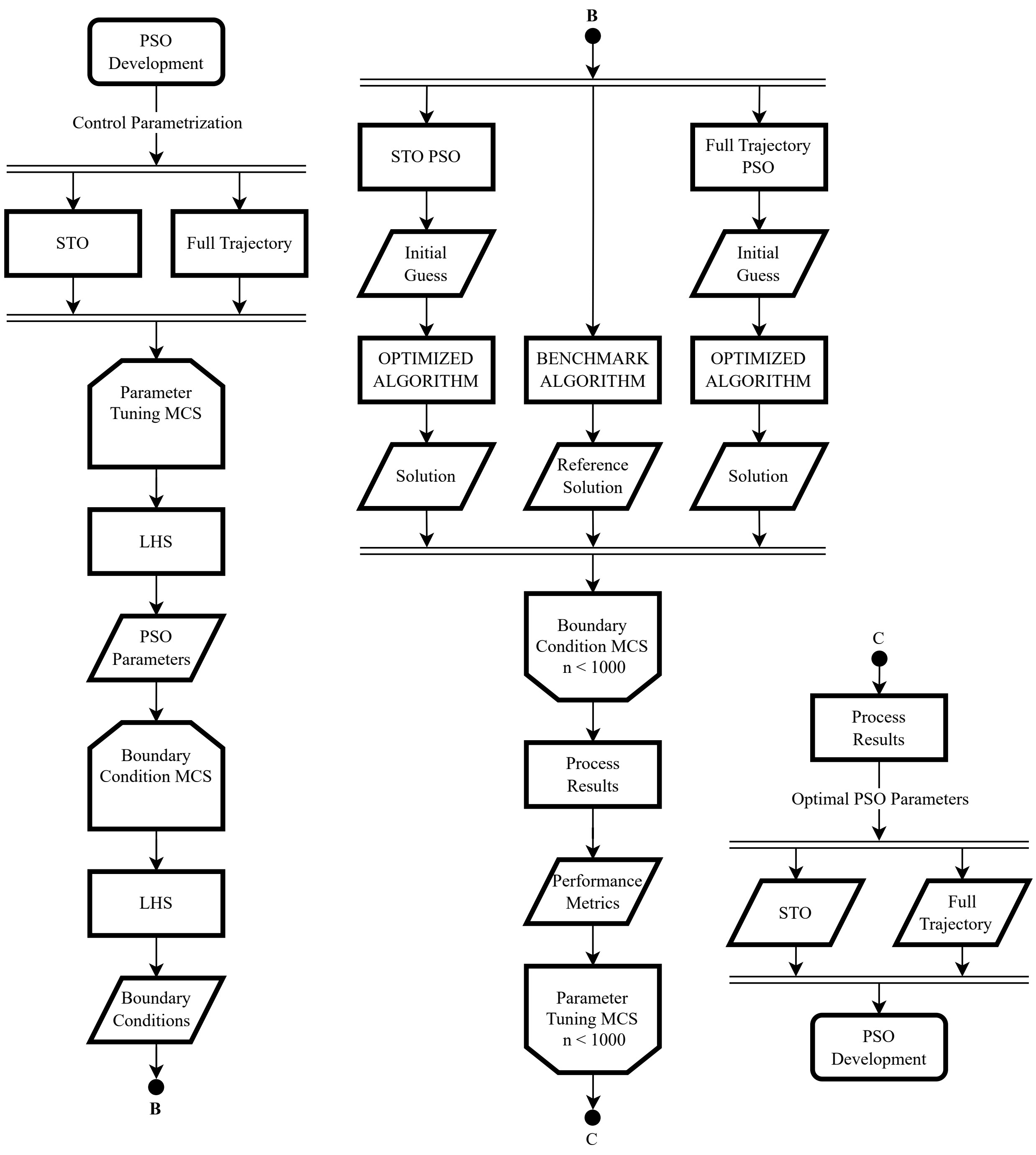


Figure 3.4 PSO development diagram.

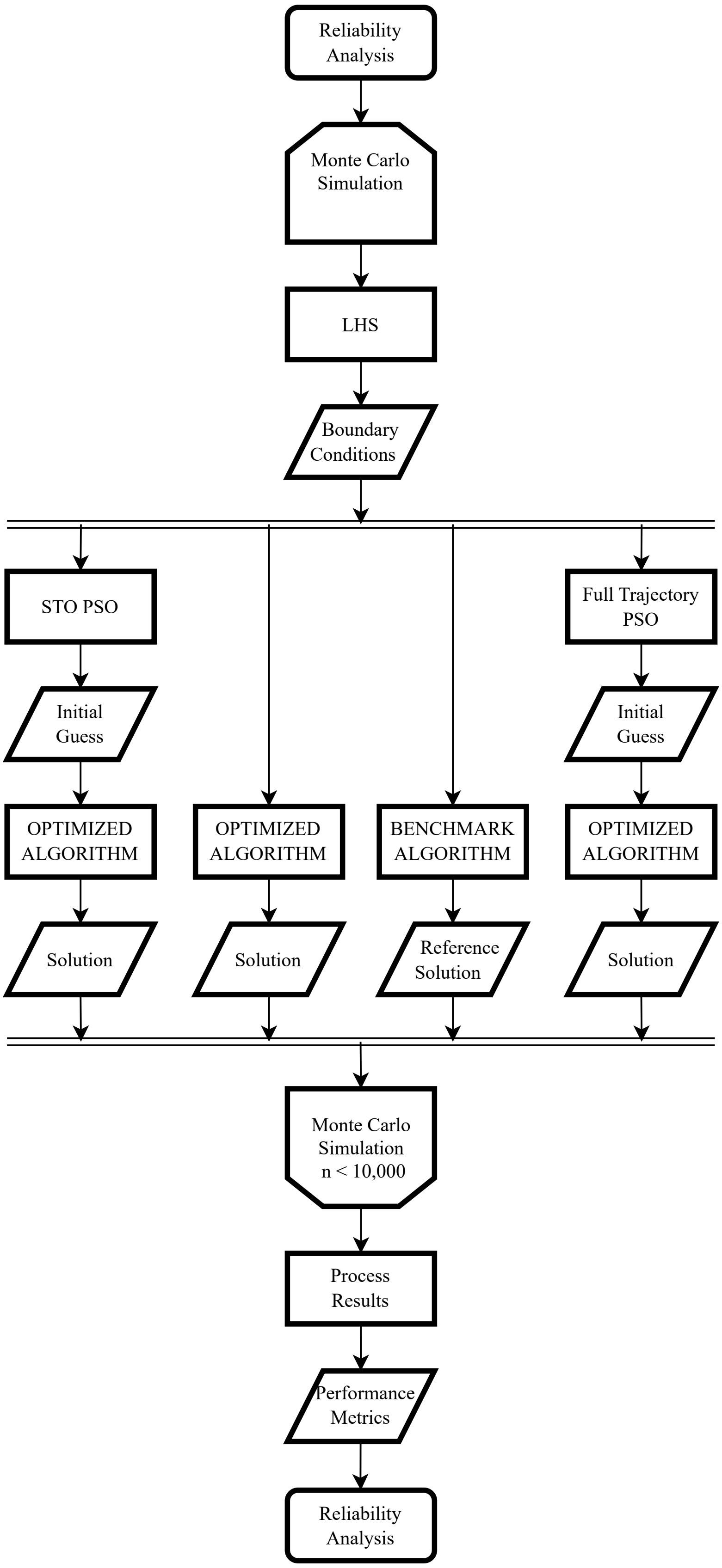


Figure 3.5 PSO reliability analysis.

4 Results

This chapter presents the numerical results obtained from the gradient-based optimization framework developed in chapter 3. The performance of different transcription methods is evaluated through benchmark maneuvers, and the computational characteristics of the implemented algorithms are analyzed.

4.1 Gradient-based Strategy Optimization Results

The numerical experiments are conducted using a symmetric spacecraft configuration with characteristic scales $J_0 = 1$ kg·m^2 and $\tau_{\max} = 1$ N·m for simplicity. The spacecraft dynamics and optimal control problem formulation follow the definitions established in section 3.2. All maneuvers consist of rest-to-rest reorientations about the first principal axis (roll axis), with initial condition $\mathbf{q}_0 = [1, 0, 0, 0]^T$.

4.1.1 Transcription Method Comparison

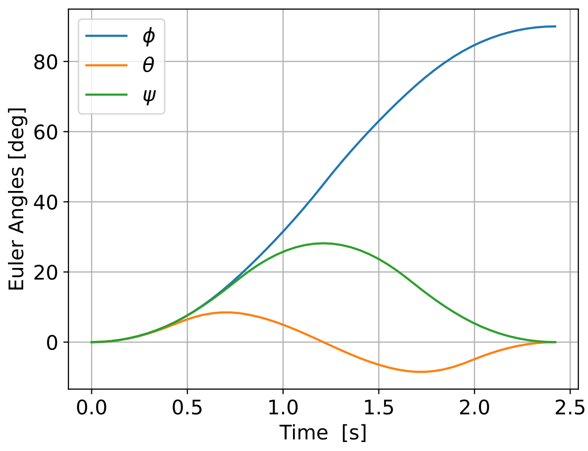
This section compares the computational performance and solution quality of multiple shooting paired with RK4 and Hermite-Simpson collocation transcription methods. Detailed framework and reasoning for method selection is found in section 3.2.1.

90° Maneuver

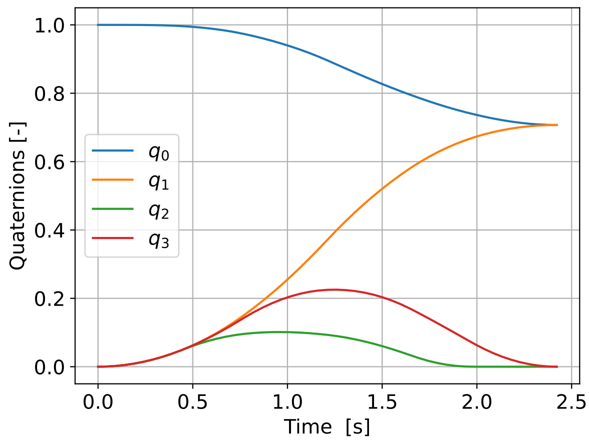
The time-optimal trajectories for the 90° rest-to-rest maneuver obtained via multiple shooting and Hermite-Simpson collocation are presented in figures 4.1 and 4.2, respectively. Both transcription methods successfully converged to optimal solutions satisfying the necessary conditions for optimality. However, despite utilizing identical initial guess trajectories, the two methods converged to distinct local optima, as evidenced by the different control switching structures visible in figures 4.1d and 4.2d. This behavior confirms the nonconvex nature of the time-optimal spacecraft reorientation problem discussed in section 2.5.2, wherein multiple locally optimal solutions exist in the optimization landscape.

The Euler angle trajectories (figures 4.1a and 4.2a) demonstrate smooth monotonic rotation about the first principal axis, while the quaternion components (figures 4.1b and 4.2b) exhibit the characteristic evolution along the unit hypersphere. Angular velocity profiles (figures 4.1c and 4.2c) display the expected symmetry for rest-to-rest maneuvers, with peak angular rates occurring near the trajectory midpoint. The control torque histories (figures 4.1d and 4.2d) exhibit bang-bang structure with 5 switches, consistent with PMP for time-optimal control of this problem class.

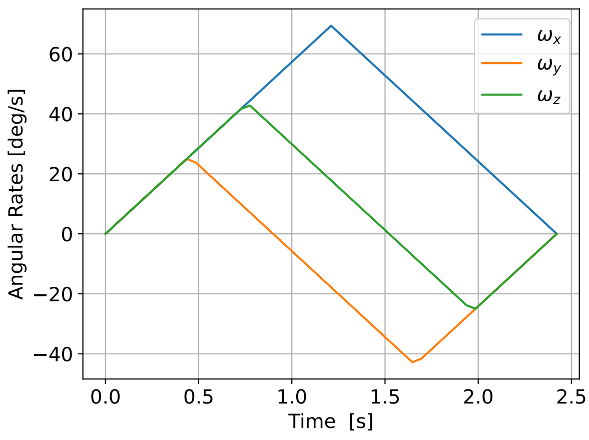
Quantitative performance metrics are summarized in table 4.1. The optimal maneuver duration serves as the baseline for error assessment. Multiple shooting achieved a maneuver duration of 2.4214 s with 0.012% error, while Hermite-Simpson collocation yielded 2.4220 s with 0.038% error. Multiple shooting demonstrated superior computational efficiency, requiring 0.961 s compared to 3.974 s for collocation (4× speedup and a 3× increase in accuracy). The increased computational efficiency and enhanced solution accuracy of multiple shooting are proof that problem sparsity is an advantage when solvers are capable of exploiting it, as is the case of IPOPT (section 2.6.1).



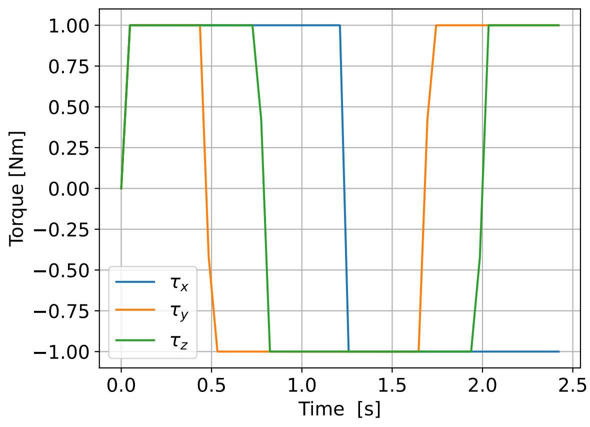
(a) Euler angles trajectory.



(b) Quaternion trajectory.



(c) Angular velocity components.

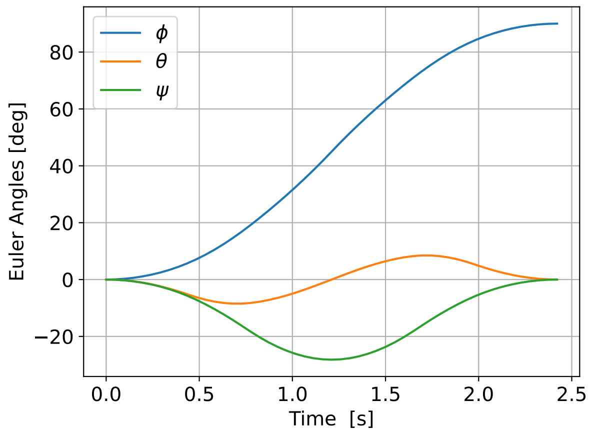


(d) Control torque inputs.

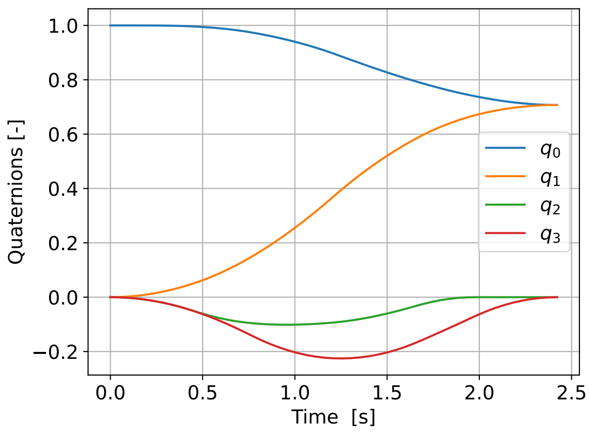
Figure 4.1 Time-optimal trajectory using multiple shooting transcription for a 90° maneuver.

Table 4.1 Performance comparison of transcription methods for a 90° maneuver.

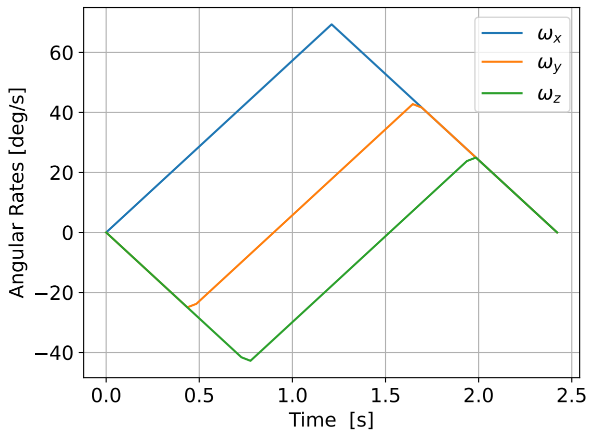
Metric	**Optimal**	**Multiple Shooting**	**HS Collocation**
Computation time [s]	–	0.961	3.974
Control switches [-]	5	5	5
Maneuver duration [s]	2.4211	2.4214	2.4220
Optimality gap Δ_{opt} [%]	0	0.012	0.038



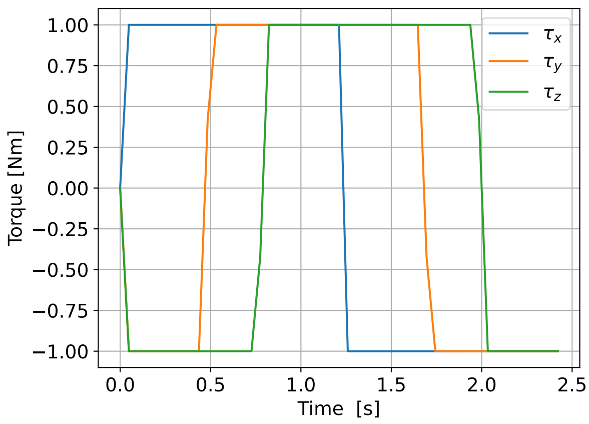
(a) Euler angles trajectory.



(b) Quaternion trajectory.



(c) Angular velocity components.



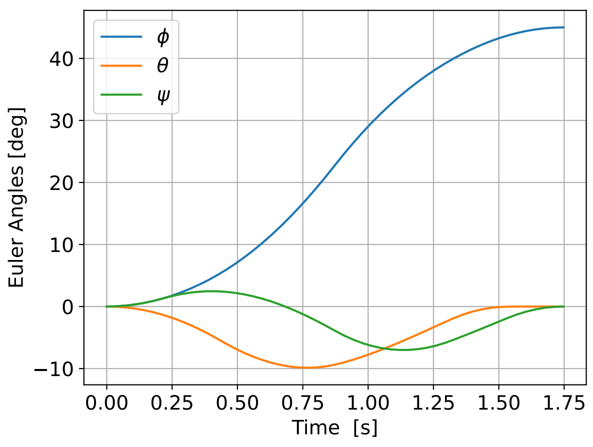
(d) Control torque inputs.

Figure 4.2 Time-optimal trajectory using Hermite-Simpson collocation transcription for a 90° maneuver.

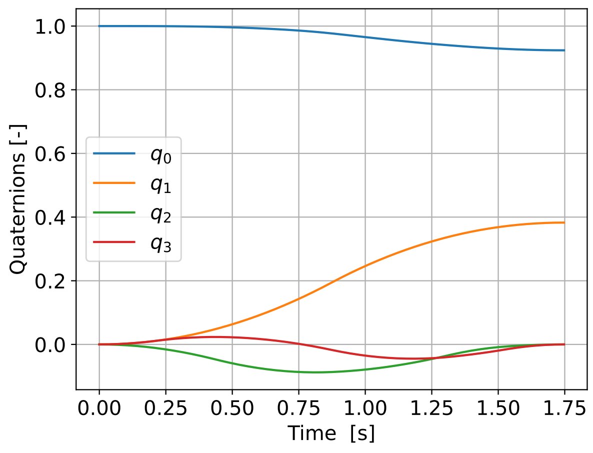
45° Maneuver

The optimal trajectories for the 45° maneuver are illustrated in figure 4.3. Unlike the 90° case, both transcription methods converged to the same optimal solution from the identical initial guess. This may indicate that the 45° problem exhibits a more favorable optimization landscape with stronger convergence properties toward the global optimum, however no certain claims can be made without further analysis.

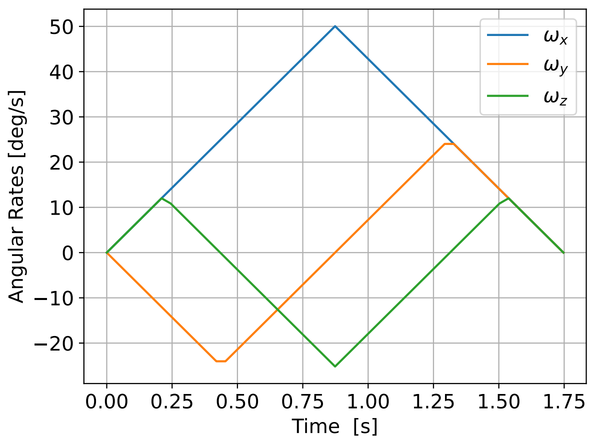
The control structure for this maneuver exhibits 6 switches, as shown in figure 4.3d. The successful identification of the 6-switch solution by both methods rather than a suboptimal 7-switch structure demonstrates the reliability of the implemented transcription approaches.



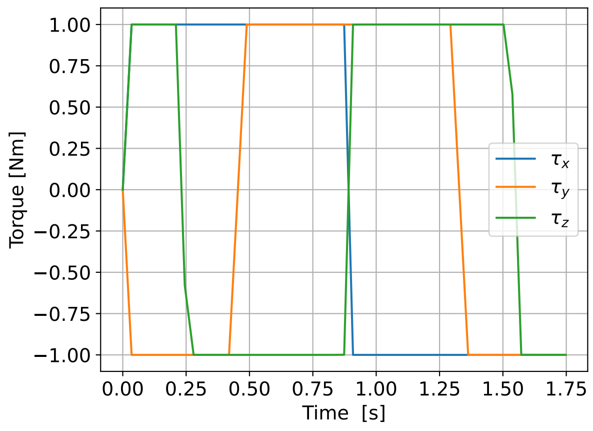
(a) Euler angles trajectory.



(b) Quaternion trajectory.



(c) Angular velocity components.



(d) Control torque inputs.

Figure 4.3 Time-optimal trajectory for a 45° maneuver.

Performance metrics in table 4.2 reveal that multiple shooting achieved 0.005% error with 1.265 s computation time, while Hermite-Simpson collocation yielded 0.031% error requiring 7.977 s: a 6× speedup and accuracy increase for multiple shooting .

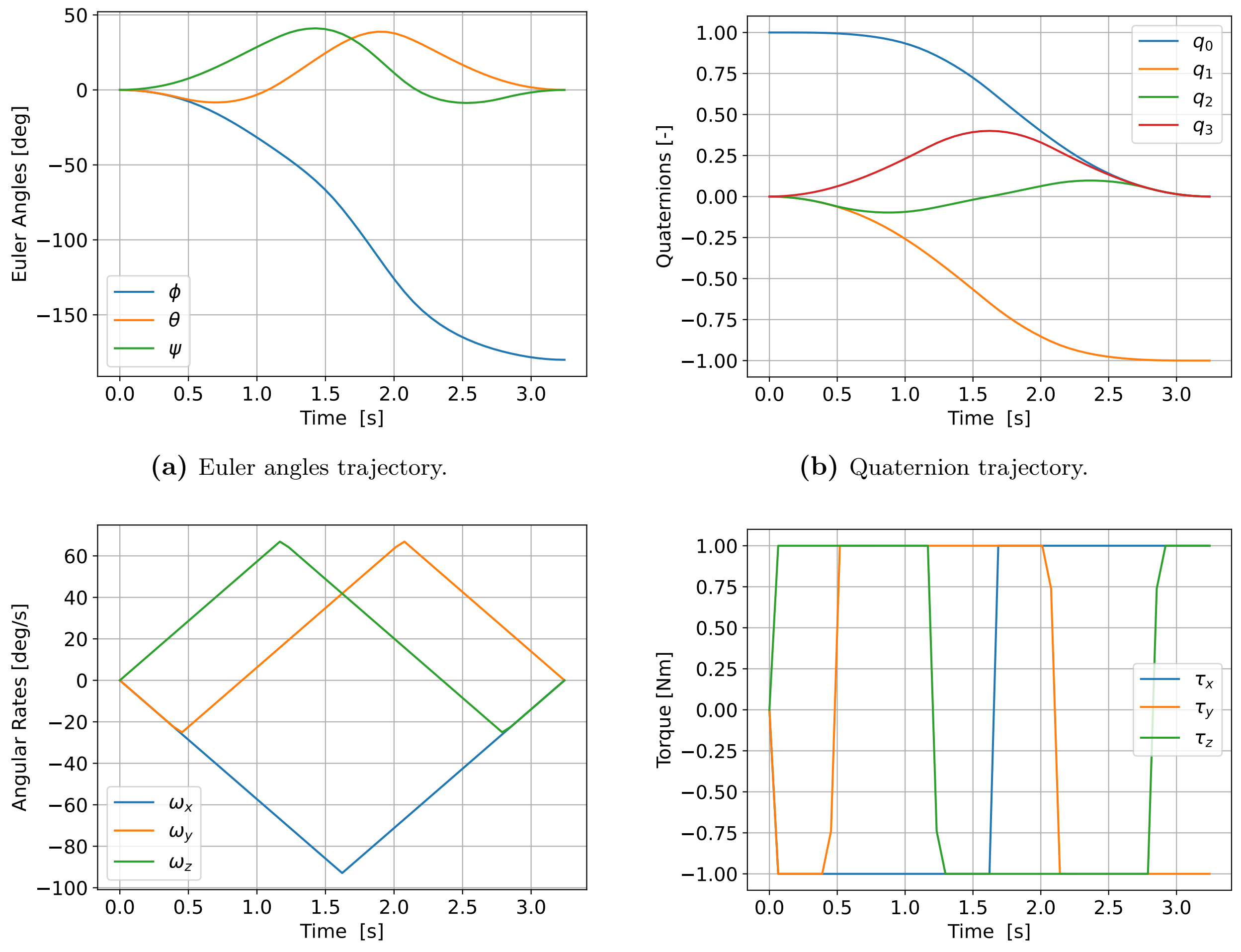
180° Maneuver

The 180° rest-to-rest maneuver represents the maximum rotation angle on SO(3) and poses particular challenges due to the quaternion sign ambiguity discussed in section 2.5.2. The optimal trajectories are presented in figure 4.4. Both transcription methods converged to identical optimal solutions exhibiting 5 control switches, as shown in figure 4.4d.

Table 4.2 Performance comparison of transcription methods for a 45° maneuver.

Metric	**Optimal**	**Multiple Shooting**	**HS Collocation**
Computation time [s]	–	1.265	7.977
Control switches [-]	6	6	6
Maneuver duration [s]	1.7471	1.7472	1.7476
Optimality gap Δ_{opt} [%]	0	0.005	0.031

The quaternion trajectory in figure 4.4b demonstrates that the sign convention enforcement procedure (equation (2.50)) successfully prevented quaternion unwinding, ensuring convergence to the minimum-time solution corresponding to rotation through 180° rather than 540°. The angular velocity profile in figure 4.4c exhibits higher peak magnitudes compared to smaller-angle maneuvers, reflecting the increased rotational momentum required for the longer slew.



(a) Euler angles trajectory.

(b) Quaternion trajectory.

(c) Angular velocity components.

(d) Control torque inputs.

Figure 4.4 Time-optimal trajectory for a 180° maneuver.

Computational performance data in table 4.3 show that multiple shooting required 2.512 s with 0.009% error, while collocation required 4.512 s with 0.046% error, corresponding to a 1.8× speedup and a 5× increase in accuracy. The reduced computational advantage of multiple shooting for this case, compared to the 45° and 90° maneuvers, may be attributed to the increased problem complexity arising from the larger rotation angle and the associated longer trajectory duration.

Table 4.3 Performance comparison of transcription methods for a 180° maneuver.

Metric	Optimal	Multiple Shooting	HS Collocation
Computation time [s]	–	2.512	4.512
Control switches [-]	5	5	5
Maneuver duration [s]	3.2431	3.2434	3.2446
Optimality gap Δ_{opt} [%]	0	0.009	0.046

Across all three test cases, multiple shooting consistently demonstrated superior computational efficiency, while simultaneously achieving lower solution errors. These results validate the selection of multiple shooting as the transcription method for the rest of the algorithm development.

4.1.2 Integration Scheme Comparison

With multiple shooting selected as the transcription method, the integration schemed used to propagate the system's dynamics is evaluated. Three methods are compared: a CPU and a GPU implementation of RK4, and a GPU implementation of CVODES. Since the solver settings are changed for this analysis (as explained in section 3.2.2), the results from the CPU-RK4 implementation in this section are different from the ones presented in the previous section.

Table 4.4 shows the results obtained for the 3 different maneuvers. From the three dynamics propagation approaches, the CPU-RK4 was consistently the fastest while also being the most accurate. Conversely, the GPU-accelerated CVODES took 53× to 1682× longer while not providing any improvement in accuracy. As for he GPU-RK4 implementation, computation times ranged from only 1.14× to 27× slower than its CPU equivalent.

Table 4.4 Performance comparison of integration methods for different maneuver angles.

Angle	Metric	Optimal	CPU-RK4	GPU-RK4	GPU-CVODES
45°	Computation time [s]	–	15.890	18.129	852.945
	Control switches [-]	6	6	6	6
	Maneuver duration [s]	1.7471	1.7472	1.7476	1.7473
	Optimality gap Δ_{opt} [%]	0	0.005	0.027	0.014
90°	Computation time [s]	–	2.147	59.575	3612.220
	Control switches [-]	5	5	5	5
	Maneuver duration [s]	2.4211	2.4214	2.4214	2.4225
	Optimality gap Δ_{opt} [%]	0	0.012	0.012	0.059
180°	Computation time [s]	–	6.957	121.336	1559.010
	Control switches [-]	5	5	5	5
	Maneuver duration [s]	3.2431	3.2434	3.2434	3.2434
	Optimality gap Δ_{opt} [%]	0	0.009	0.009	0.009

The GPU-accelerated CVODES implementation was abandoned due to its bad performance combined with its higher complexity interface and reduced costumization possibilities. As for the GPU-RK4 implementation, its performance was further studied. The findings are presented in table 4.5. CPU-GPU synchronization overhead constitutes 71.3% of the total computational budget, while GPU kernel execution and memory transfers account for 13.0% of runtime. The ratio of CPU-side CUDA API blocking time (83.760 s) to actual GPU active time (12.902 s) yields a synchronization efficiency of 15.4%, indicating that the implementation is synchronization-bound rather than compute-bound. This bottleneck originates from synchronous memory transfer operations that serialize CPU-GPU execution, preventing concurrent operation of host and device. The

IPOPT nonlinear programming solver contributes 6.343 s of computational overhead for constraint Jacobian evaluations and KKT system assembly, representing 6.4% of total runtime.

Table 4.5 Computational performance breakdown of the GPU-accelerated optimization framework using RK4 for a 180° rotation.

Operation Category	Time (s)	Percentage (%)
Total Optimization Runtime	99.383	100.0
GPU Operations		
Kernel Execution	7.739	7.8
Memory Transfers	5.163	5.2
Total GPU Active Time	12.902	13.0
CPU Operations		
IPOPT Solver Overhead	6.343	6.4
Function Evaluation Overhead	8.588	8.6
Total CPU Computation Time	14.931	15.0
CPU-GPU Synchronization Overhead	70.858	71.3

The profiling data also revealed 995,866 kernel invocations distributed across 1,656 gradient evaluations. Considering that the solver is using finite differences to compute the Jacobian, and that there are 11 decision variables (7 states, 3 controls and the time step size) for every of the 50 discretization points, $50 \times 11 = 550$ kernel calls are made for every Jacobian evaluation. This means that $550 \times 1,656 = 910,800$ out of the 995,866 kernel calls stem from Jacobian evaluation. Since every discretization point is independent from every other point, finite differences analysis can be implemented at every of he 50 shooting intervals simultaneously reducing the number of calls per Jacobian evaluation from 550 to 11. Table 4.6 compares the performance of this new approach with the CPU-RK4 implementations using finite differences (FD) and automatic differentiation (AD) for gradient computation. A clear performance improvement is seen, with results being comparable with the CPU implementation using FD. However, computation times still remain shy of the ones achieved when using AD. In order to achieve results comparable with the ones achieved with AD, exact jacobian and hessian formulations would need to be implemented.

Table 4.6 Performance comparison of different RK4 integration implementations.

Angle	Metric	CPU-RK4 FD	GPU-RK4	CPU-RK4 AD
45°	Computation time [s]	15.890	8.5087	1.265
	Control switches [-]	6	6	6
	Maneuver duration [s]	1.7472	1.7472	1.7473
	Optimality gap Δ_{opt} [%]	0.005	0.005	0.005
90°	Computation time [s]	2.147	5.8627	0.961
	Control switches [-]	5	5	5
	Maneuver duration [s]	2.4214	2.4214	2.4214
	Optimality gap Δ_{opt} [%]	0.012	0.012	0.012
180°	Computation time [s]	6.957	5.753	2.512
	Control switches [-]	5	5	5
	Maneuver duration [s]	3.2434	3.2434	3.2434
	Optimality gap Δ_{opt} [%]	0.009	0.009	0.009

The performance analysis presented in table 4.7 demonstrates that the optimized GPU implementation achieves a 17.3× speedup compared to the previous GPU implementation (5.753 s versus 99.383 s), reducing IPOPT gradient evaluations from 1,656 to 142 gradient evaluations. The GPU active time constitutes 7.1% of total runtime (0.406 s), comprising 0.248 s for kernel execution across 31,895 invocations and 0.158 s for memory transfers. Pure CPU computation accounts for 43.4% of runtime, with the IPOPT solver framework consuming 1.815 s (31.5%) for gradient evaluations and constraint processing. The CPU-GPU synchronization overhead represents 49.6% of total runtime (2.853 s). Despite achieving a 31.2× reduction in kernel invocations relative to the baseline, the GPU implementation (5.753 s) remains 2.3× slower than the CPU-RK4 version employing automatic differentiation (2.512 s). This performance deficit is attributed to insufficient problem scale for effective GPU utilization: with 50 discretization intervals and 7 states, only $50 \times 7 = 350$ threads are active per invocation, achieving approximately 3.4% of theoretical thread capacity. The computational overhead associated with kernel launches and memory transfers is not amortized at this problem scale, rendering the GPU approach inefficient for this use-case.

Table 4.7 Computational performance breakdown of the GPU-accelerated optimization framework using RK4 with structure-exploiting gradient computation for a 180° rotation.

Operation Category	**Time (s)**	**Percentage (%)**
Total Optimization Runtime	5.753	100.0
GPU Operations		
Kernel Execution	0.248	4.3
Memory Transfers	0.158	2.7
Total GPU Active Time	0.406	7.1
CPU Operations		
IPOPT Solver Overhead	1.815	31.5
Function Evaluation Overhead	0.679	11.8
Total CPU Computation Time	2.494	43.4
CPU-GPU Synchronization Overhead	2.853	49.6

4.1.3 NLP Solver Comparison

Using a multiple shooting discretization scheme with $N = 50$ intervals, employing RK4 integration with automatic differentiation on the CPU for dynamics propagation, two interior-point NLP solvers are evaluated: IPOPT and FATROP, a structured optimal control problem solver exploiting the special block-sparse structure of multiple shooting formulations (section 2.6.1). The comparative performance analysis presented in table 4.8 demonstrates that both solvers converge to equivalent solutions across all tested maneuver angles, with identical control switching structures and maneuver durations within 0.005–0.012% of the theoretical optimal values. However, FATROP achieves computation times of 0.085–0.191 s compared to IPOPT's 0.961–2.512 s, representing a consistent 9.0–14.9× speedup factor. This performance advantage is attributed to FATROP's exploitation of the optimal control problem structure through specialized condensing and Riccati recursion algorithms, which reduce computational complexity relative to general-purpose interior-point methods. The sub-200 ms computation times achieved by FATROP satisfy the real-time feasibility requirements set in section 1.2. Consequently, FATROP is selected as the NLP solver for all subsequent computational experiments in this thesis.

Table 4.8 Performance comparison of IPOPT and FATROP NLP solvers for different maneuver angles.

Angle	Metric	Optimal	IPOPT	FATROP
45°	Computation time [s]	–	1.265	0.085
	Control switches [-]	6	6	6
	Maneuver duration [s]	1.7471	1.7472	1.7472
	Optimality gap Δ_{opt} [%]	0	0.005	0.005
90°	Computation time [s]	–	0.961	0.107
	Control switches [-]	5	5	5
	Maneuver duration [s]	2.4211	2.4214	2.4214
	Optimality gap Δ_{opt} [%]	0	0.012	0.012
180°	Computation time [s]	–	2.512	0.191
	Control switches [-]	5	5	5
	Maneuver duration [s]	3.2431	3.2434	3.2434
	Optimality gap Δ_{opt} [%]	0	0.009	0.009

4.2 Algorithmic Reliability Analysis

To assess algorithmic reliability and computational performance, three distinct scenarios were evaluated through MC simulation. Scenario A represents symmetric spacecraft executing rest-to-rest maneuvers. Scenario B corresponds to symmetric spacecraft performing track-to-track maneuvers. Scenario C examines asymmetric spacecraft executing track-to-track maneuvers, representing the most challenging configuration due to gyroscopic coupling effects.

For each scenario, 10,000 boundary condition sets were randomly sampled using LHS to ensure uniform coverage of the parameter space. All trials employed an identical fixed initial guess corresponding to a 90° rest-to-rest minimum-time rotation.

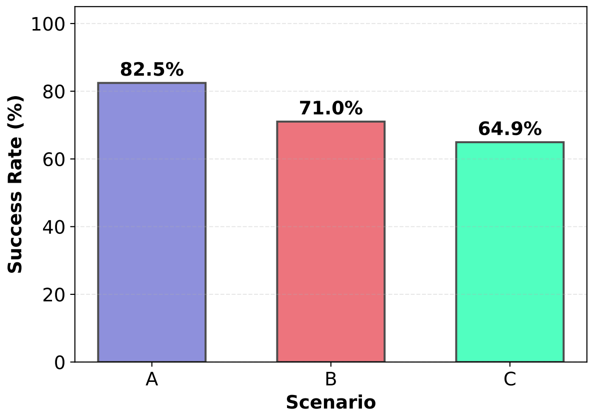
4.2.1 Success Rate

Figure 4.5 presents the convergence success rates for both algorithms across all scenarios. The CGPOPS benchmark algorithm demonstrates sensitivity to initialization quality, with success rates declining from 82.5% in scenario A to 64.9% in scenario C. This degradation reflects the increasing difficulty of the optimization landscape as problem complexity increases through the introduction of non-zero boundary velocities and inertial asymmetry.

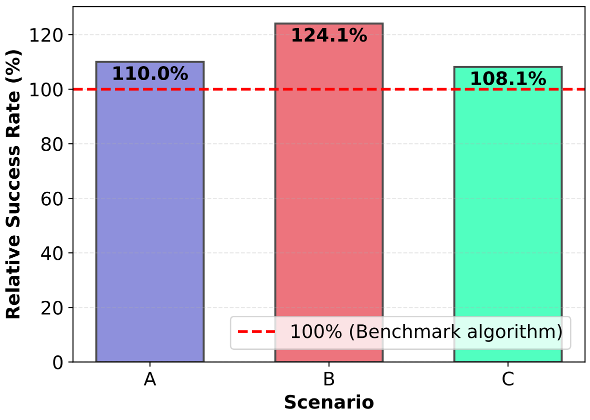
The optimized algorithm exhibits substantially improved reliability, achieving relative success rates exceeding 100% in all scenarios. This counterintuitive result arises from the employed evaluation methodology: when CGPOPS failed to converge, solutions from the optimized algorithm were classified as successful if they satisfied convergence tolerances for both optimality and constraint violation. The relative success rate is computed as the ratio of successful trials from the optimized algorithm to successful trials from CGPOPS, multiplied by 100%. Values exceeding 100% indicate that the optimized algorithm converged to feasible, optimal solutions in cases where the benchmark failed. Scenario B exhibits the highest relative improvement (124.1%), corresponding to an absolute convergence rate of 88%, while scenario A experiences the highest absolute convergence rate of 91%. Scenario C, although still outperforming the benchmark algorithm, only has a 70% absolute convergence rate.

4.2.2 Computation Time

Figure 4.6 illustrates the computational performance characteristics of both algorithms. The CGPOPS benchmark algorithm exhibits median computation times ranging from approximately 2 s in



(a) CGPOPS benchmark algorithm.



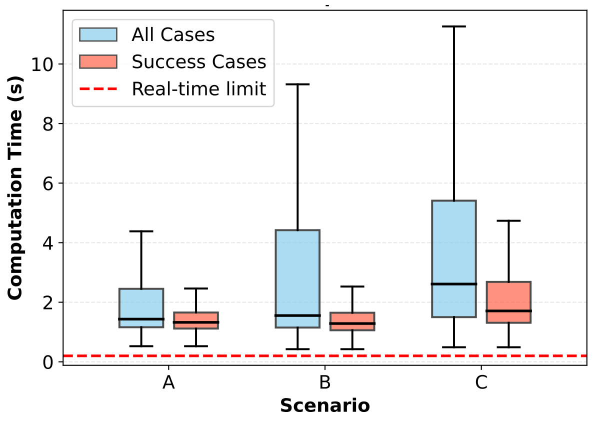
(b) Optimized algorithm.

Figure 4.5 Success rate of the developed algorithms over the 3 different scenarios.

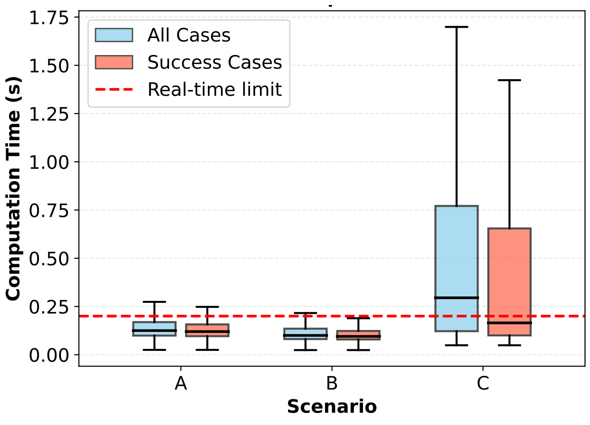
scenario A to 3 s in scenario C, with individual trials occasionally exceeding 10 s. This computational burden renders the benchmark unsuitable for real-time applications.

The optimized algorithm demonstrates dramatically reduced computation times across all scenarios. For scenarios A and B, the median computation time remains below the 0.2 s real-time threshold indicated by the dashed red line, with median values of approximately 0.125 s and 0.100 s respectively. Scenario C exhibits increased computational demand due to asymmetric inertia characteristics, with median computation time 0.295 s. However, when considering only successful convergence cases, the median computation time for scenario C remains 0.165 s, still satisfying real-time requirements. The computational overhead in scenario C stems from the more complex optimization landscape introduced by gyroscopic coupling terms in Euler's equations.

The substantially tighter interquartile ranges observed in the optimized algorithm, compared to the benchmark, indicate more consistent and predictable computational performance, which is critical for operational deployment where deterministic timing guarantees are essential.



(a) CGPOPS benchmark algorithm.



(b) Optimized algorithm.

Figure 4.6 Computation time of the developed algorithms over the 3 different scenarios.

4.2.3 Solution Optimality

To quantify solution quality in cases where the optimized algorithm failed to converge but CGPOPS succeeded, the optimality gap was computed as the percentage difference in maneuver time relative to the CGPOPS solution. Figure 4.7 presents the distribution of this metric across non-converged trials. The goal of this analysis is to understand if the non-optimal solutions are acceptable even if they do not correspond to the fastest trajectory.

Scenario A exhibits minimal optimality degradation, with median deviation of approximately 0.2% and with the 75th percentile remaining below 1%. This near-optimal performance suggests that even non-converged solutions from the optimized algorithm provide trajectories of practical utility for symmetric rest-to-rest maneuvers.

Scenario B demonstrates increased optimality gap, with median deviation near 0.5%. The elevated gap reflects the additional complexity introduced by non-zero boundary velocities, which expand the feasible solution space and increase the probability of convergence to suboptimal local minima.

Scenario C presents the largest optimality gaps, with median deviation of approximately 1% and the third quarter percentile reaching 8.8%. The increased gap in asymmetric configurations correlates with the more rugged optimization landscape characterized by multiple local minima separated by regions of infeasibility, consequence of gyroscopic coupling effects.

Notably, even in worst-case scenarios, the optimality gap remains below 9%, indicating that the optimized algorithm consistently identifies solutions in the vicinity of the global optimum. Nevertheless, it is decided to improve these results with the introduction of PSO initialization strategies to understand how dependent they are on the initial guess.

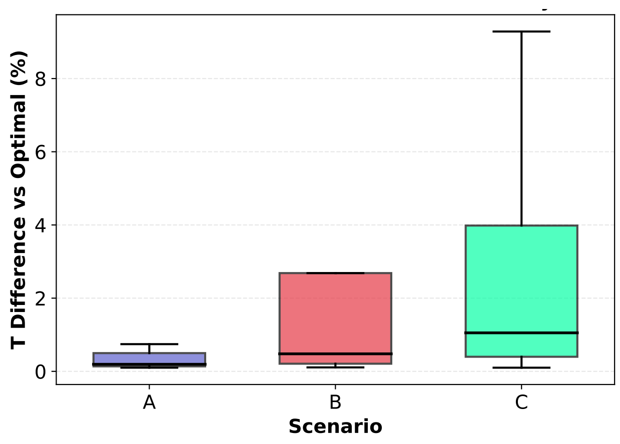


Figure 4.7 Optimality gap for non-converged runs of the optimized algorithm.

4.3 PSO Development

The reliability analysis presented in section 4.2 demonstrated that while the optimized algorithm achieves substantial improvements over the CGPOPS benchmark, convergence success rates decline in complex scenarios, particularly for asymmetric spacecraft configurations. Additionally, the optimality gap analysis revealed that non-converged solutions can deviate from the global optimum by up to 9%. These observations motivate the investigation of metaheuristic initialization strategies to enhance both reliability and solution quality.

This section presents the development of two PSO-based initialization approaches: full control trajectory parameterization and STO parameterization. The objective is to position the NLP solver within the convergence basin of the global optimum, thereby improving both success rate and solution quality across all scenario classes.

4.3.1 Parameter Tuning Results

Following the LHS-based parameter tuning methodology detailed in section 3.4.2, optimal PSO configurations for both parameterization approaches were identified through evaluation of 1000 parameter combinations across 1000 independent maneuver scenarios each. Table 4.9 presents the resulting optimal parameter sets, which exhibit substantial differences reflecting the distinct search space characteristics of full control trajectory versus STO parameterization.

Table 4.9 PSO optimal parameters.

Category	**Parameter**	**Full Control**	**STO**
Population	Population size $\mathcal{N}$	3200	3200
	Iteration count $\mathcal{K}$	10	100
Inertia Weight	Initial weight w_0	3.5	5.6
	Minimum weight $w_{0,\text{min}}$	0.0	3.2
Cognitive Parameter	Initial weight w_1	4.8	4.8
	Minimum weight $w_{1,\text{min}}$	1.5	1.7
Social Parameter	Initial weight w_2	1.7	6.0
	Minimum weight $w_{2,\text{min}}$	1.5	6.0
Binary PSO	Sigmoid scaling α	—	7.9
	Sigmoid saturation ϵ	—	0.57

The optimal configurations reveal fundamentally different search strategies driven by the dimensionality disparity between parameterization approaches. Full control trajectory parameterization, with its $O(151)$-dimensional search space, performs optimally with only 10 iterations, which is compensated with an aggressive inertia weight decay ($w_{0,\text{min}} = 0.0$, representing complete transition to exploitation). This rapid convergence strategy reflects the challenge of high-dimensional optimization where exhaustive exploration is computationally prohibitive.

In contrast, STO parameterization exploits its substantially reduced $O(10)$-dimensional search space through sustained exploration over 100 iterations with maintained inertia ($w_{0,\text{min}} = 3.2$). The tenfold increase in iteration count, while remaining within computational budget, enables more thorough coverage of the compact search space. The preserved high social parameter ($w_2 = w_{2,\text{min}} = 6.0$) promotes convergence toward global best positions, appropriate for initialization strategies, whose focus is purely finding the global minimum basin and not the minimum itself.

Both parameterizations converge on identical population sizes $\mathcal{N} = 3200$–five times the available GPU core count–balancing parallel efficiency with population diversity, important for good parameter space exploration. The cognitive parameter initialization ($w_1 = 4.8$) remains consistent across both approaches, indicating that individual particle memory contributes similarly to search effectiveness regardless of dimensionality.

The binary PSO parameters for STO parameterization ($\alpha = 7.9$, $\epsilon = 0.57$) indicate preference for sharp sigmoid transitions with moderate stochasticity. The high sigmoid scaling creates decisive initial control sign switching decisions while the saturation threshold slightly above the minimum ($\epsilon = 0.57 > 0.5$) leads to an almost purely stochastic behavior, which forces exploration of discrete control directions even with high particle velocities. Again this is expected for an initialization strategy whose goal is to properly explore the solution space and find a global optimum basin.

4.3.2 Trajectory Visualization

To visualize the difference between the trajectories generated by both approaches, they are are evaluated on a 90° rest-to-rest reorientation about the roll axis, corresponding to the optimal maneuver analyzed in figure 4.1 and table 3.2. From analytical solutions it is known that it exhibits 5 control switches and a maneuver duration of 2.4211 s.

Figure 4.8 presents the trajectory generated by PSO with full control trajectory parameterization. The Euler angle trajectory (figure 4.8a) demonstrates successful completion of the 90° rotation, though the quaternion evolution (figure 4.8b) exhibits deviations from the smooth hyperspheric arc characteristic of optimal solutions. The angular velocity profile (figure 4.8c) displays significant oscillations throughout the maneuver rather than the symmetric acceleration-deceleration pattern observed in optimal trajectories. Most notably, the control torque history (figure 4.8d) exhibits highly oscillatory behavior with frequent switching, deviating substantially from the optimal bang-bang structure. Despite satisfying the final state constraints, the maneuver duration is 5.9078 s, a 144% increase relative to the optimal value.

The excessive oscillations in the full control parameterization trajectory reflect the high-dimensional optimization landscape ($O(151)$ decision variables) and the limited exploitation phase (10 iterations with aggressive inertia decay to $w_{0,\min} = 0.0$). The PSO algorithm prioritizes exploration over refinement, yielding feasible but inefficient trajectories. However, the key advantage of this approach lies in its ability to satisfy terminal constraints while providing the gradient-based solver with a feasible starting point within the global convergence basin.

Figure 4.9 presents the trajectory generated by PSO with STO parameterization. The control torque history (figure 4.9d) exhibits the prescribed bang-bang structure with exactly 5 switches, matching the optimal control topology. The Euler angle trajectory (figure 4.9a) demonstrates smooth monotonic rotation, and the angular velocity profile (figure 4.9c) displays no oscillations, approaching the symmetric profile of the optimal solution. The maneuver duration of 2.2880 s represents only a 5.5% deviation from the optimal solution, indicating proximity to the global optimum. However, final state constraints are violated, with the spacecraft failing to achieve the required zero angular velocity and 90° roll attitude at the final time.

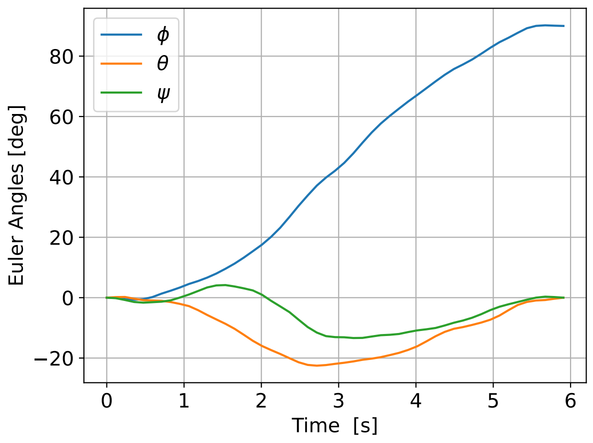
The superior trajectory structure and near-optimal maneuver duration of the STO parameterization stem from the substantially reduced search space ($O(10)$ decision variables) and extended exploration phase (100 iterations with maintained inertia $w_{0,\min} = 3.2$). However, the constraint violation indicates that while STO parameterization excels at identifying the correct control structure, the discrete nature of the switching topology introduces challenges in precisely satisfying boundary conditions.

These observations suggest that PSO serves most effectively as an initialization strategy rather than a standalone solver. The metaheuristic phase positions the gradient-based solver within the appropriate convergence basin (either a feasible region (full control) or a structurally correct region (STO)) from which local refinement can converge to the constrained optimum. The subsequent reliability analysis (section 4.4) quantifies the extent to which these initialization strategies enhance algorithmic performance across diverse scenario classes.

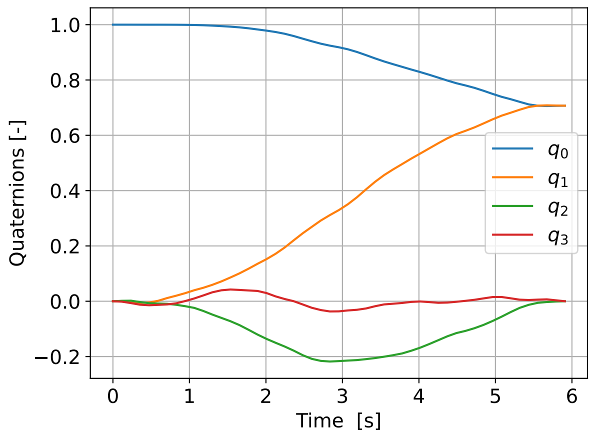
Figure 4.9 presents the trajectory generated by PSO with STO parameterization. The control torque history (figure 4.9d) exhibits the prescribed bang-bang structure with exactly 5 switches, matching the optimal control topology. The Euler angle trajectory (figure 4.9a) demonstrates smooth monotonic rotation, and the angular velocity profile (figure 4.9c) displays no oscillations, approaching the symmetric profile of the optimal solution. The maneuver duration of 2.2880 s represents only a 5.5% deviation from the optimal solution, indicating proximity to the global optimum. However, final state constraints are violated, with the spacecraft failing to achieve the required zero angular velocity and 90° roll attitude at the final time.

The superior trajectory structure and near-optimal maneuver duration of the STO parameterization stem from the substantially reduced search space ($O(10)$ decision variables) and extended exploration phase (100 iterations with maintained inertia $w_{0,\min} = 3.2$). However, the constraint violation indicates that while STO parameterization excels at identifying the correct control structure, the discrete nature of the switching topology introduces challenges in precisely satisfying boundary conditions.

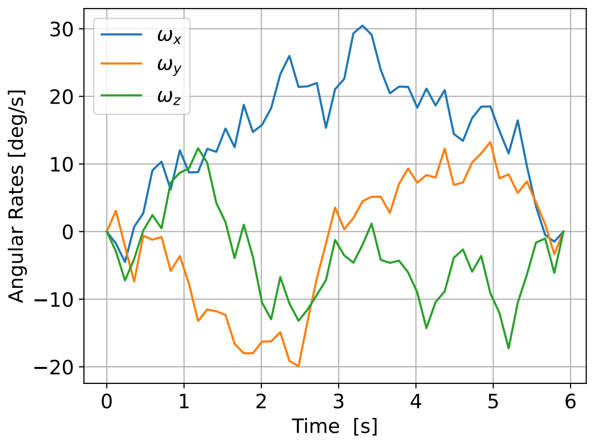
These observations suggest that PSO serves most effectively as an initialization strategy rather than a standalone solver. The metaheuristic phase positions the gradient-based solver within the appropriate convergence basin (either a feasible region (full control) or a structurally correct region (STO)) from which local refinement can converge to the constrained optimum. The subsequent



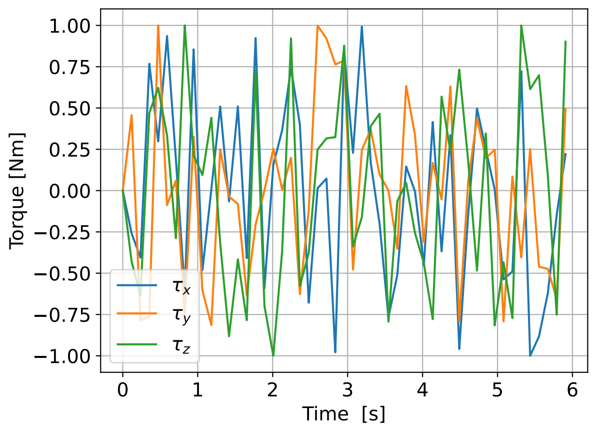
(a) Euler angles trajectory.



(b) Quaternion trajectory.

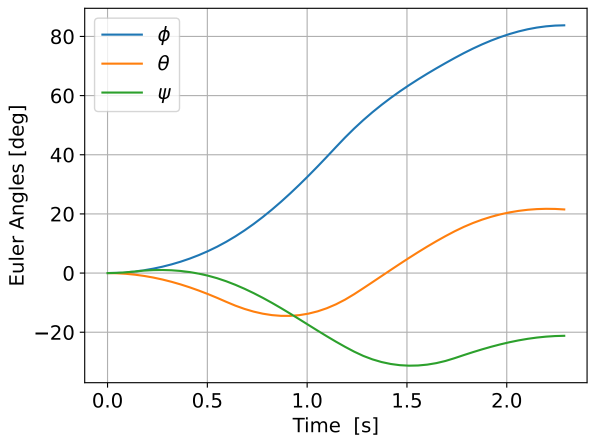


(c) Angular velocity components.

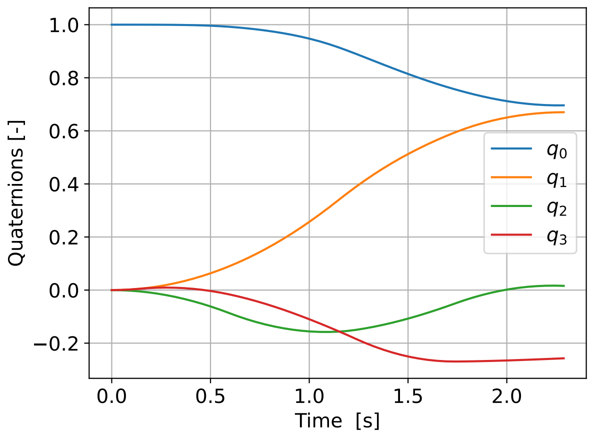


(d) Control torque inputs.

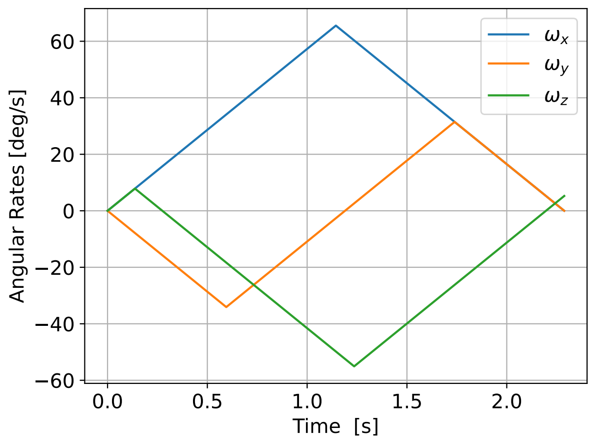
Figure 4.8 Output trajectory generated by the PSO with full control trajectory parameterization.



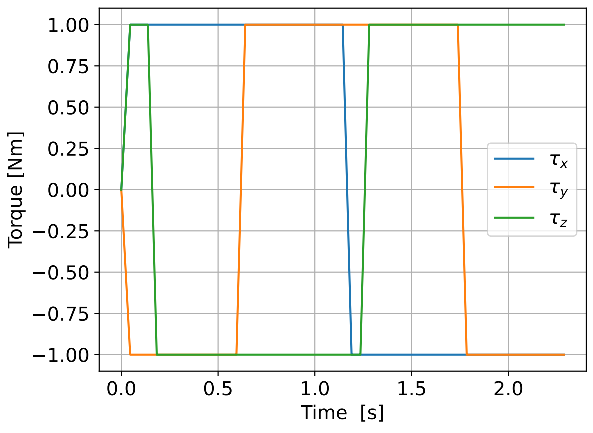
(a) Euler angles trajectory.



(b) Quaternion trajectory.



(c) Angular velocity components.



(d) Control torque inputs.

Figure 4.9 Output trajectory generated by the PSO with STO parameterization.

reliability analysis (section 4.4) quantifies the extent to which these initialization strategies enhance algorithmic performance across diverse scenario classes.

4.4 PSO Reliability Analysis

The reliability analysis presented in section 4.2 is extended to evaluate the performance of PSO-based initialization strategies. The two developed PSO variants are examined, where: "PSO Full" refers to the optimized algorithm initialized with PSO with full control trajectory parameterization, "PSO STO" refers to PSO with switch time optimization parameterization. The analysis compares these approaches against both the benchmark CGPOPS algorithm and the optimized algorithm without PSO initialization, referred to as "No PSO".

4.4.1 Success Rate

Figure 4.10 presents the success rates achieved by the different initialization strategies across all three scenarios. The initialization approach demonstrates varying effects on convergence reliability depending on the scenario complexity and spacecraft configuration.

For scenarios A and B, "PSO Full" achieves slightly better success rates than "No PSO". Contrarily, PSO STO exhibits a more pronounced reduction in success rates when compared to "No PSO", however, it still outperforms the benchmark algorithm.

The most significant deviation occurs in scenario C, where PSO-based initialization demonstrates unexpected behavior. "No PSO" achieves a higher success rate than both PSO-initialed approaches. Most notably, "PSO STO" shows the lowest success rate among all methods for this scenario, even falling below the benchmark baseline by 2%.

The reduced success rates observed with "PSO STO", particularly for asymmetric spacecraft configurations, can be attributed to the constrained solution space imposed by the fixed bang-bang control structure. The STO parameterization inherently restricts the control trajectory to a predefined structure, creating steeper convergence basins in the solution landscape. When PSO-STO converges to a suboptimal basin, the subsequent gradient-based refinement encounters difficulty escaping due to the high quality of the non-optimal minimum, resulting in convergence to local minima rather than the global optimum. This effect is amplified in scenario C, where the asymmetric inertia distribution increases the complexity of the optimal control structure.

The counterintuitive result that the PSO initialization, implemented specifically to enhance reliability for asymmetric spacecraft, performs worst in scenario C suggests that the current PSO implementations may require additional tuning of exploration-exploitation trade-offs. The PSO tuning was carried out with rest-to-rest maneuvers performed with symmetric spacecraft. The results show that the tuning parameters may be dependent on spacecraft configuration and need personalized tuning.

4.4.2 Computation Time

Figures 4.11 and 4.12 present the computation time distributions for all cases and just for successfully converged cases, and breakdown of time spent in PSO versus gradient-based solver phases. The results reveal distinct performance characteristics across scenarios and parameterization approaches.

For scenarios A, both PSO variants achieve faster overall computation times compared to "No PSO". As shown in figures 4.11a and 4.11b, "PSO Full" demonstrates a median computation time 0.007 s slower than "PSO STO" and 0.005 s faster than "No PSO". For scenario B both PSO implementations have a median 0.006 s skower than "No PSO". Both methods maintain computation times substantially below the 0.2 s real-time threshold.

The time breakdown analysis in figure 4.12 reveals that PSO computation time remains approximately constant across scenarios, with "PSO Full" requiring 0.08 s of PSO time, while "PSO STO"

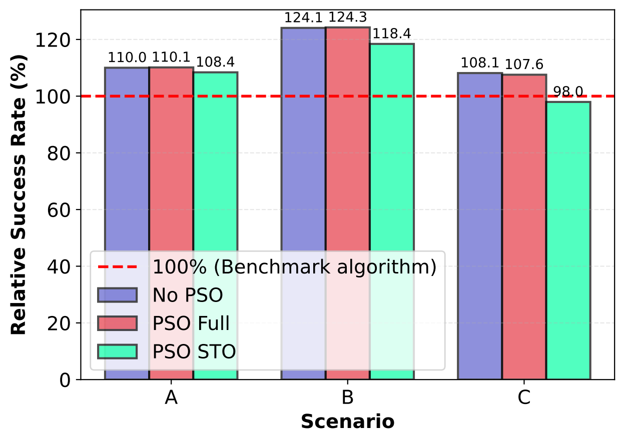
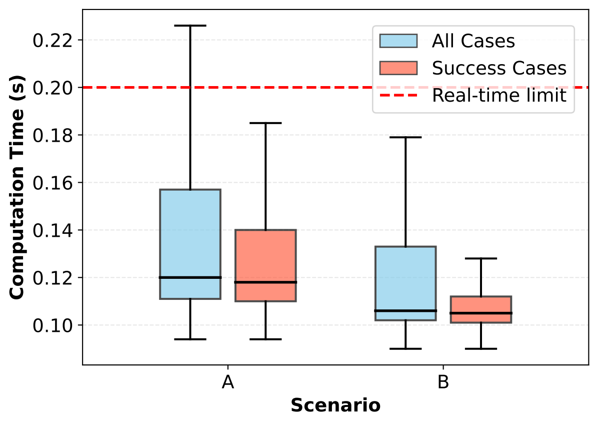
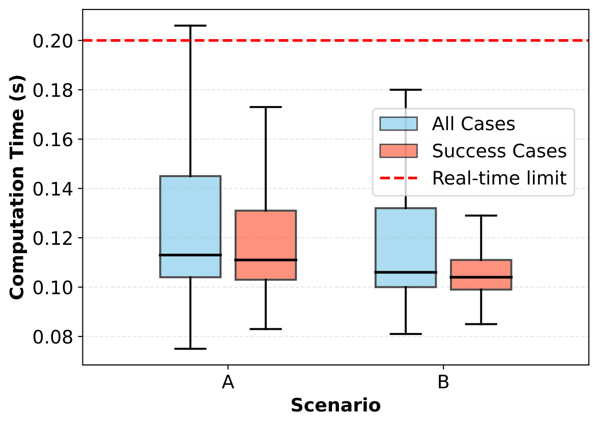


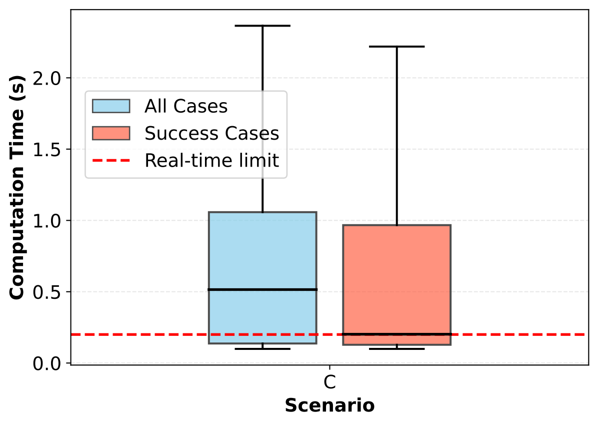
Figure 4.10 Success rate of the developed algorithm with different initialization strategies.



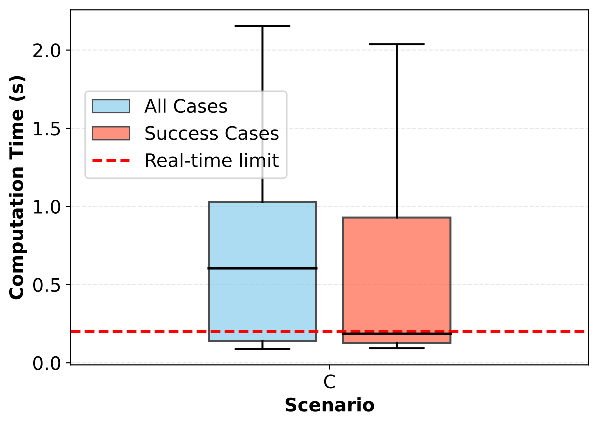
(a) PSO with full control trajectory parameterization (scenarios A and B).



(b) PSO with STO control parameterization (scenarios A and B).



(c) PSO with full control trajectory parameterization (scenario C).

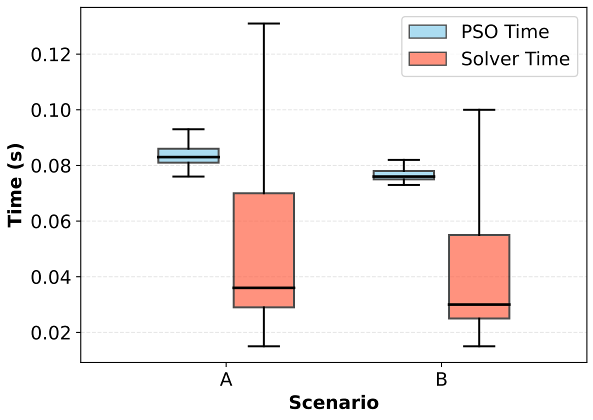


(d) PSO with STO control parameterization (scenario C).

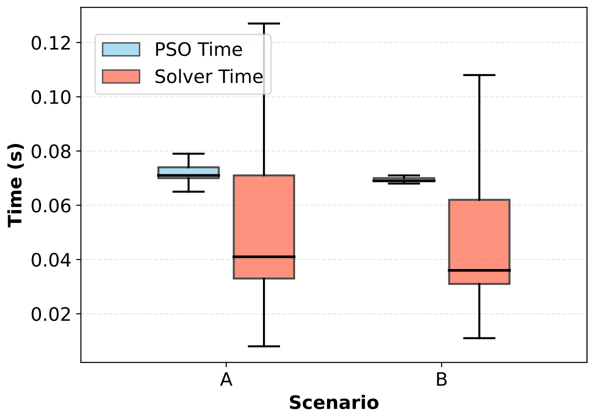
Figure 4.11 Computation time of the optimized algorithm with PSO initialization.

requires 0.07 s. The consistency of the PSO phase duration demonstrates the predictable computational cost of the metaheuristic initialization phase. Notably, the gradient-based solver time following PSO initialization is significantly reduced compared to the approximately 0.12 s of solver time in the "No PSO" for scenarios A and B, with "PSO Full" achieving a median solver time of 0.03 s, and "PSO STO" 0.04 s, indicating that improved initialization quality reduces the refinement burden on the NLP solver.

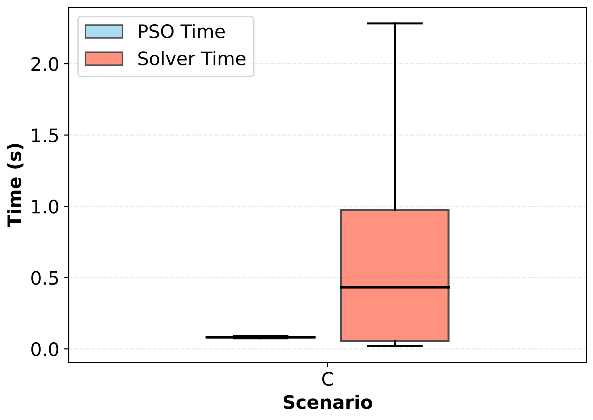
Scenario C presents a contrasting behavior, as shown in figures 4.11c, 4.11d, 4.12c and 4.12d. Despite PSO maintaining consistent computation times, the gradient-based solver requires substantially longer refinement periods, with median solver times of approximately 0.4 s for "PSO Full" and 0.5 s for "PSO STO". This increased solver time results in overall median computation times of 0.515 s and 0.605 s, respectively, exceeding the 0.2 s real-time threshold in 62.01% ("PSO Full") and 62.96% ("PSO STO") of cases. The extended solver times in scenario C, combined with the previously discussed success rate reductions, indicate that PSO initialization provides less effective starting points for asymmetric configurations, requiring more extensive gradient-based refinement and potentially converging to suboptimal local minima.



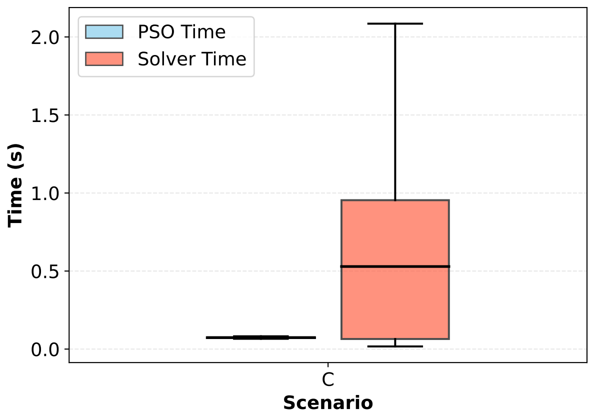
(a) PSO with full control trajectory parameterization (scenarios A and B).



(b) PSO with STO control parameterization (scenarios A and B).



(c) PSO with full control trajectory parameterization (scenario C).



(d) PSO with STO control parameterization (scenario C).

Figure 4.12 Discriminated computation time between time spent on PSO and on gradient-based solver.

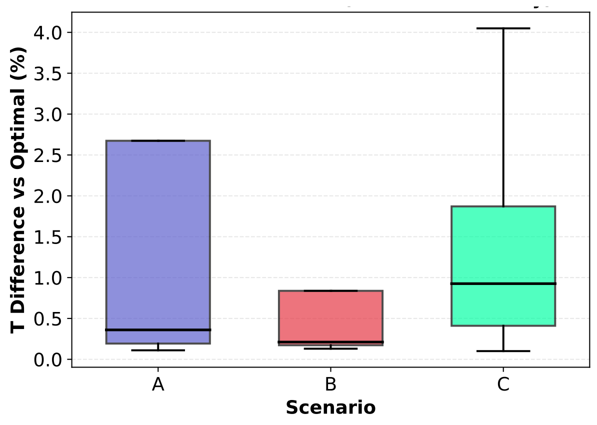
4.4.3 Solution Optimality

Figure 4.13 presents the optimality gap analysis for non-converged runs, quantified as the difference in maneuver time relative to converged CGPOPS solutions. The results demonstrate that parameterization approach significantly influences solution quality in failed convergence cases.

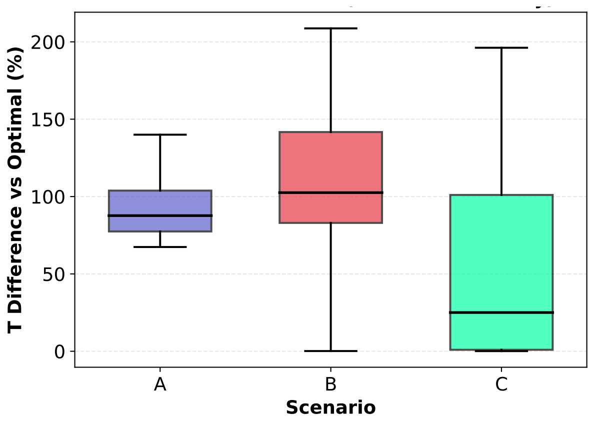
"PSO Full" exhibits relatively modest optimality gaps across all scenarios, as shown in figure 4.13a. Even scenario C, despite having the lowest success rate, maintains a median optimality gap near 1.0% though with increased variability extending to approximately 4.0% in extreme cases. These results indicate that when PSO Full fails to converge, the resulting suboptimal solutions remain relatively close to the optimal maneuver time.

In contrast, "PSO STO" demonstrates substantially larger optimality gaps, as illustrated in figure 4.13b. All scenarios exhibit a dramatic increase, with medians near 100% and the error extending to 200% in scenarios B and C. These pronounced optimality gaps directly correlate with the structural constraints imposed by switch time optimization parameterization. The fixed bang-bang control structure creates distinct convergence basins in the solution space, each corresponding to a particular switching sequence. When the PSO-STO converges to an incorrect switching sequence the gradient-based solver encounters steep basin walls, resulting in convergence to a locally optimal solution within the incorrect basin.

This behavior suggests that "PSO STO" would benefit from parameter tuning that emphasizes solution refinement over exploration, potentially allowing the swarm to escape suboptimal basins before transitioning to gradient-based refinement. Additionally, implementing diversity-preserving mechanisms or multi-start strategies within the PSO phase could improve the probability of identifying the correct convergence basin, particularly for asymmetric configurations where the optimal switching structure is less intuitive.



(a) PSO with full control trajectory parameterization.



(b) PSO with STO control parameterization.

Figure 4.13 Optimality gap for non-converged runs of the optimized algorithm with PSO initialization.

5 Conclusion

This thesis addresses the challenge of computing time-optimal spacecraft attitude slew maneuvers with computation times compatible with real-time onboard implementation. The research question is answered through investigation of algorithmic optimization strategies across transcription methods, integration schemes, NLP solvers, and initialization approaches.

The optimization framework employs multiple shooting transcription with RK4 integration, automatic differentiation, and the FATROP structured NLP solver. This configuration achieves median computation times of 0.100–0.115 s for symmetric spacecraft executing rest-to-rest and track-to-track maneuvers, satisfying the 200 ms real-time threshold established in section 1.2. For asymmetric spacecraft configurations, the median computation time is 0.165 s for successfully converged cases marginally satisfying real-time requirements, though the overall median including failed convergences is 0.295 s.

Convergence success rates of 91% for symmetric rest-to-rest maneuvers, 88% for symmetric track-to-track maneuvers, and 70% for asymmetric track-to-track maneuvers are achieved with the optimized algorithm. These results represent substantial improvements over the CGPOPS benchmark algorithm, which exhibits success rates of 82.5%, 71.0%, and 64.9% for the corresponding scenarios while requiring computation times of 2–3 s. The optimized algorithm thus achieves approximately 10–15$\times$ computational speedup with 8–26% relative improvement in convergence reliability.

Non-optimal solution quality verification confirmed that even non-converged solutions provide trajectories of practical utility, with 75% of optimality gaps remaining below 9% and median deviations of 0.2–1.0% depending on the scenario.

The investigation of PSO-based metaheuristic initialization strategies revealed configuration-dependent performance characteristics. For symmetric spacecraft, PSO with full control trajectory parameterization achieved negligible improvements in success rate (0.1–0.2%) while requiring similar computation time, despite reducing gradient-based solver time by approximately 75%. However, for asymmetric configurations, PSO initialization degraded both success rates and computation times relative to gradient-based optimization from fixed initial guesses. This suggests that the PSO parameter tuning, conducted exclusively on symmetric rest-to-rest maneuvers, does not generalize effectively to asymmetric spacecraft dynamics. The optimal configuration for general application therefore omits PSO initialization in favor of gradient-based optimization from standardized initial guesses.

The research demonstrates that real-time time-optimal spacecraft attitude control is achievable with the developed optimized algorithm. The sub-200 ms computation times attained for symmetric spacecraft configurations enable closed-loop trajectory replanning at star tracker update rates, facilitating autonomous response to mission requirements and environmental disturbances.

5.1 Limitations

Although the optimized algorithm was developed with the objective of being as inclusive as possible of every operational scenario, several limitations inherent to the current implementation warrant acknowledgment.

5.1.1 Modeling Assumptions and Scope

The spacecraft dynamics formulation assumes rigid body motion, neglecting flexibility effects that may be significant for spacecraft with large appendages or low structural frequencies. The applicability of the developed algorithm is therefore restricted to spacecraft configurations where rigid body assumptions provide adequate fidelity. Additionally, the optimal control problem formulation does not incorporate control rate constraints, despite actuator slew rate limitations being present in physical reaction wheel and control moment gyroscope systems. The omission of control rate constraints may result in trajectories that, while theoretically time-optimal, are infeasible for implementation on systems with finite actuator bandwidth.

The implemented framework requires spacecraft inertia tensor properties to be specified at compile time as constants. This design decision, while enabling certain compiler optimizations, precludes runtime adaptation to time-varying mass properties during mission execution. For spacecraft experiencing significant propellant consumption or configuration changes through deployment mechanisms, this limitation necessitates either acceptance of degraded solution accuracy as inertia properties deviate from compiled nominal values, or periodic recompilation and redeployment of the optimization algorithm with updated parameters.

5.1.2 Numerical Integration and Asymmetric Spacecraft

While the developed algorithm demonstrates robust performance for symmetric spacecraft configurations across both rest-to-rest and track-to-track maneuvers, the reliability degradation observed with an asymmetric spacecraft suggests that additional investigation is warranted.

The RK4 integration method, while computationally efficient and accurate for symmetric and mildly asymmetric configurations, has not been verified for spacecraft with highly disparate principal moments of inertia. The cross-coupling terms in Euler's equations become increasingly dominant, potentially requiring tighter integration tolerances or higher-order schemes to maintain solution accuracy. The impact of extreme inertia asymmetry on integration error accumulation, constraint satisfaction, and convergence basin geometry requires further investigation.

5.1.3 PSO Parameter Tuning Scope

The PSO parameter optimization was conducted using only 1000 parameter combinations, each evaluated across 1000 maneuver scenarios. While this approach identified configurations yielding good performance for scenarios A and B, the results suggest that optimal PSO parameters exhibit sensitivity to spacecraft configuration. The parameter tuning was performed exclusively for symmetric spacecraft executing rest-to-rest maneuvers, and the resulting configurations demonstrated suboptimal performance when applied to asymmetric configurations. The scope of the conducted tuning study was constrained by available computational resources and project timeline.

5.1.4 Computational Hardware

The computational performance results presented throughout this thesis were obtained using a consumer-grade 2017 gaming laptop platform. The reported computation times therefore represent performance on hardware substantially less capable than modern spacecraft flight computers or contemporary space-qualified CPU and GPU architectures in terms of core count, memory bandwidth, and floating-point throughput.

5.2 Future Work

Several avenues for extending and enhancing the developed optimization framework have been identified through the research conducted in this thesis.

5.2.1 Extended Dynamical Models

The rigid body assumption employed in the current formulation could be relaxed to incorporate flexible body dynamics for spacecraft with significant structural flexibility. This extension would require coupling of attitude dynamics with structural deformation modes, increasing problem dimensionality but enabling application to a broader class of spacecraft configurations. Additionally, the inclusion of control rate constraints in the optimal control problem formulation would produce trajectories that account for finite actuator bandwidth limitations, improving implementability on physical spacecraft systems.

5.2.2 Runtime Inertia Adaptation

The compile-time specification of inertia tensor properties could be replaced with a runtime parameterization approach that accepts time-varying mass properties as inputs to the optimization algorithm. This modification would require careful consideration of potential performance implications from reduced compiler optimization opportunities.

5.2.3 Adaptive PSO Parameterization

A more comprehensive PSO parameter tuning methodology could be developed involving configuration-specific optimization with adaptive parameter scheduling based on inertia properties. Implementation of adaptive metaheuristic strategies or development of inertia-ratio-dependent parameter selection heuristics may yield improved convergence reliability for asymmetric spacecraft configurations. Expansion of the parameter search space beyond the 1000 combinations evaluated in this thesis, combined with evaluation across asymmetric spacecraft scenarios, would provide more robust parameter sets.

5.2.4 Path Constraint Integration

Extension of the optimal control problem formulation to incorporate state-dependent inequality constraints along the trajectory such as keep-out cones would enable modeling of operational constraints. This enhancement would require evaluation of the computational cost and convergence reliability implications of increasing problem dimensionality through path constraint addition.

5.2.5 Hardware-in-the-Loop Validation

Integration of the developed optimization framework with representative spacecraft attitude determination and control system components would provide higher-fidelity assessment of real-time feasibility and help identify potential implementation challenges not evident in purely numerical simulation environments. Hardware-in-the-loop validation should include actual sensor suites with measurement noise and bias characteristics, actuator dynamics models with saturation and slew rate limits, and flight software architectures with realistic timing constraints and computational resource limitations.

5.3 Closing Remark

The framework developed in this thesis establishes a foundation for autonomous spacecraft that can compute and execute time-optimal maneuvers within sensor update cycles, closing the control loop around optimal trajectory generation. This represents a significant step toward spacecraft systems capable of autonomous, optimally responsive operations in dynamic mission environments. The open-source implementation and comprehensive verification methodology provide a platform for continued advancement in real-time aerospace trajectory optimization, with potential extensions to multi-spacecraft coordination, path-constrained problems, and coupled translational-rotational dynamics.

Bibliography

[1] M. Wilde, J. Harder, and E. Stoll, “Editorial: On-orbit servicing and active debris removal: Enabling a paradigm shift in spaceflight,” *Frontiers in Robotics and AI*, vol. 6, p. 136, 2019.

[2] C. D. Petersen, “Advances in underactuated spacecraft control,” Ph.D. dissertation, University of Michigan, 2016.

[3] K. D. Bilimoria and B. Wie, “Time-optimal three-axis reorientation of a rigid spacecraft,” *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 3, pp. 446–452, May/Jun. 1993.

[4] X. Bai and J. L. Junkins, “New results for time-optimal three-axis reorientation of a rigid spacecraft,” *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 4, pp. 1071–1076, Jul./Aug. 2009.

[5] S. L. Scrivener and R. C. Thompson, “Survey of time-optimal attitude maneuvers,” *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 2, pp. 225–233, mar 1994.

[6] B. Pain, B. Hancock, C. Liebe, and J. Mellstrom, “Stellar gyroscope for determining attitude of a spacecraft,” Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, United States, NASA Tech Brief NPO-30481, Oct. 2005, document ID: 2011016302. [Online]. Available: http://www.techbriefs.com/component/content/518?task=view

[7] Y. M. Agamawi and A. V. Rao, “Cgpops: A c++ software for solving multiple-phase optimal control problems using adaptive gaussian quadrature collocation and sparse nonlinear programming,” 2019. [Online]. Available: https://arxiv.org/abs/1905.11898

[8] M. D. Shuster, “A survey of attitude representations,” *Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.

[9] R. M. Byers, “Time optimal attitude control of asymmetric rigid spacecraft,” *Journal of Vibration and Control*, vol. 2, no. 1, pp. 17–32, 1996, received 12 June 1995; Accepted 5 September 1995.

[10] A. Fleming, P. Sekhavat, and I. M. Ross, “Minimum-time reorientation of a rigid body,” *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 1, pp. 160–170, January–February 2010.

[11] T. Lee, M. Leok, and N. H. McClamroch, “Optimal attitude control of a rigid body using geometrically exact computations on SO(3),” *Journal of Optimization Theory and Applications*, 2006, arXiv:math/0601424.

[12] S. Kulumani and T. Lee, “Constrained geometric attitude control on SO(3),” *International Journal of Control, Automation, and Systems*, vol. 15, no. 6, pp. 2674–2685, December 2017, arXiv:1711.10992.

[13] S. T. McDonald, T. L. Griszel, and Z. Wang, “A real-time approach to minimum-energy reorientation of an asymmetric rigid body spacecraft,” in *AIAA Scitech 2020 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, January 2020, p. 1261.

[14] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes.* New York: Wiley-Interscience, 1962.

[15] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*, 3rd ed., ser. Advances in Design and Control. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2020, vol. 36.

[16] R. Kress, "Initial value problems," in *Numerical Analysis*, ser. Graduate Texts in Mathematics. New York, NY: Springer, 1998, vol. 181, ch. 10, pp. 225–257.

[17] D. J. Gardner, D. R. Reynolds, C. S. Woodward, and C. J. Balos, "Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, 2022.

[18] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.

[19] R. Serban and A. C. Hindmarsh, "CVODES: The sensitivity-enabled ODE solver in SUNDIALS," in *Proceedings of the ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 6. Long Beach, California, USA: ASME, September 2005, pp. 257–269.

[20] C. J. Balos, D. J. Gardner, C. S. Woodward, and D. R. Reynolds, "Enabling gpu accelerated computing in the sundials time integration library," *Parallel Computing*, vol. 108, p. 102836, 2021.

[21] 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*, vol. 41, no. 1, pp. 1:1–1:37, 2014.

[22] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[23] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Review*, vol. 47, no. 1, pp. 99–131, 2005.

[24] D. Spiller, L. Ansalone, and F. Curti, "Particle swarm optimization for time-optimal spacecraft reorientation with keep-out cones," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 312–325, February 2016.

[25] H. Seywald, "Trajectory optimization based on differential inclusion," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 3, pp. 480–487, 1994.

[26] G. A. Boyarko, M. Romano, and O. A. Yakimenko, "Time-optimal reorientation of a spacecraft using an inverse dynamics optimization method," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1197–1208, July-August 2011. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.49449

[27] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, "Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 036–10 043.

[28] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer, 2006.

[29] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[30] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control**this research was supported by the german federal ministry for economic affairs and energy (bmwi) via eco4wind (0324125b) and dyconpv (0324166b), and by dfg via research unit for 2401." *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320303293

[31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[32] D. Malyuta, Y. Yu, P. Elango, and B. Açıkmeşe, "Advances in trajectory optimization for space vehicle control," *Annual Reviews in Control*, vol. 52, pp. 282–315, 2021. [Online]. Available: https://doi.org/10.1016/j.arcontrol.2021.04.013

[33] J. A. E. Andersson, "A general-purpose toolbox for nonlinear optimization," CasADi Documentation, 2018, available: https://web.casadi.org.

[34] G. R. Hecht and E. M. Botta, "Particle swarm optimization-based co-state initialization for low-thrust minimum-fuel trajectory optimization," *Acta Astronautica*, vol. 211, pp. 416–430, 2023.

[35] R. G. Melton, "Hybrid methods for determining time-optimal, constrained spacecraft reorientation maneuvers," *Acta Astronautica*, vol. 94, pp. 294–301, 2014.

[36] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5. Orlando, FL, USA: IEEE, October 1997, pp. 4104–4108.

[37] C.-C. Wang, C.-Y. Ho, C.-H. Tu, and S.-H. Hung, "cuPSO: GPU parallelization for particle swarm optimization algorithms," 2023, arXiv preprint arXiv:2205.01313v2.

[38] Y. Zhuo, T. Zhang, F. Du, and R. Liu, "A parallel particle swarm optimization algorithm based on GPU/CUDA," *Applied Soft Computing*, vol. 144, p. 110499, 2023.

[39] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *ACM Queue*, vol. 6, no. 2, pp. 40–53, Mar./Apr. 2008.

[40] NVIDIA Corporation, *CUDA C++ Programming Guide*, NVIDIA Corporation, 2024, version 12.x. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/

[41] S. H. Jeon, S. Hong, H. J. Lee, C. Khazoom, and S. Kim, "CusADi: A GPU parallelization framework for symbolic expressions and optimal control," *arXiv preprint arXiv:2408.09662*, 2024. [Online]. Available: https://arxiv.org/abs/2408.09662

[42] N. F. Gade-Nielsen, "Interior point methods on GPU with application to model predictive control," PhD thesis, Technical University of Denmark, Department of Applied Mathematics and Computer Science, Kongens Lyngby, Denmark, Apr. 2014. [Online]. Available: https://orbit.dtu.dk

[43] J. L. Loeppky, J. Sacks, and W. J. Welch, "Choosing the sample size of a computer experiment: A practical guide," *Technometrics*, vol. 51, no. 4, pp. 366–376, 2009.

[44] M. D. McKay, R. J. Beckman, and W. J. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[45] M. Stein, “Large sample properties of simulations using Latin hypercube sampling,” *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[46] W. Zhang, W. Quan, and L. Guo, “Blurred star image processing for star sensors under dynamic conditions,” *Sensors*, vol. 12, no. 5, pp. 6712–6726, 2012. [Online]. Available: https://doi.org/10.3390/s120506712